INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, BHOPAL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BACHELORS OF TECHNOLOGY

DIGITAL LOGIC AND DESIGN

II YEAR, III SEM

Submitted by -

Gaurav Singla(20U02034)

Submitted to -

Dr. Deepa Sharma

NOVEMBER, 2021

EXPERIMENT LIST

1.) To verify and interpret the logic and truth table for AND, OR, NOT, NAND, NOR, Ex - OR, Ex - NOR gates using RTL (Resistor Transistor Logic), DTL (Diode Transistor Logic) and TTL (Transistor Transistor Logic).

2.) To design and implement 8:1 MUX using IC - 74LS153 and verify its truth table.

3.) To design and implement the given 4 variable function using IC74LS153 and verify its truth table.

4.) To design and implement 3-bit Gray to Binary code converter using IC-74LS138.

5.) To design and implement 3-bit Binary to Gray code converter using IC-74LS138

6.) To construct logic circuit of washing machine control using a generalized simulator and verify its output.

7.) To apply the basic OR gate logic in industrial control process and verify the truth table.

8.) To verify the truth table of half adder and full adder by using XOR and NAND gates respectively and analyse the working of half adder and full adder circuit with the help of LEDs and verify the truth table only of half adder and full adder.

9.) 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.

10.) To verify the truth table and timing diagram of NOR gate latch using NOR gate IC and analyse the circuit of NOR gate latch with the help of LEDs display.

11.) To verify the truth table and timing diagram of RS, JK, T and D flip-flops by using NAND & NOR gates ICs and analyse the circuit of RS, JK, T and D flip-flops with the help of LEDs display.

Experiment - 1

Aim: To verify and interpret the logic and truth table for AND, OR, NOT, NAND, NOR, Ex – OR, Ex – NOR gates using RTL (Resistor Transistor Logic), DTL (Diode Transistor Logic) and TTL (Transistor Transistor Logic) logics in simulator 1 and verify the truth table for AND, OR, NOT, NAND, NOR, Ex – OR, Ex – NOR gates in simulator 2.

Apparatus Required: Perform the experiment using this link: https://de-iitr.vlabs.ac.in/exp/truth-table-gates

Theory:

Logic gates are the basic building blocks of any digital system. Logic gates are electronic circuits having one or more than one input and only one output. The relationship between the input and the output is based on a certain logic. Based on this, logic gates are named as

1. AND gate
2. OR gate
3. NOT gate
4. NAND gate
5. NOR gate
6. Ex-OR gate
7. Ex-NOR gate

1. AND gate:

The AND gate is an electronic circuit that gives a high output (1) only if all its inputs are high. A dot (.) is used to show the AND operation i.e. A.B or can be written as AB.

$$Y = A.B$$

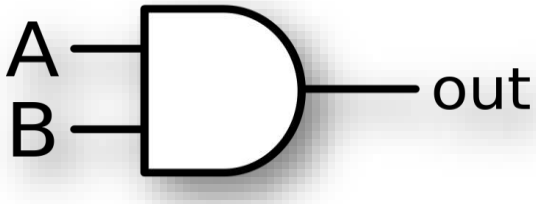


Fig 1: Logic Symbol of AND Gate

Input		Output
A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

Fig 2: Truth Table of AND Gate

A simple 2-input logic AND gate can be constructed using RTL (Resistor-Transistor-Logic) switches connected together as shown below with the inputs connected directly to the transistor bases. Both transistors must be saturated "ON" for an output at Q.

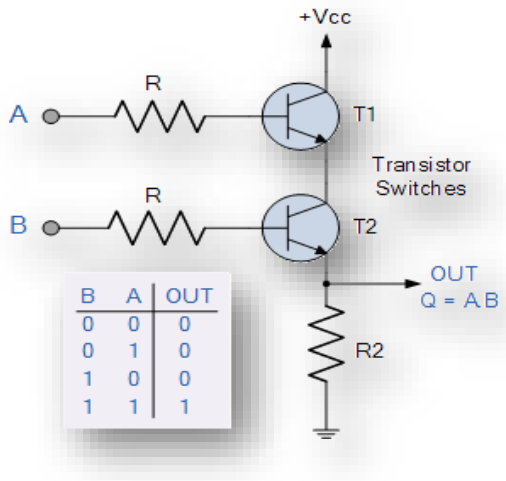


Fig 3: AND Gate through RTL Logic

2. OR Gate:

The OR gate is an electronic circuit that gives a high output (1) if one or more of its inputs are high. A plus (+) is used to show the OR operation.

$$Y = A + B$$

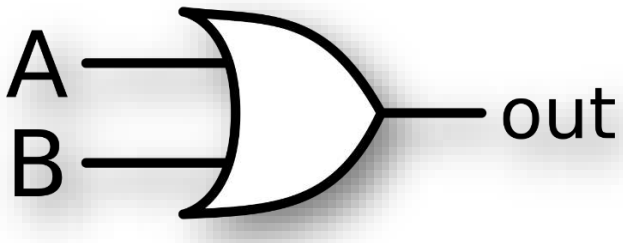


Fig 4: Logic Symbol of OR Gate

Input		Output
A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Fig 5: Truth Table of OR Gate

OR gate can be realized by DRL (Diode-Resistance-Logic) or by TTL (Transistor-Transistor-Logic). Presently, we will learn how to implement the OR gate using DRL (Diode-Resistance-Logic). To realise OR gate, we will use a diode at every input of the OR gate. The anode part of diode is connected with input while the cathode part is joined together and a resistor, connected with the cathode is grounded. In

this case, we have taken two inputs which can be seen in the circuit below.

When both the inputs are at logic 0 or low state then the diodes D1 and D2 become reverse biased. Since the anode terminal of diode is at lower voltage level than the cathode terminal, so diode will act as open circuit so there is no voltage across resistor and hence output voltage is same as ground. When either of the diodes is at logic 1 or high state then the diode corresponding to that input is forward bias. Since this time anode is at high voltage than cathode therefore current will flow through forward biased diode and this current then appears on resistor causing high voltage at output terminal also. Hence at output we get high or logic 1 or +5V. So, if any or both inputs are high, the output will be high or "1".

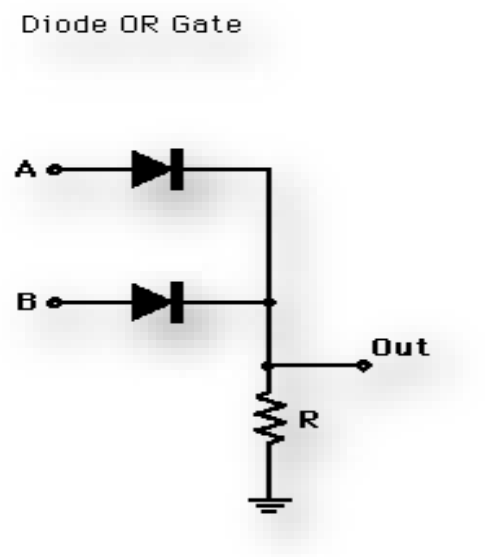


Fig 6: OR Gate through DRL Logic

3. NOT Gate:

The NOT gate is an electronic circuit that produces an inverted version of the input at its output. It is also known as an inverter. If the input variable is A, the inverted output is known as NOT A. This is also shown as A' or A with a bar over the top, as shown at the outputs.

Y = A’

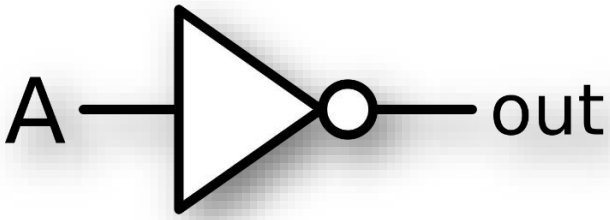


Fig 7: Logic Symbol of NOT Gate

Input	Output
A	Y
0	1
1	0

Fig 8: Truth Table of NOT Gate

NOT gate can be realized through transistor. The input is connected through resistor R2 to the transistor’s base. When no voltage is present on the input, the transistor turns off. When the transistor is off, no current flows through the collector-emitter path. Thus, current from the supply voltage (Vcc) flows through resistor R1 to the output. In this way, the circuit’s output is high when its input is low.

When voltage is present at the input, the transistor turns on, allowing current to flow through the collector-emitter circuit directly to ground. This ground path creates a shortcut that bypasses the output, which causes the output to go low.

In this way, the output is high when the input is low and low when the input is high.

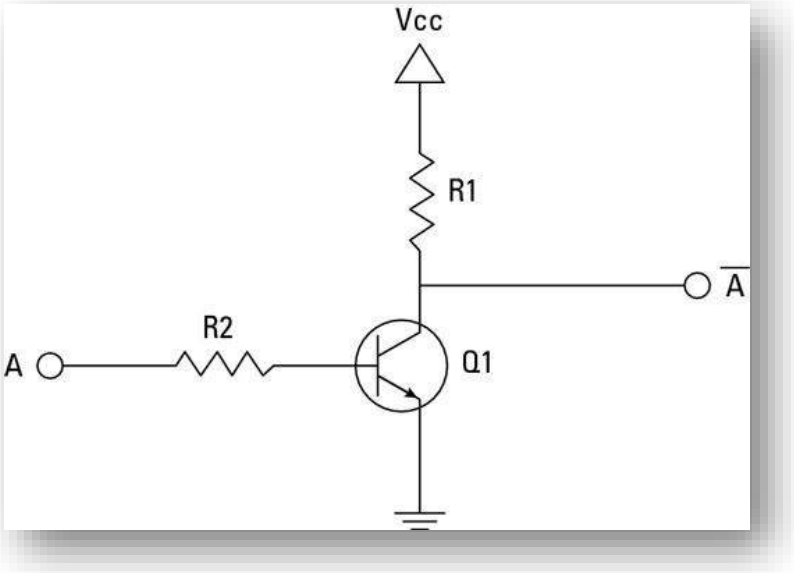


Fig 9: NOT Gate through transistor

4. NAND Gate:

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. The outputs of all NAND gates are high if any of the inputs are low. The symbol is an AND gate with a small circle on the output. The small circle represents inversion.

$$Y = (AB)'$$

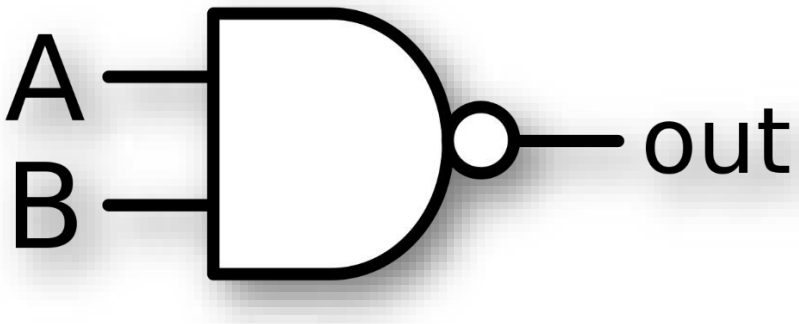


Fig 10: Logic Symbol of NAND Gate

Input	Input	Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Fig 11: Truth Table of NAND Gate

A simple 2-input logic NAND gate can be constructed using RTL (Resistor-transistor-logic) switches connected together as shown below with the inputs connected directly to the transistor bases. Either transistor must be cut-off or "OFF" for an output at Q.

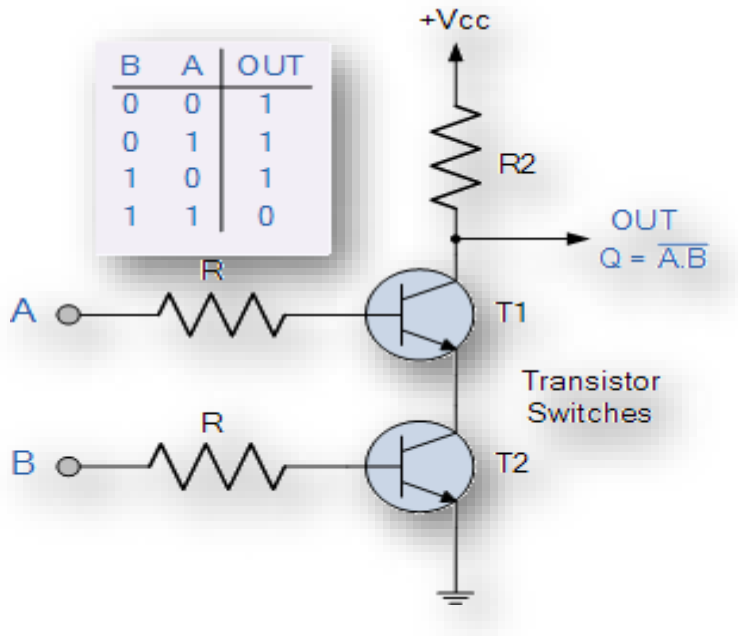


Fig 12: NAND Gate through RTL Logic

5. NOR Gate:

This is a NOT-OR gate which is equal to an OR gate followed by a NOT gate. The outputs of all NOR gates are low if any of the inputs are high. The symbol is an OR gate with a small circle on the output. The small circle represents inversion.

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

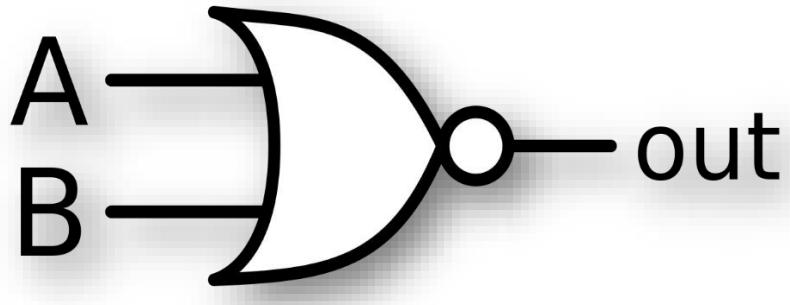


Fig 13: Logic Symbol of NOR Gate

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

Fig 14: Truth Table of NOR Gate

A simple 2-input logic NOR gate can be constructed using RTL (Resistor-transistor-logic) switches connected together as shown below with the inputs connected directly to the transistor bases. Both transistors must be cut-off or "OFF" for an output at Q.

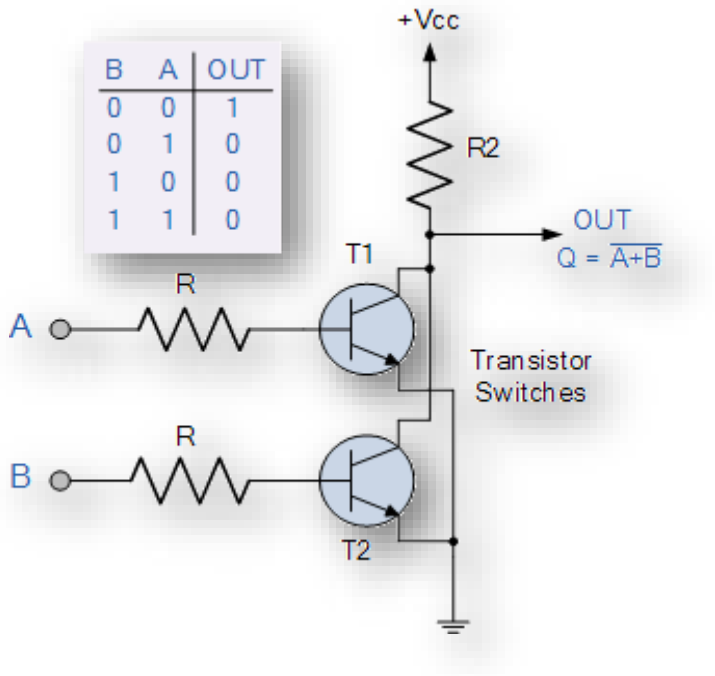


Fig 15: NOR Gate through RTL Logic

6. Ex – OR:

The 'Exclusive-OR' gate is a circuit which will give a high output if either, but not both of its two inputs are high. An encircled plus sign (⊕) is used to show the Ex-OR operation.

$$Y = A \oplus B$$

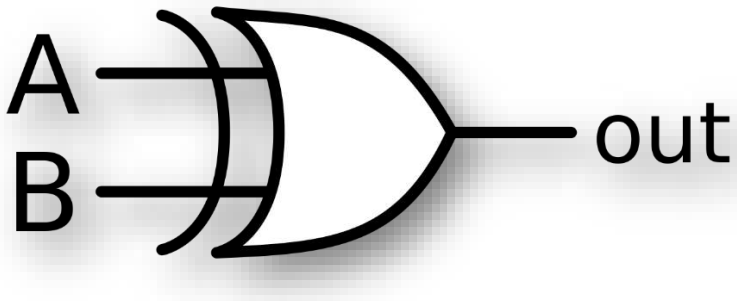


Fig 16: Logic Symbol of Ex – OR Gate

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

Fig 17: Truth Table of Ex – OR Gate

Ex-OR gate is created from AND, NAND and OR gates. The output is high only when both the inputs are different.

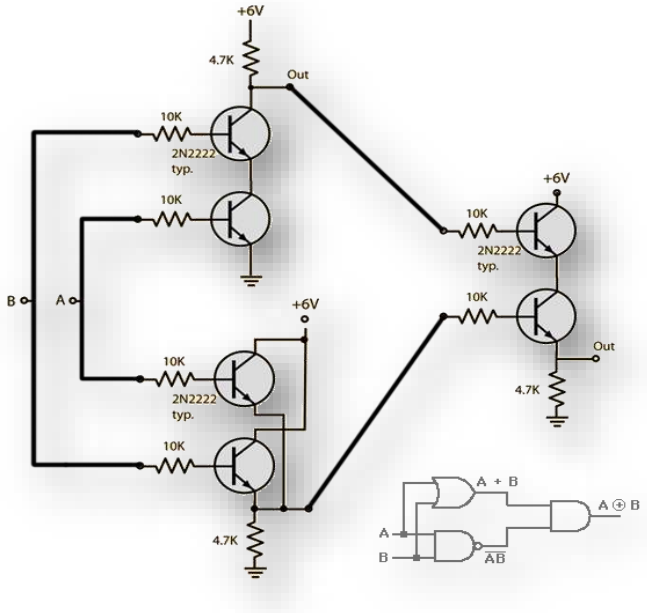


Fig 18: Ex – OR through RTL Logic

7. Ex – NOR Gate:

The 'Exclusive-NOR' gate circuit does the opposite to the EX-OR gate. It will give a low output if either, but not both of its two inputs are

high. The symbol is an EX-OR gate with a small circle on the output. The small circle represents inversion.

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

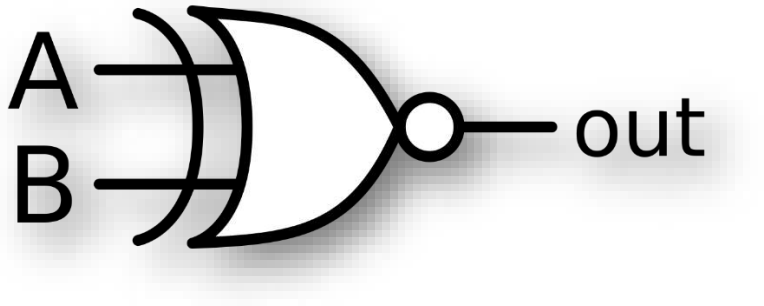


Fig 19: Logic Symbol of Ex – NOR Gate

XNOR Truth Table		
A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

Fig 20: Truth Table of Ex – NOR Gate

Ex-NOR gate is created from AND, NOT and OR gates. The output is high only when both the inputs are same.

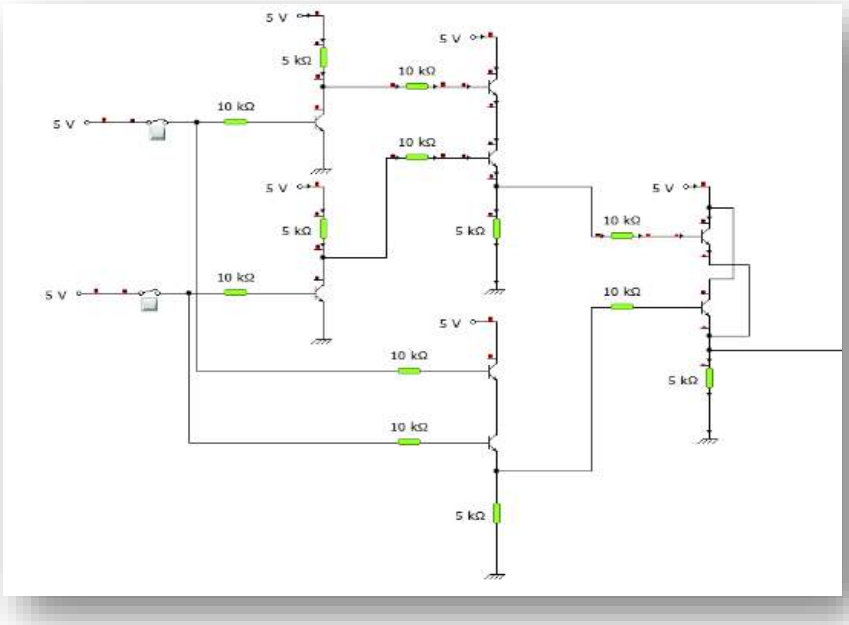
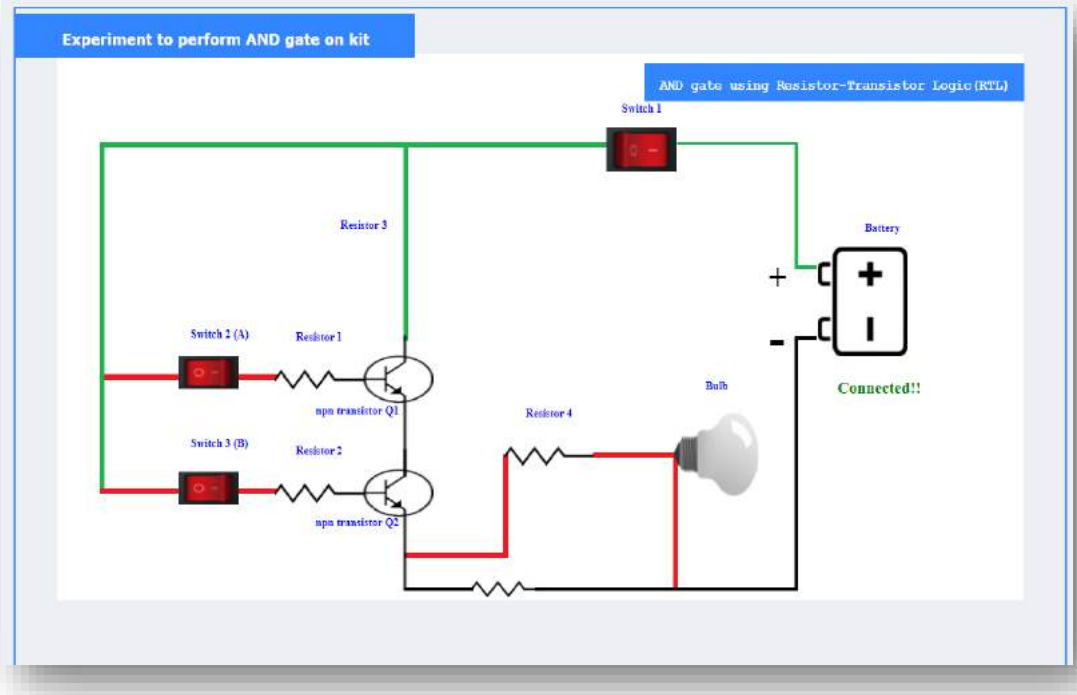
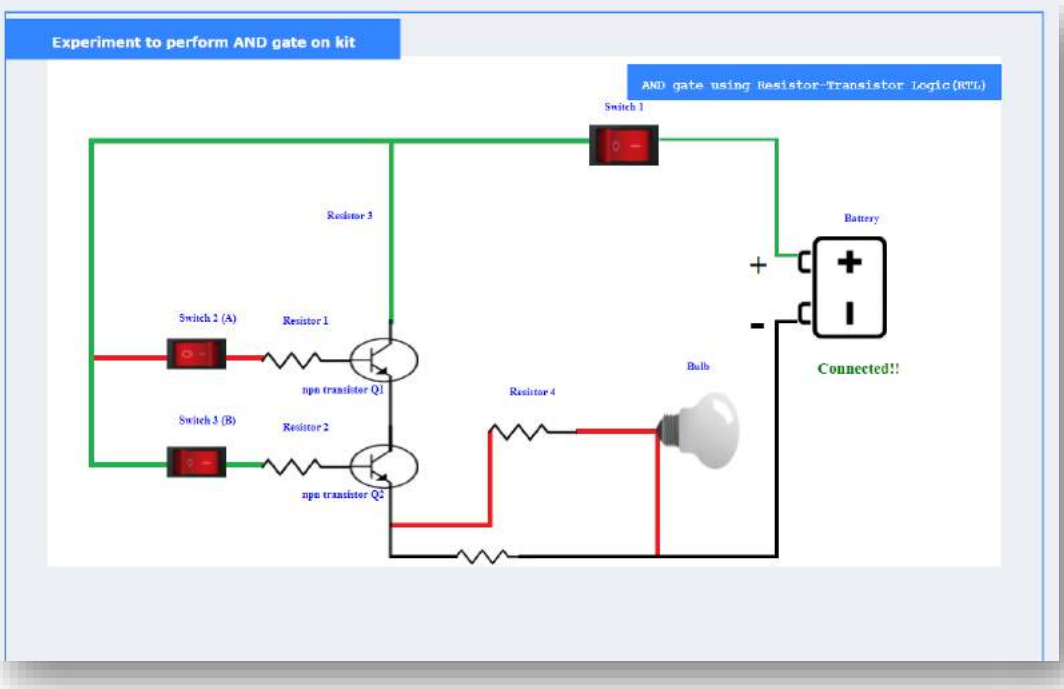


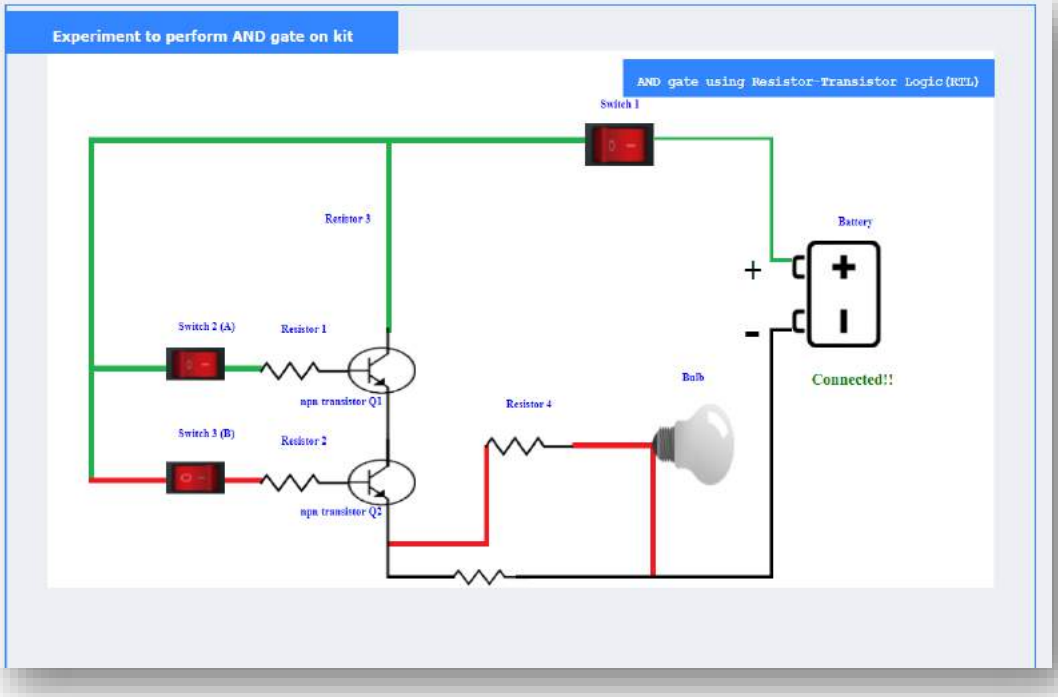
Fig 21: Ex – NOR Gate through RTL Logic

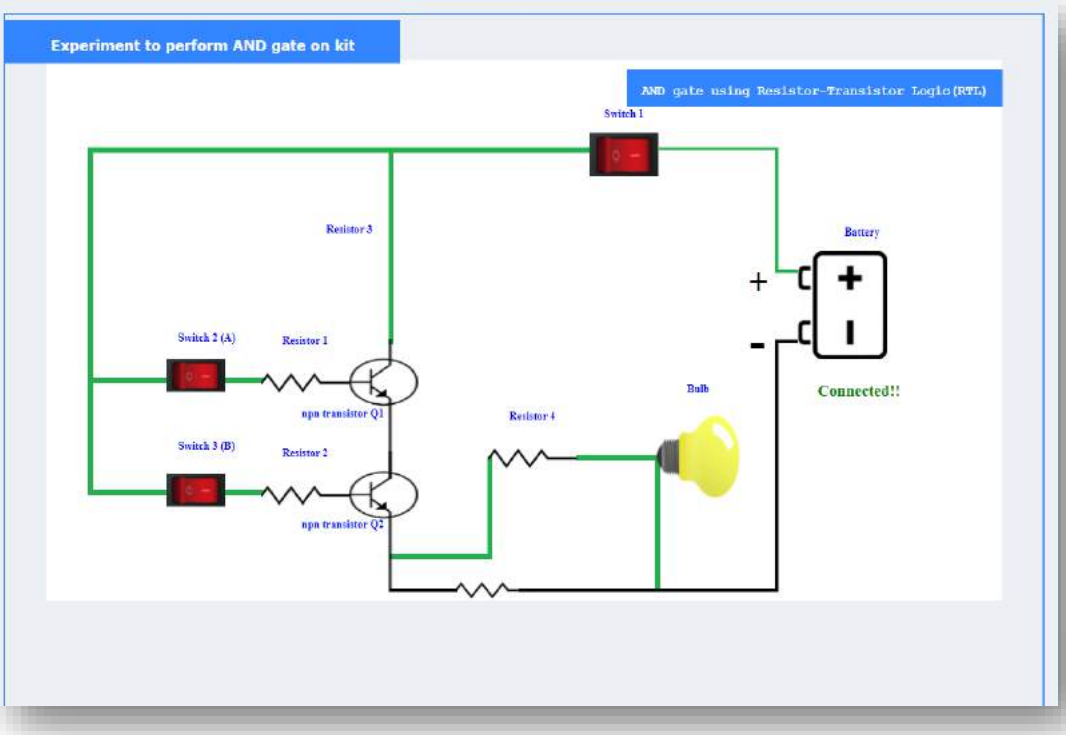
Observations :-

1. AND Gate :-

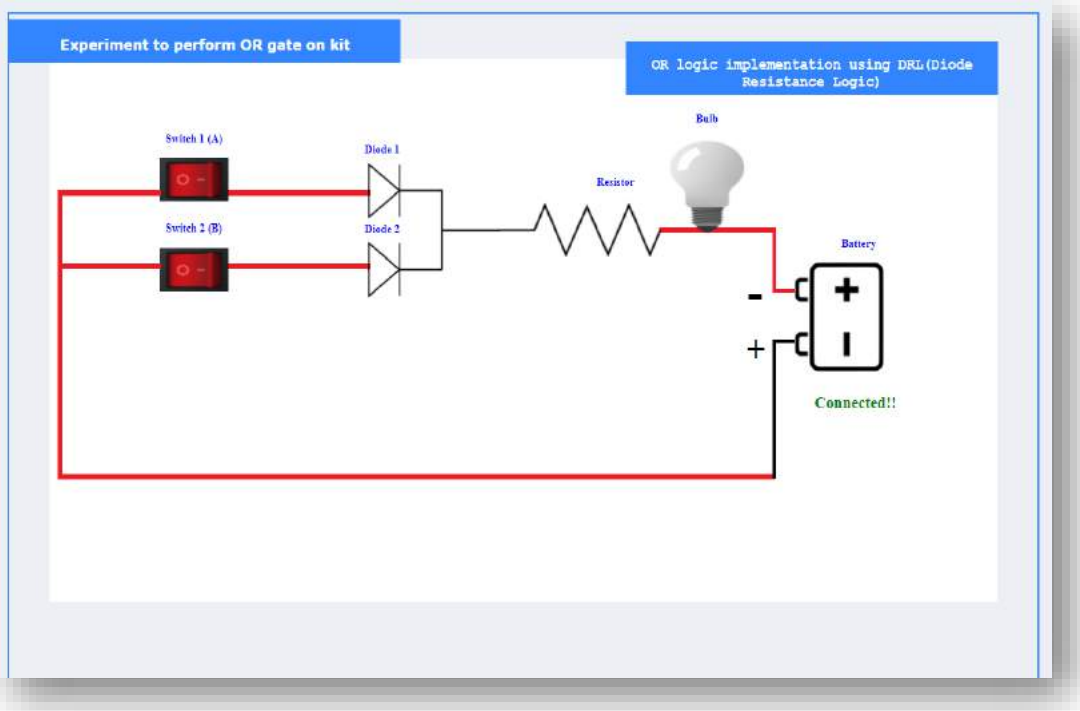


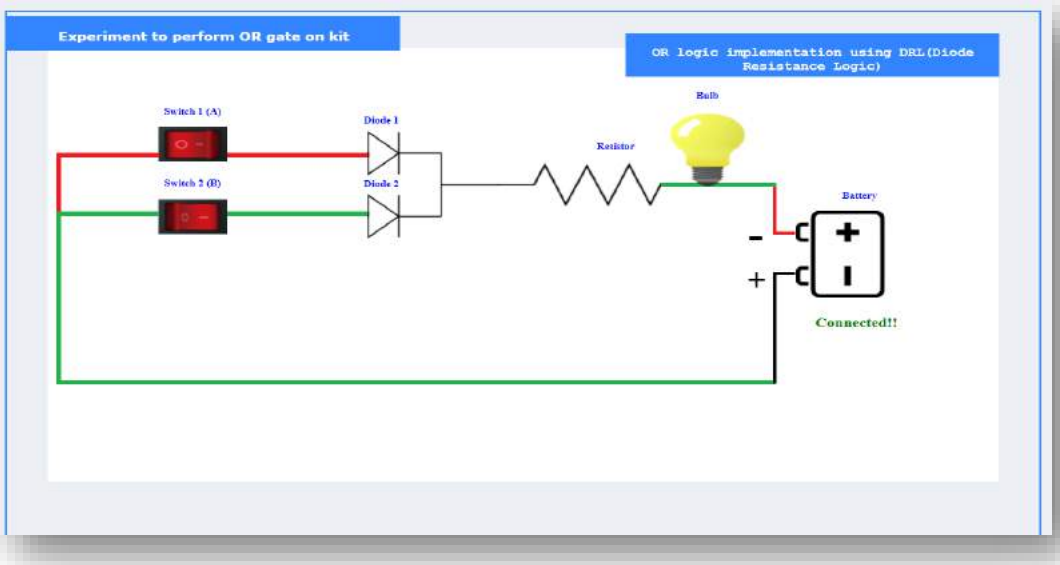

Experiment to perform AND gate on kit
AND gate using Resistor-Transistor Logic(RTL)
Switch 1
Resistor 3
Battery
+
-
Switch 2 (A)
Resistor 1
npn transistor Q1
Bulb
Connected!!
Resistor 4
Switch 3 (B)
Resistor 2
npn transistor Q2

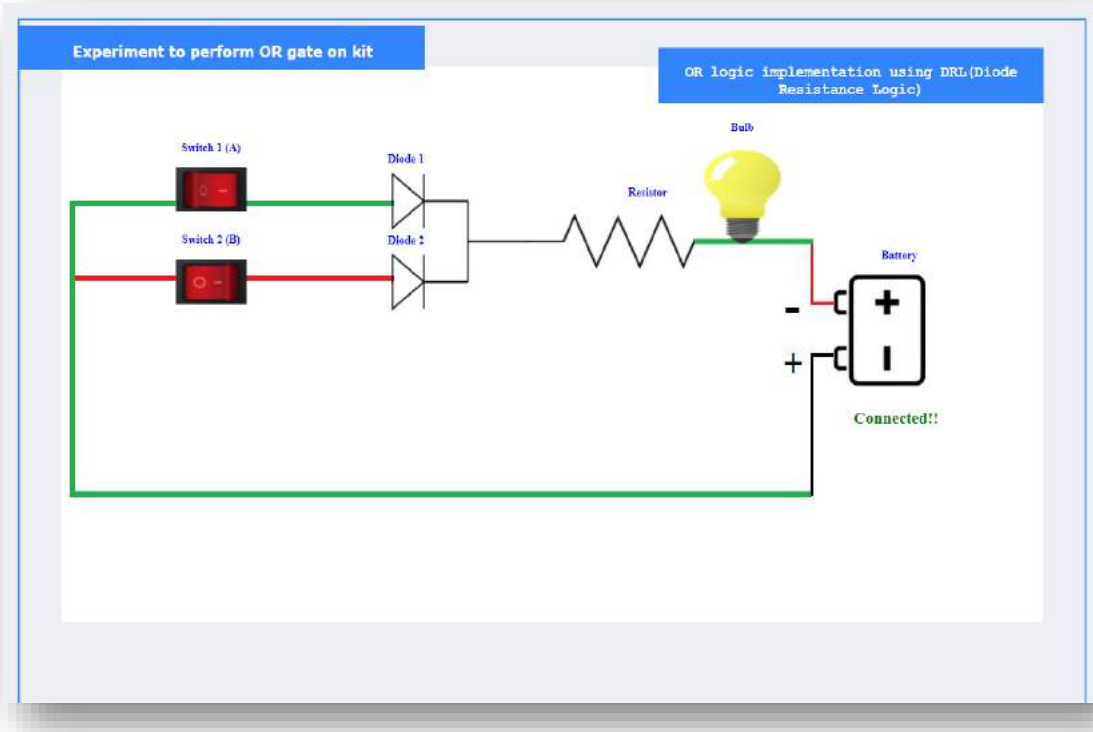

Experiment to perform AND gate on kit
AND gate using Resistor-Transistor Logic(RTL)
Switch 1
Resistor 3
Battery
+
-
Switch 2 (A)
Resistor 1
npn transistor Q1
Bulb
Connected!!
Resistor 4
Switch 3 (B)
Resistor 2
npn transistor Q2

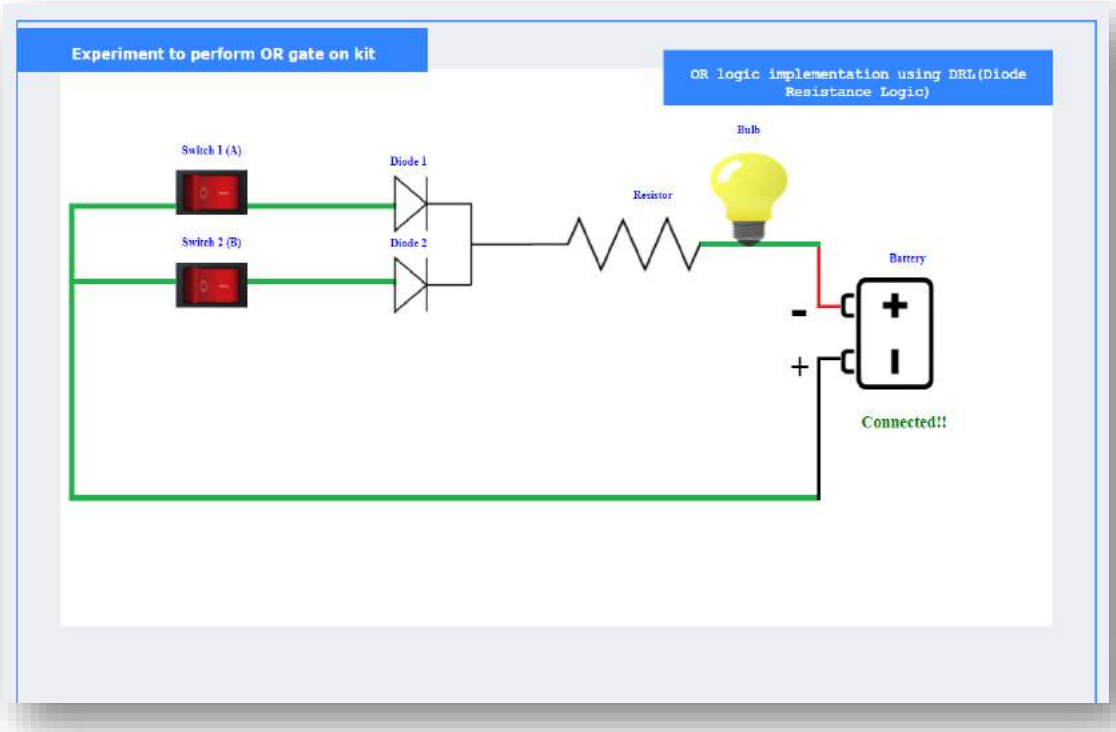

Experiment to perform AND gate on kit
AND gate using Resistor-Transistor Logic(RTL)
Switch 1
Resistor 3
Battery
+
-
Switch 2 (A)
Resistor 1
npn transistor Q1
Bulb
Connected!!
Resistor 4
Switch 3 (B)
Resistor 2
npn transistor Q2

2. OR Gate :-

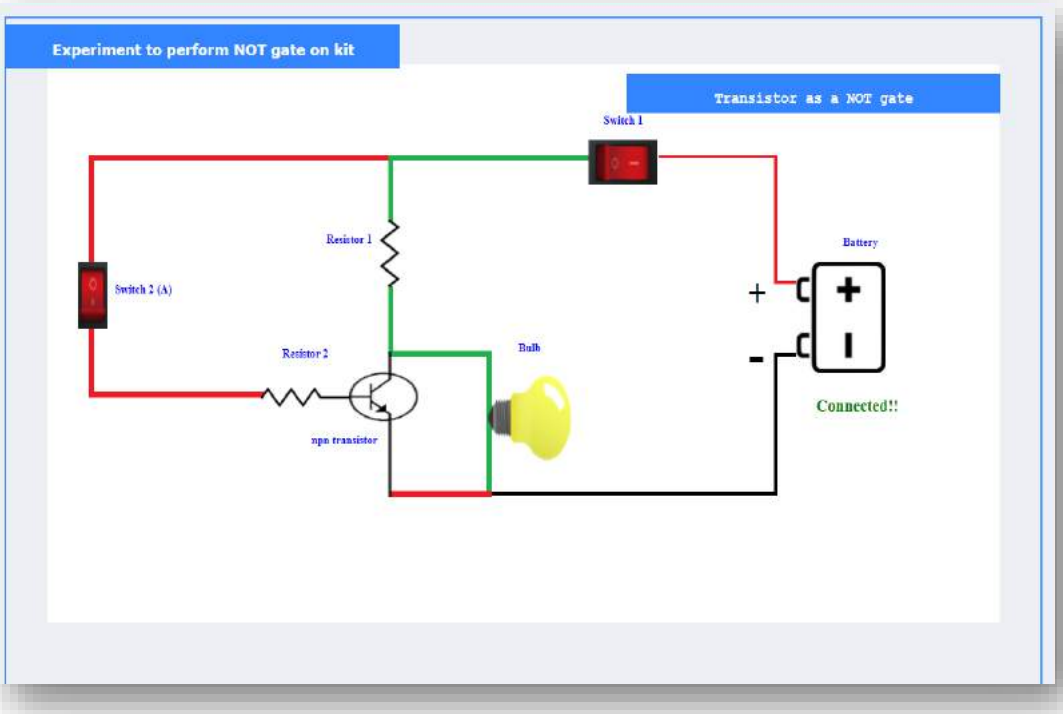

Experiment to perform OR gate on kit
OR logic implementation using DRL(Diode Resistance Logic)
Bulb
Switch 1 (A)
Diode 1
Resistor
Switch 2 (B)
Diode 2
Battery
-
+
Connected!!

