AIM: Logic gate is a basic building block of a digital circuit. So verify the truth tables of all the logic gates on trainer kit using TTL ICs. Also verify them using multisim.

APPARATUS REQUIRED: Bread board, IC's (7400,7402,7404,7408,7432,7486,74266), single stand connecting wires**,** Digital Trainer kit and 4mm connecting leads.

USING SIMULATOR

https://de-iitr.vlabs.ac.in/exp/truth-table-gates/

OBSERVATION

Inputs		Output						
A	B	AND Gate	OR Gate	NOT Gate	NAND Gate	NOR Gate	EX-OR Gate	EX-NOR Gate
0	0	0	0	1	1	1	0	1
0	1	0	1	---	1	0	1	0
1	0	0	1	---	1	0	1	0
1	1	1	1	0	0	1	0	1

USING MULTISIM:-

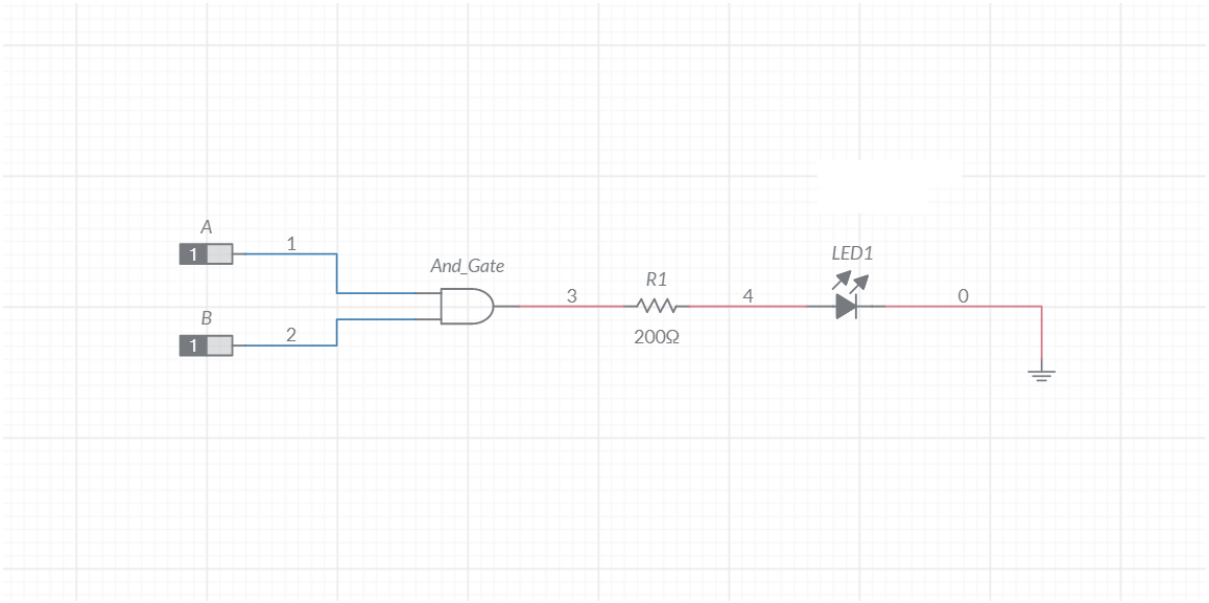


Figure: - 1. AND gate

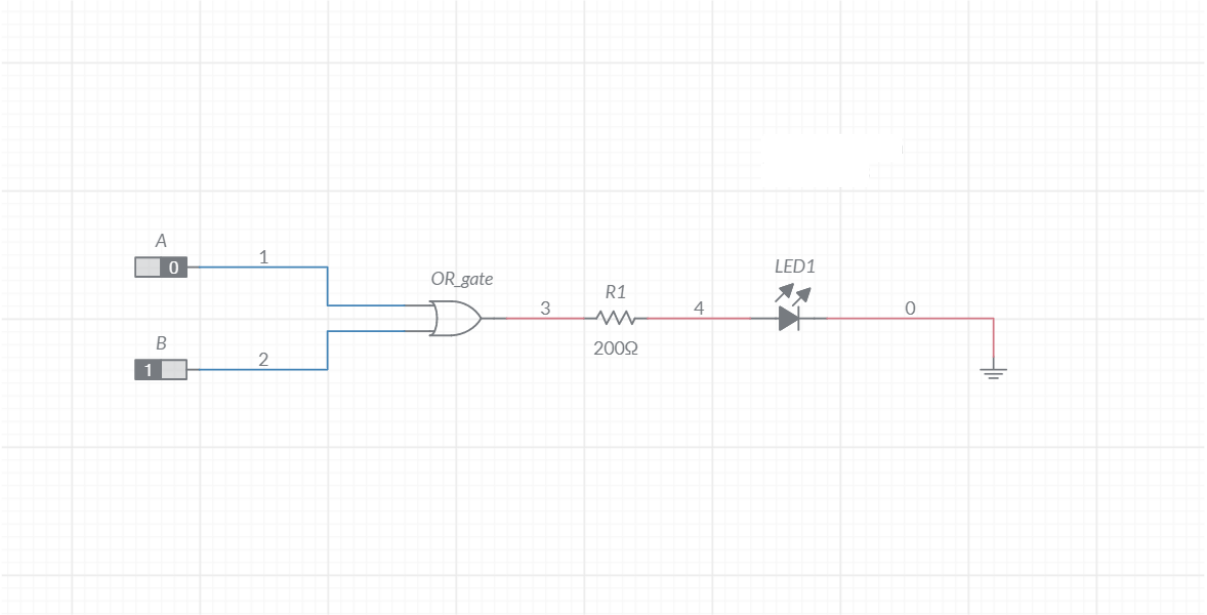


Figure: - 2. OR gate

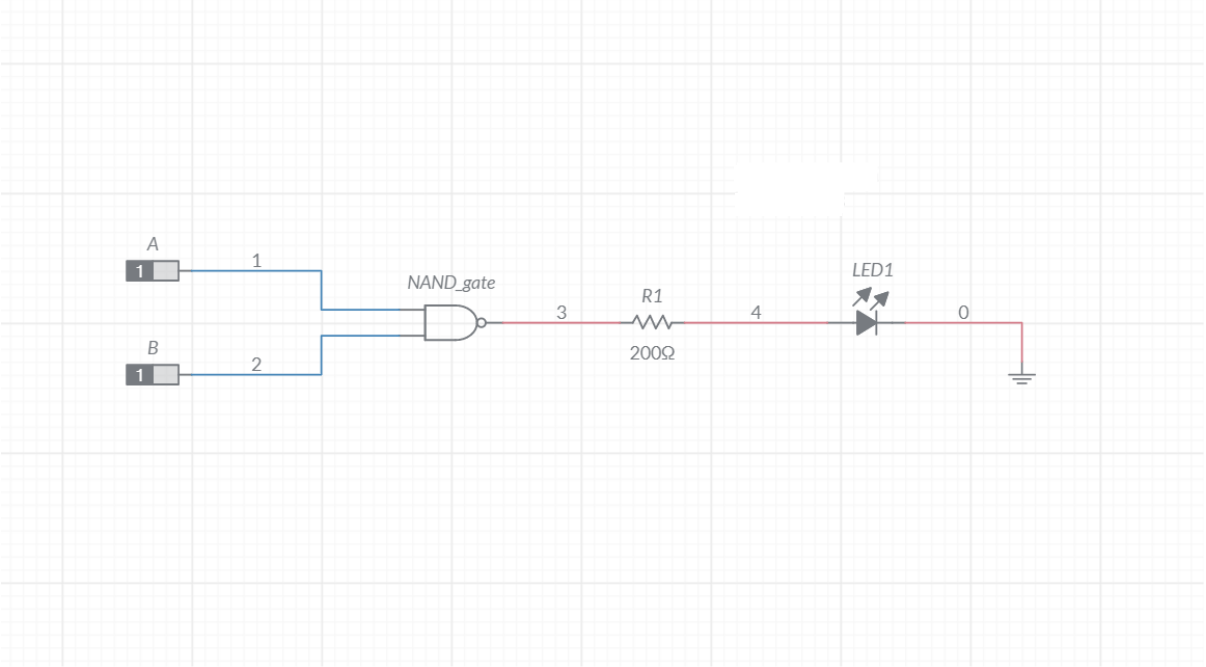


Figure: - 3. NAND gate

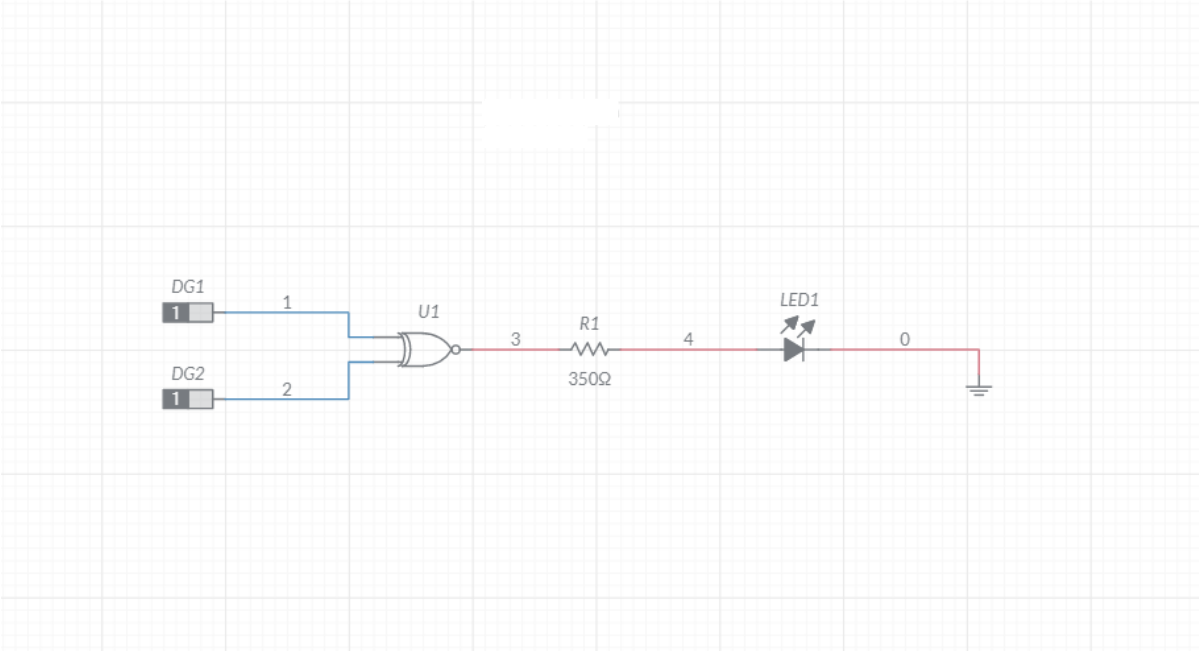
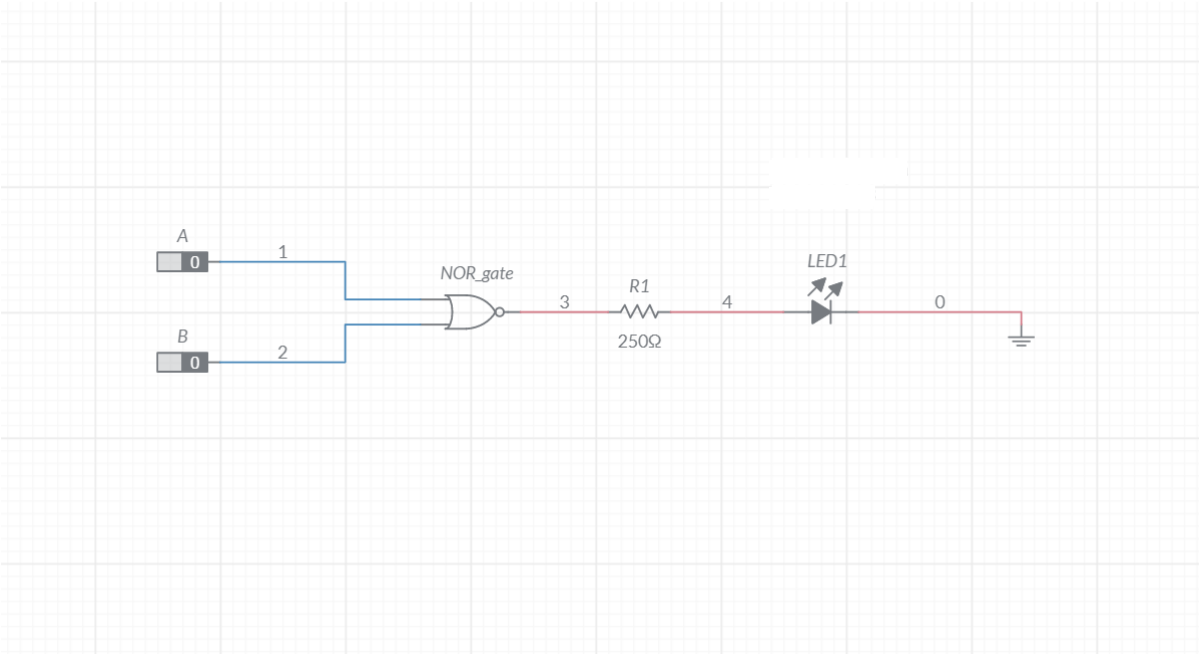