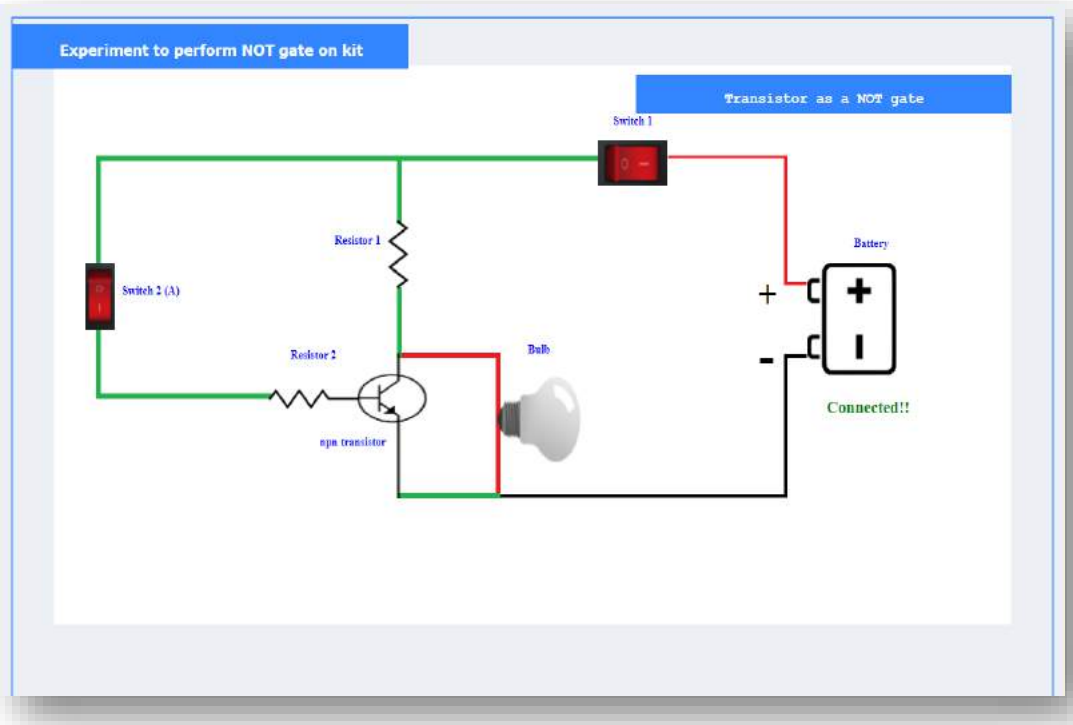

Experiment to perform OR gate on kit
OR logic implementation using DRL(Diode Resistance Logic)
Switch 1 (A)
Switch 2 (B)
Diode 1
Diode 2
Resistor
Bulb
Battery
-
+
Connected!!


Experiment to perform OR gate on kit
OR logic implementation using DRL(Diode Resistance Logic)
Switch 1 (A)
Switch 2 (B)
Diode 1
Diode 2
Resistor
Bulb
Battery
-
+
Connected!!

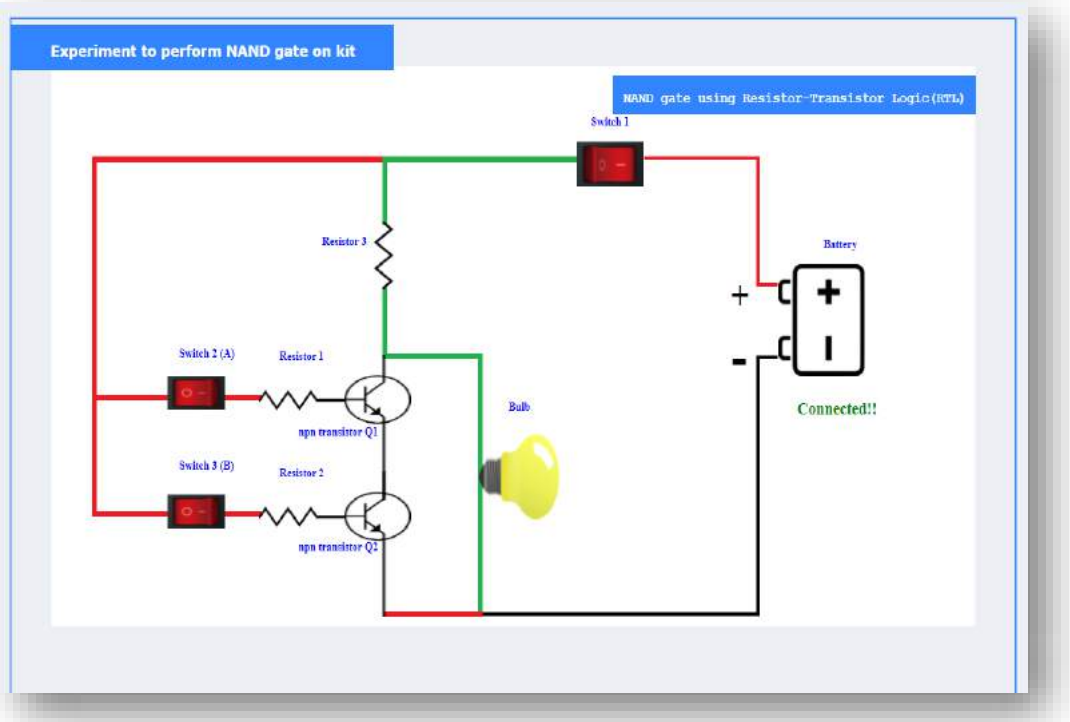

Experiment to perform OR gate on kit
OR logic implementation using DRL(Diode Resistance Logic)
Switch 1 (A)
Switch 2 (B)
Diode 1
Diode 2
Resistor
Bulb
Battery
-
+
Connected!!

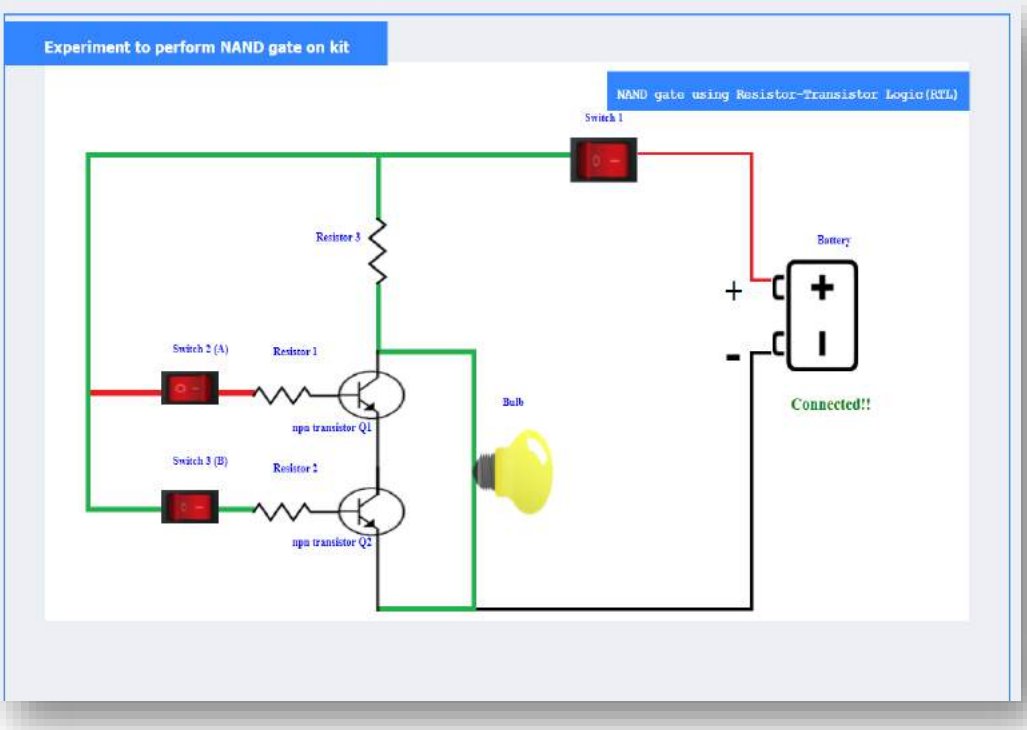
3. NOT Gate :-

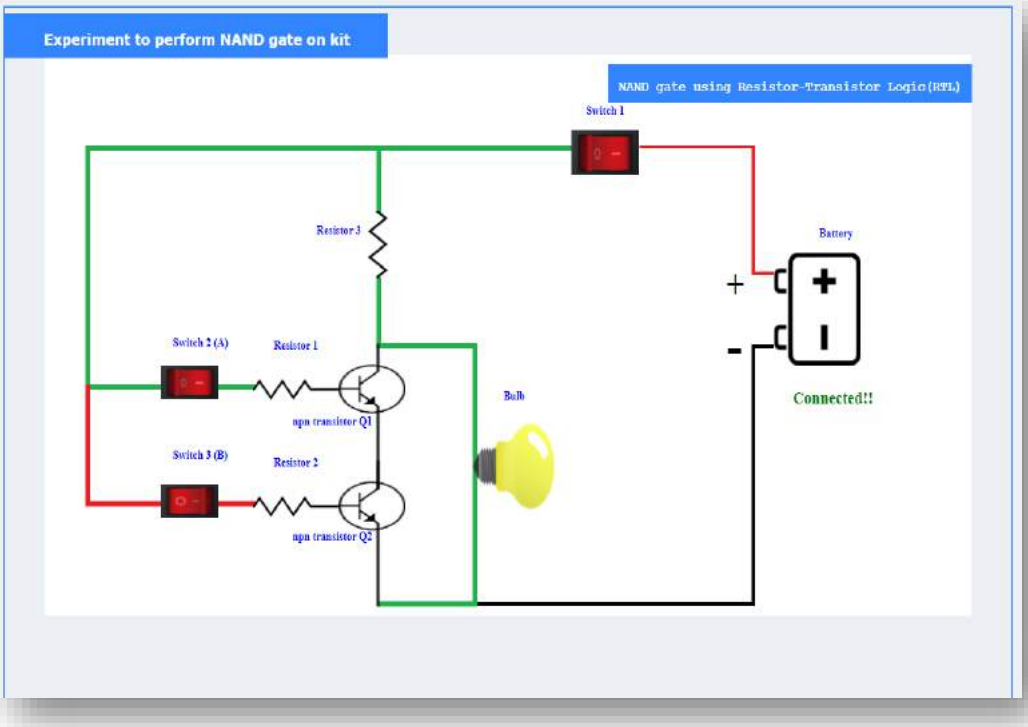

Experiment to perform NOT gate on kit
Transistor as a NOT gate
Switch 1
Switch 2 (A)
Resistor 1
Resistor 2
npn transistor
Bulb
Battery
+
-
Connected!!

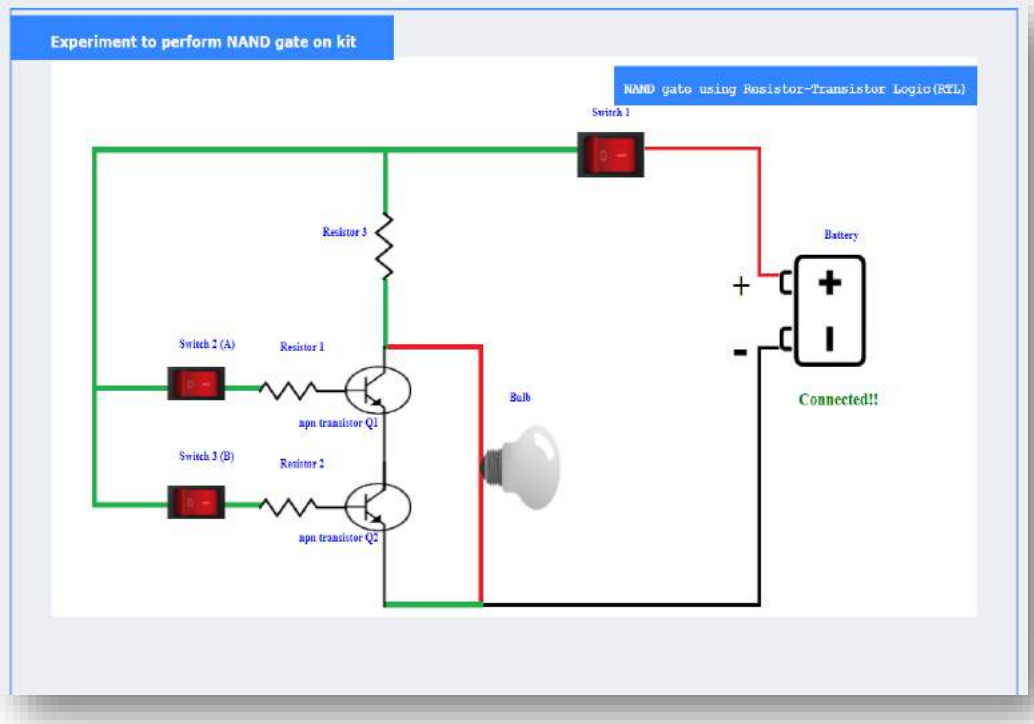

Experiment to perform NOT gate on kit
Transistor as a NOT gate
Switch 1
Resistor 1
Battery
Switch 2 (A)
+
Resistor 2
Bulb
-
npn transistor
Connected!!

4. NAND Gate :-

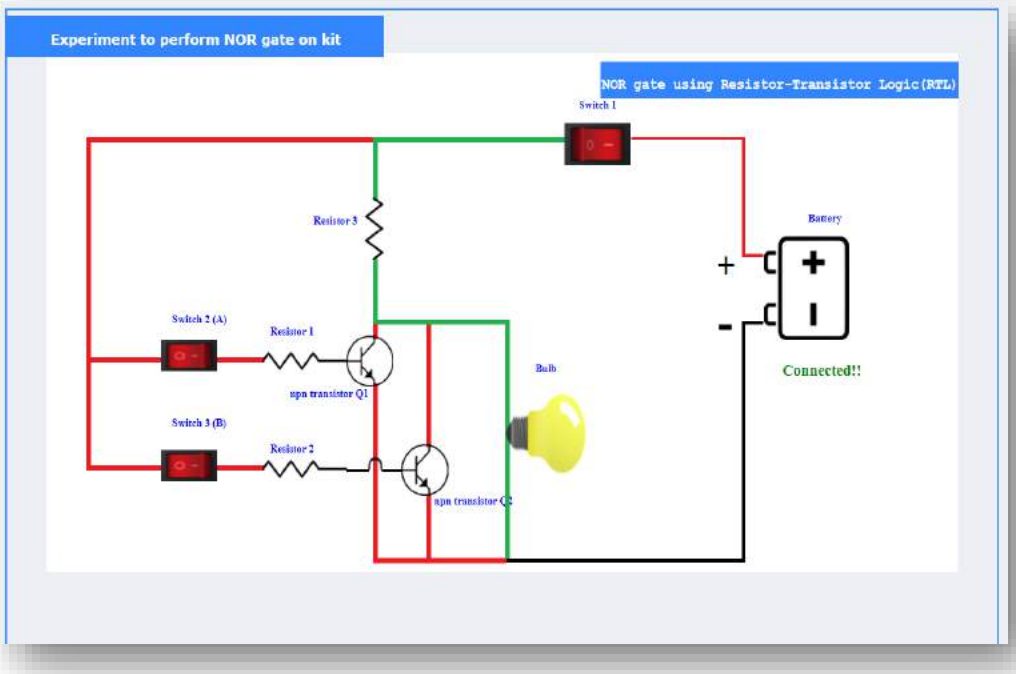

Experiment to perform NAND gate on kit
NAND gate using Resistor-Transistor Logic(RTL)
Switch 1
Resistor 3
Battery
+
Switch 2 (A)
Resistor 1
Bulb
-
npn transistor Q1
Connected!!
Switch 3 (B)
Resistor 2
npn transistor Q2

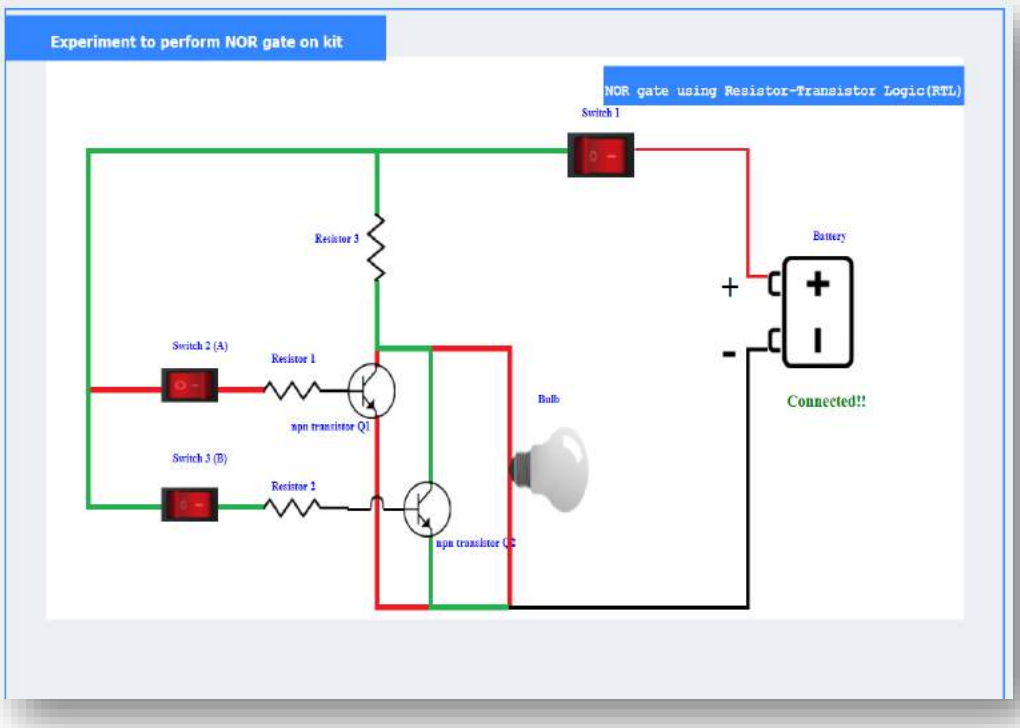

Experiment to perform NAND gate on kit
NAND gate using Resistor-Transistor Logic(RTL)
Switch 1
Resistor 3
Battery
+
-
Switch 2 (A)
Resistor 1
Bulb
Connected!!
npn transistor Q1
Switch 3 (B)
Resistor 2
npn transistor Q2

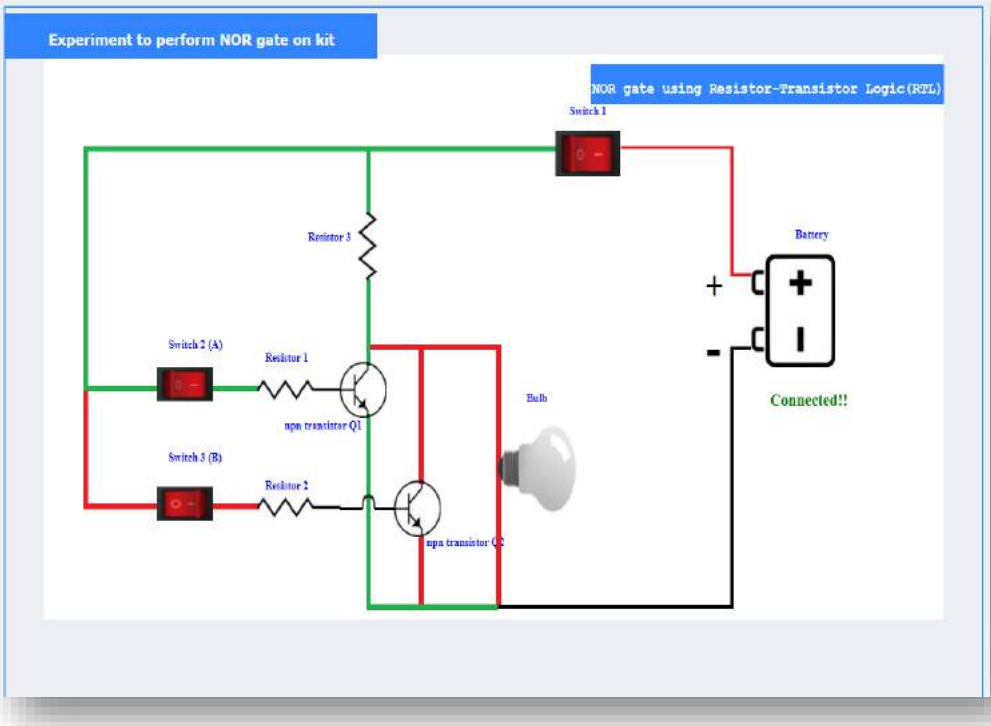

Experiment to perform NAND gate on kit
NAND gate using Resistor-Transistor Logic(RTL)
Switch 1
Resistor 3
Battery
+
-
Switch 2 (A)
Resistor 1
Bulb
Connected!!
npn transistor Q1
Switch 3 (B)
Resistor 2
npn transistor Q2

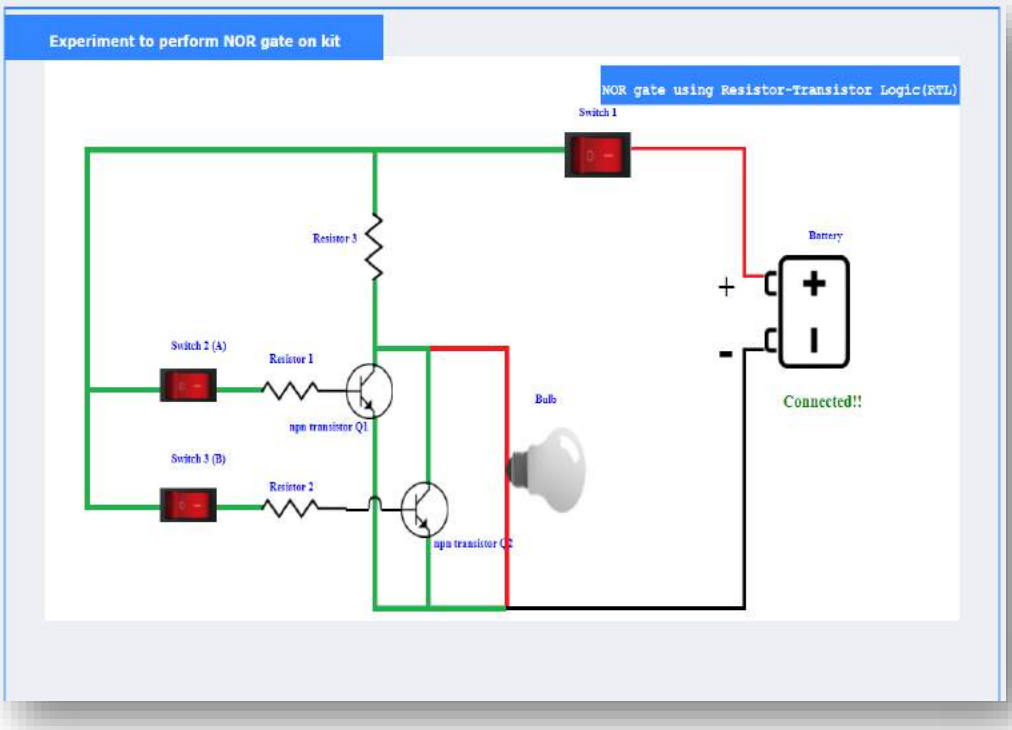

Experiment to perform NAND gate on kit
NAND gate using Resistor-Transistor Logic(RTL)
Switch 1
Resistor 3
Battery
+
-
Switch 2 (A)
Resistor 1
Bulb
Connected!!
npn transistor Q1
Switch 3 (B)
Resistor 2
npn transistor Q2

5. NOR Gate:-

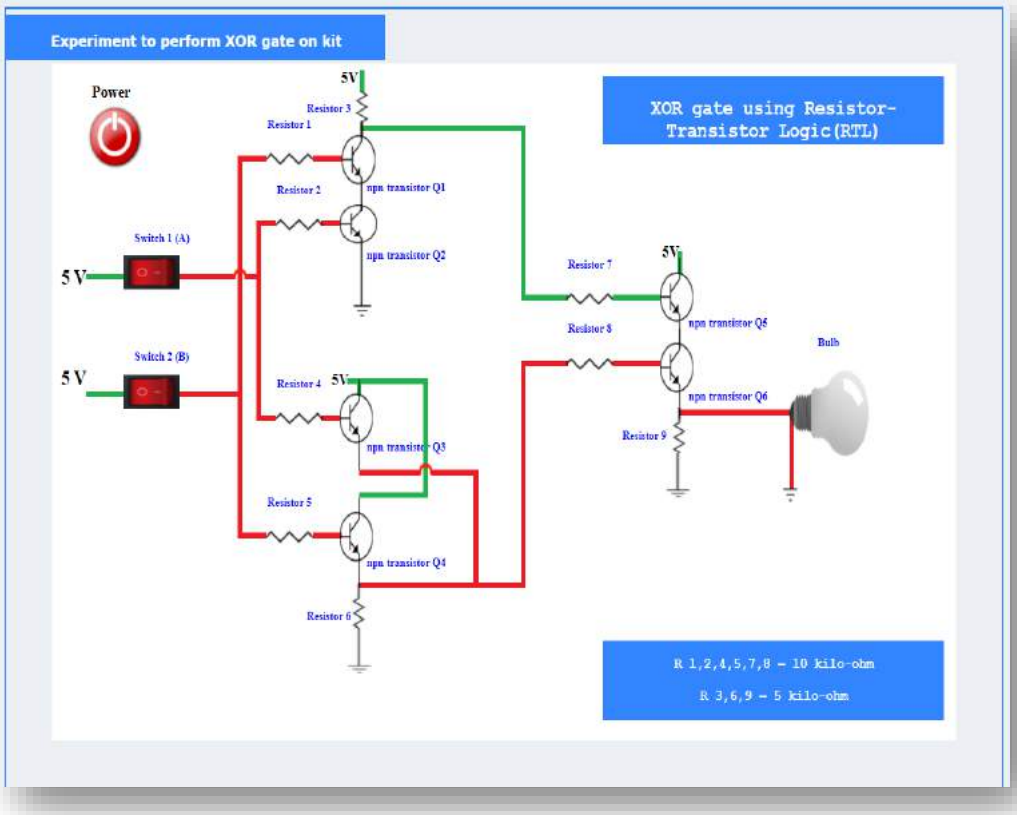

Experiment to perform NOR gate on kit
NOR gate using Resistor-Transistor Logic(RTL)
Switch 1
Resistor 3
Battery
+
-
Switch 2 (A)
Resistor 1
Bulb
Connected!!
npn transistor Q1
Switch 3 (B)
Resistor 2
npn transistor Q2

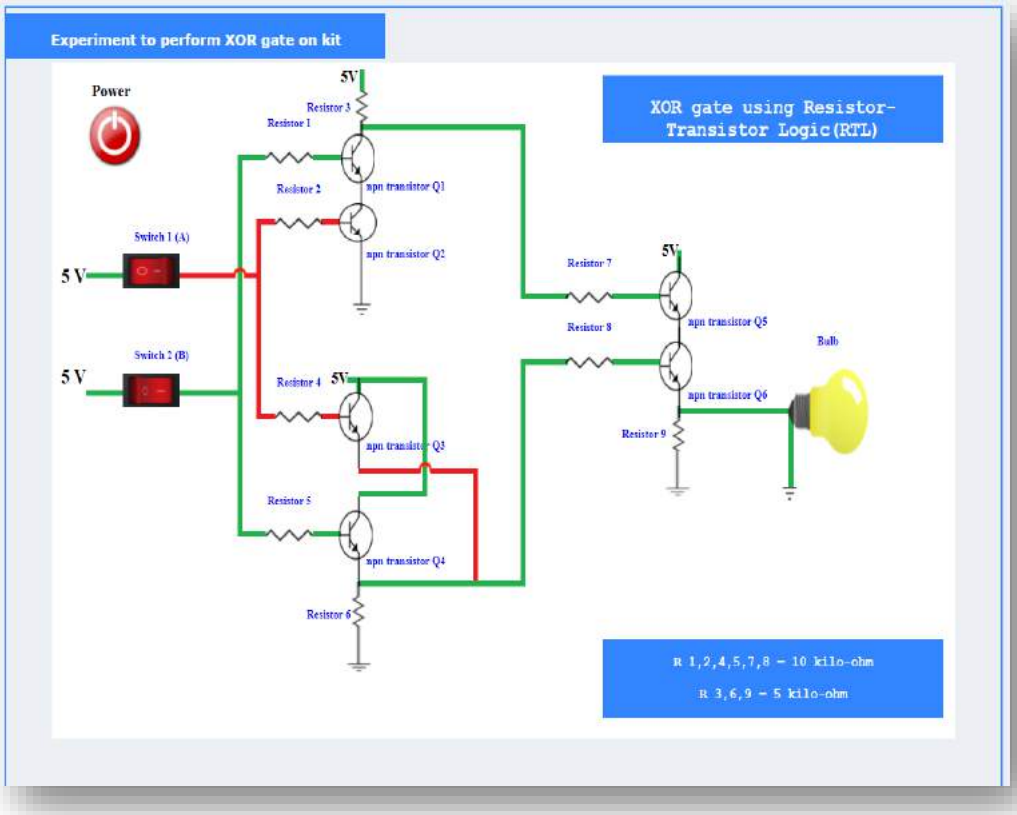

Experiment to perform NOR gate on kit
NOR gate using Resistor-Transistor Logic(RTL)
Switch 1
Resistor 3
Battery
+
-
Switch 2 (A)
Resistor 1
npn transistor Q1
Bulb
Connected!!
Switch 3 (B)
Resistor 2
npn transistor Q2

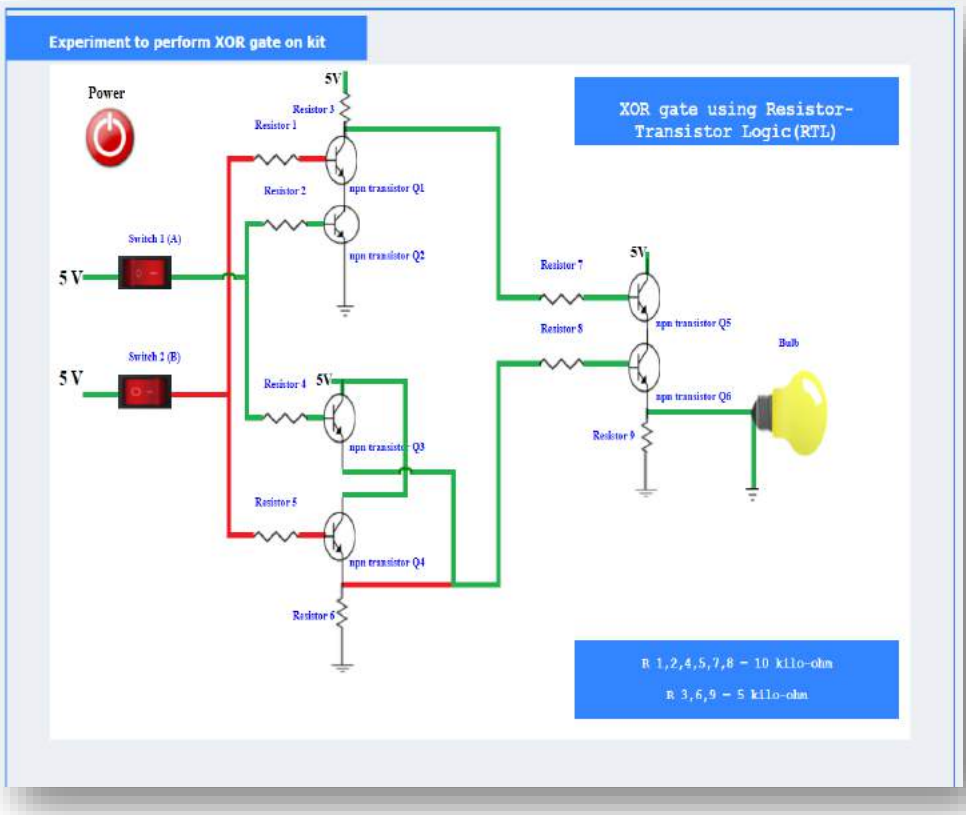

Experiment to perform NOR gate on kit
NOR gate using Resistor-Transistor Logic(RTL)
Switch 1
Resistor 3
Battery
+
-
Switch 2 (A)
Resistor 1
npn transistor Q1
Bulb
Connected!!
Switch 3 (B)
Resistor 2
npn transistor Q2

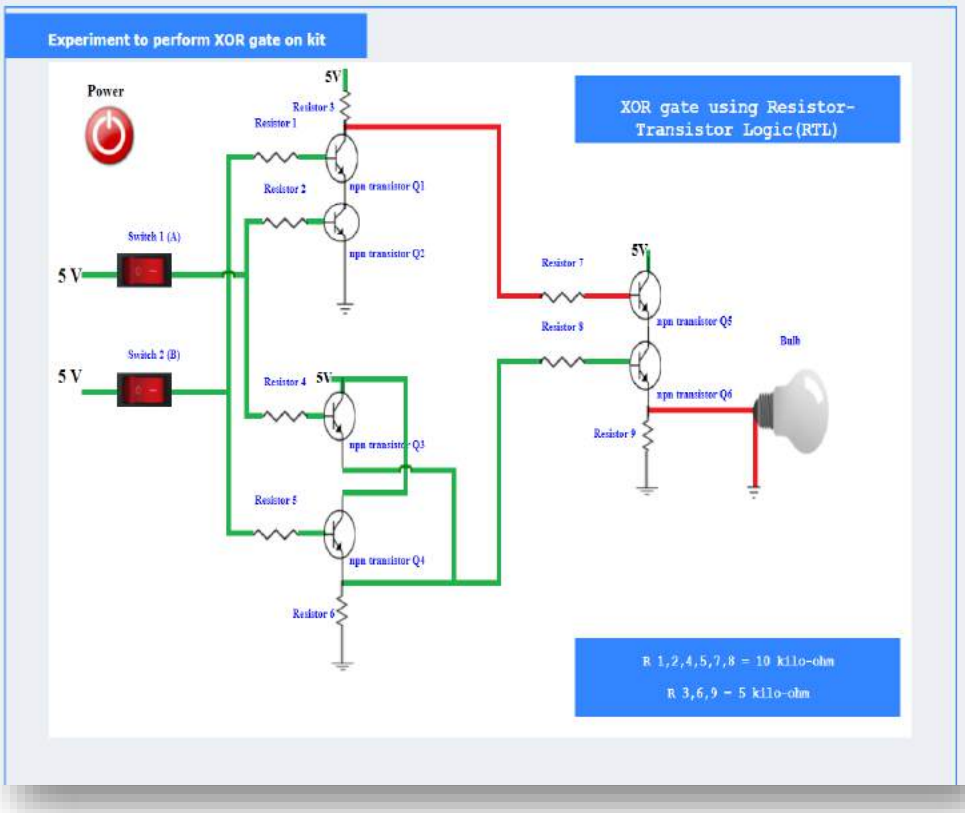

Experiment to perform NOR gate on kit
NOR gate using Resistor-Transistor Logic(RTL)
Switch 1
Resistor 3
Battery
+
-
Switch 2 (A)
Resistor 1
Bulb
npn transistor Q1
Connected!!
Switch 3 (B)
Resistor 2
npn transistor Q2

6. Ex – OR Gate :-

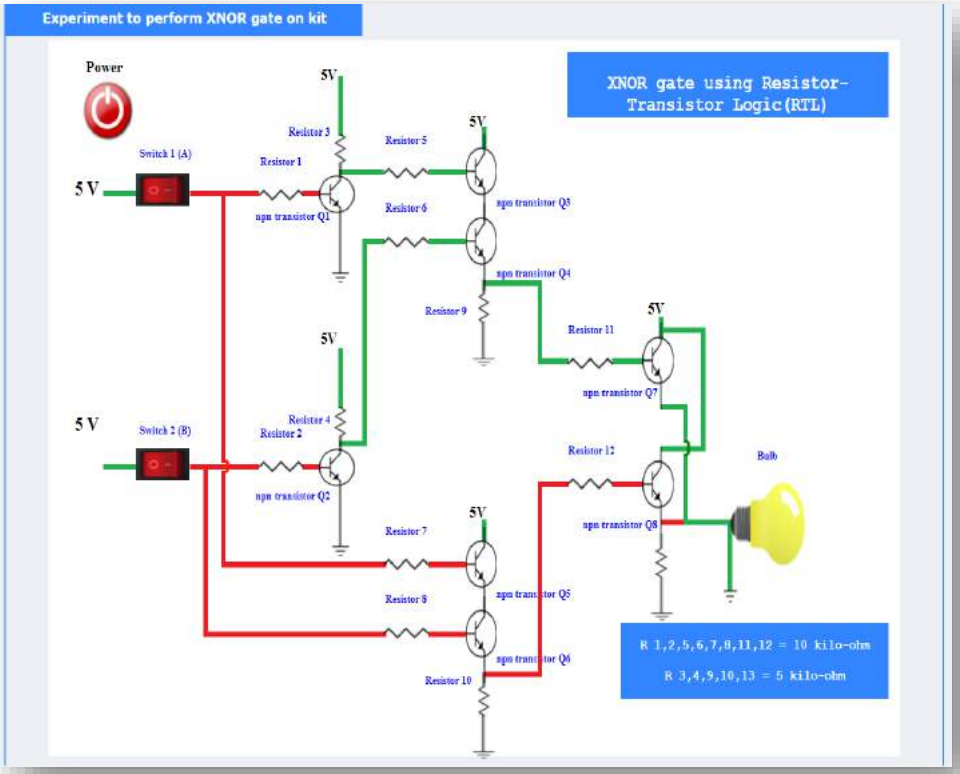

Experiment to perform XOR gate on kit
Power
5V
Resistor 3
Resistor 1
XOR gate using Resistor-Transistor Logic(RTL)
Resistor 2
npn transistor Q1
npn transistor Q2
Switch 1 (A)
5 V
Resistor 7
5V
npn transistor Q5
Resistor 8
Bulb
Switch 2 (B)
5 V
Resistor 4
5V
npn transistor Q6
Resistor 9
npn transistor Q3
Resistor 5
npn transistor Q4
Resistor 6
R 1,2,4,5,7,8 = 10 kilo-ohm
R 3,6,9 = 5 kilo-ohm

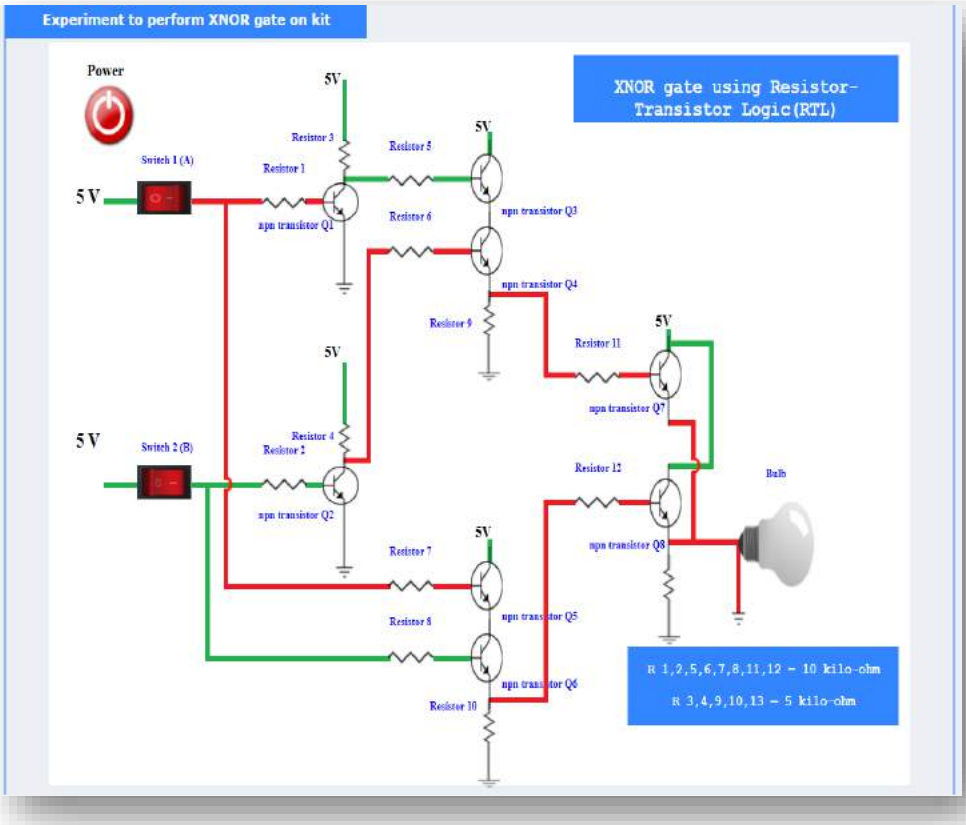

Experiment to perform XOR gate on kit
Power
5V
Resistor 3
Resistor 1
Resistor 2
npn transistor Q1
npn transistor Q2
Switch 1 (A)
5 V
Switch 2 (B)
5 V
Resistor 4
5V
npn transistor Q3
Resistor 5
npn transistor Q4
Resistor 6
Resistor 7
5V
npn transistor Q5
Resistor 8
Bulb
npn transistor Q6
Resistor 9
XOR gate using Resistor-Transistor Logic(RTL)
R 1,2,4,5,7,8 = 10 kilo-ohm
R 3,6,9 = 5 kilo-ohm

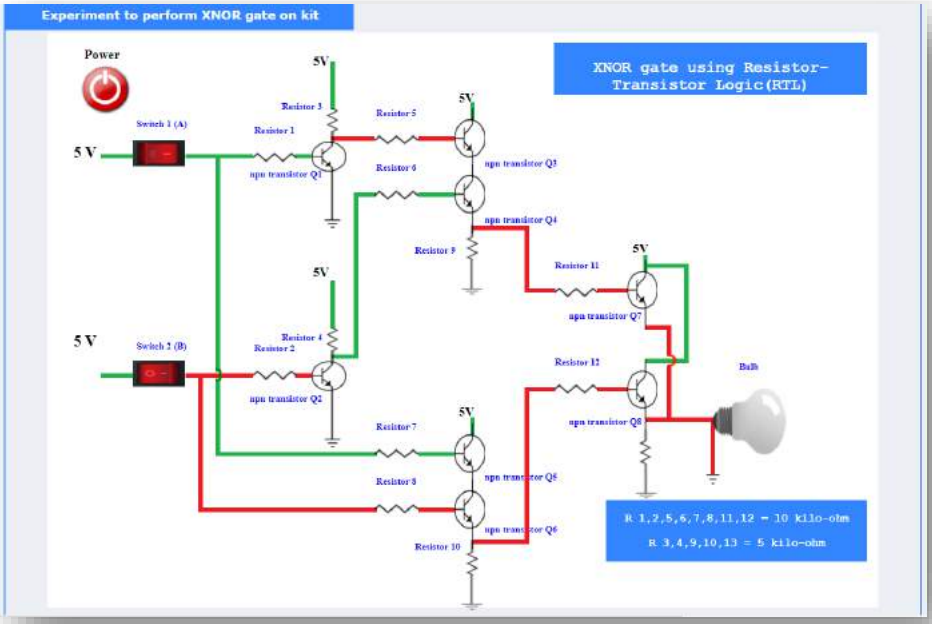

Experiment to perform XOR gate on kit
Power
5V
Resistor 3
Resistor 1
Resistor 2
npn transistor Q1
npn transistor Q2
Switch 1 (A)
5 V
Switch 2 (B)
5 V
Resistor 4
5V
npn transistor Q3
Resistor 5
npn transistor Q4
Resistor 6
Resistor 7
5V
npn transistor Q5
Resistor 8
Bulb
npn transistor Q6
Resistor 9
XOR gate using Resistor-Transistor Logic(RTL)
R 1,2,4,5,7,8 = 10 kilo-ohm
R 3,6,9 = 5 kilo-ohm

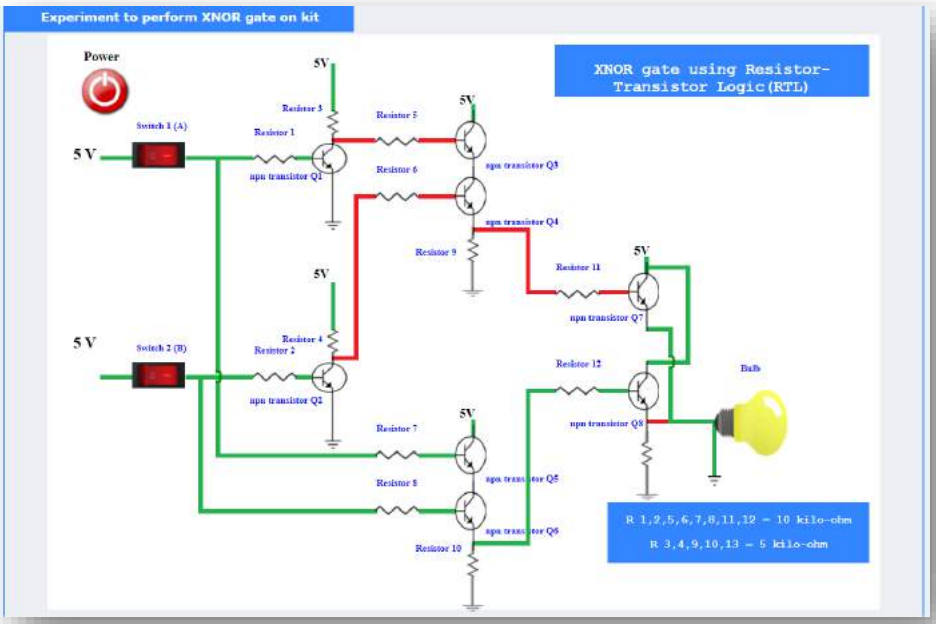

Experiment to perform XOR gate on kit
Power
5V
Resistor 3
Resistor 1
Resistor 2
npn transistor Q1
npn transistor Q2
Switch 1 (A)
5 V
Switch 2 (B)
5 V
Resistor 4
5V
npn transistor Q3
Resistor 5
npn transistor Q4
Resistor 6
Resistor 7
5V
npn transistor Q5
Resistor 8
Bulb
npn transistor Q6
Resistor 9
XOR gate using Resistor-Transistor Logic(RTL)
R 1,2,4,5,7,8 = 10 kilo-ohm
R 3,6,9 = 5 kilo-ohm

7. Ex - NOR Gate :-


Experiment to perform XNOR gate on kit
XNOR gate using Resistor-Transistor Logic(RTL)
Power
5V
Switch 1 (A)
Switch 2 (B)
Resistor 1
Resistor 2
Resistor 3
Resistor 4
Resistor 5
Resistor 6
Resistor 7
Resistor 8
Resistor 9
Resistor 10
Resistor 11
Resistor 12
npn transistor Q1
npn transistor Q2
npn transistor Q3
npn transistor Q4
npn transistor Q5
npn transistor Q6
npn transistor Q7
npn transistor Q8
Bulb
R 1,2,5,6,7,8,11,12 = 10 kilo-ohm
R 3,4,9,10,13 = 5 kilo-ohm


Experiment to perform XNOR gate on kit
XNOR gate using Resistor-Transistor Logic(RTL)
Power
5V
Switch 1 (A)
Switch 2 (B)
Resistor 1
Resistor 2
Resistor 3
Resistor 4
Resistor 5
Resistor 6
Resistor 7
Resistor 8
Resistor 9
Resistor 10
Resistor 11
Resistor 12
npn transistor Q1
npn transistor Q2
npn transistor Q3
npn transistor Q4
npn transistor Q5
npn transistor Q6
npn transistor Q7
npn transistor Q8
Bulb
R 1,2,5,6,7,8,11,12 = 10 kilo-ohm
R 3,4,9,10,13 = 5 kilo-ohm





Conclusion:

Logic and truth table for AND, OR, NOT, NAND, NOR, Ex – OR, Ex – NOR gates is verified.

Experiment - 2

Aim: To design and implement 8:1 MUX using IC – 74LS153 and verify its truth table.

Apparatus Required : Perform the experiment using this link : http://vlabs.iitb.ac.in/vlabs-dev/labs/dldesignlab/experiments/implementation-multiplexer-using-msi/

Equipment Used: IC – 74LS153

Theory :

A modern home stereo system may have a switch that selects music from one or four sources: a radio tuner, a cassette tape, a compact disc (CD) or an auxiliary input. The auxiliary input could be an audio from VCR or DVD or a smart phone. A MUX is a combinational logic block that selects one-of-N inputs and directs the information to a single output. It acts like a multi-position switch. Mux is a well-known MSI (medium-scale integration) IC.

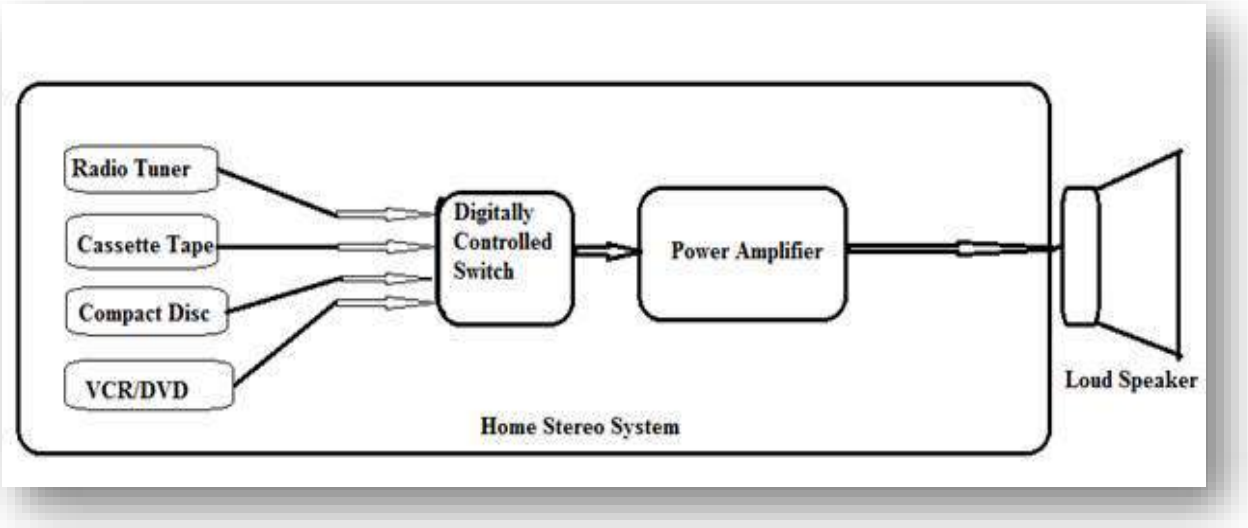


Fig.1. Block schematic: Home Stereo System

Multiplexer or Data Selector is a very widely used combinational circuit. It has multiple inputs and one output. It accepts several data inputs and allows only one of them at a time to get through to the output. The routing of the desired input to the output is controlled by the select lines. M select lines can select one of the 2^M input channels. The generalized block schematic of a multiplexer is shown in Fig. 2. Mechanical rotary switch is a good analogy to explain the MUX concept.

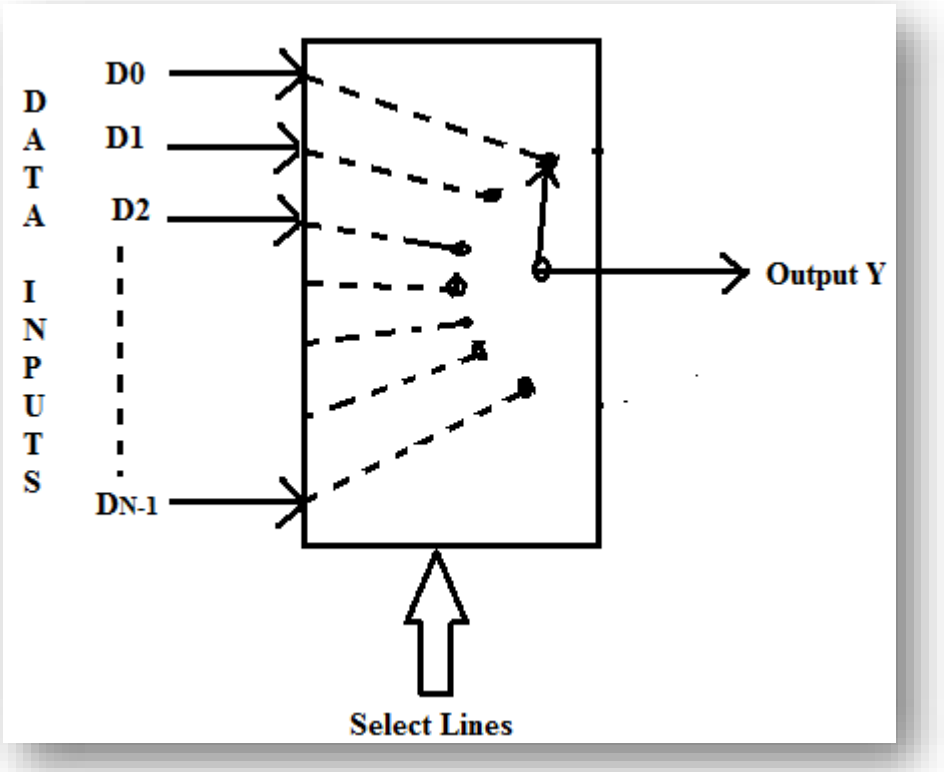


Fig.2. Multiplexer Schematic : As a digitally controlled multiposition switch

The multiplexer selects 1 out of N input data sources and sends selected data to single output channel. This Many as to One function is called as Multiplexing. A data selector is a sort of one-package-solution to a complicated logic problem. It consists of large number of gates packaged inside a single integrated circuit (IC). It can be considered to belong to medium scale integration - MSI technology. MUX Applications: Multiplexers basically can be used as universal logic elements. It provides a low-cost reliable and compact solution to many logic problems with three to five input variables. The MUX applications include:

1. Data Selection
2. Data Routing
3. Operation Sequencing
4. Parallel –to-Serial Conversion

5. Waveform Generation
6. Logic Function Generation.

Design:

The functional name of IC 74LS153 is Dual 4 line to 1 Line Data Selector/Multiplexer.

It implies that there are two 4: Multiplexers (Mux) inside the IC. It is a 16 pin Dual-In-Line Package (DIP) IC. The outputs of the two Muxes then can be given to an OR gate to produce the single final output Y as shown in Fig.3. The function table of the 8:1 MUX is given in Table 1. The Boolean expression defining the output is given by:

Y = C'. [B'.A'.D0 + B'.A.D1 + B.A'.D2 + B.A.D3] C. [B'.A'.D4 + B'.A.D5 + B.A'.D6 + B.A.D7]

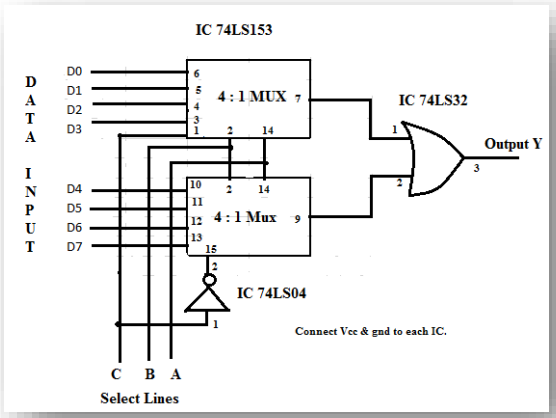


Fig. 3. Eight to One Multiplexer using IC 74LS153

The logic diagram, for 8:1 MUX in IC 74LS153 is:

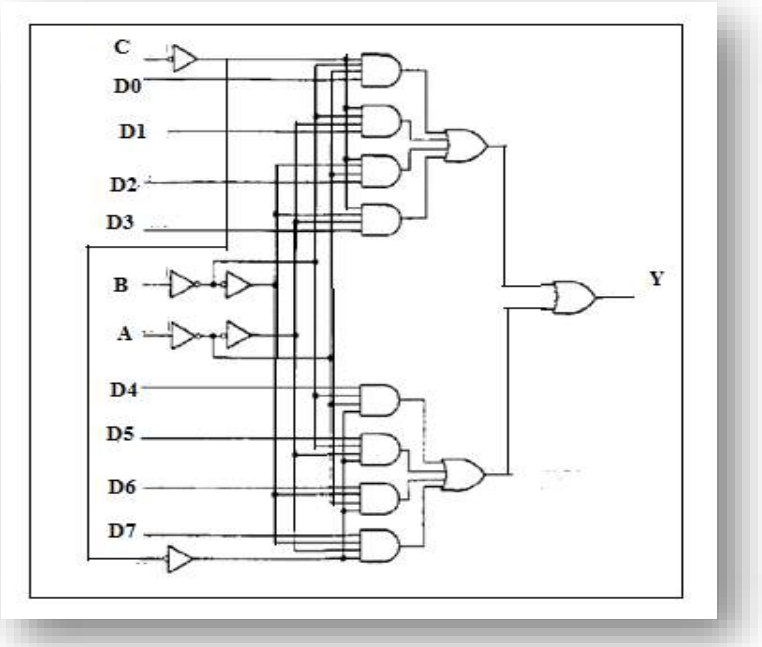
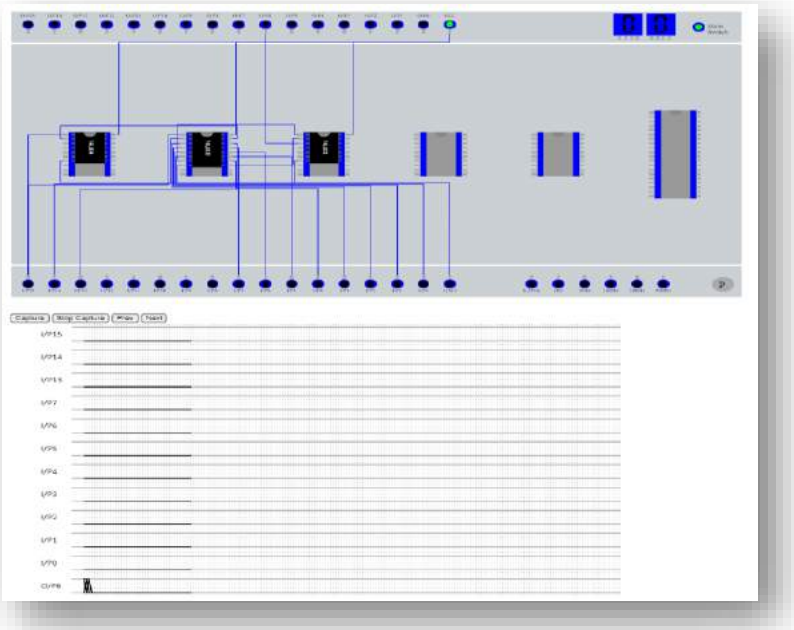
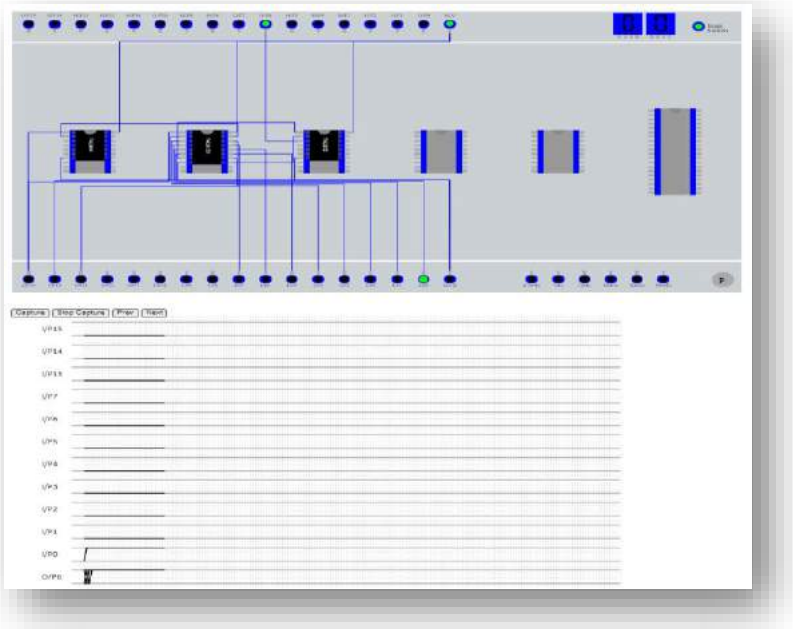
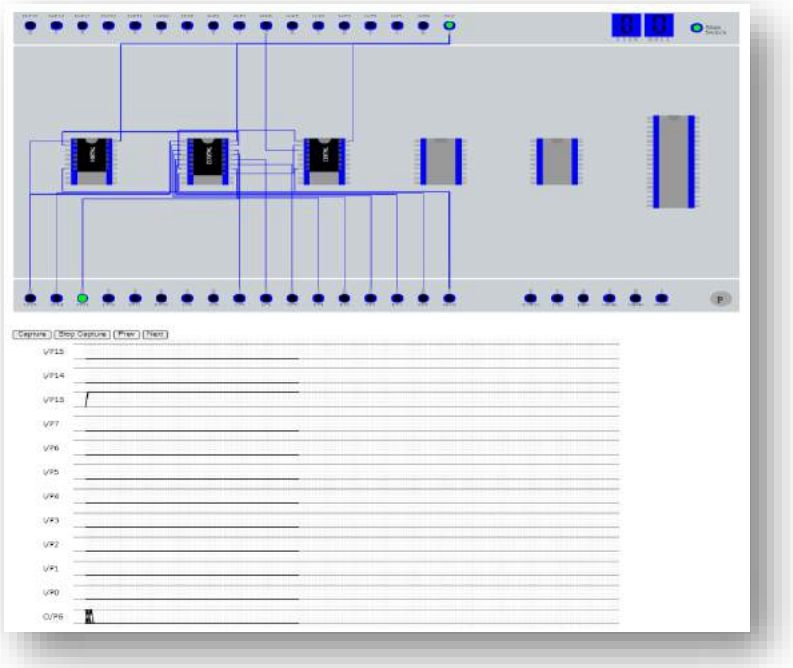


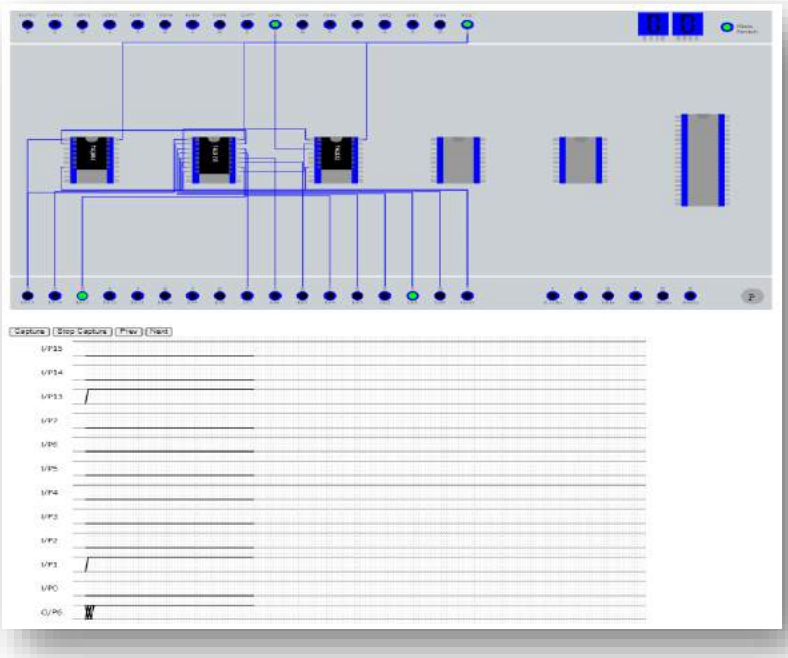
Fig. 4. Logic Diagram

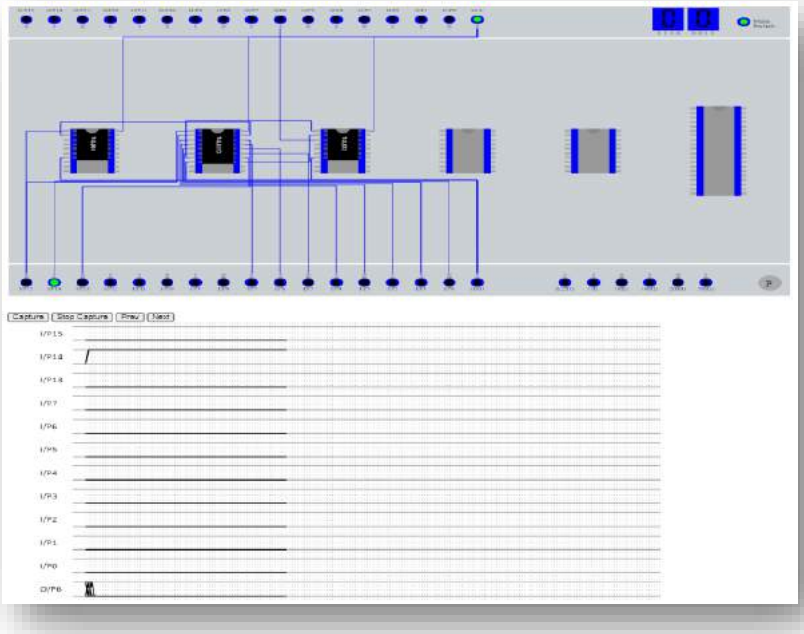
Observations:

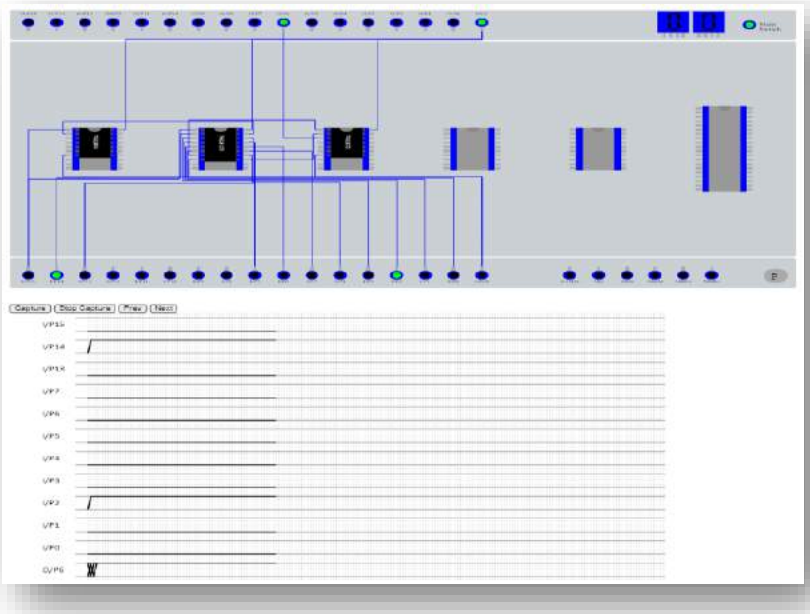


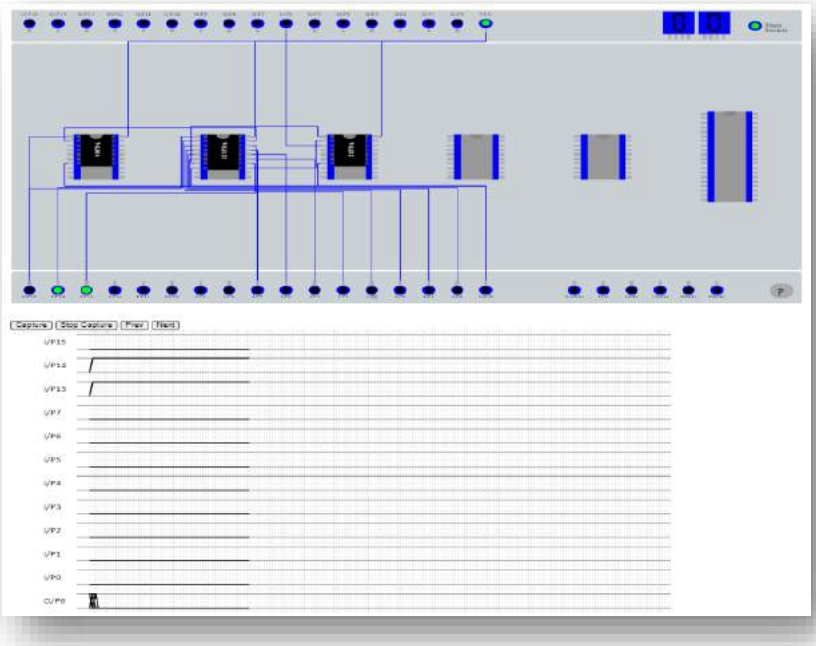


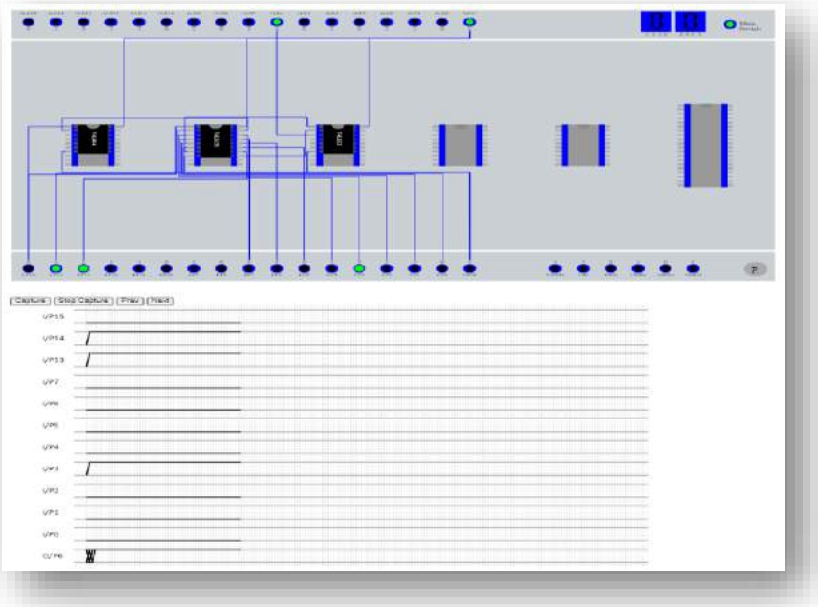


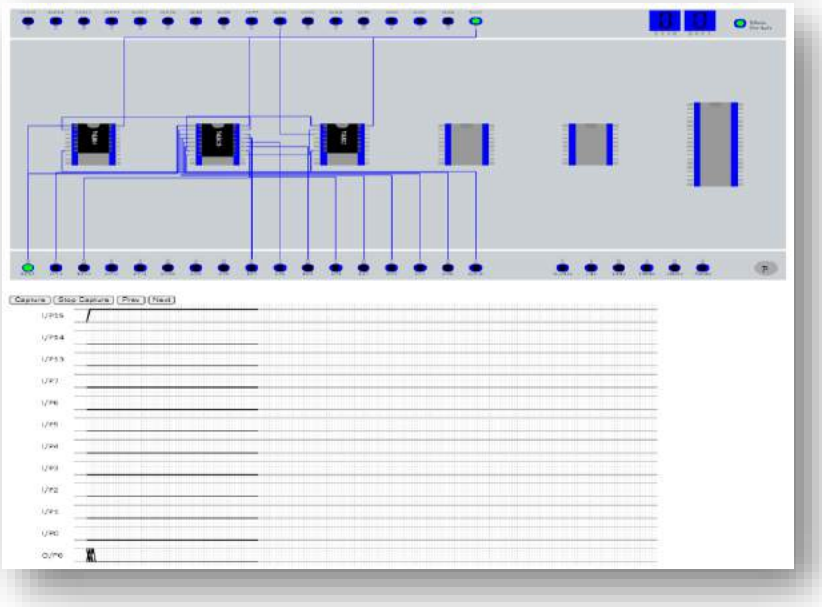


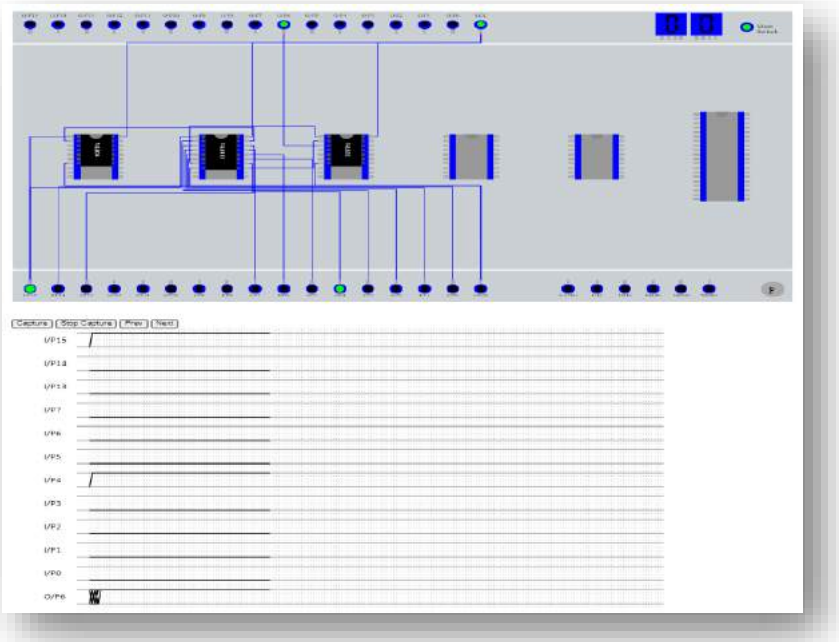


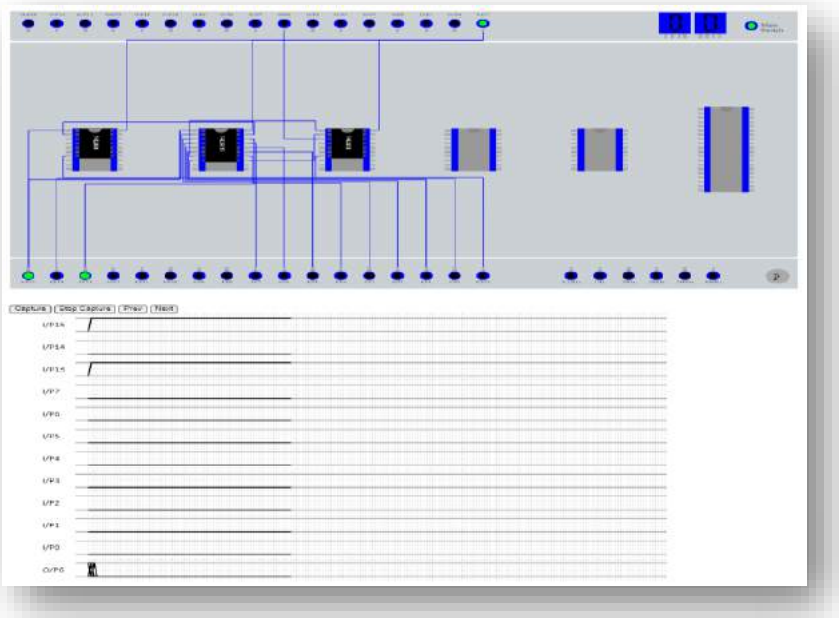


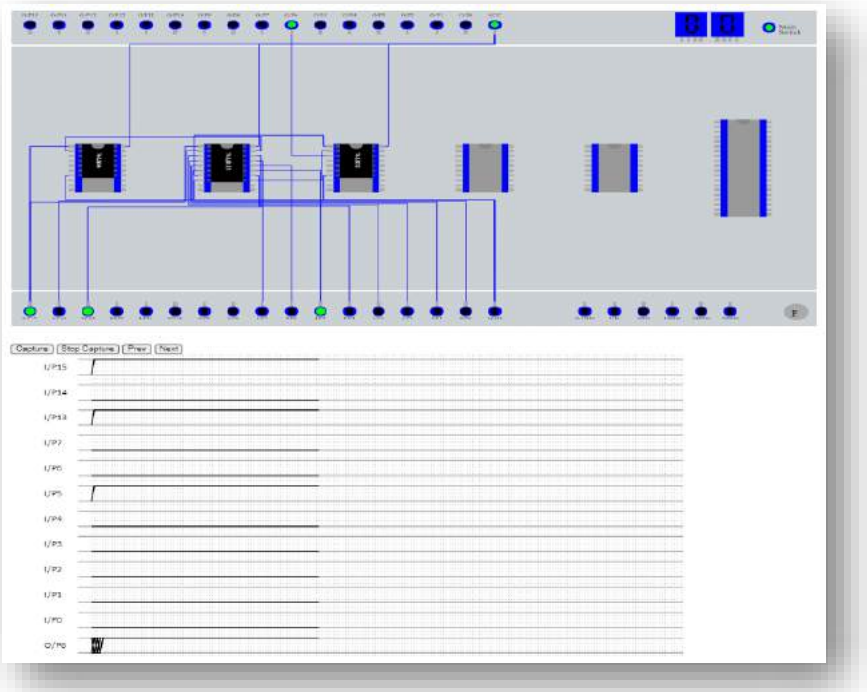


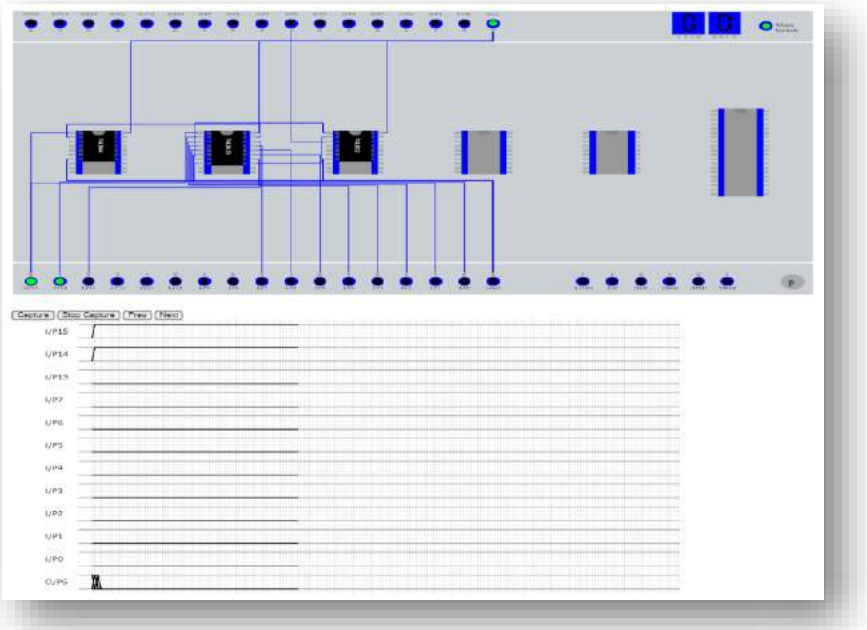


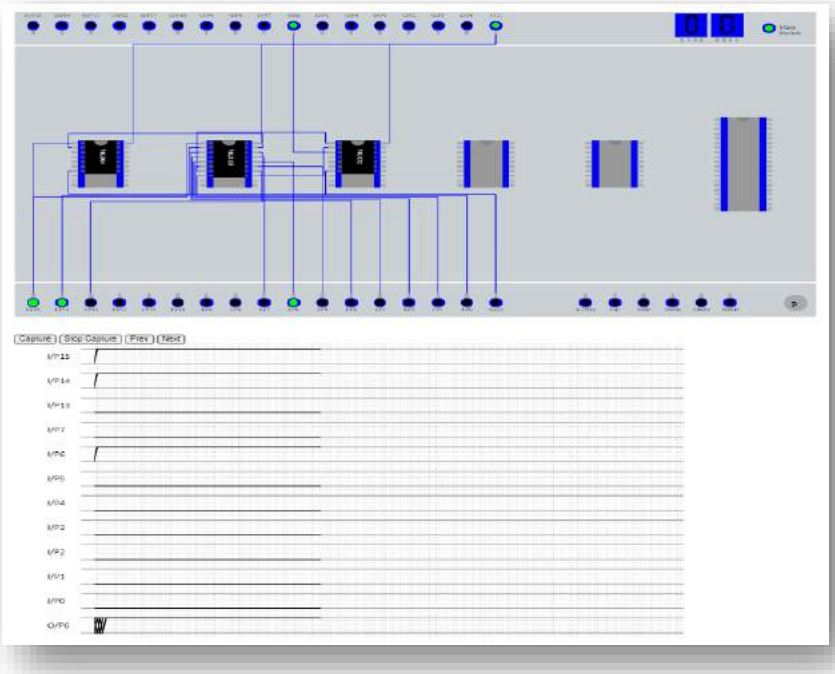


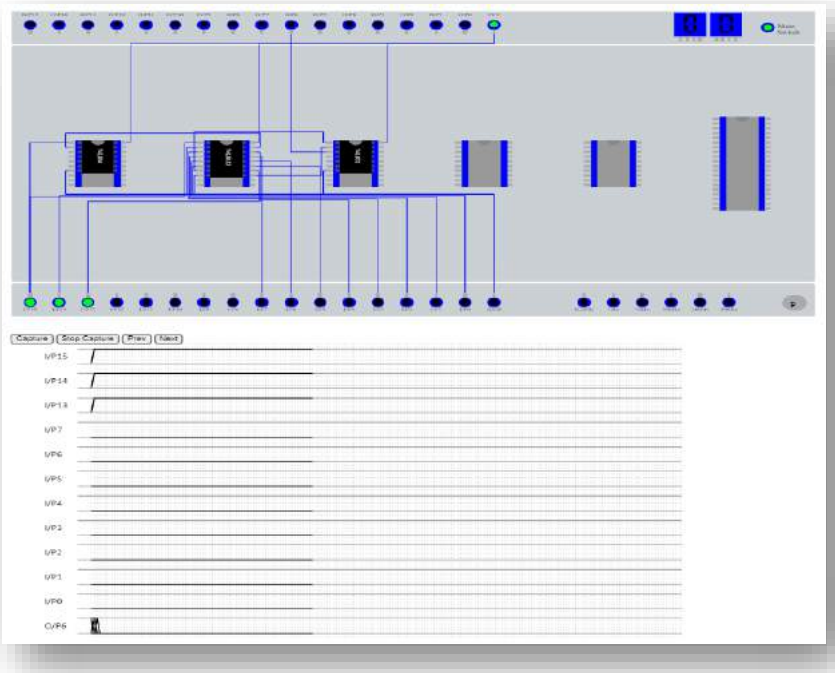


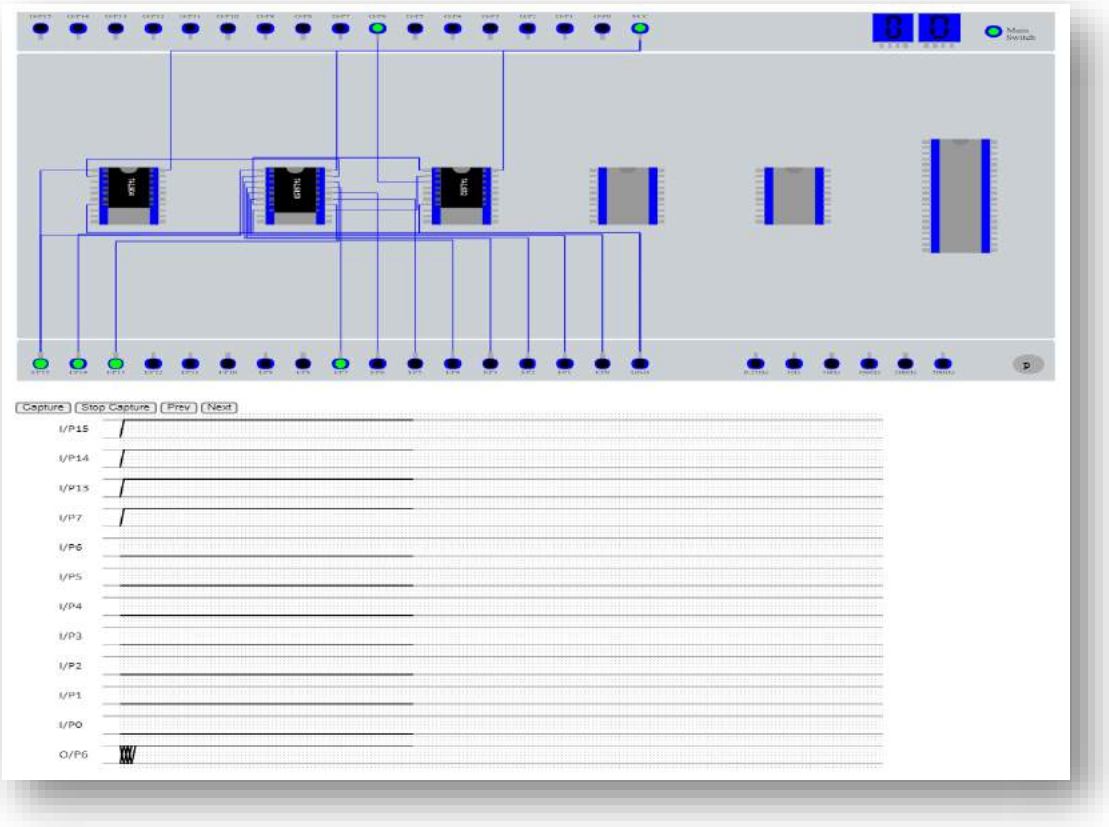



Capture
Stop Capture
Prev
Next
I/P15
I/P14
I/P13
I/P7
I/P6
I/P5
I/P4
I/P3
I/P2
I/P1
I/P0
O/P0


Capture
Stop Capture
Prev
Next
I/P15
I/P14
I/P13
I/P7
I/P6
I/P5
I/P4
I/P3
I/P2
I/P1
I/P0
O/P0


Capture
Stop Capture
Prev
Next
I/P15
I/P14
I/P13
I/P7
I/P6
I/P5
I/P4
I/P3
I/P2
I/P1
I/P0
O/P6


Capture
Stop Capture
Prev
Next
I/P15
I/P14
I/P13
I/P7
I/P6
I/P5
I/P4
I/P3
I/P2
I/P1
I/P0
O/P6



Conclusion :- It verifies the design and implement 8:1 MUX using IC – 74LS153 and its truth table.

Experiment - 3

Aim :- To design and implement the given 4 variable function using IC74LS153 and verify its truth table.

Apparatus Required :- Perform the experiment using this link : http://vlabs.iitb.ac.in/vlabs-dev/labs/dldesignlab/experiments/four-variable-function-pvg/

Equipment Used :- IC - 74LS153

Theory :-

In Experiment # 1 we studied the use of IC 74LS153 as a Dual 4:1 MUX. We required only one IC 74LS153 to design an 8:1 MUX. Now in Experiment # 2 we shall study the application of MUX as a Universal Logic Generator for a four variable system.

To implement a four variable Universal Function Generator using IC 74LS153, let us follow these steps:

1. To begin with we shall design a 16:1 MUX using multiple 74LS153 ICs.
2. IC 74LS153 has two 4:1 MUXes within it. Each MUX has two select lines and four data input lines. So in all eight data inputs D0 to D7 are available from first IC # U1.
3. Similarly second IC # U2 provides another eight data inputs D8 to D15. Thus in all 16 input lines are now available.
4. To select one of these 16 data inputs, we need four select lines; let us label them as A B C & D; where A is the most significant bit (MSB) and D is the least significant bit (LSB).
5. Each IC has 4 select lines.
6. All eight select lines of U1 and U2 are connected together and labelled as select lines C & D.
7. To get additional two select lines we introduce the third IC U3.

Only one 4:1 MUX of U3 is required. The two select lines of U3 are labelled as A & B.

8. The strobe inputs 1G' & 2G' of all the ICs is connected to ground so that all the IC's are enabled for operation.

9. Thus in all we need three IC's for the design of 4-variable Universal Logic Generator as shown in Fig.1.

The output function F can be written as:

F = A'.B'.Y0 + A'.B.Y1 +A.B'.Y2 +A.B.Y3

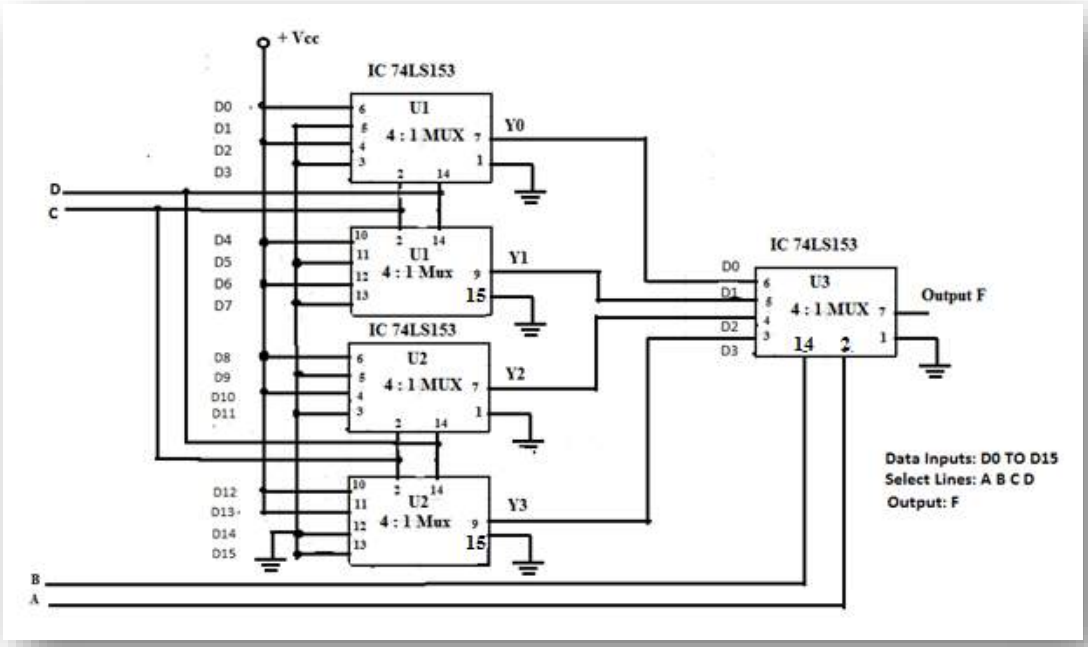


Fig.1. Four variable - Universal Logic Function Generator

For the logic diagram & pin diagram, refer Fig.2. The function table for IC 74LS153 is also provided for reference.

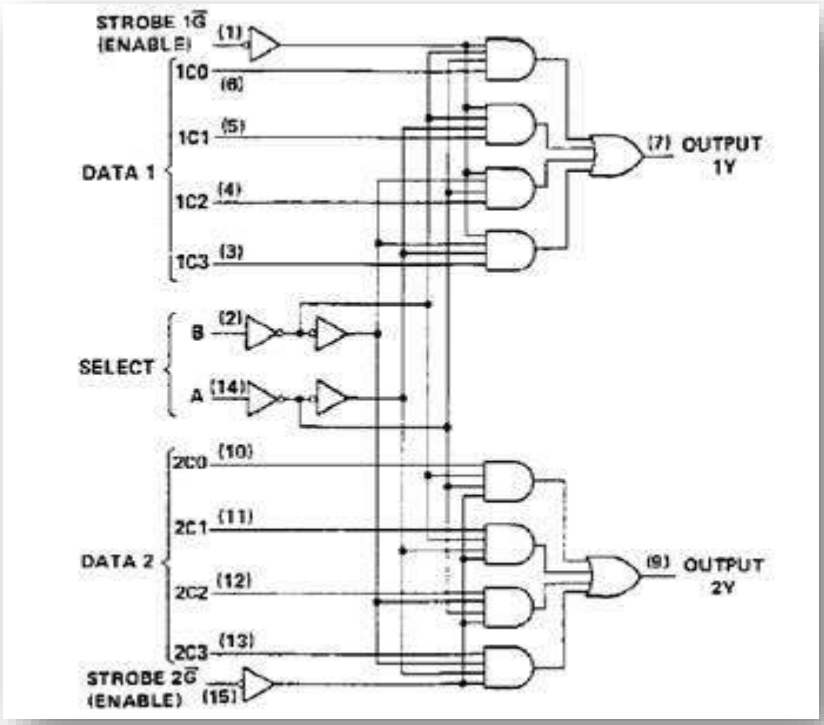
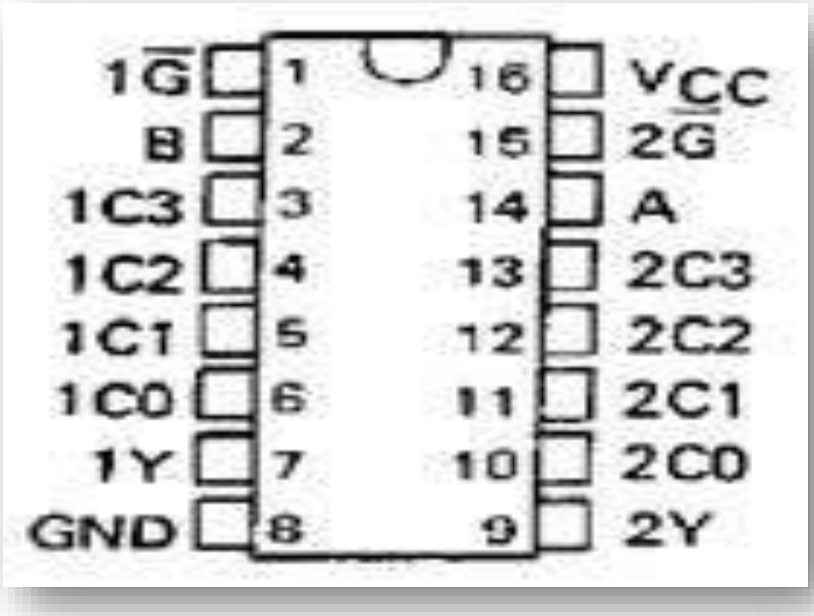


Fig. 2. Internal Structure & Pin Diagram of IC74LS153



FUNCTION TABLE

SELECT INPUTS		DATA INPUTS				STROBE	OUTPUT
B	A	C0	C1	C2	C3	$\bar{G}$	Y
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

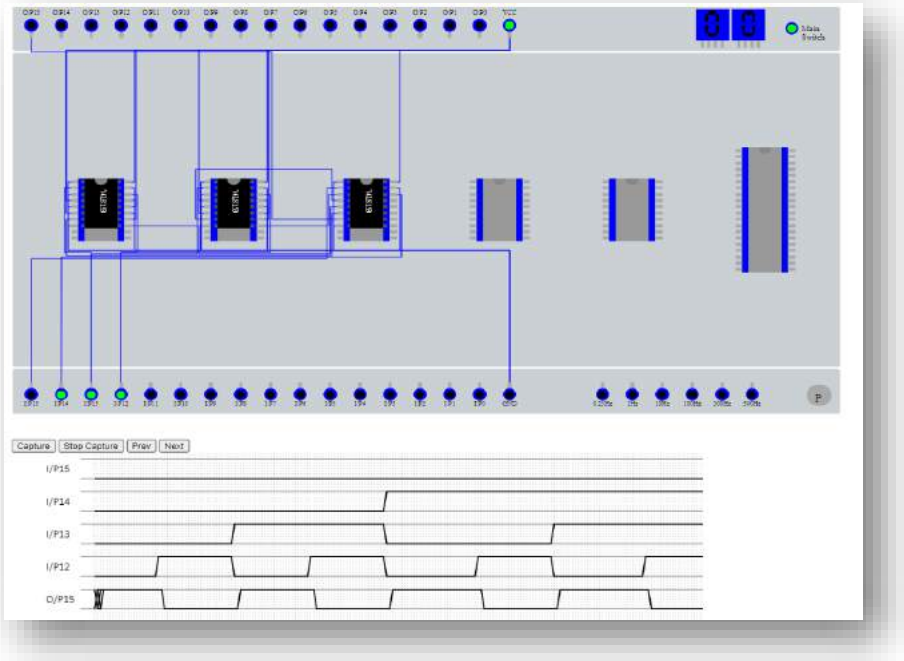
Select inputs A and B are common to both sections.
H = high level, L = low level, X = irrelevant

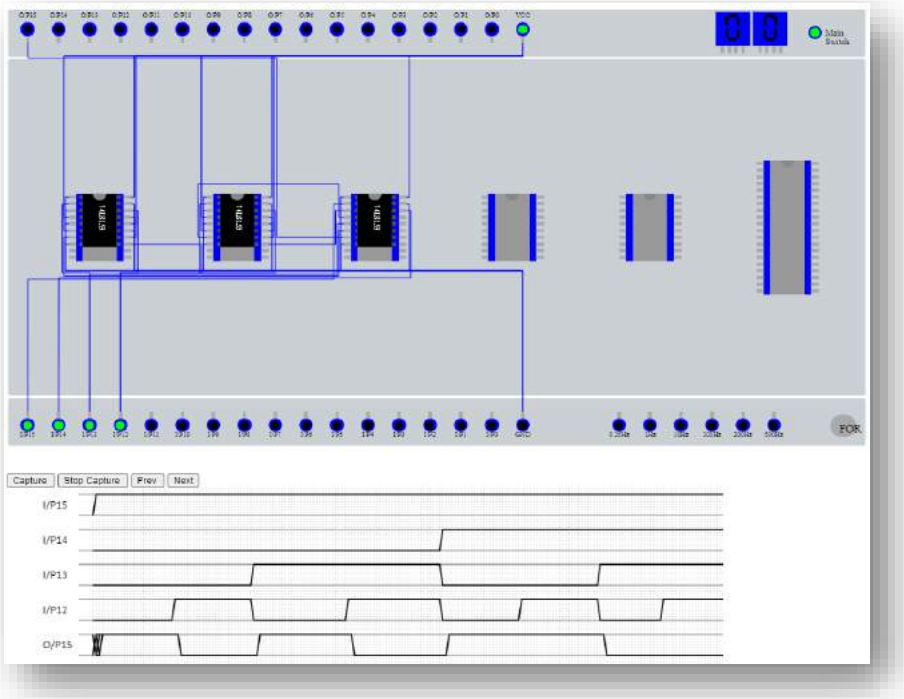
Fig. 3.Function Table

So the resultant truth table is :

Inputs				Output
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Observations :-





Conclusion :- It verifies the design and implement the given 4 variable functions using IC – 74LS153 and its truth table.

Experiment - 4

Aim :- To design and implement 3-bit Gray to Binary code converter using IC – 74LS138.

Apparatus Required :- Perform the experiment using this link :

http://vlabs.iitb.ac.in/vlabs-dev/labs/dldesignlab/experiments/gray-to-binary-code-converter-pvg/index.html

Device Used :- IC – 74LS138

Theory :-

Gray Code :-

The Gray code is unweighted code. Gray code exhibits only a single bit change from one code word to the next in sequence. Due to this specific feature Gray code is important in applications such as shaft position encoders, where error susceptibility increases with the number of bit changes between adjacent numbers in a sequence.

The relation between these two codes can be understood from the following diagram and equations :

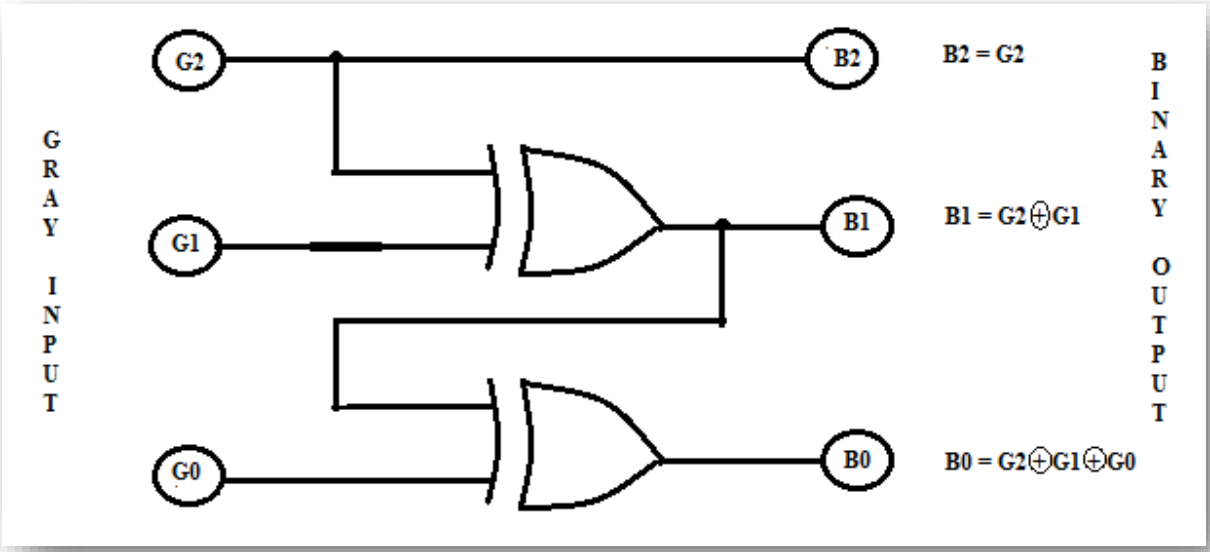


Fig.1. Gray Code to Binary Conversion

Decimal Equivalent	Gray Code			Binary Output		
	G2	G1	G0	B2	B1	B0
0	0	0	0	0	0	0
1	0	0	1	0	0	1
3	0	1	1	0	1	0
2	0	1	0	0	1	1
6	1	1	0	1	0	0
7	1	1	1	1	0	1
5	1	0	1	1	1	0
4	1	0	0	1	1	1

From the truth table it is observed that the output B2 is HIGH for minterms m_4, m_5, m_6 and m_7. Hence equation defining B2 output can be written as: $B2 = \Sigma\, m\,(4, 5, 6, 7)$. The output B1 is HIGH for

minterms m_2 and m_3 m_4 & m_5. Hence equation defining B1 output can be written as: B1 = Σ m (2, 3, 4, 5). The output B0 is HIGH for minterms m_1, m_2 m_4 m_7. Hence equation defining B0 output can be written as: B0 = Σ m (1, 2, 4, 7). The Binary outputs for given 3 bit Gray code are summarized as follows :

$$B2 = \Sigma m (4, 5, 6, 7)$$

$$B1 = \Sigma m (2, 3, 4, 5)$$

$$B0 = \Sigma m (1, 2, 4, 7)$$

These equations can be implemented by using IC74LS138 as a 3:8 decoder as shown in Fig.2. Every Binary output has 4-minterms each. Hence IC74LS20- four input NAND gate ICs are required to produce the final Binary equivalent.

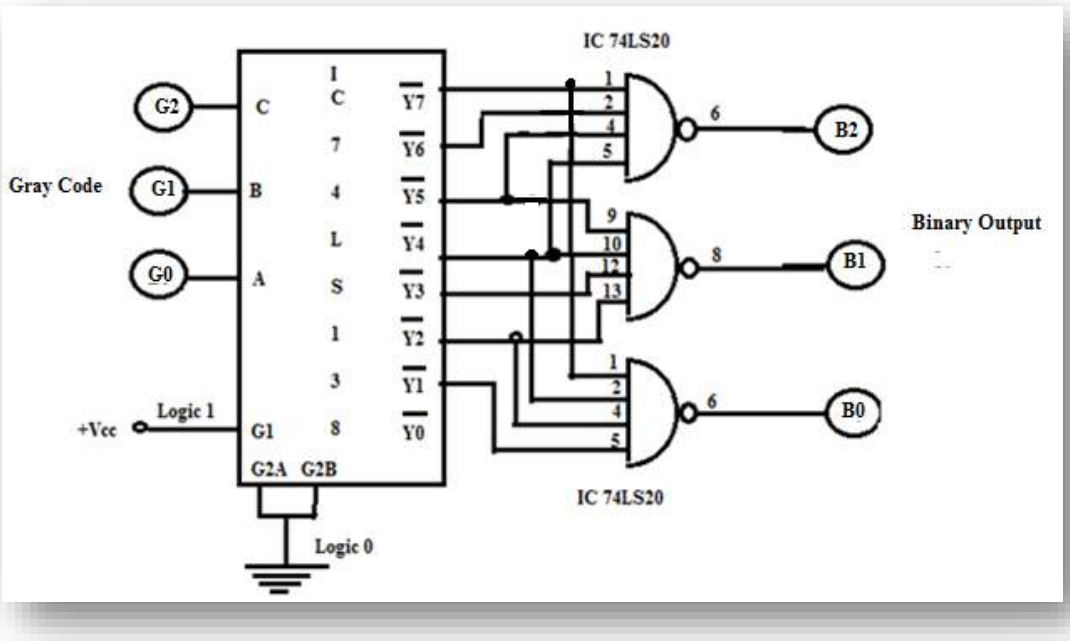
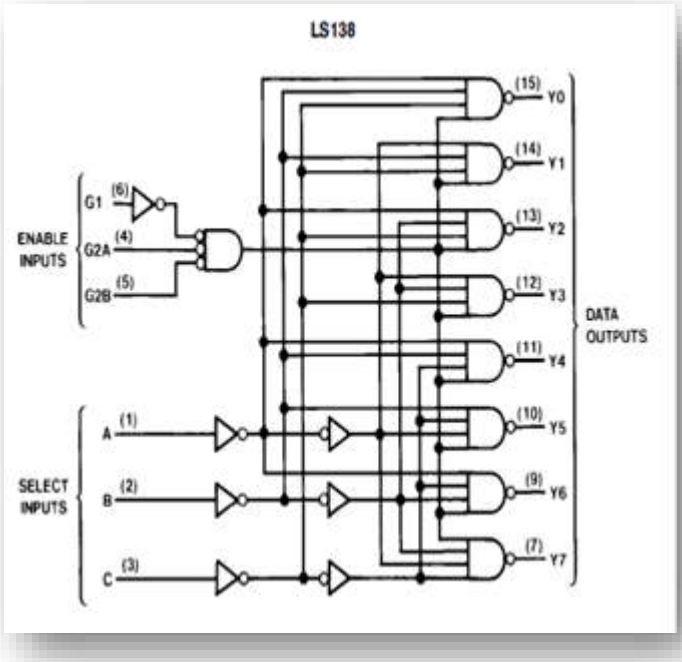
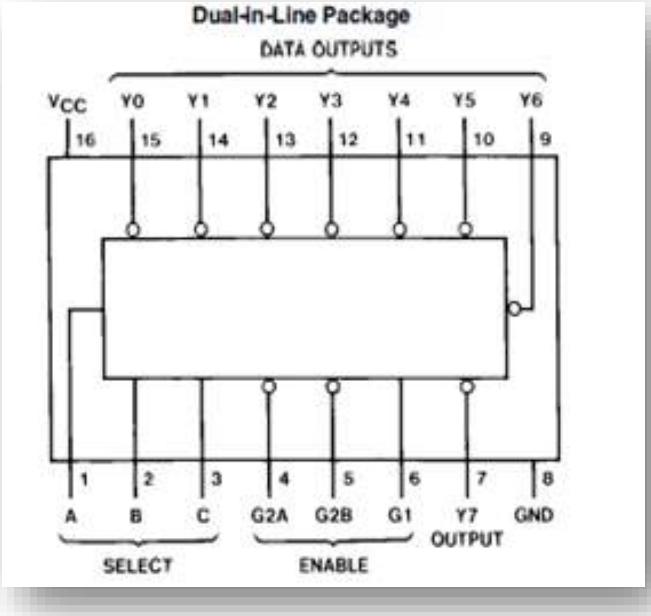


Fig.2. Implementation of Gray code to Binary Converter using IC74LS138 Decoder

The logic diagram, connection diagram and function table for IC 74LS138 are given below :


LS138
ENABLE INPUTS
G1
G2A
G2B
SELECT INPUTS
A
B
C
DATA OUTPUTS
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7


Dual-in-Line Package
DATA OUTPUTS
VCC
Y0
Y1
Y2
Y3
Y4
Y5
Y6
A
B
C
G2A
G2B
G1
Y7
GND
SELECT
ENABLE
OUTPUT

Function Table

LS138

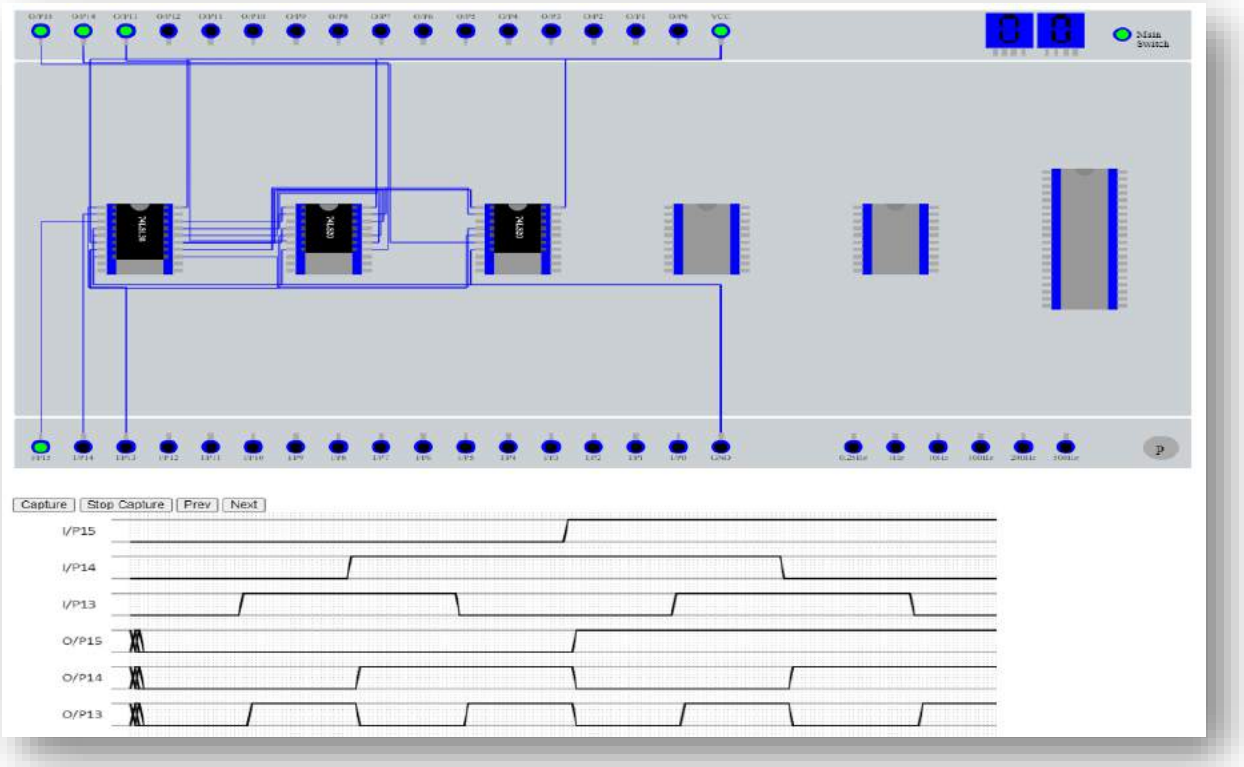
Inputs					Outputs							
Enable		Select										
G1	G2*	C	B	A	YO	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

* G2 = G2A + G2B

H = High Level, L = Low Level, X = Don't Care

Fig.3. Logic Diagram, Connection Diagram & Function Table of IC74LS138.

Observations :-



Conclusion :- It verifies the design and implement the 3-bit Gray code to Binary code converter using IC – 74LS138.

Experiment - 5

Aim :- To design and implement 3-bit Binary to Gray Code converter using IC - 74LS138.

Apparatus Required :- Perform the experiment using this link :

http://vlabs.iitb.ac.in/vlabs-dev/labs/dldesignlab/experiments/binary-to-gray-code-converter-pvg/index.html

Device Used :- IC - 74LS138

Theory :-

Binary Code : It is weighted code i.e. it is a code in which weight is assigned to every symbol position in the code word. The positional weights in binary code are shown below :

Binary Code :-----> ..... 2^4 2^3 2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3} 2^{-4} ...

Decimal ----------> ..... 16 8 4 2 1 1/2 1/4 1/8 1/16 ...

Gray Code : It is a non-weighted code i.e. it does not have any specific/fixed weight assigned to each symbol position in the code word. The unique feature of Gray code is that at a time only “one” bit changes. In other words, in Gray code every new code differs from the previous in terms of one single bit.

Use of Gray Code: For correct measurement of angular position of shaft.

Design : The binary and their equivalent Gray Codes are related as shown in Fig.1

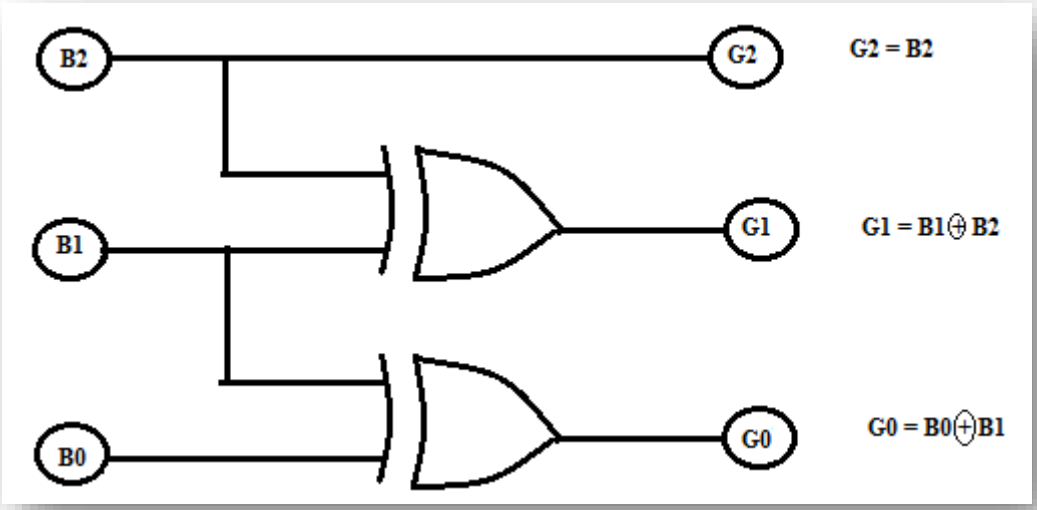


Fig.1. Logic Diagram showing relation between Binary and Gray Code.

Let us now prepare the 3 bit- Binary to 3 bit Gray code truth table :

Decimal Equivalent	Binary			Gray Code		
	B2	B1	B0	G2	G1	G0
0	0	0	0	**0**	**0**	**0**
1	0	0	1	**0**	**0**	**1**
2	0	1	0	**0**	**1**	**1**
3	0	1	1	**0**	**1**	**0**
4	1	0	0	**1**	**1**	**0**
5	1	0	1	**1**	**1**	**1**
6	1	1	0	**1**	**0**	**1**
7	1	1	1	**1**	**0**	**0**

From the truth table it is observed that the output G2 is HIGH for minterms m4, m5, m6 and m7. Hence equation defining G2 output

can be written as : G2 = Σ m (4, 5, 6, 7). The output G1 is HIGH for miniterms m2, m3, m4 & m5. Hence equation defining G1 output can be written as : G1 = Σ m (2, 3, 4, 5). The output G0 is HIGH for minterms m1, m2, m5 and m6 . Hence equation defining G1 output can be written as : G0 = Σ m (1, 2, 5, 6). The Gray outputs for given 3 bit Binary are summarized as follows :

G2 = Σ m (4, 5, 6, 7)

G1 = Σ m (2, 3, 4, 5)

G0 = Σ m (1, 2, 5, 6)

The above Boolean expressions can be implemented using IC 74LS138 as a 3:8 decoder by just connecting its relevant outputs to 4- input NAND gates as shown in Fig.1. Four input IC 74LS20 is used to produce the final Gray equivalent.

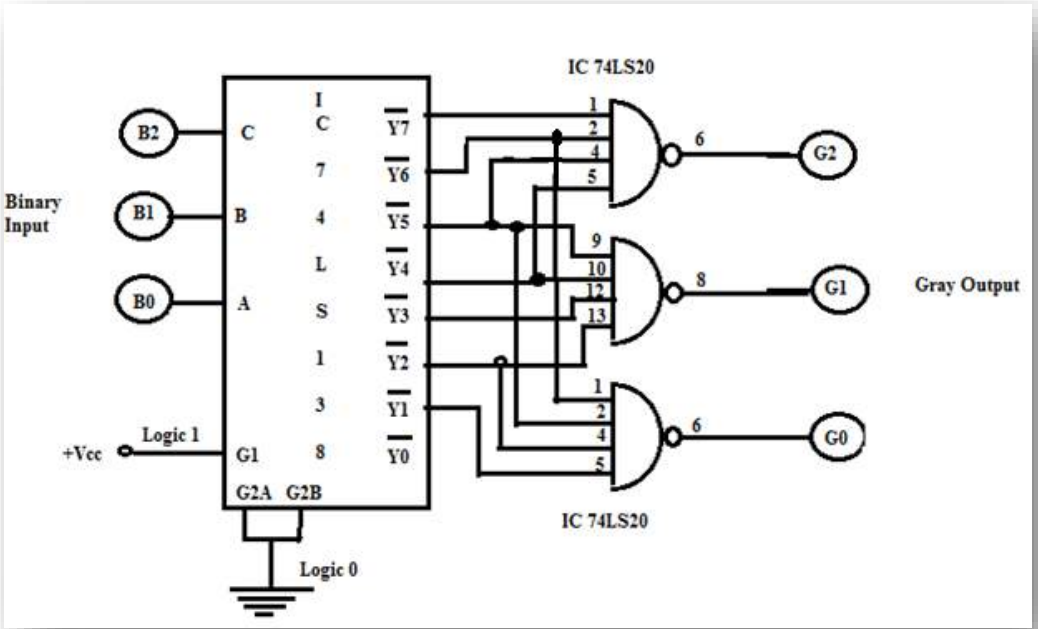


Fig.2. Binary to Gray Code Converter using IC 74LS138

The logic diagram, connection diagram and function table for IC 74LS138 are given below :

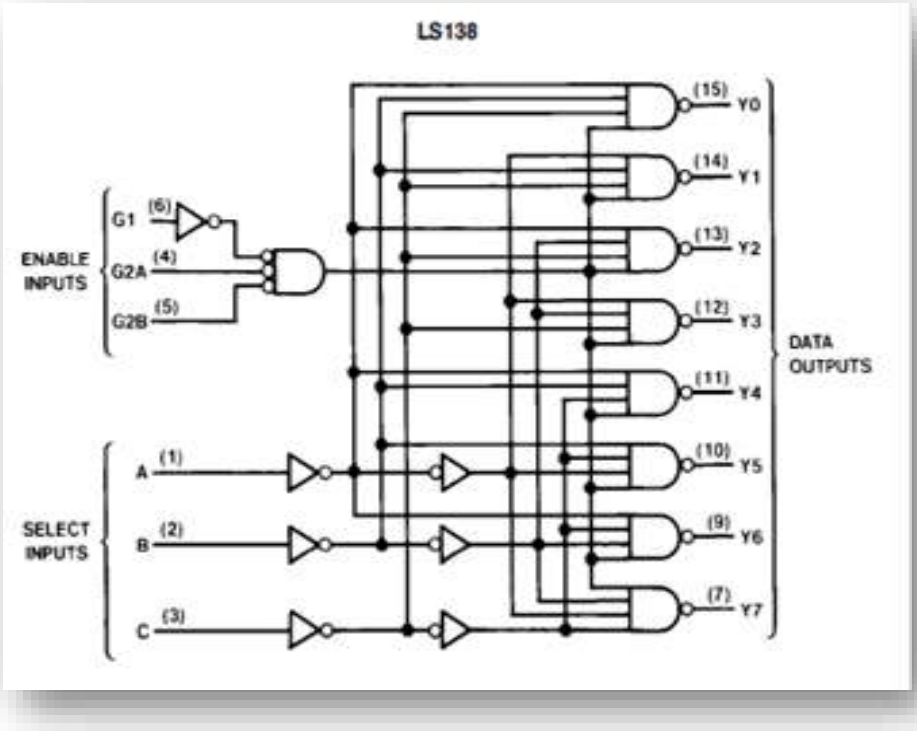
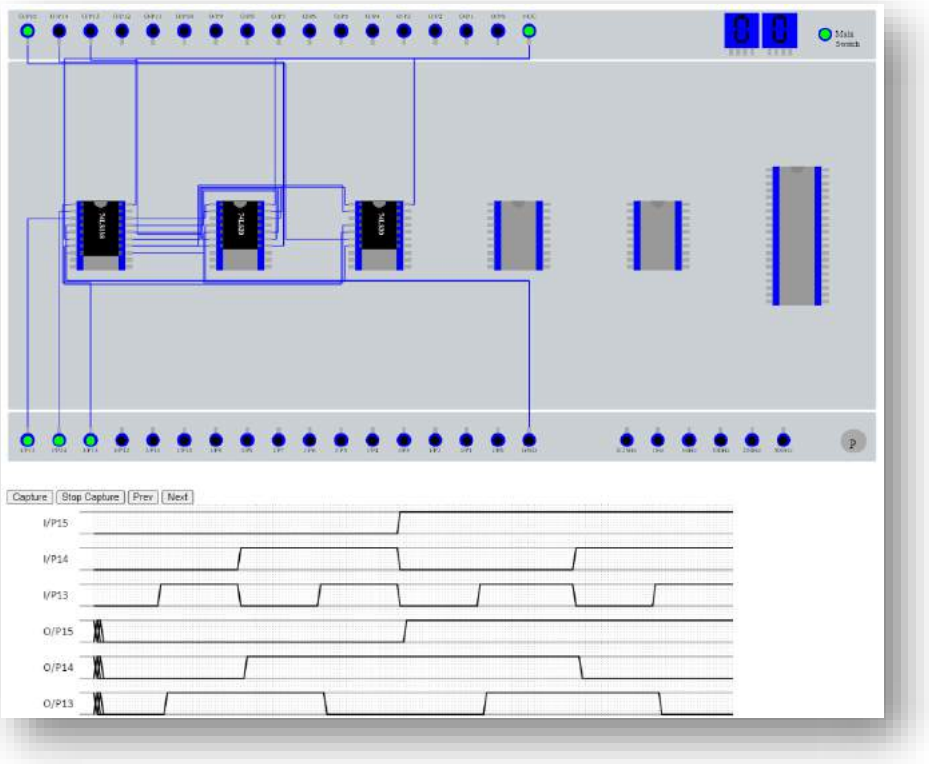


Fig.3. Logic Diagram, Connection Diagram & Function Table of IC 74LS138

Observations :-



Conclusion :- Design and implementation of 3-bit Binary to Gray Code converter using IC - 74LS138 is verified.

Experiment - 6

Aim :- To construct logic circuit of washing machine control using a generalized simulator and verify its output.

Apparatus Required :- Perform the experiment using this link :

https://da-iitb.vlabs.ac.in/exp/washin-machine-control/

Device Used :- AND and NOT Gate

Theory :-

Introduction :

When logic gates are connected together to produce a specific output for certain specific combinations of input variables, with no storage involved, the resulting circuit is called as a *Combinational logic* circuit. The combination of basic gates can be used for a variety of applications such as washing machine control, level monitoring and indicating applications in manufacturing processes, elevator control applications, a warning indicating applications and binary addition - subtraction and multiplication circuits.

Application : Washing Machine Controller

For simplicity, consider a three-sensor based washing machine controller namely *Door Sensor, Water Level Sensor and Temperature Sensor* that produce *digital outputs*. Let the controlling action include control of *Water Valve, Heater and Motor*. All these are digitally controlled devices.

Concept :

The motor of the washing machine turns ON when the *right temperature*, the *right water level* and obviously when *the door of the machine is closed*.

The system design involves three inputs: D, L & T representing Door position, Level & Temperature respectively. It controls three output devices: W, H & M representing Water Valve, Heater & Motor respectively. Let us decide the logics behind the system:

D = 0 ------- Door Open;

D = 1-------- Door Closed (desired)

L = 0 --------Water Level is LOW;

L = 1 --------Water Level is HIGH (satisfactory)

T = 0 --------Temperature is LOW

T = 1 --------Temperature is HIGH (right value)

The truth table for this application can be developed by logical reasoning:

1. For turning ON of any of the output devices, *the washing machine door/lid should be closed* at any point of time, so only last four cases of the truth table should to be considered where D takes a value 1.
2. If door is closed & water level is LOW, the water valve should be turned ON.
3. If door is closed, water level is satisfactory (HIGH) & the temperature is low, the heater should be turned ON.
4. Whereas when the door is closed, water level is satisfactory and the temperature is right, the motor should turn ON.

Door(D)	Level(L)	Temperature(T)	Valve(V)	Heater(H)	Motor(M)
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	0	0
1	0	0	1	0	0
1	1	0	0	1	0
1	1	1	0	0	1

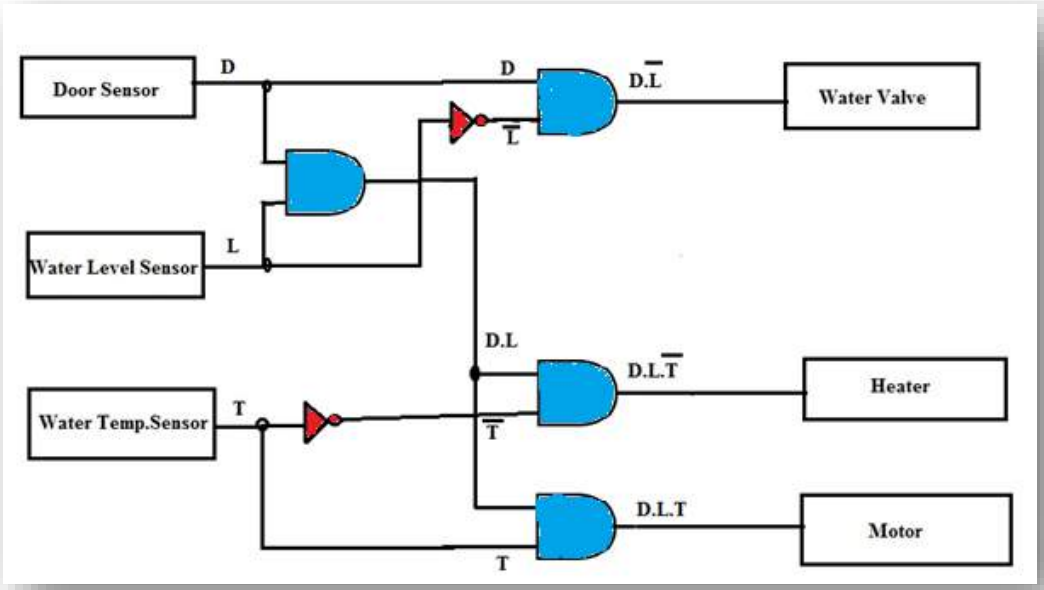
Considering only those input conditions that produce a HIGH output, we get the reduced Boolean expressions for controlling as follows:

Water Valve (V) = D.L'

Heater (H) = D.L.T'

Motor (M) = D.L.T

The corresponding combinational logic circuit is as shown in Figure 1.

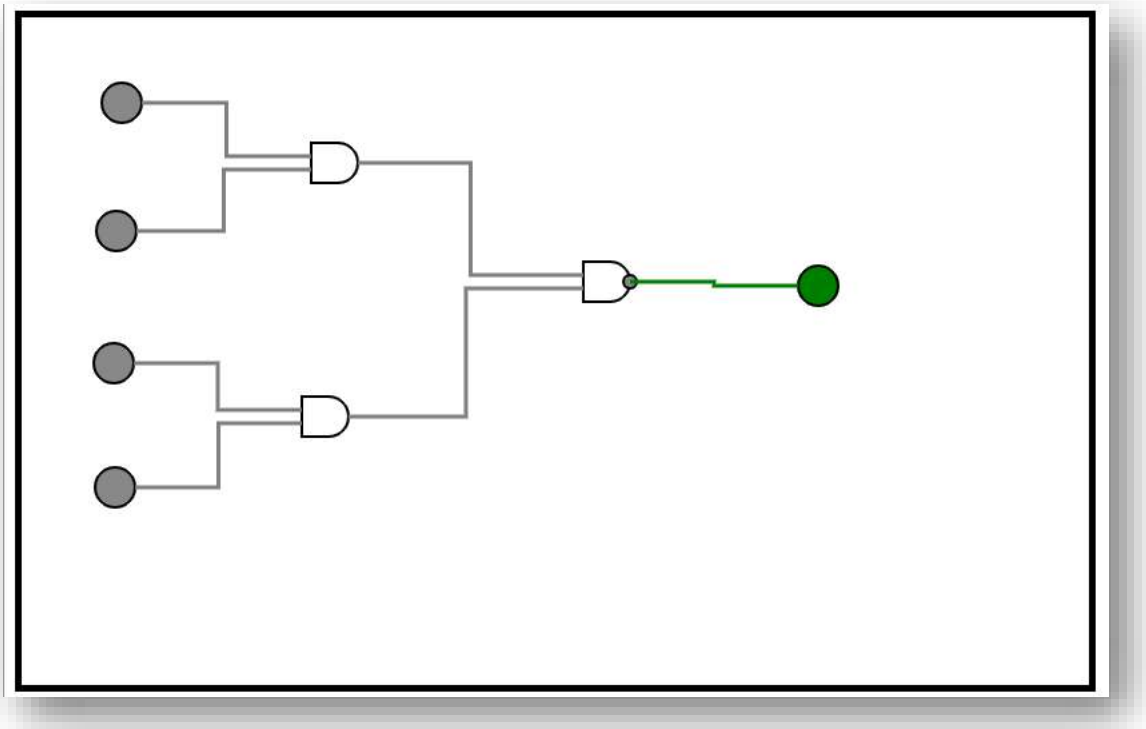


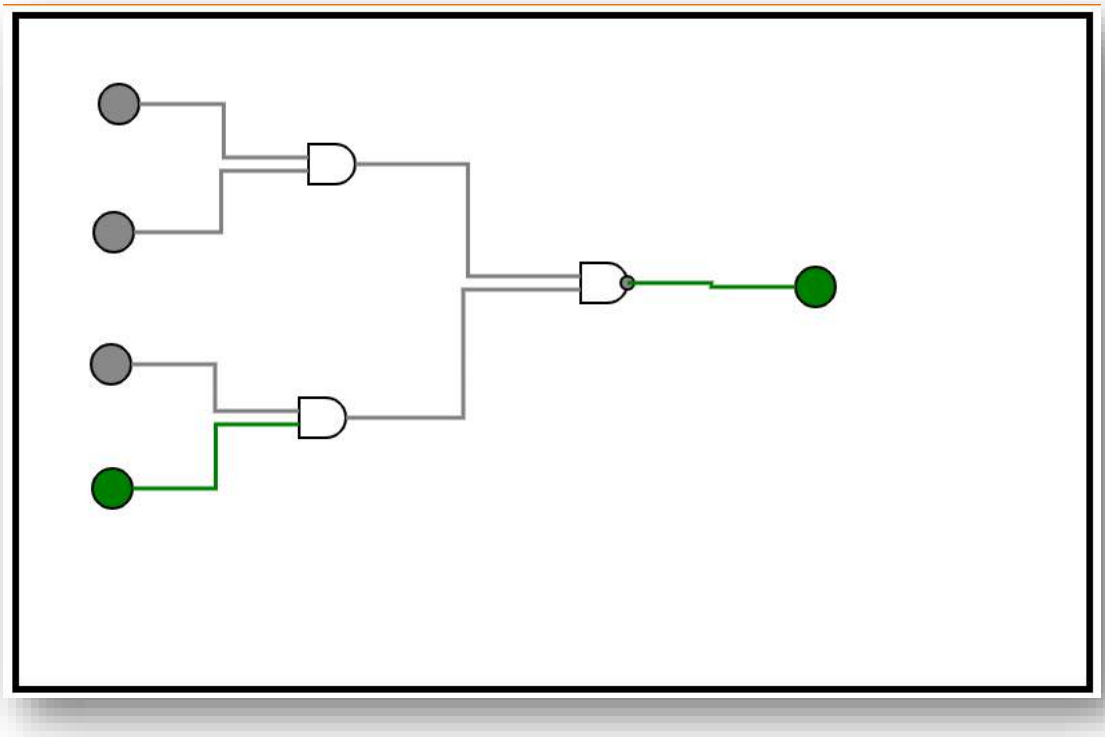
Observations :-

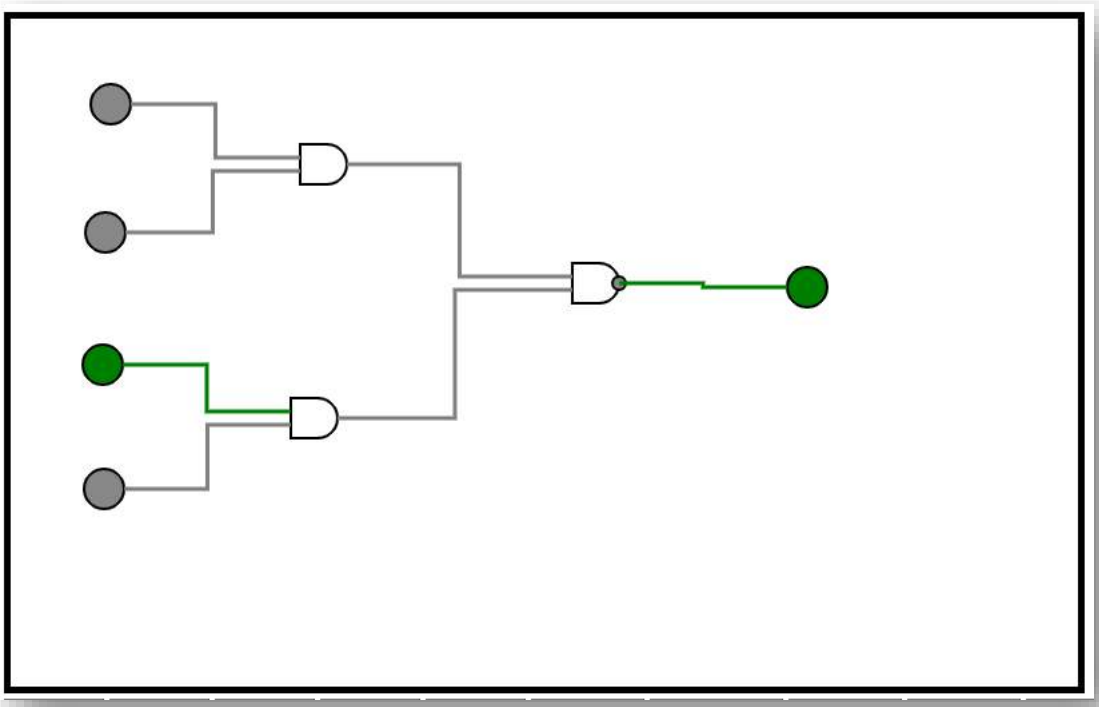
Truth Table :

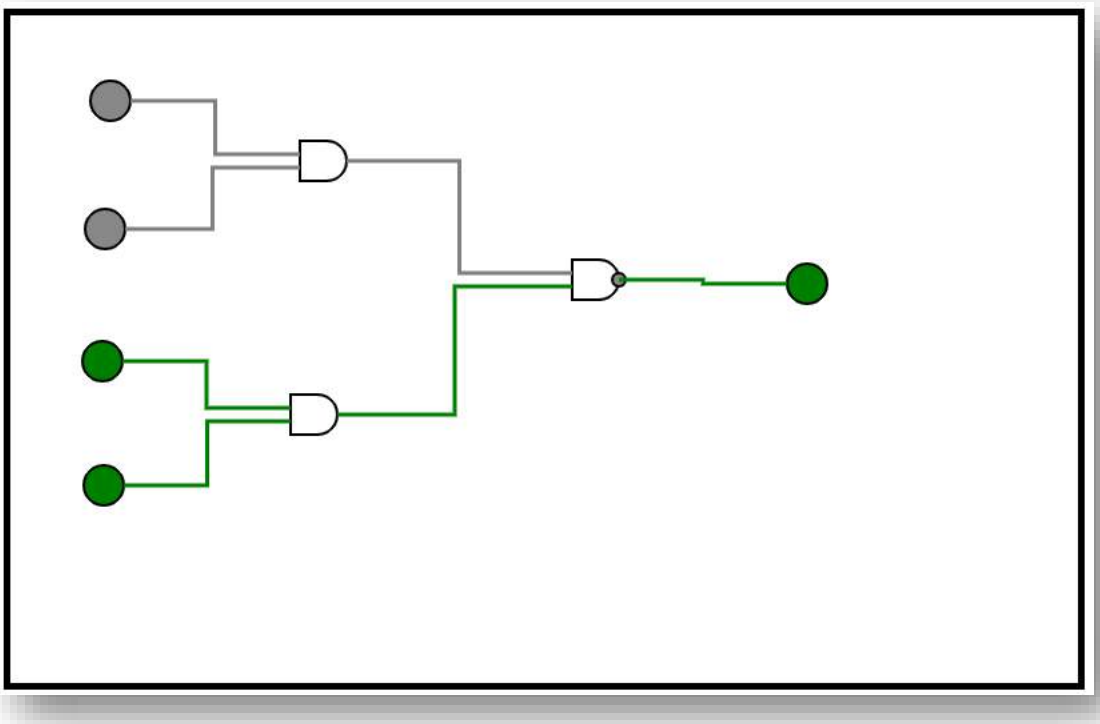
A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

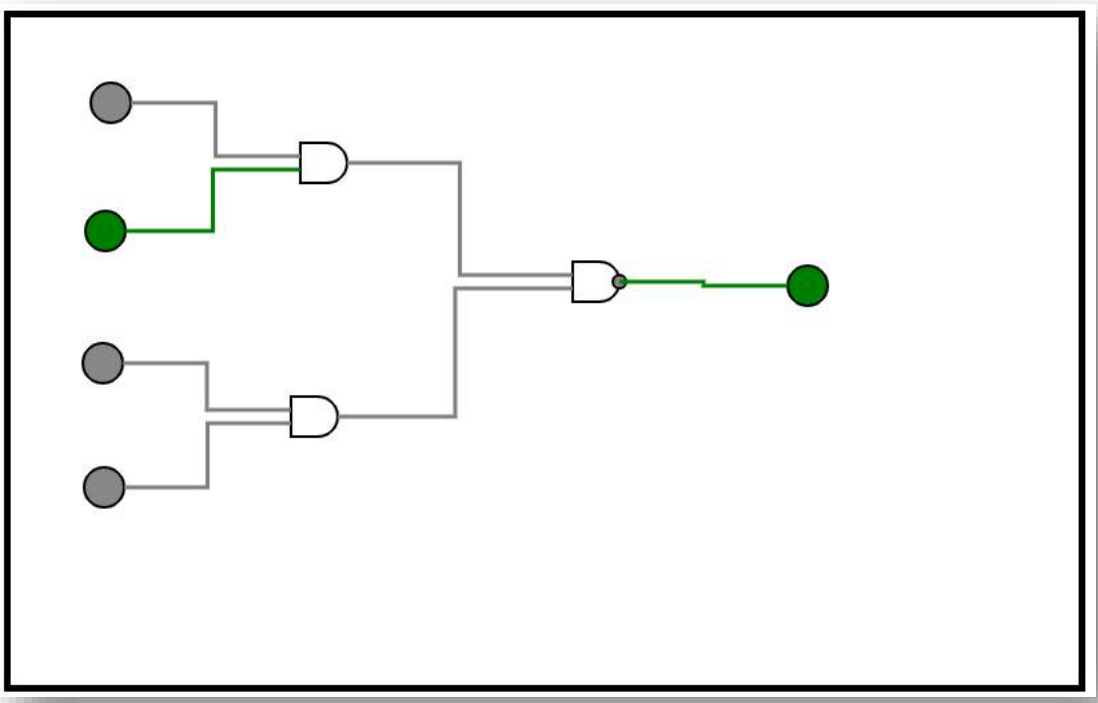
Circuit Diagram:

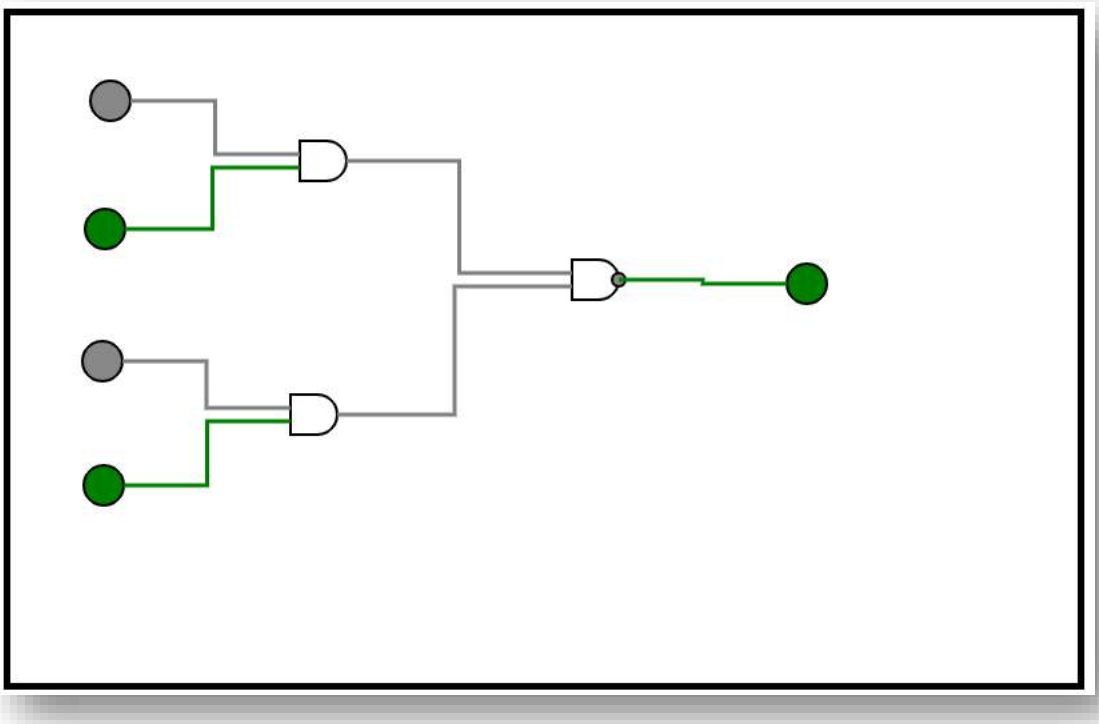


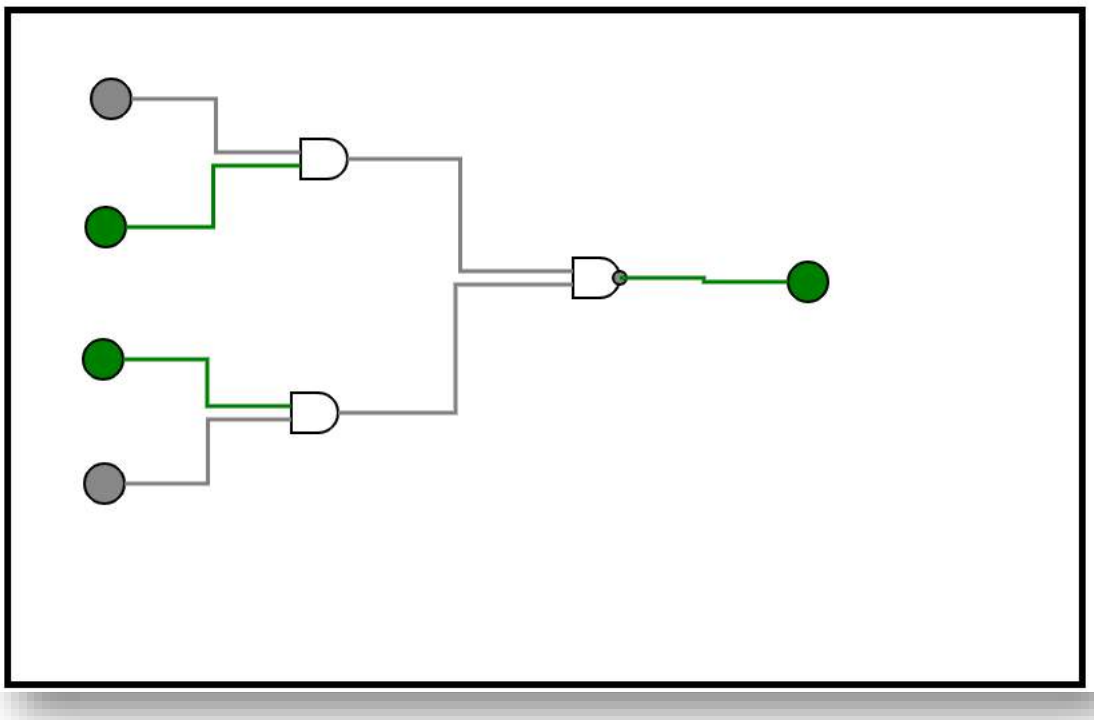


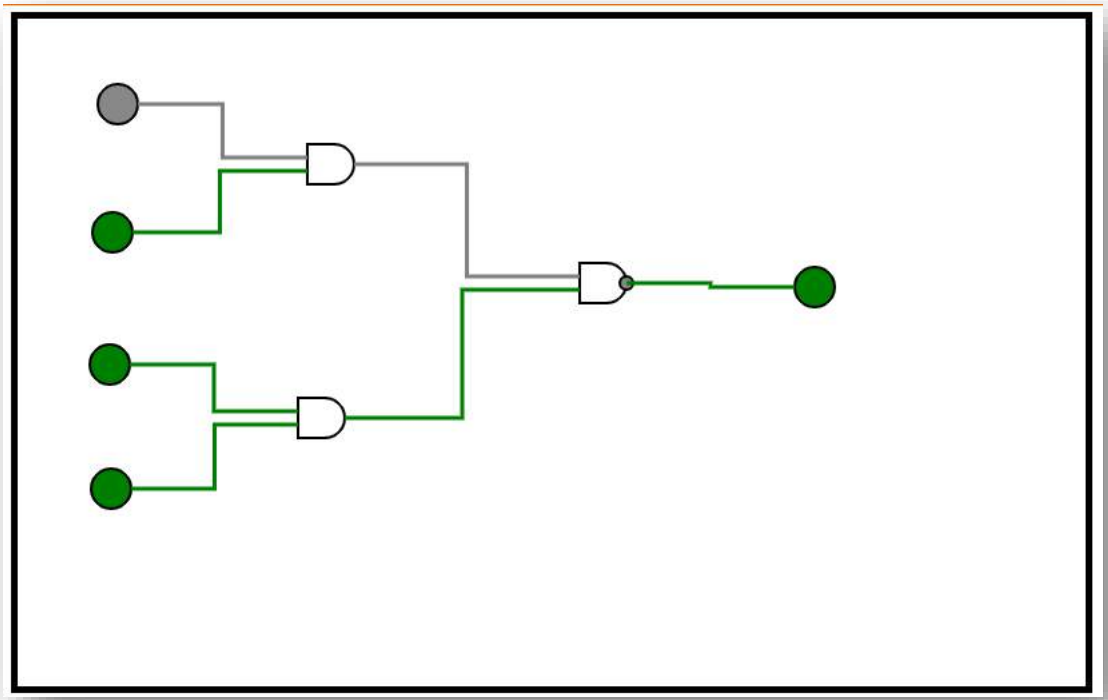


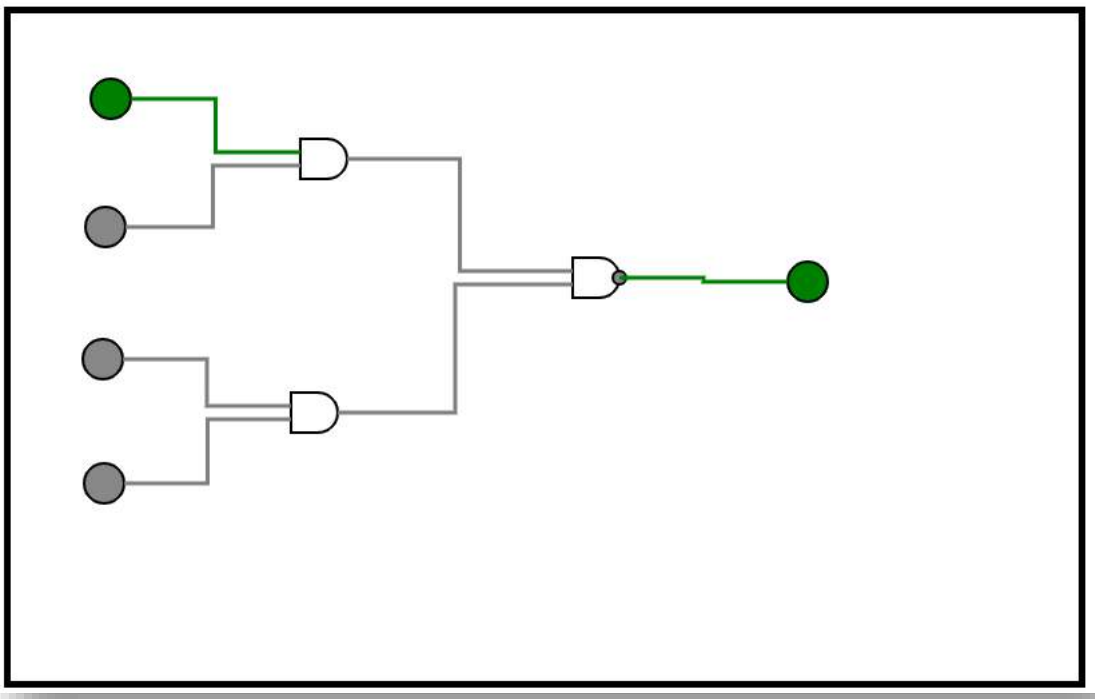


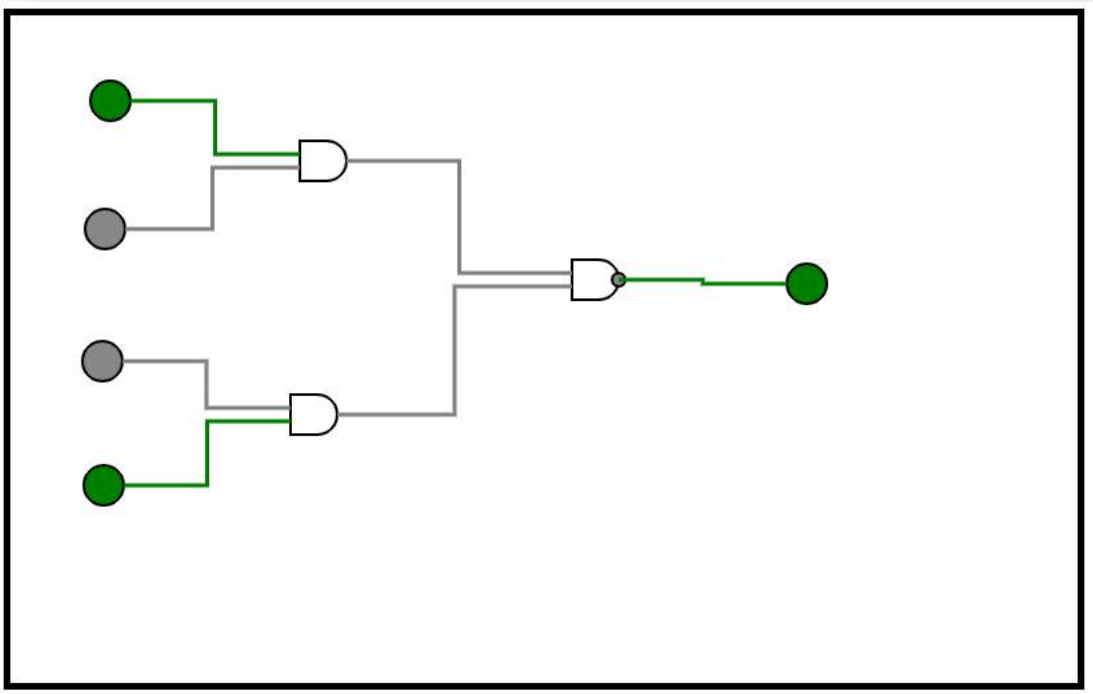


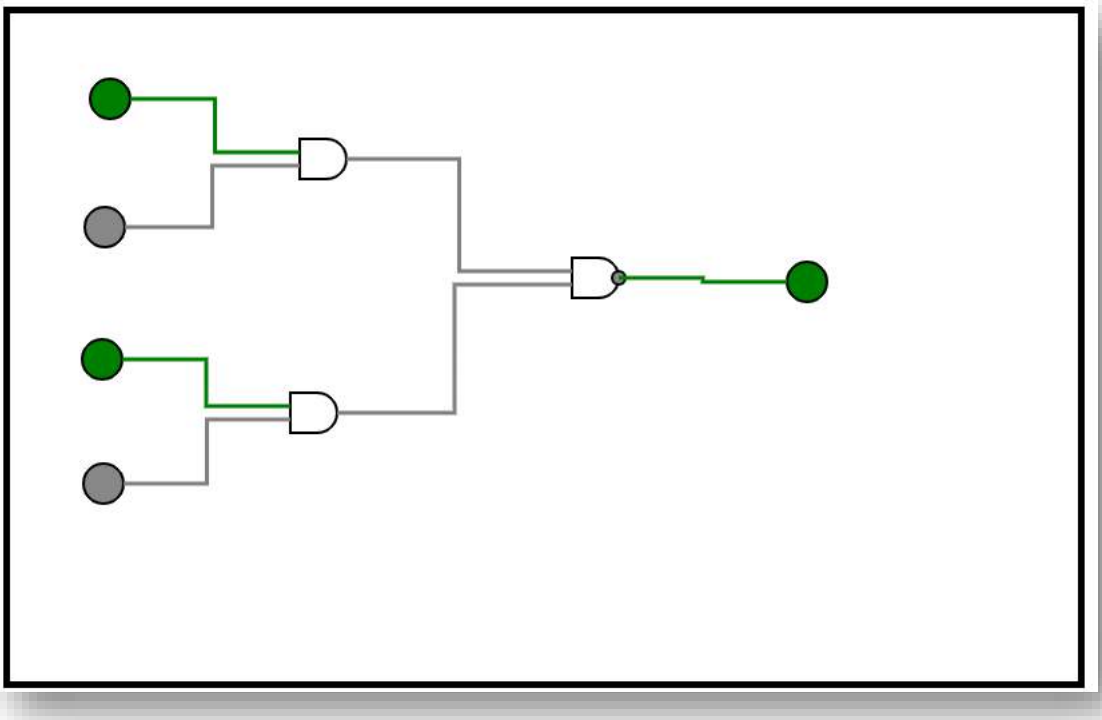


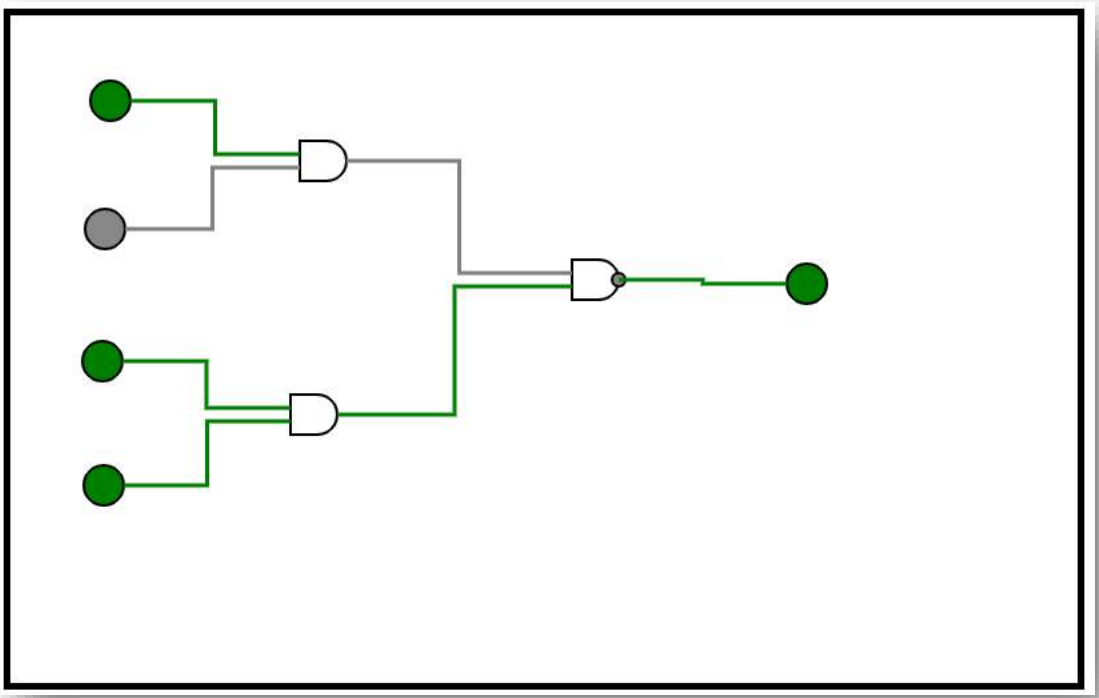


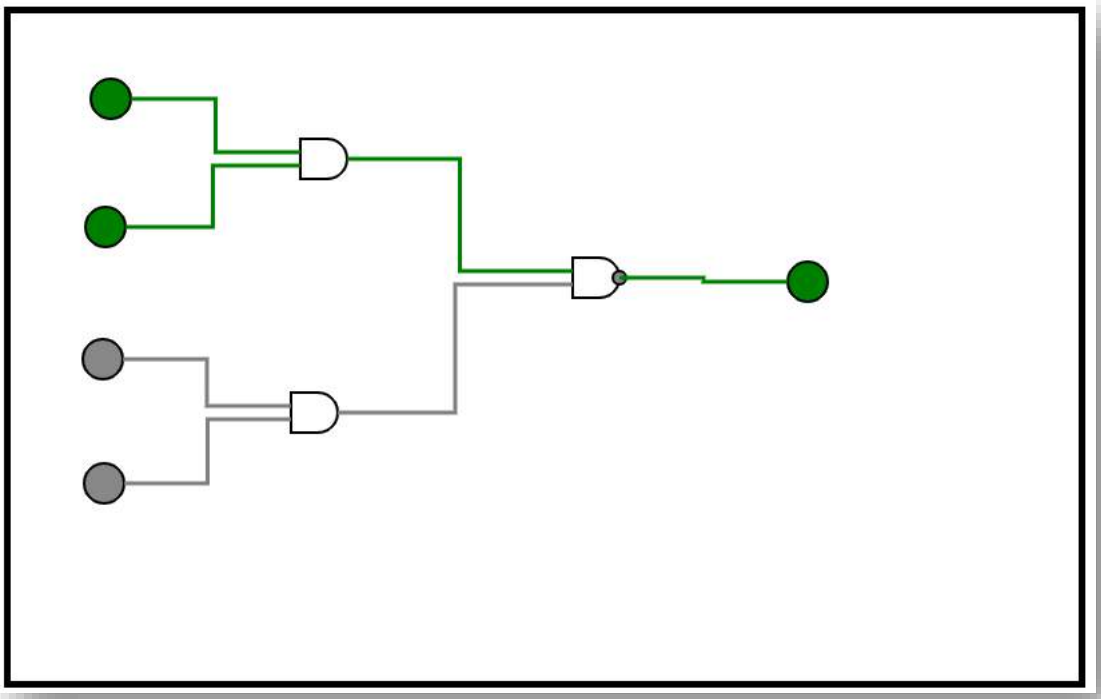


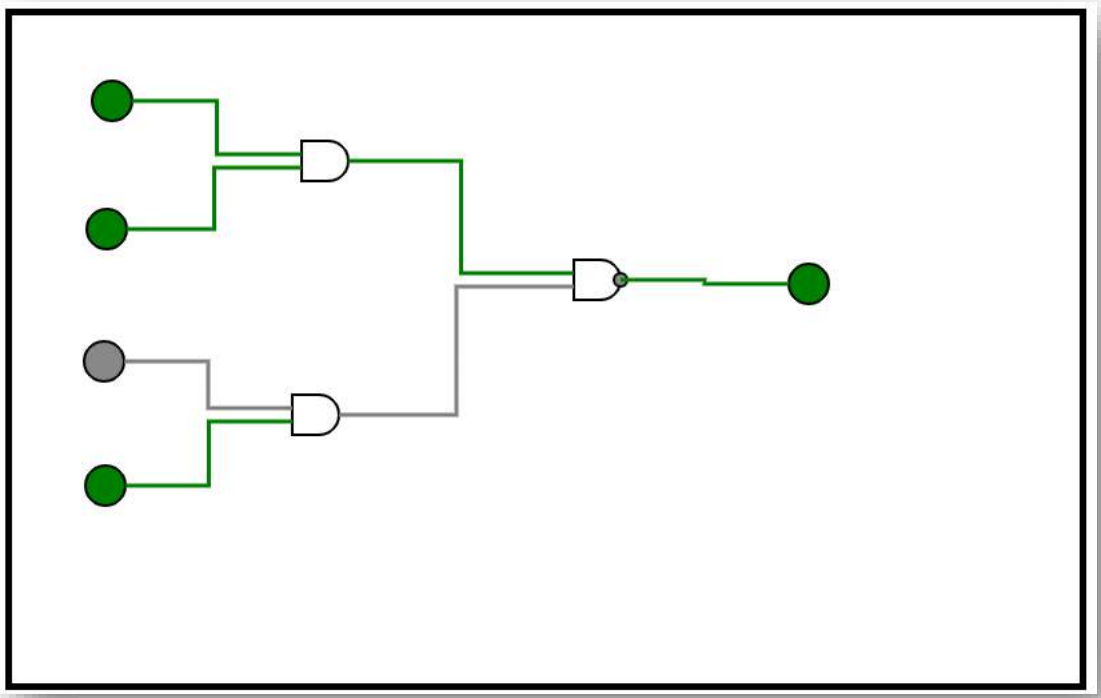


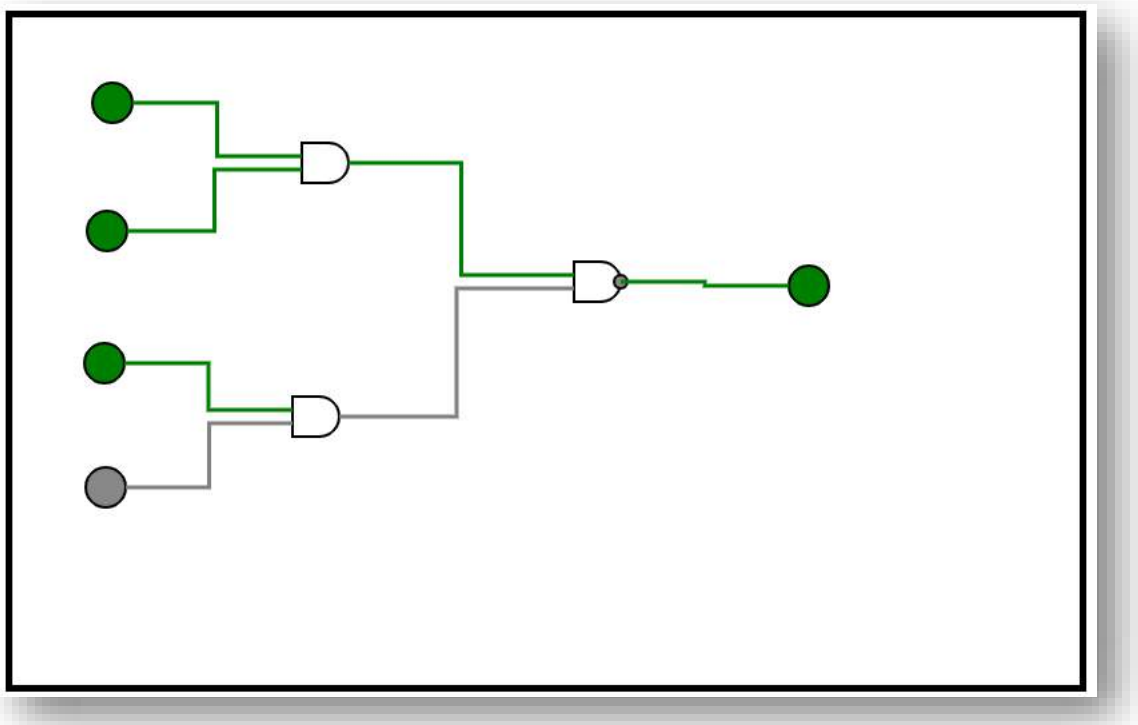


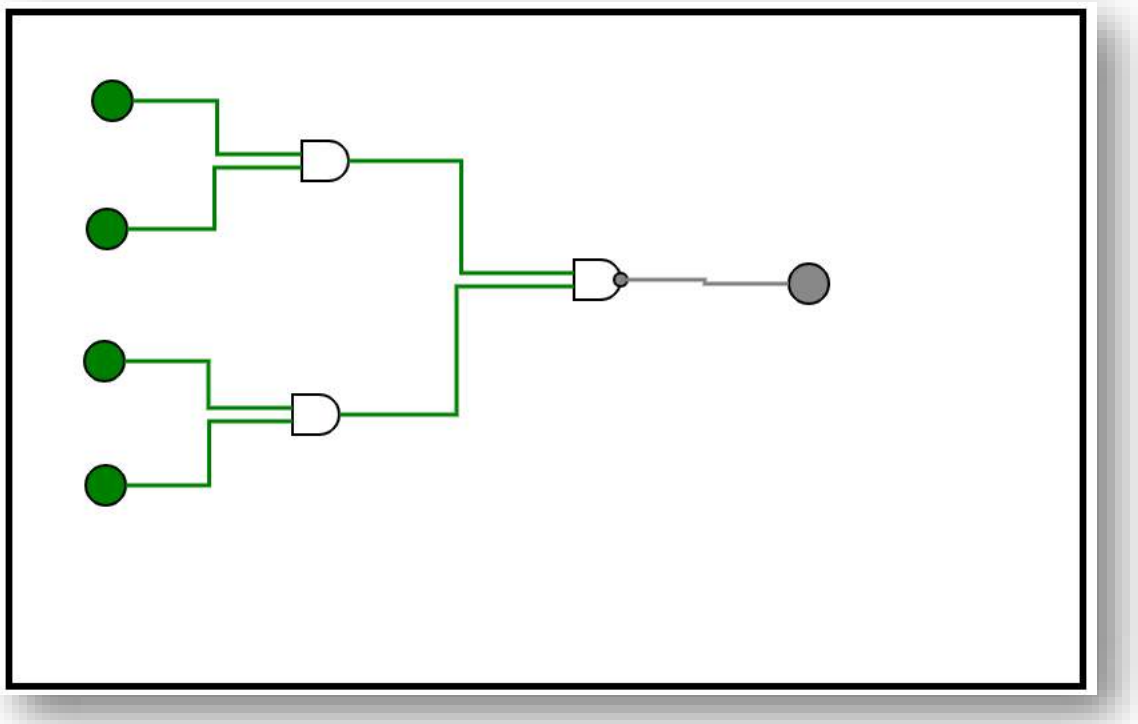












Conclusion :- It verifies logic circuit of washing machine control using a generalized simulator and its output.

Experiment - 7

Aim :- The aim of this experiment is to apply the basic OR gate logic in industrial control process. The user will be able to drag and drop the OR gate on the blank canvas and make input-output connections and verify the truth table.

Apparatus Required :- Perform the experiment using this link :

http://vlabs.iitb.ac.in/vlabs-dev/labs/digital_application/experiments/industrial-control-pvg/index.html

Device Used :- OR Gate

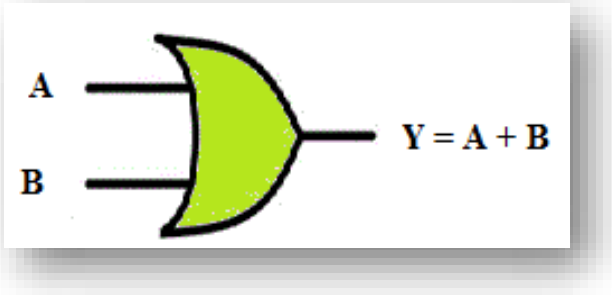
Theory :-

Introduction :-

AND, OR and NOT gates are basic logic gates and are fundamental building blocks from which all other logic circuits and digital systems can be constructed. Basic logic gates such as AND and OR can be used to control the passage of an input signal through to the output. This controlling action is why these circuits are referred to as *gates*.

OR Gate :-

Symbol :



Truth Table :-

A	B	Y = A + B
0	0	0
0	1	1
1	0	1
1	1	1

Application : Industrial Control System

All applications in real world where the occurrence of *any one or more than one* event is needed to be detected, OR gates can be used. For instance, In a chemical process or an industrial plant, if one or more parameters exceeds the safe limit, it is desirable that an alarm should be activated along with protective measure such as turning off the valve or shutting down the plant/process. This is possible by using the OR operation along with some additional circuitry.

Let us consider the example of an industrial control system. In an industrial process, it may be desired that when the temperature and pressure exceeds the safe limits, an alarm needs to be activated and the process needs to be stopped.

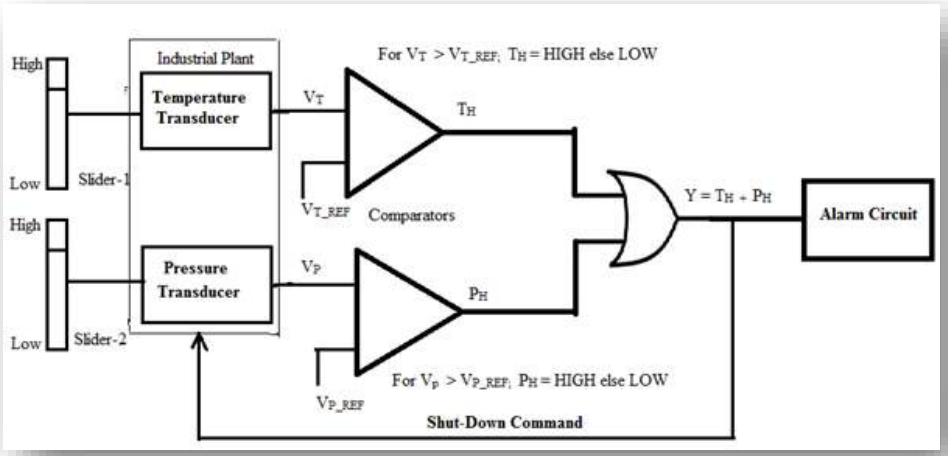


Fig.1. Use of OR gate in Industrial Control System

Concept :-

To detect whether the process temperature or pressure exceeds the safe limits and indicate the same via alarm and generate a process shutdown signal for the system.

The temperature and pressure transducers produce output voltages VT and VP proportional to the process temperature and pressure respectively. These voltages are compared with pre-set values of temperature reference VT_REF and pressure reference VP_REF in a voltage comparator circuit. Let TH and PH represent the comparator outputs.

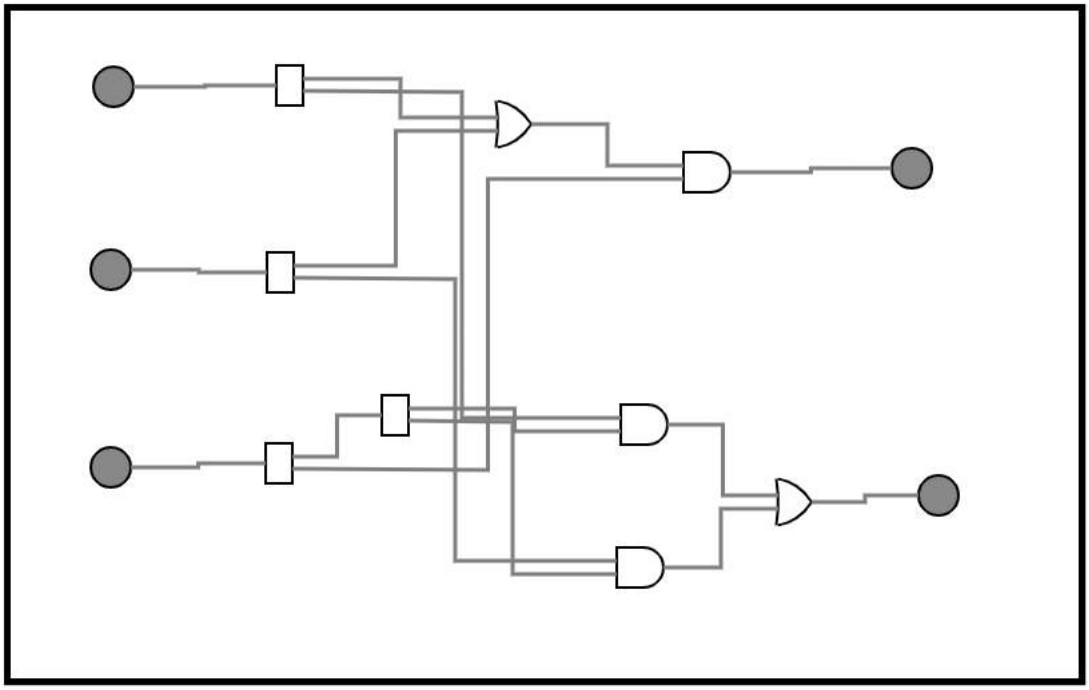
Whenever VT > VT_REF or VP > VP_REF then the upper and the lower comparators produce a HIGH output respectively at points TH and PH respectively. When either of the parameters exceeds the safe limit, the alarm is activated and the process shut down command is generated.

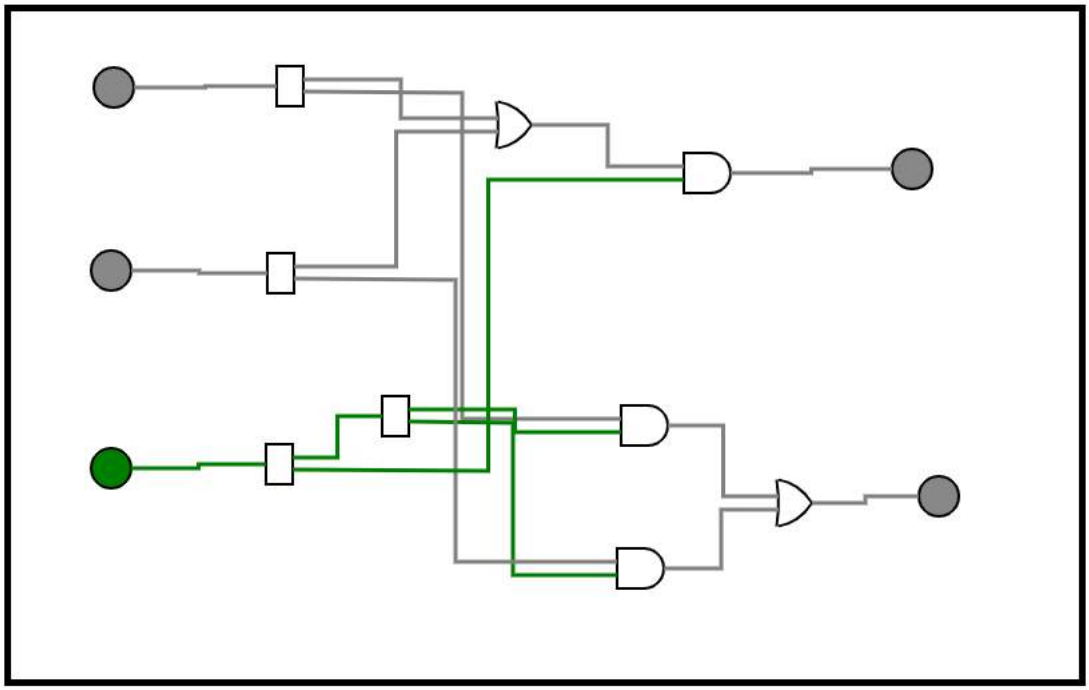
Observations :-

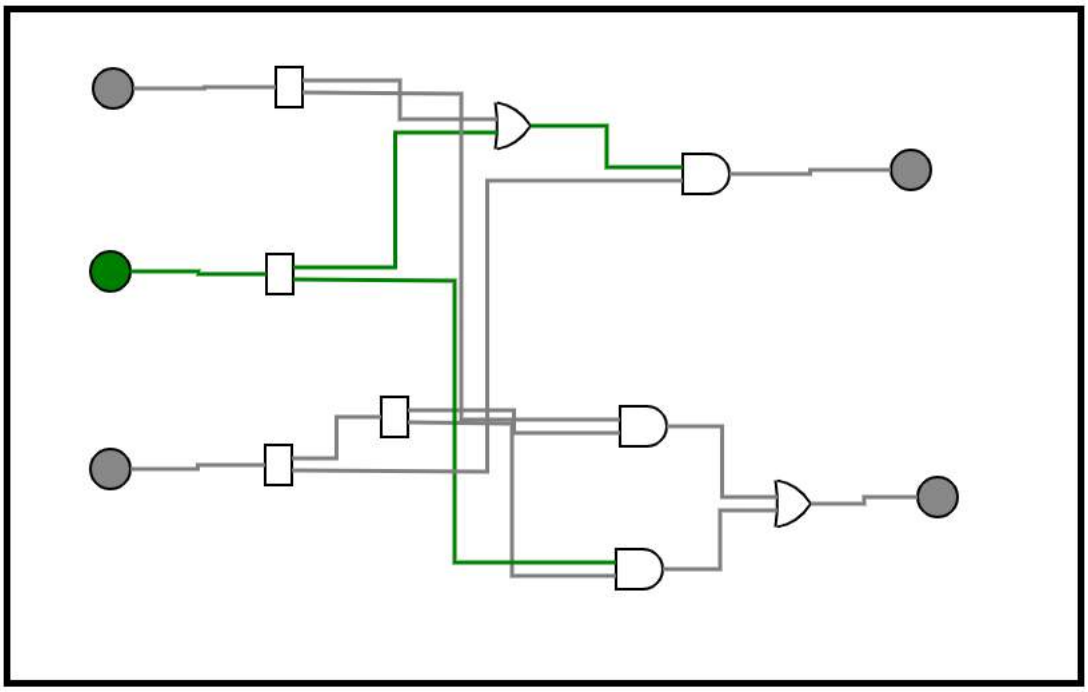
Truth Table :-

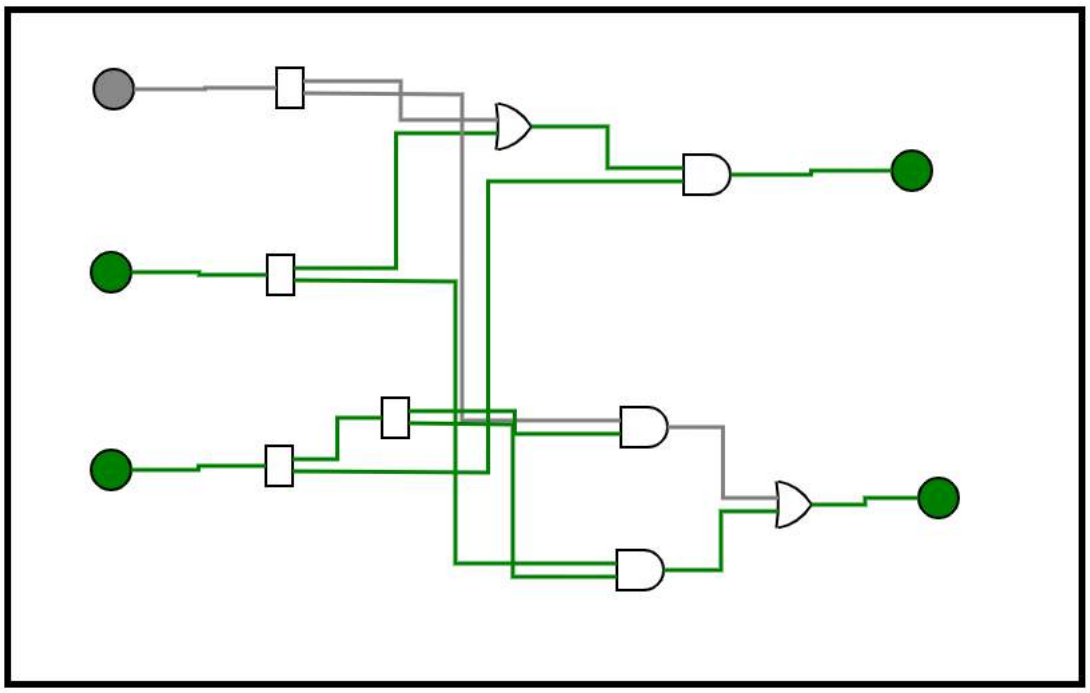
A	B	C	(A + B).C	A.C + B.C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

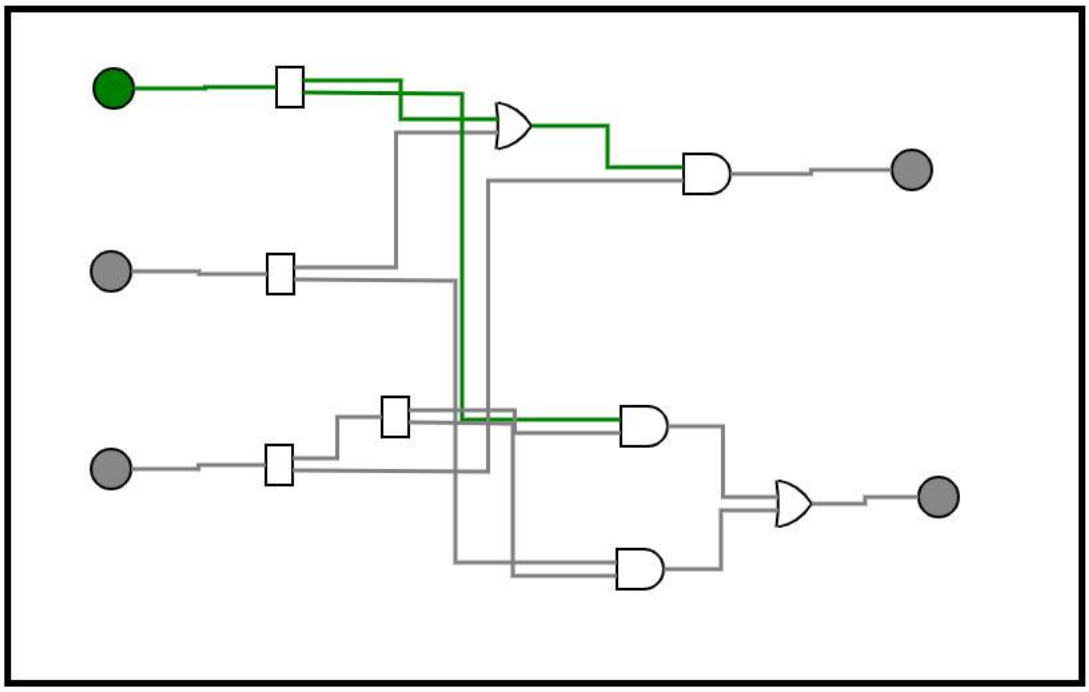
Circuit Diagram :-

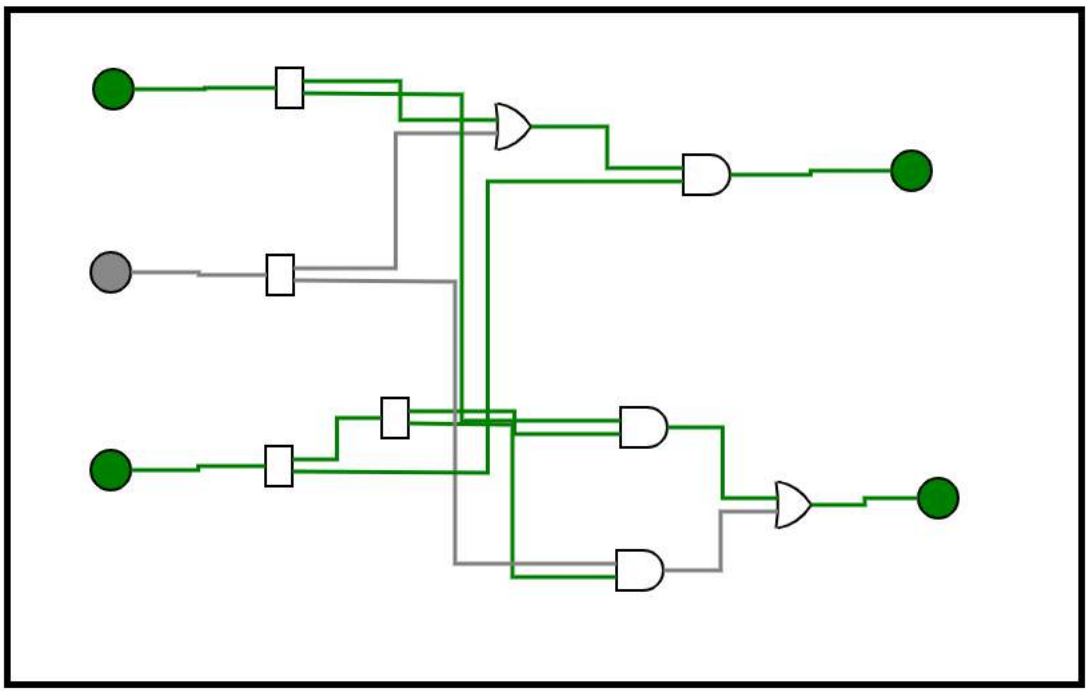


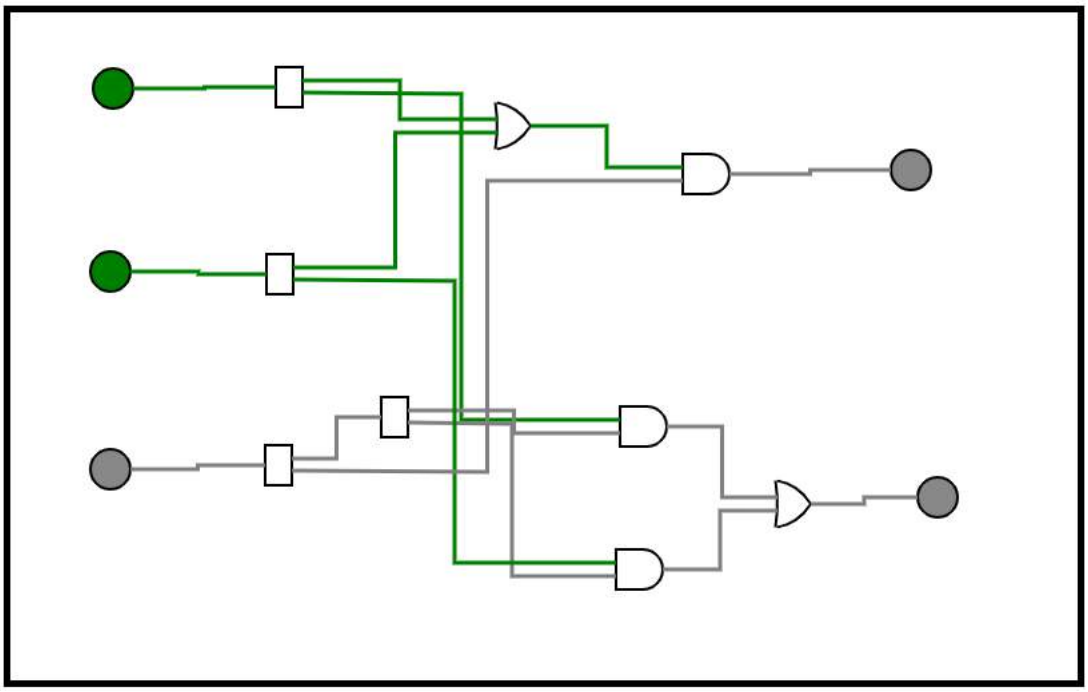


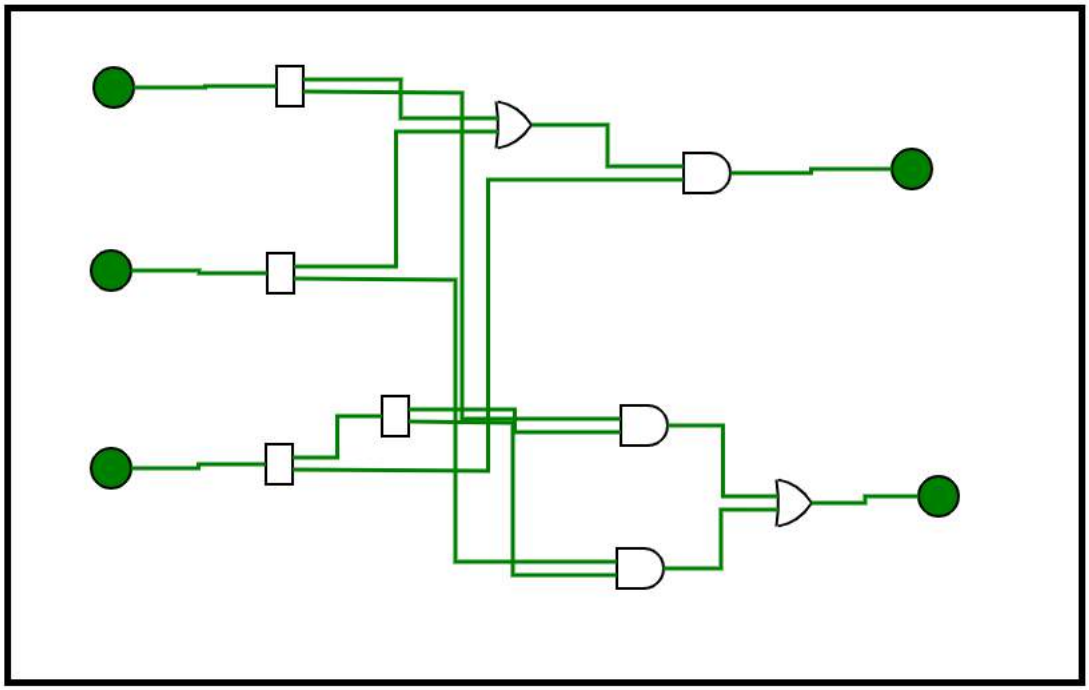












Conclusion :- It verifies the Boolean expression (A + B).C = A.C + B.C by its truth table and by constructing OR Gates.

Experiment - 8

Aim :- To verify the truth table of half adder and full adder by using XOR and NAND gates respectively and analyse the working of half adder and full adder circuit with the help of LEDs in simulator 1 and verify the truth table only of half adder and full adder in simulator 2.

Apparatus Required :- Perform the experiment using this link :

https://de-iitr.vlabs.ac.in/exp/half-full-adder/

Device Used :- XOR and NAND gates

Theory :-

Introduction :

Adders are digital circuits that carry out addition of numbers. Adders are a key component of arithmetic logic unit. Adders can be constructed for most of the numerical representations like Binary Coded Decimal (BCD), Excess – 3, Gray code, Binary etc. out of these, binary addition is the most frequently performed task by most common adders. Apart from addition, adders are also used in certain digital applications like table index calculation, address decoding etc.

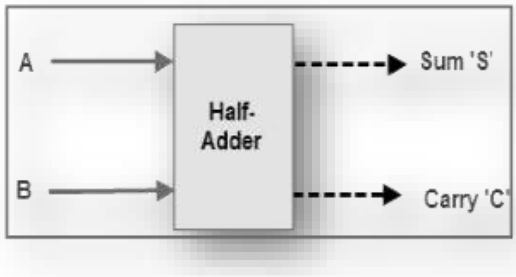
Binary addition is similar to that of decimal addition. Some basic binary additions are shown below.

0	0	1	1
+0	+1	+0	+1
0	1	1	(carry) 1 0

Fig.1 Schematic representation of half adder

1. Half Adder :-

Half adder is a combinational circuit that performs simple addition of two binary numbers. If we assume A and B as the two bits whose addition is to be performed, the block diagram and a truth table for half adder with A, B as inputs and Sum, Carry as outputs can be tabulated as follows.



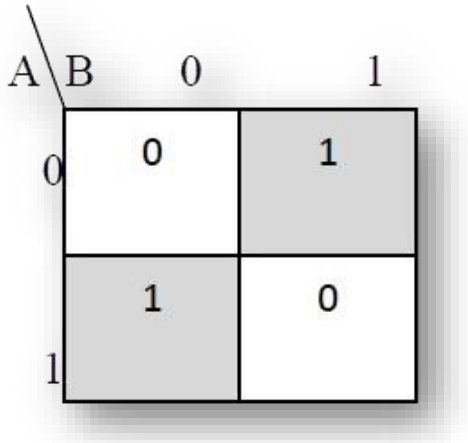
Truth Table			
Input		Output	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

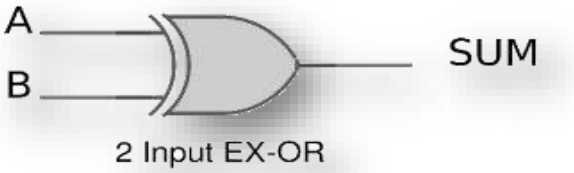
Fig.2 Block diagram and truth table of half adder

The sum output of the binary addition carried out above is similar to that of an Ex-OR operation while the carry output is similar to that of an AND operation. The same can be verified with help of Karnaugh Map.

The truth table and K Map simplification and logic diagram for sum output is shown below.

Truth Table		
A	B	SUM
0	0	0
0	1	1
1	0	1
1	1	0





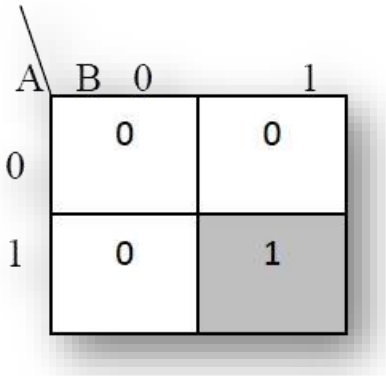
2 Input EX-OR

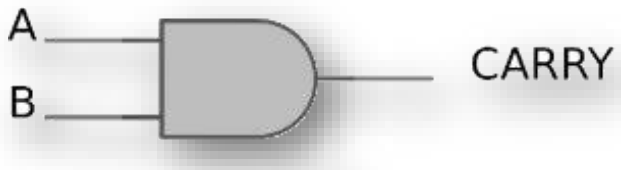
Fig.3 Truth table, K Map simplification and Logic diagram for sum output of half adder

$$Sum = A B' + A' B$$

The truth table and K Map simplification and logic diagram for carry is shown below.

Truth Table		
A	B	Carry
0	0	0
0	1	0
1	0	0
1	1	1





2 Input AND

Fig.4 Truth table, K Map simplification and Logic diagram for sum output of adder

Carry = AB

If A and B are binary inputs to the half adder, then the logic function to calculate sum S is Ex – OR of A and B and logic function to calculate carry C is AND of A and B. Combining these two, the logical circuit to implement the combinational circuit of half adder is shown below.

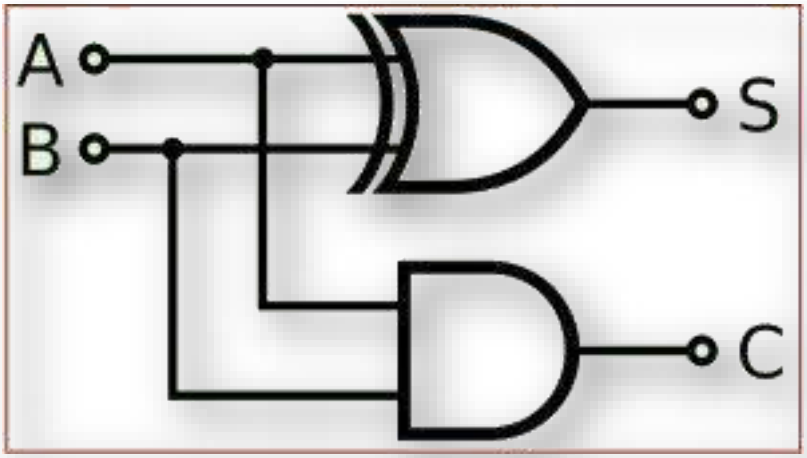


Fig.5 Half Adder Logic Diagram

As we know that NAND and NOR are called universal gates as any logic system can be implemented using these two, the half adder circuit can also be implemented using them. We know that a half adder circuit has one Ex – OR gate and one AND gate.

1.1 Half Adder using NAND gates :-

Five NAND gates are required in order to design a half adder. The circuit to realize half adder using NAND gates is shown below.

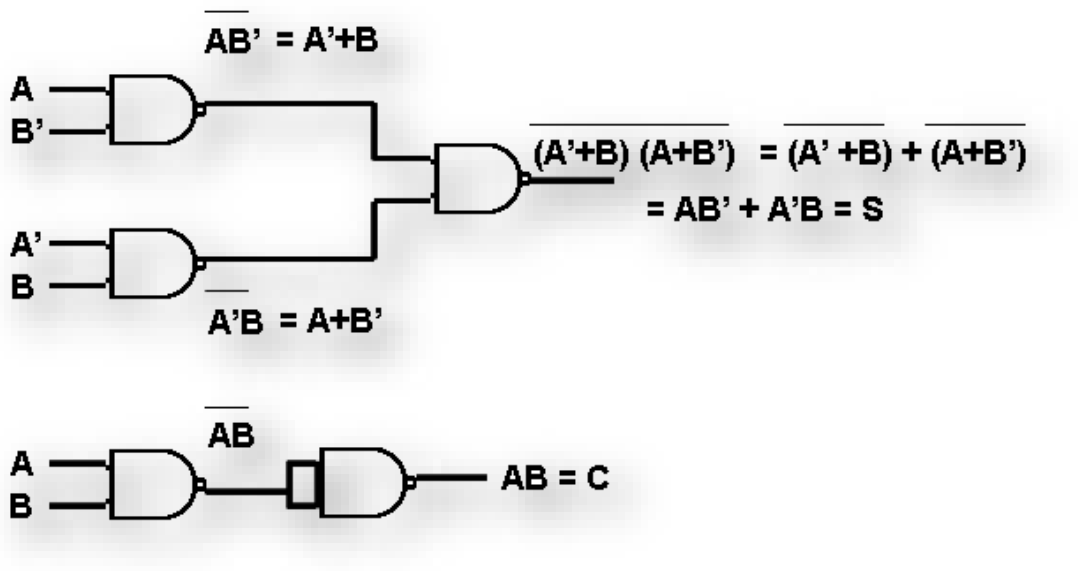


Fig.6 Realization of half adder using NAND gates

1.2 Half Adder using NOR gates

Five NOR gates are required in order to design a half adder. The circuit to realize half adder using NOR gates is shown below.

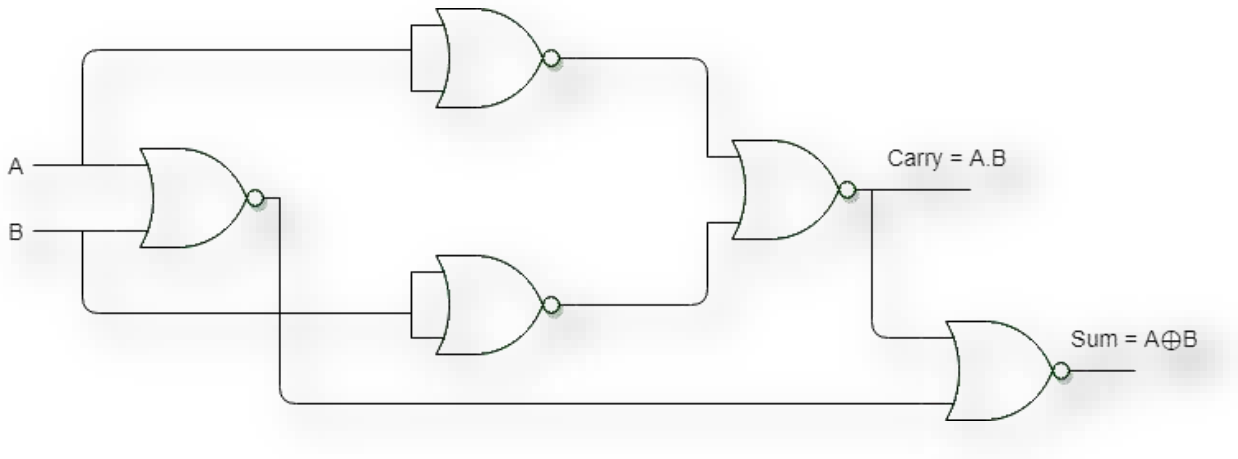
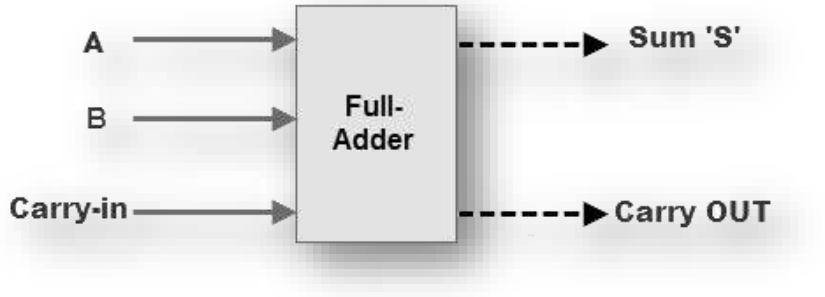


Fig.7 Realization of half adder using NOR Gates

2. Full Adder :-

Full adder is a digital circuit used to calculate the sum of three binary bits. Full adders are complex and difficult to implement when compared to half adders. Two of the three bits are same as before which are A, the augend bit and B, the addend bit. The additional third bit is carry bit from the previous stage and is called 'Carry' – in generally represented by CIN. It calculates the sum of three bits along with the carry. The output carry is called Carry – out and is represented by Carry OUT.
The block diagram of a full adder with A, B and CIN as inputs and S, Carry OUT as outputs is shown below.



Input			Output	
A	B	Cin	Sum	Carry
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

Fig.8 Full Adder Block Diagram and Truth Table

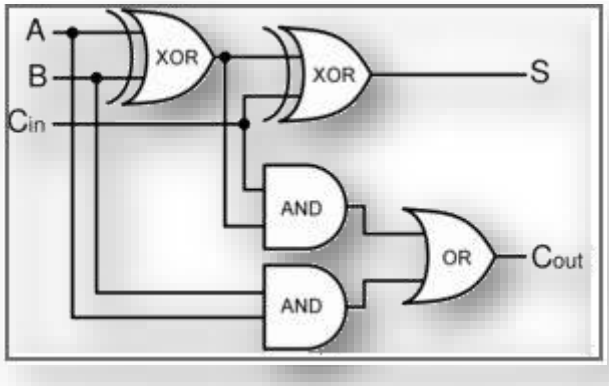


Fig.9 Full Adder Logic Diagram

Based on the truth table, the Boolean functions for Sum (S) and Carry – out (COUT) can be derived using K – Map.

A \ BC_{IN}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Fig.10 The K-Map simplified equation for sum is

$$S = A'B'C_{in} + A'BC_{in}' + ABC_{in}$$

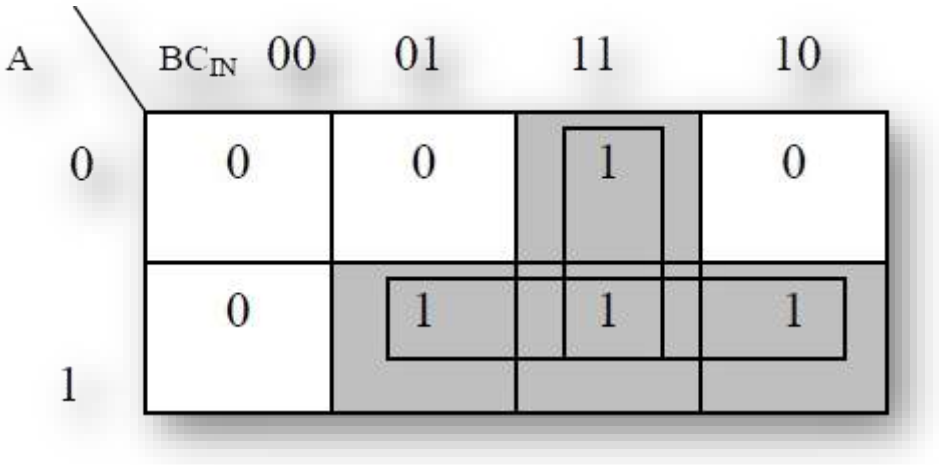


Fig.11 The K-Map simplified equation for COUT is

$$COUT = AB + AC_{IN} + BC_{IN}$$

In order to implement a combinational circuit for full adder, it is clear from the equations derived above, that we need four 3-input AND gates and one 4-input OR gates for Sum and three 2-input AND gates and one 3-input OR gate for Carry – out.

2.1 Full Adder using NAND gates :-

As mentioned earlier, a NAND gate is one of the universal gates and can be used to implement any logic design. The circuit of full adder using only NAND gates is shown below.

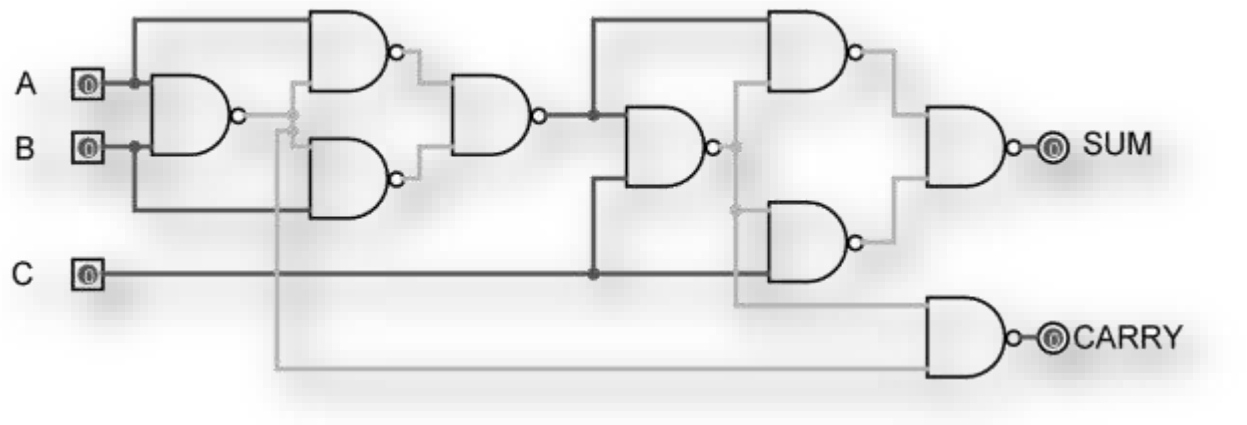


Fig.12 Full Adder using NAND gates

2.2 Full Adder using NOR gates :-

As mentioned earlier, a NOR gate is one of the universal gates and can be used to implement any logic design. The circuit of full adder using only NOR gates is shown below.

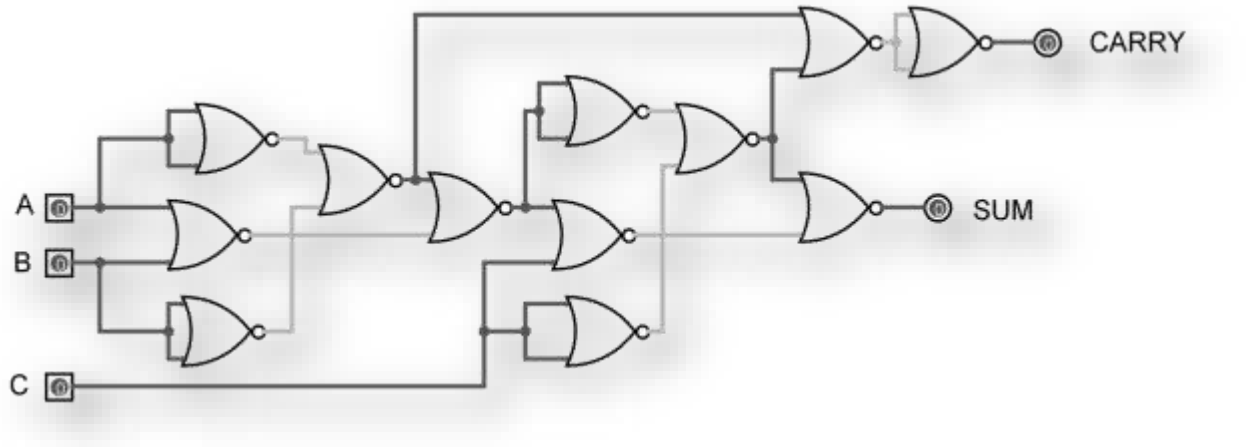
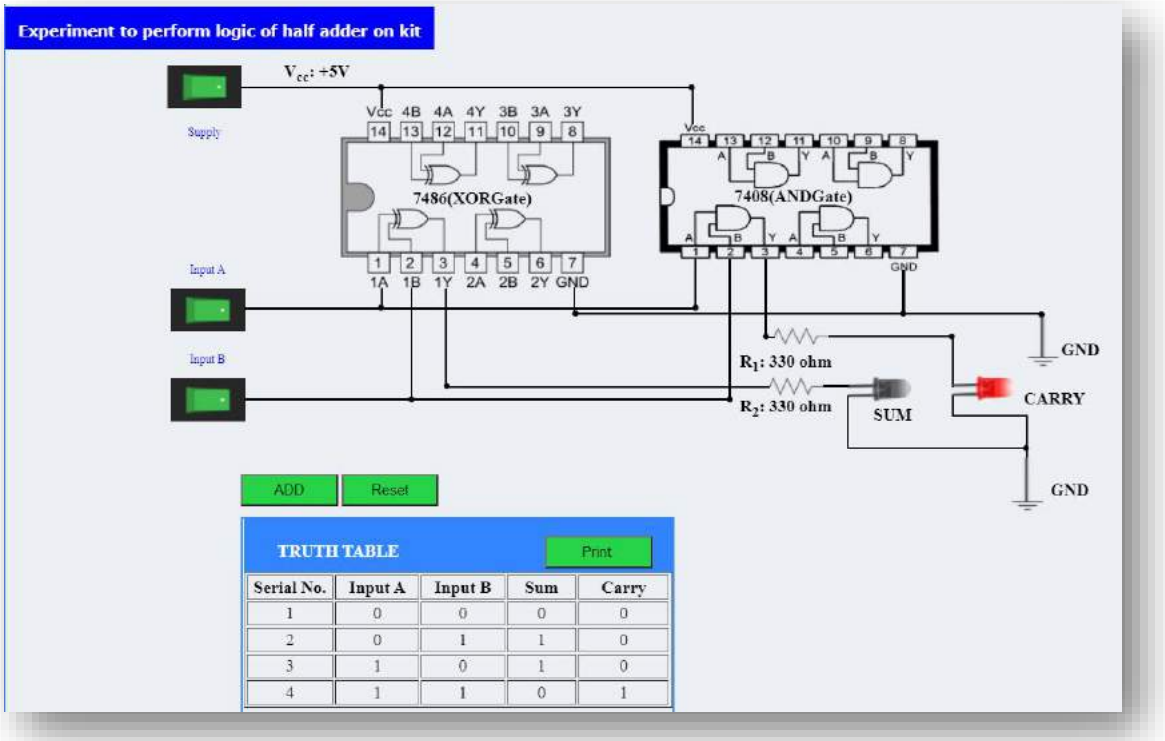
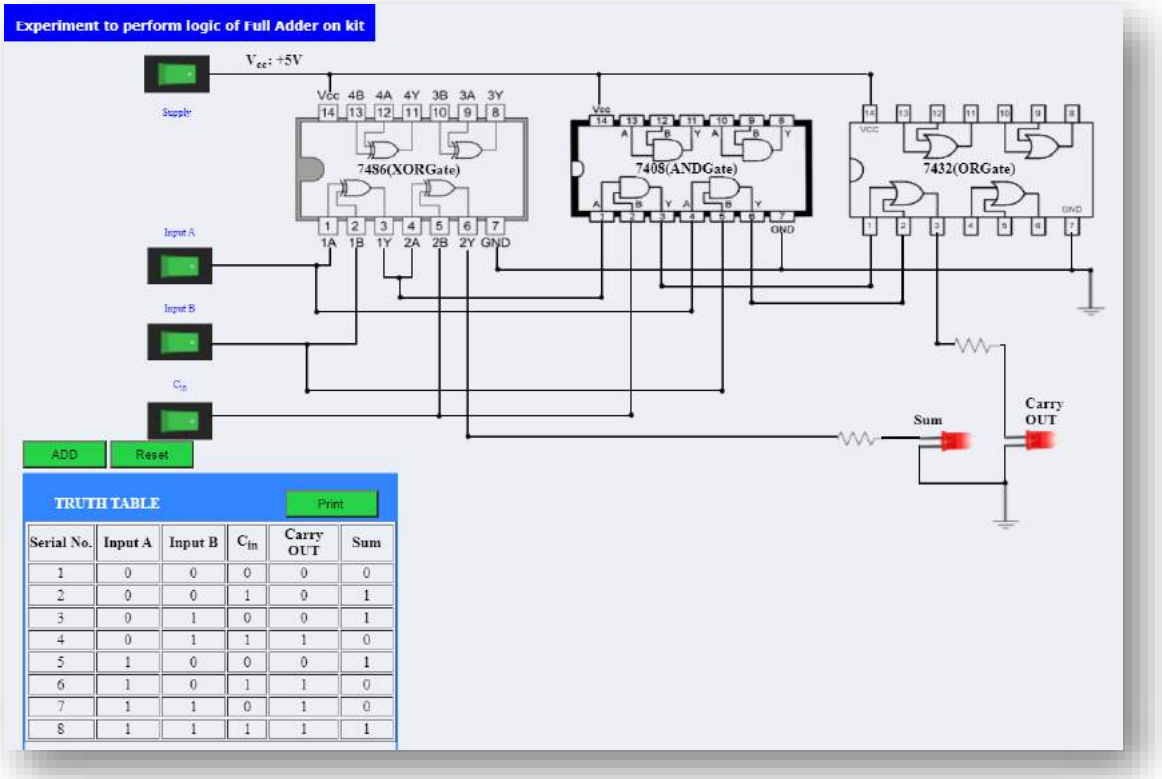


Fig.13 Full Adder using NOR gates

Observations :-





Serial No.	Input A	Input B	C_{in}	Carry OUT	Sum
1	0	0	0	0	0
2	0	0	1	0	1
3	0	1	0	0	1
4	0	1	1	1	0
5	1	0	0	0	1
6	1	0	1	1	0
7	1	1	0	1	0
8	1	1	1	1	1

Conclusion :- It verifies the truth table of half adder and full adder by using XOR and NAND gates respectively and analysation of the working of half adder and full adder circuit.

Experiment - 9

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.

Apparatus Required :- Perform the experiment using the link :

http://vlabs.iitb.ac.in/vlabs-dev/labs/digital-electronics/experiments/realization-of-logic-functions-iitr/

Device Used :- NAND and NOR universal gates

Theory :-

Introduction :-

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 :-

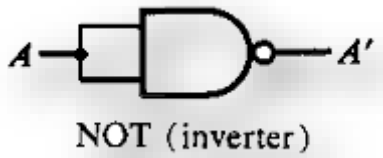
NAND gate is actually a combination of two logic gates i.e. AND gate followed by NOT gate. So its output is complement of the output of an AND gate. This gate can have minimum two inputs, output is always one. By using only NAND gates, we can realize all logic functions: AND, OR, NOT, X-OR, X-NOR, NOR.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

1.1 NAND gates as NOT gate :

A NOT produces complement of the input. It can have only one input, tie the inputs of a NAND gate together. Now it will work as a NOT gate. Its output is

$$Y = (A.A)'$$

$$Y = (A)'$$



NOT (inverter)

Fig1. NAND gates as NOT gate

Input	Output
A	A'
0	1
1	0

Fig2. Truth table of NOT

1.2 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.

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

$$Y = (A.B)$$

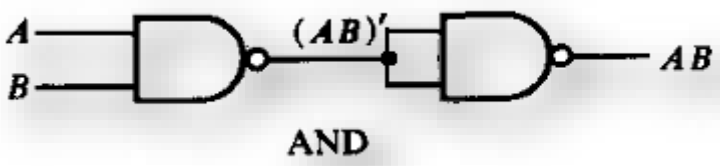


Fig3. NAND gates as AND gate

Input		Output
A	B	F = A.B
0	0	0
0	1	0
1	0	0
1	1	1

Fig4. Truth table of AND

1.3 NAND gates as OR gate :

From De Morgan's theorems:
(A.B)' = A' + B'
(A'.B')' = A'' + B'' = A + B
So, give the inverted inputs to a NAND gate, obtain OR operation at output.

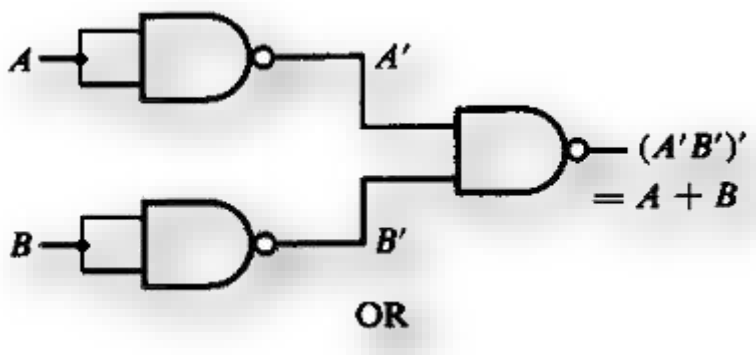


Fig5. NAND gates as OR gate

A	B	X = A+B
0	0	0
0	1	1
1	0	1
1	1	1

Fig6. Truth table of OR

1.4 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.

Gate No.	Inputs	Output
1	A, B	(AB)'
2	A, (AB)'	(A (AB)')'
3	(AB)', B	(B (AB)')'
4	(A (AB)')', (B (AB)')'	A'B + AB'

Now the ouput from gate no. 4 is the overall output of the configuration.

$$
\begin{aligned}
Y &= ((A\,(AB)')'\,(B\,(AB)')')' \\
&= (A(AB)'')'' + (B(AB)'')'' \\
&= (A(AB)'') + (B(AB)'') \\
&= (A(A' + B)') + (B(A' + B')) \\
&= (AA' + AB') + (BA' + BB') \\
&= (0 + AB' + BA' + 0) \\
&= AB' + BA' \\
\Rightarrow \quad Y &= AB' + A'B
\end{aligned}
$$

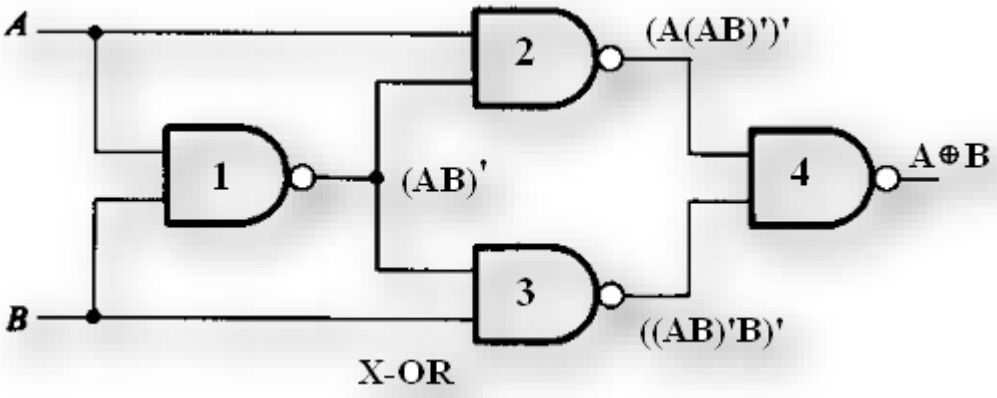


Fig7. NAND gates as Ex-OR gate

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

Fig8. Truth table of Ex-OR

1.5 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 output is that of an Ex-NOR gate.

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

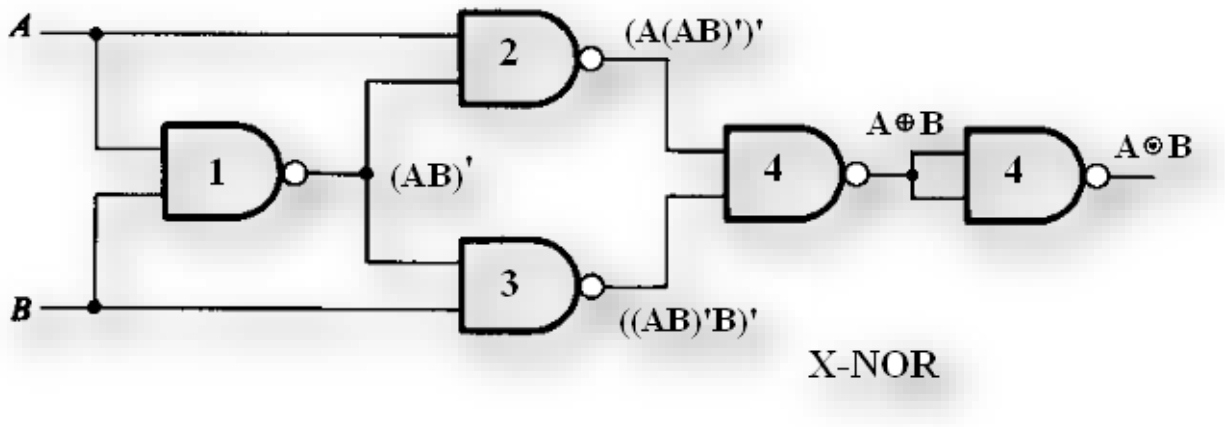


Fig9. NAND gates as Ex-NOR gate

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

Fig10. Truth table of Ex-NOR

1.6 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 use a 3-input NAND gate. Each of the three terms is, itself, a NAND expression. Finally, negated single terms can be generates with a 2-input NAND gate acting as an inverted. Figure 8 illustrates a circuit using NAND gates only.

$$F = ((C'.B.A)'(D'.C.A)'(C.B'.A)')'$$

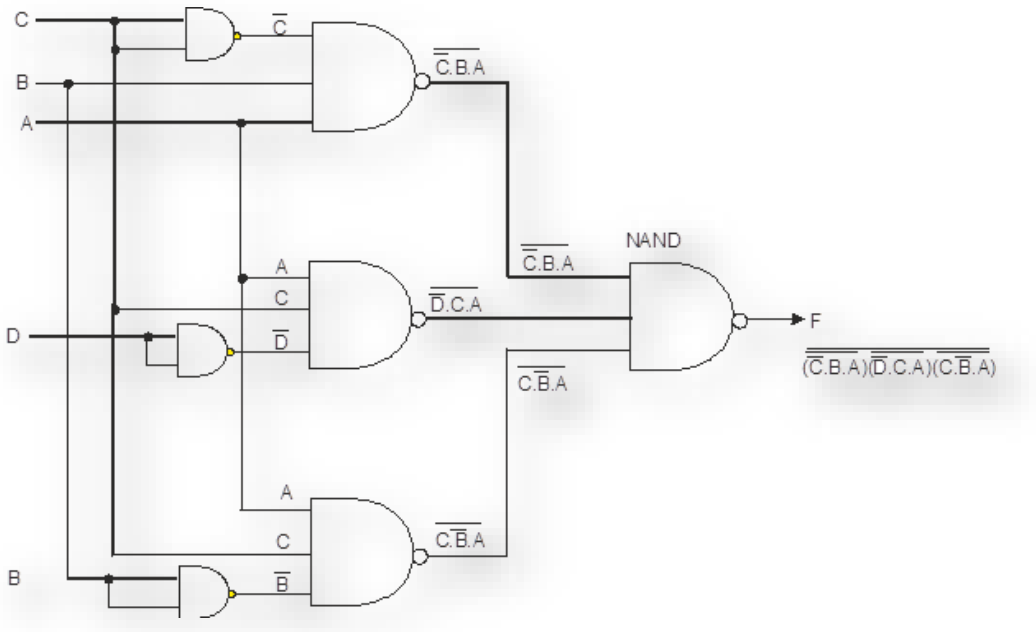


Fig11. Implementing the simplified function with NAND gates only

2. Nor gate as Universal gate :-

NOR gate is actually a combination of two logic gates: OR gate followed by NOT gate. So its output is complement of the 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 universal gate.

2.1 NOR gates as NOT gate :

A NOT produces complement of the input. It can have only one input, tie the inputs of a NOR gate together. Now it will work as a NOT gate. Its output is

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

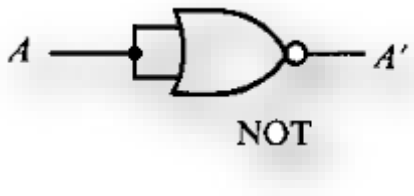


Fig12. NOR gates as NOT gate

Input	Output
A	A'
0	1
1	0

Fig13. Truth table of NOT

2.2 NOR gates as OR gate :

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

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

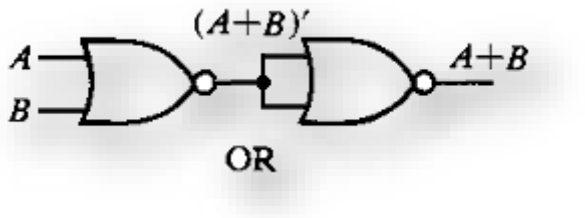


Fig14. NOR gates as OR gate

A	B	X = A+B
0	0	0
0	1	1
1	0	1
1	1	1

Fig15. Truth table of OR

2.3 NOR gates as AND gate :

From De Morgan's theorems:
(A+B)' = A'B'
(A'+B')' = A''B'' = AB

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

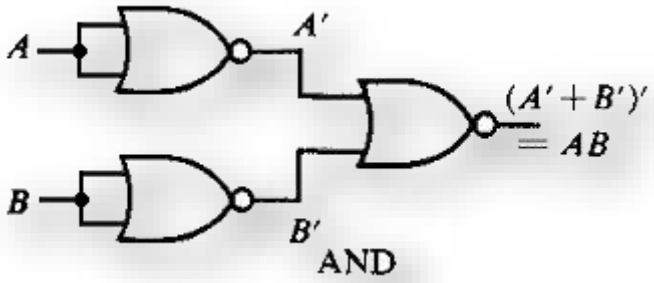


Fig16. NOR gates as AND gate

Input		Output
A	B	F = A.B
0	0	0
0	1	0
1	0	0
1	1	1

Fig17. Truth table of AND

2.4 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 side.

Gate No.	Inputs	Output
1	A, B	(A + B)'
2	A, (A + B)'	(A + (A+B)')'
3	(A + B)', B	(B + (A+B)')'
4	(A + (A + B)')', (B + (A+B)')'	AB + A'B'

Now the ouput from gate no. 4is the overall output of the configuration.

Y = ((A + (A+B)')' (B +(A+B)')')'
= (A+(A+B)'')'.(B+(A+B)'')''
= (A+(A+B)').(B+(A+B)')
= (A+A'B').(B+A'B')
= (A + A').(A + B').(B+A')(B+B')
= 1.(A+B').(B+A').1
= (A+B').(B+A')
= A.(B + A') +B'.(B+A')
= AB + AA' +B'B+B'A'
= AB + 0 + 0 + B'A'
= AB + B'A'

=> Y = AB + A'B'

Fig18. NOR gates as Ex-NOR gate

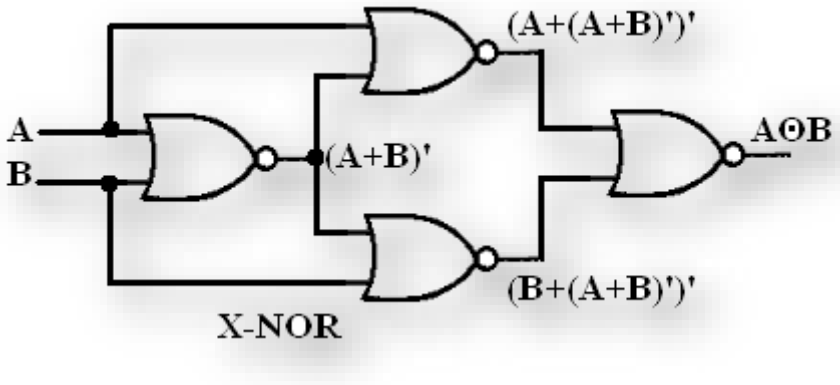


Fig19. Truth table of Ex-NOR

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

2.5 NOR gates as Ex-OR gate :

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

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

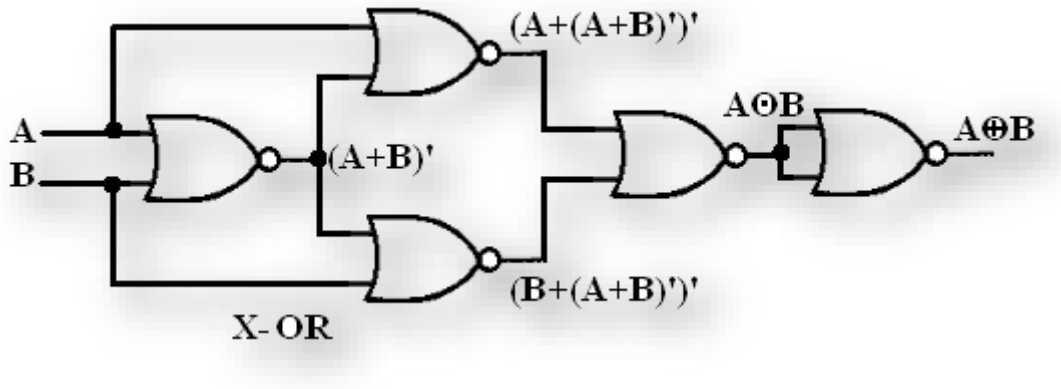


Fig20. NOR gates as Ex-OR gate

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

Fig21. Truth table of Ex-OR

2.6 Constructing a circuit with NOR gates only :

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

F=(((C.B'.A)+(D.C'.A)+(C.B'.A))')'

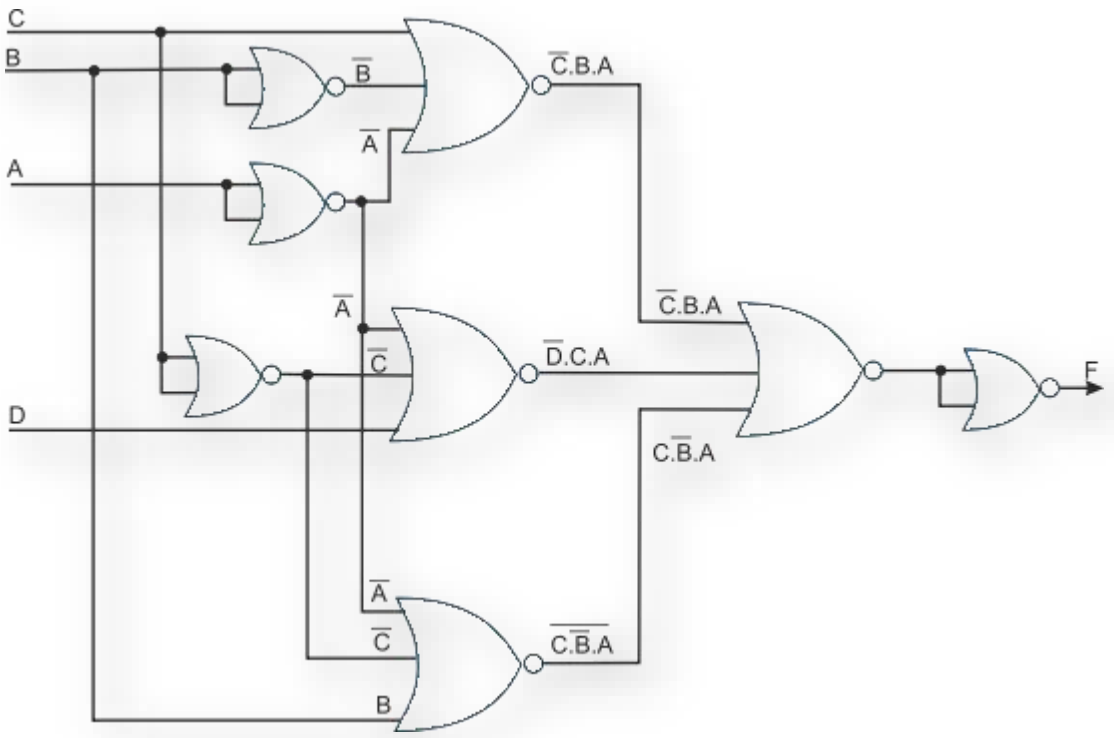
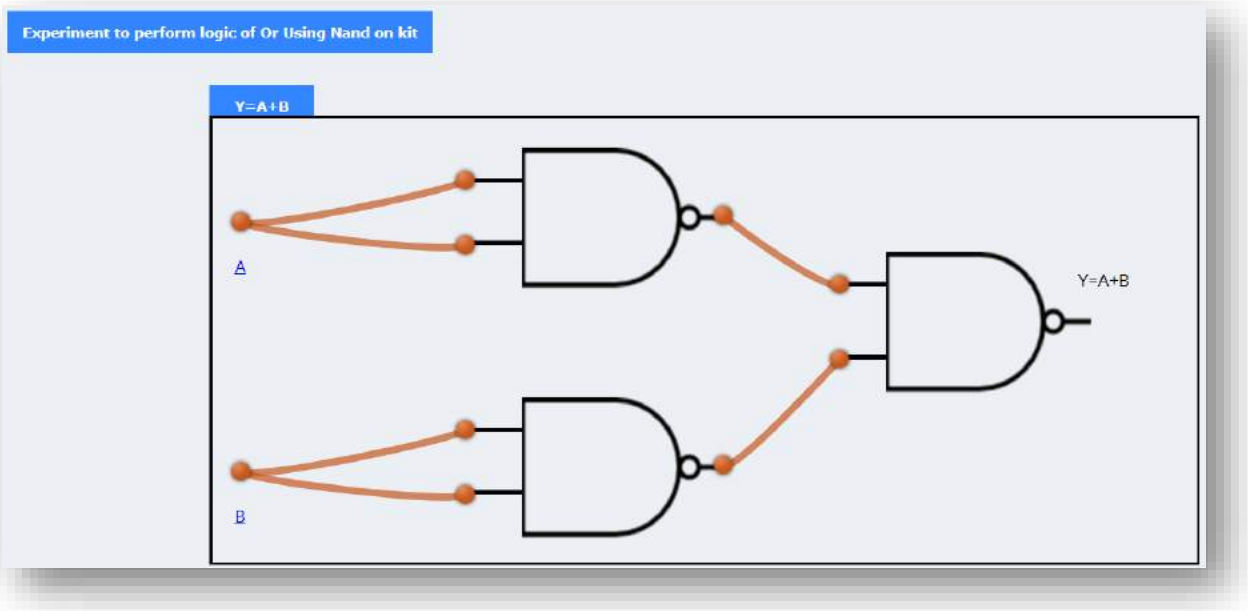
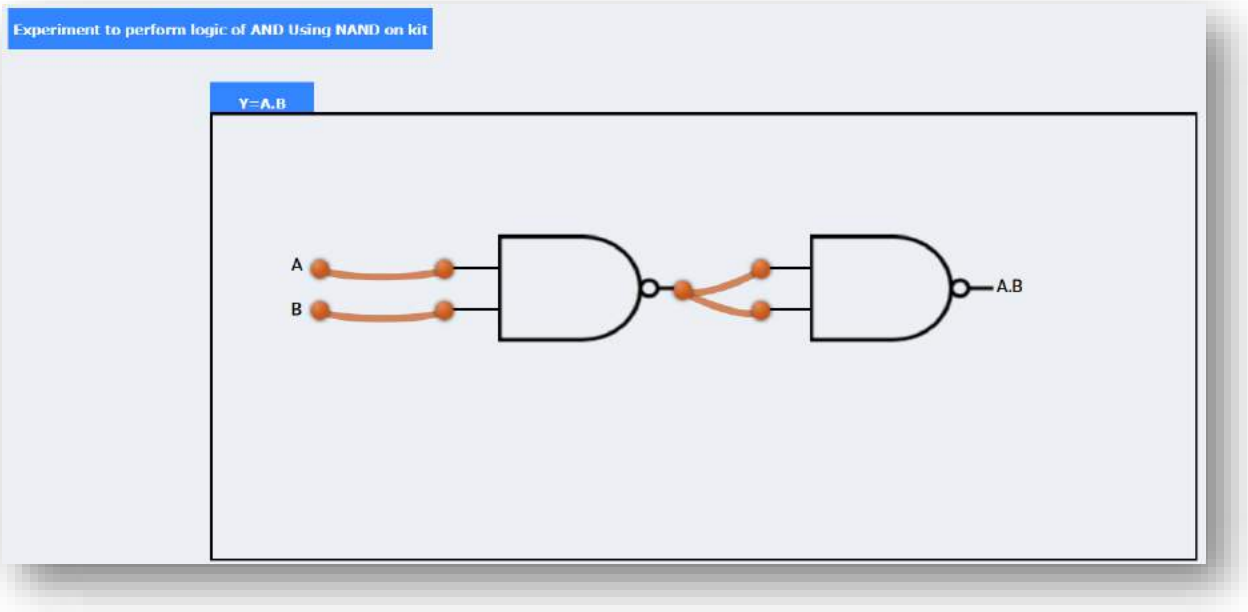


Fig22. Implementing the simplified function with NOR gates only

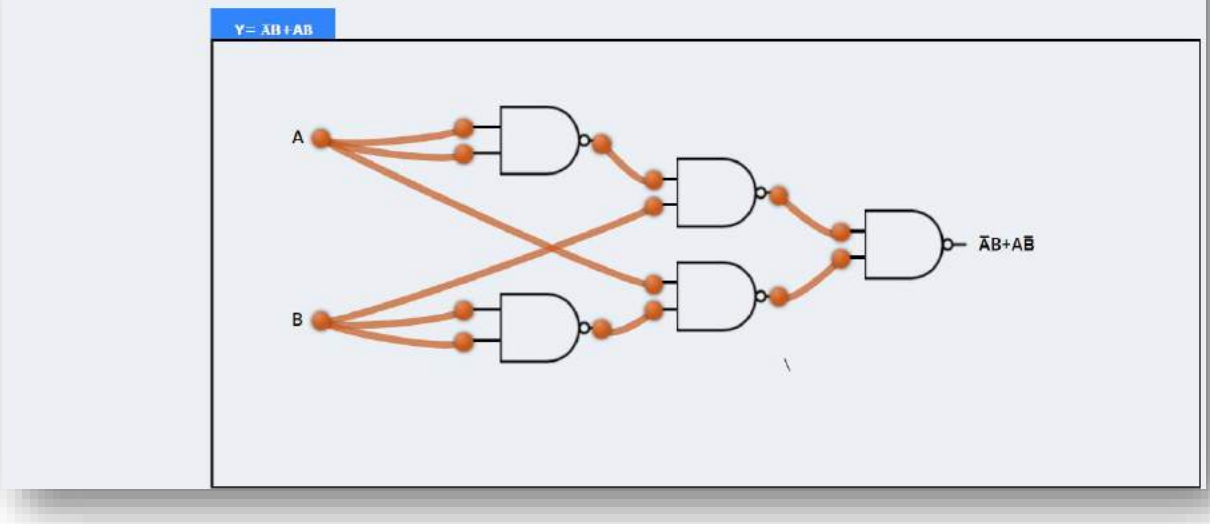
Observations :-

NAND Gate as Universal gate :

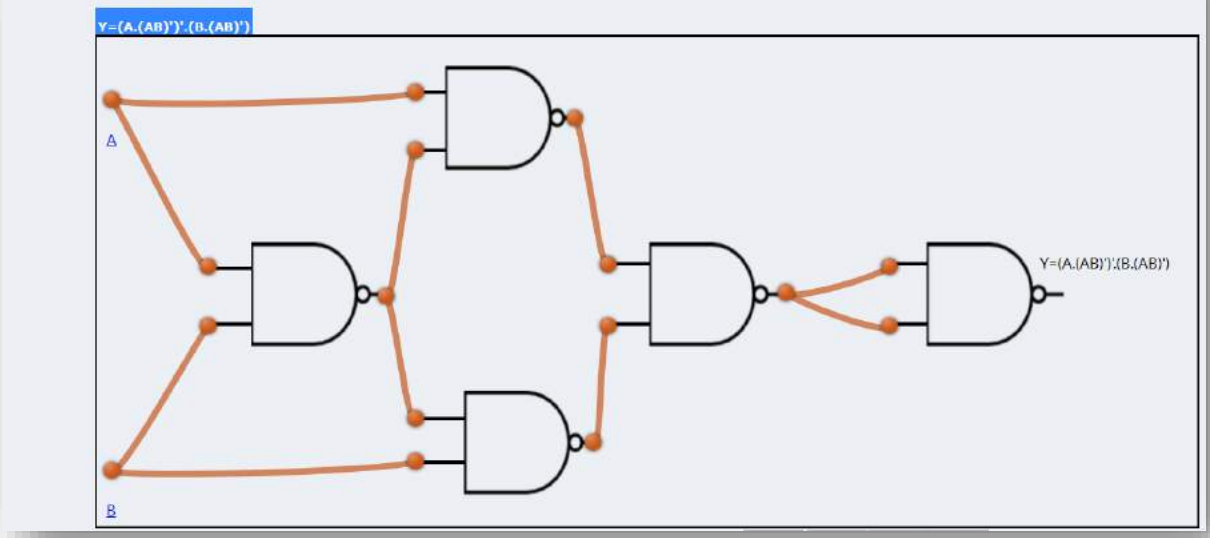


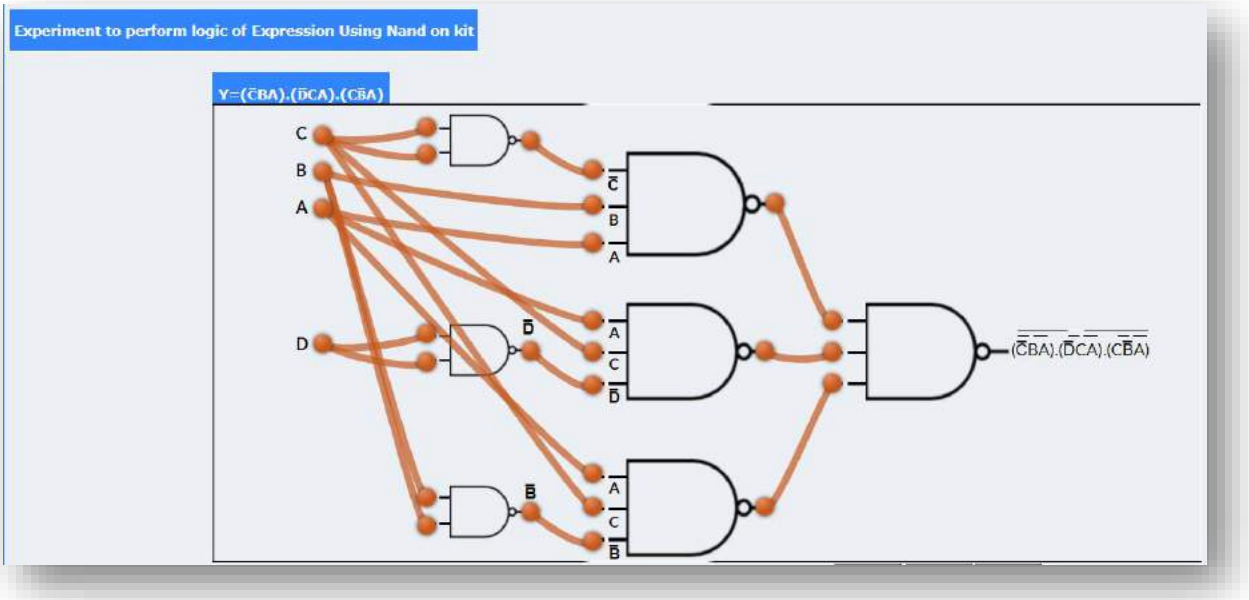


Experiment to perform logic of Ex-OR Using Nand on kit

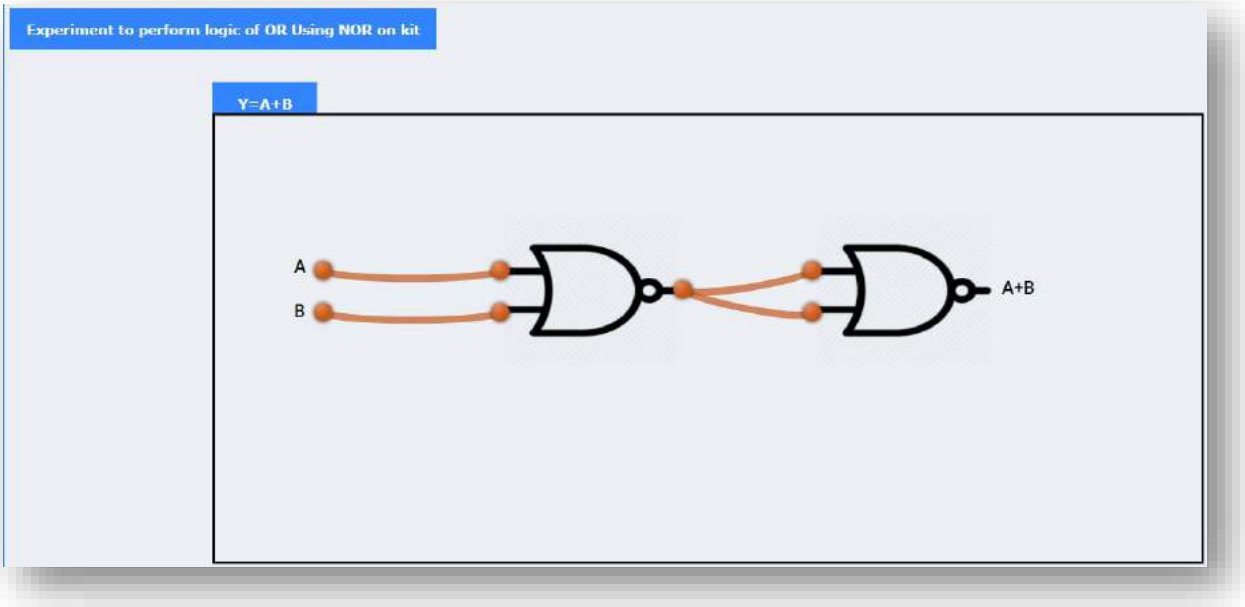


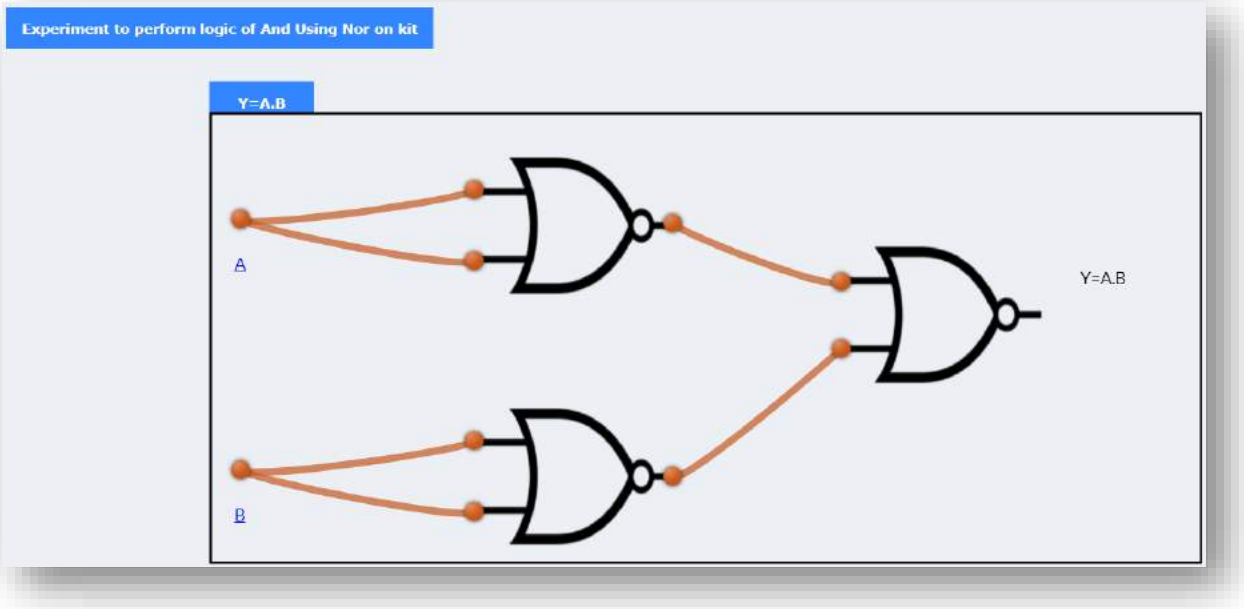
Experiment to perform logic of Ex-NOR Using Nand on kit



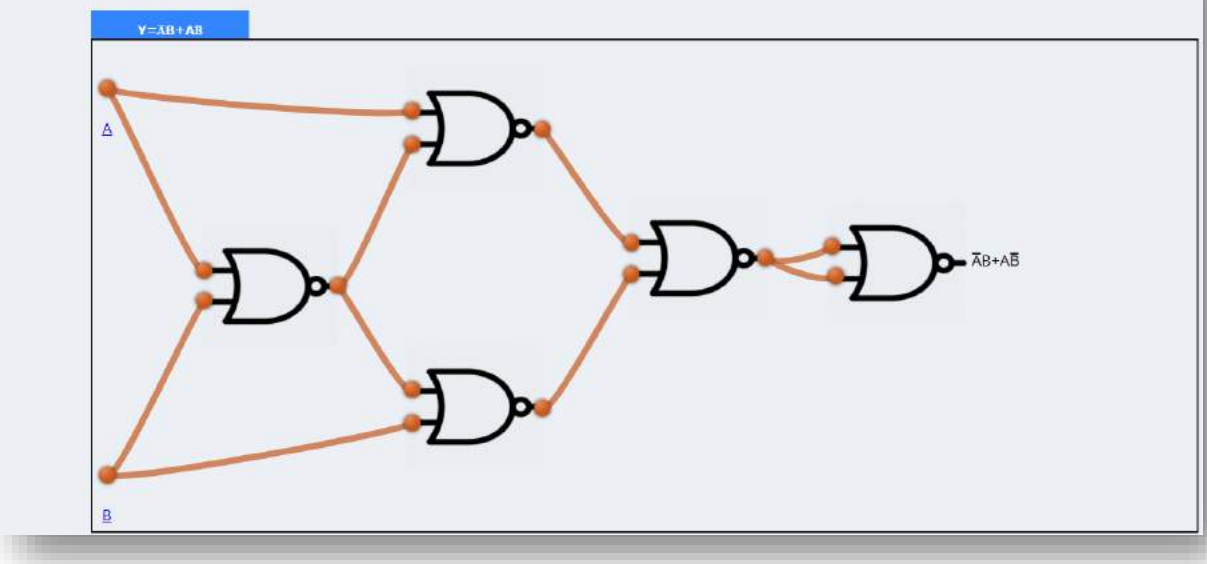

Experiment to perform logic of Expression Using Nand on kit
Y=(C̄BA).(D̄CA).(CB̄A)
C
B
A
D
C̄
B
A
D̄
A
C
D̄
B̄
A
C
B̄
(C̄BA).(D̄CA).(CB̄A)

NOR gate as Universal gate :-

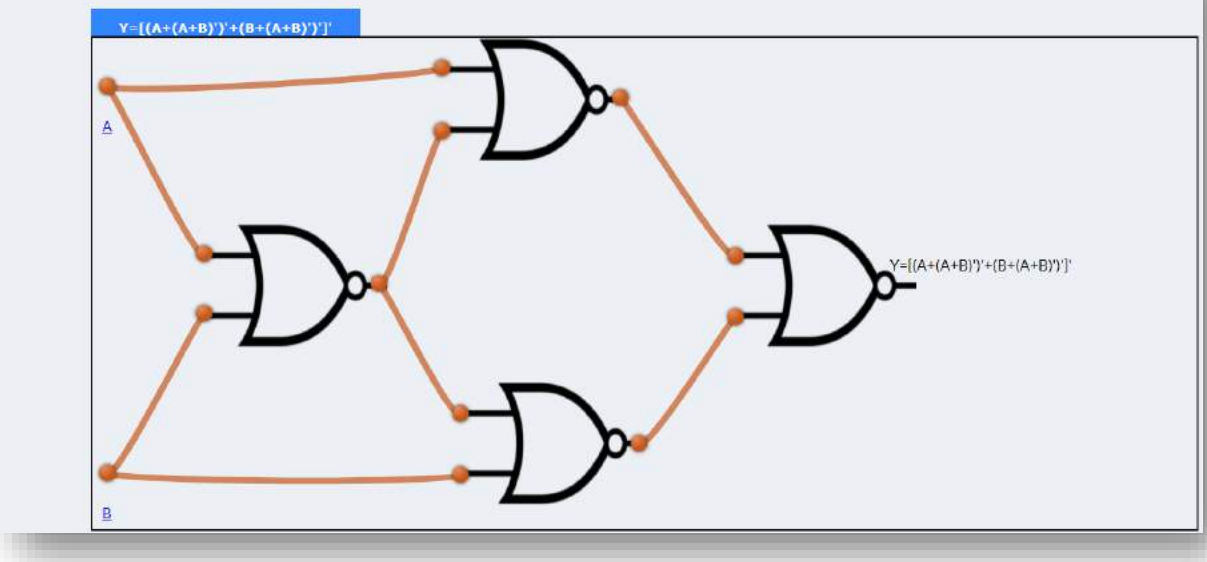


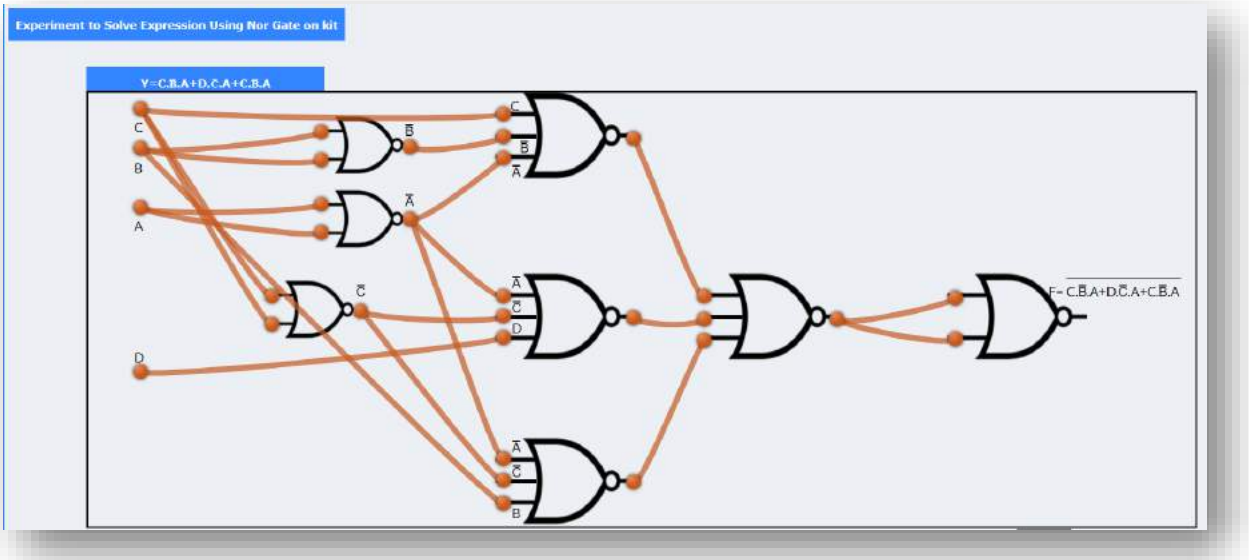


Experiment to perform logic of Ex-OR using NOR on kit



Experiment to perform logic of Ex-NOR Using Nor on kit





Conclusion :- It verifies the implementation of the logic functions i.e. AND , OR, NOT, Ex-OR, Ex-NOR and a logical expression.

Experiment - 10

Aim :- To verify the truth table and timing diagram of NOR gate latch using NOR gate IC and analyse the circuit of NOR gate latch with the help of LEDs display.

Apparatus Required :- Perform the experiment using this link :

http://vlabs.iitb.ac.in/vlabs-dev/labs/digital-electronics-12022020/labs/exp5/index.html

Device Used :- NOR Gate

Theory :-

Latches are basic storage elements that operate with signal levels (rather than signal transitions). Latches controlled by a clock transition are flip-flops. Latches are edge-sensitive devices. Latches are useful for the design of the asynchronous sequential circuit.

SR (Set-Reset) Latch – SR Latch is a circuit with:

(i) 2 cross-coupled NOR gates or 2 cross-coupled NAND gates.
(ii) 2 inputs S for SET and R for RESET.
(iii) 2 outputs Q, Q.

The SR Latch using NOR gate is shown below with its truth table :

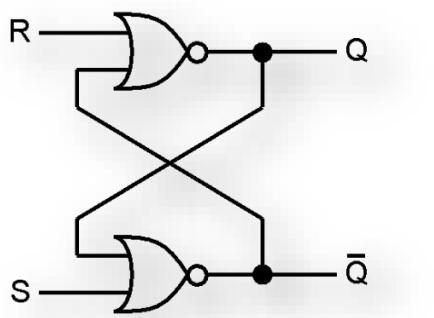


Fig 1. Logic Symbol of NOR gate latch

R	S	Q(n)	Q'(n)
0	0	Q(n-1)	Q'(n-1)
0	1	1	0
1	0	0	1
1	1	Invalid outputs	

Fig 2. Truth Table of NOR gate latch

While the R and S inputs are both low, feedback maintains the Q and Q outputs in a constant state, with Q the complement of Q. If S (Set) is pulsed high while R (Reset) is held low, then the Q output is forced high, and stays high when S returns to low; similarly, if R is pulsed high while S is held low, then the Q output is forced low, and stays low when R returns to low. The R = S = 1 combination is called a restricted combination or a forbidden state because, as both NOR gates then output zeros, it breaks the logical equation Q = Q. The combination is also inappropriate in circuits where both inputs may go low simultaneously (i.e. a transition from restricted to keep). The output would lock at either 1 or 0 depending on the propagation time relations between the gates (a race condition).

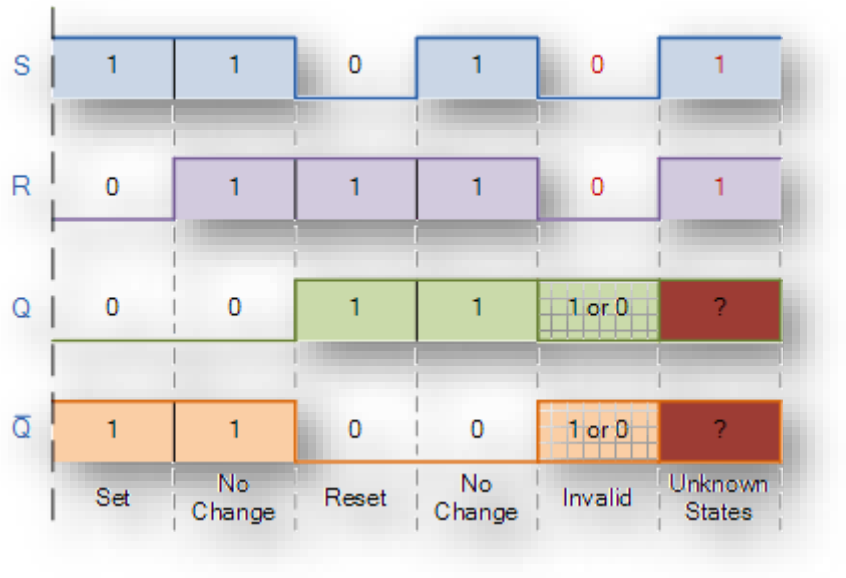
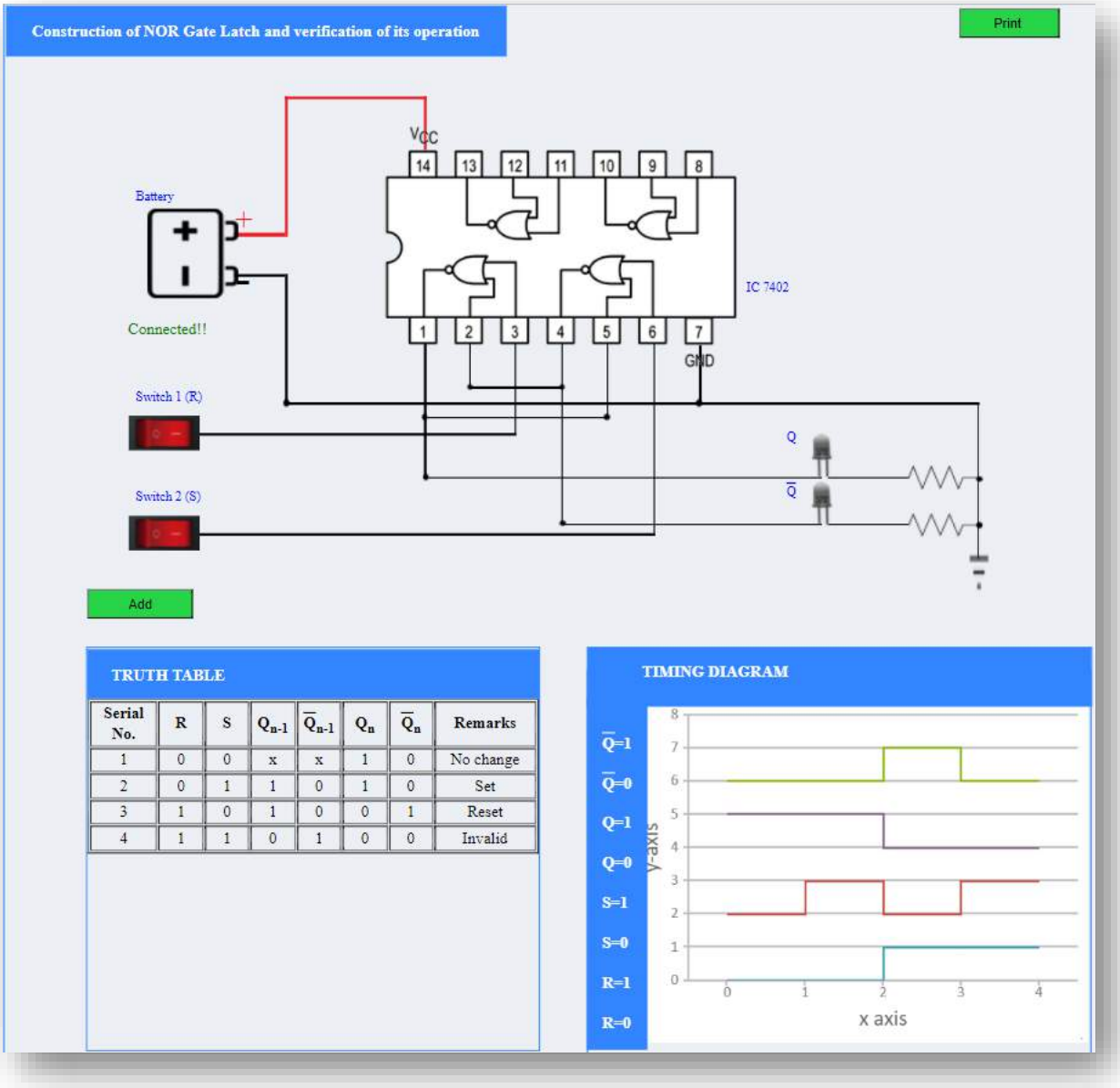


Fig 3. Timing Diagram of NOR gate latch

Observations :-



Serial No.	R	S	Q_{n-1}	$\overline{Q}_{n-1}$	Q_n	$\overline{Q}_n$	Remarks
1	0	0	x	x	1	0	No change
2	0	1	1	0	1	0	Set
3	1	0	1	0	0	1	Reset
4	1	1	0	1	0	0	Invalid

Conclusion :- It verifies the truth table and timing diagram of NOR gate latch and analyses the circuit of NOR gate latch.

Experiment - 11

Aim :- To verify the truth table and timing diagram of RS, JK, T and D flip-flops by using NAND & NOR gates ICs and analyse the circuit of RS, JK, T and D flip-flops with the help of LEDs display.

Apparatus Required :- Perform the experiment using this link :

http://vlabs.iitb.ac.in/vlabs-dev/labs/digital-electronics-12022020/labs/exp6/index.html

Device Used :- NAND and NOR gates

Theory :-

A flip flop is an electronic circuit with two stable states that can be used to store binary data. The stored data can be changed by applying varying inputs. Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems.

1) R-S flip flop
2) D flip flop
3) J-K flip flop
4) T flip flop

1. R - S flip flop :-

The basic NAND gate RS flip flop circuit is used to store the data and thus provides feedback from both of its outputs again back to its inputs. The RS flip flop actually has three inputs, SET, RESET and its current output Q relating to its current state as shown in figure below.

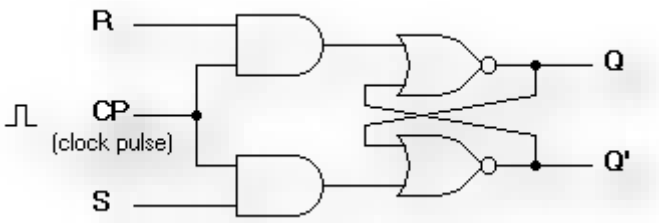


Fig 1. R-S flip flop circuit diagram

INPUTS			OUTPUT	STATE
CLK	S	R	Q	
X	0	0	No Change	Previous
↑	0	1	0	Reset
↑	1	0	1	Set
↑	1	1	-	Forbidden

Fig 2. Characteristics table of R-S flip flop

2. D flip flop :-

A D flip flop has a single data input. This type of flip flop is obtained from the SR flip flop by connecting the R input through an inverter, and the S input is connected directly to data input. The modified clocked SR flip-flop is known as D-flip-flop and is shown below. From the truth table of SR flip-flop we see that the output of the SR flip-flop is in unpredictable state when the inputs are same and high. In many practical applications, these input conditions are not required. These input conditions can be avoided by making them complement of each other.

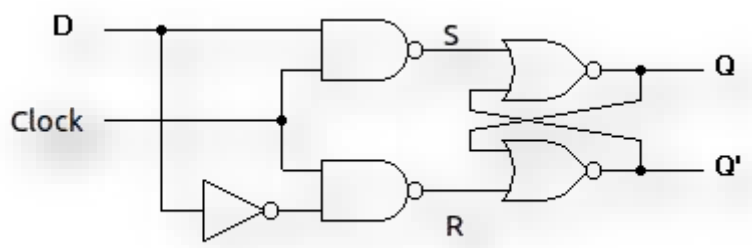


Fig 3. Circuit diagram of D flip flop

Input			Output	
D	reset	clock	Q	Q’
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

Fig 4. Characteristics table of D flip flop

3. J - K flip flop :-

In a RS flip-flop the input R=S=1 leads to an indeterminate output. The RS flip-flop circuit may be re-joined if both inputs are 1 than also the outputs are complement of each other as shown in characteristics table below.

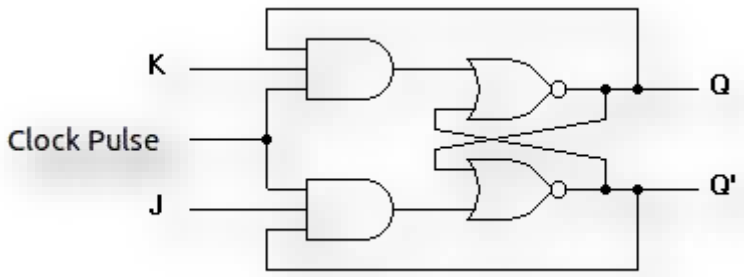


Fig 5. Circuit diagram of J-K flip flop

Trigger	Inputs		Output				Inference
			Present State		Next State		
CLK	J	K	Q	Q'	Q	Q'	
⊠	x	x	-		-		Latched
⊓	0	0	0	1	0	1	No Change
⊓			1	0	1	0	
⊓	0	1	0	1	0	1	Reset
⊓			1	0	0	1	
⊓	1	0	0	1	1	0	Set
⊓			1	0	1	0	
⊓	1	1	0	1	1	0	Toggles
⊓			1	0	0	1	

Fig 6. Characteristics table of J-K flip flop

4. T flip flop :-

T flip-flop is known as toggle flip-flop. The T flip-flop is modification of the J-K flip-flop. Both the JK inputs of the JK flip – flop are held at logic 1 and the clock signal continuous to change as shown in table below.

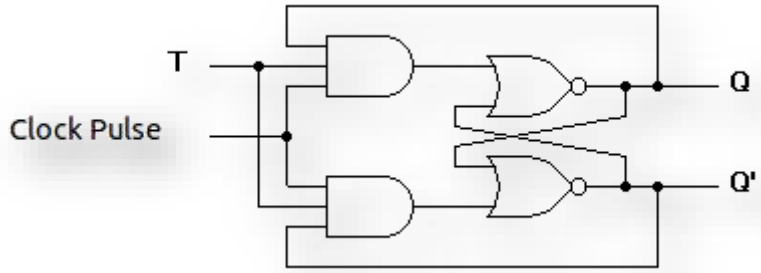


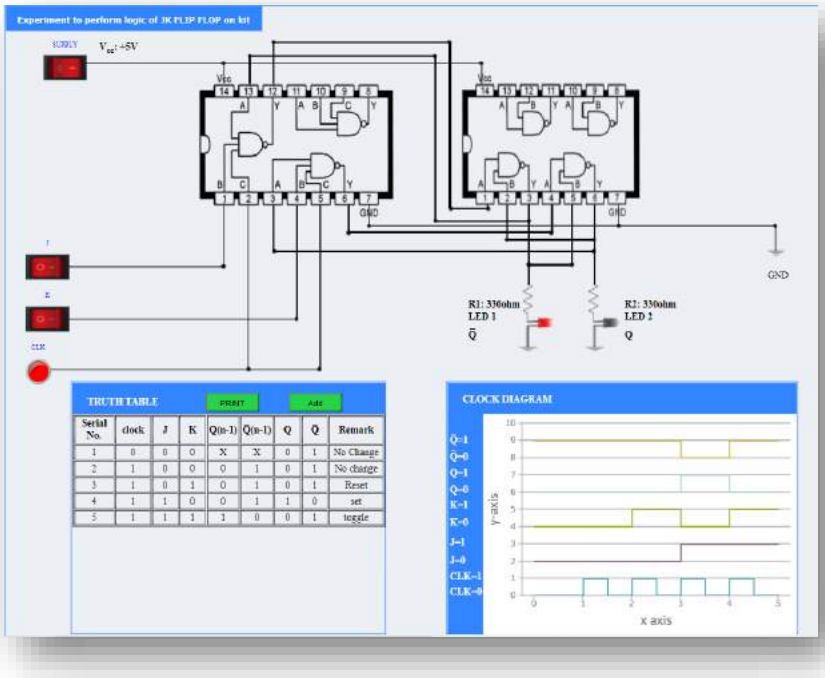
Fig 7. Circuit diagram of T flip flop

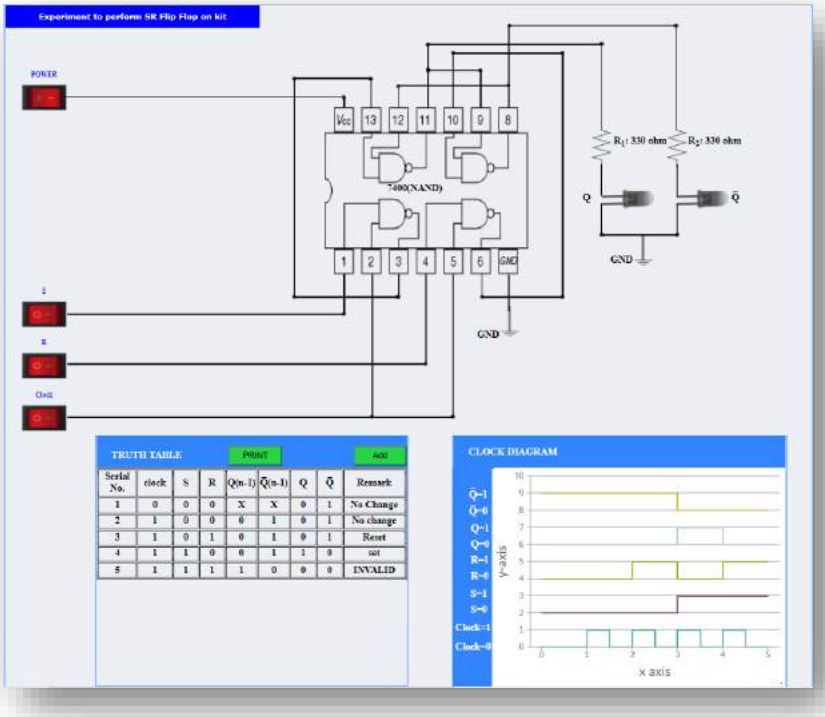
T flip-flop

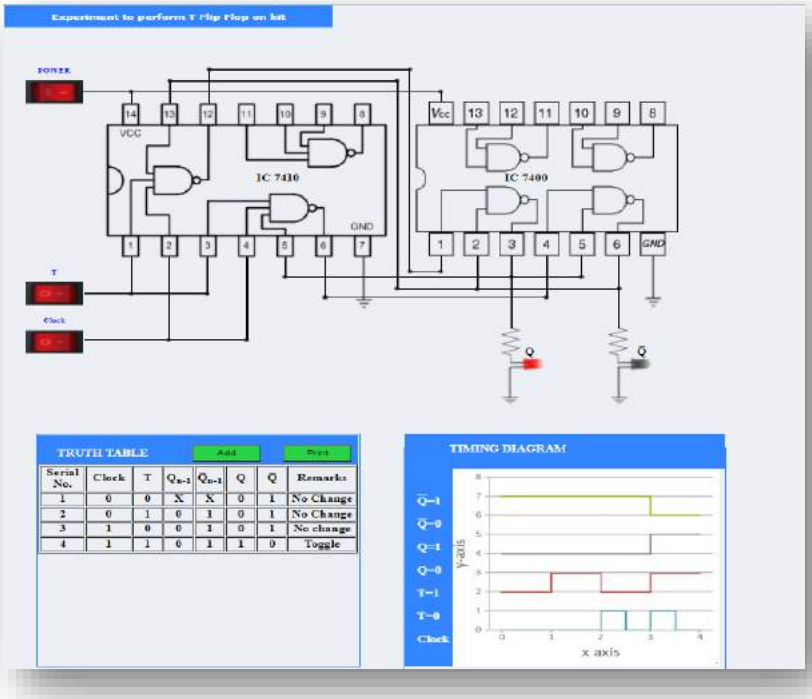
T	Clock	Q	Q'
0	↑	Q	Q'
1	↑	Q'	Q
X	↓	Q	Q'

Fig 8. Characteristics table of T flip flop

Observations :-







Conclusion :- It verifies the truth table and timing diagram of RS, JK, T and D flip-flops and analyses the circuit of RS, JK, T and D flip-flops.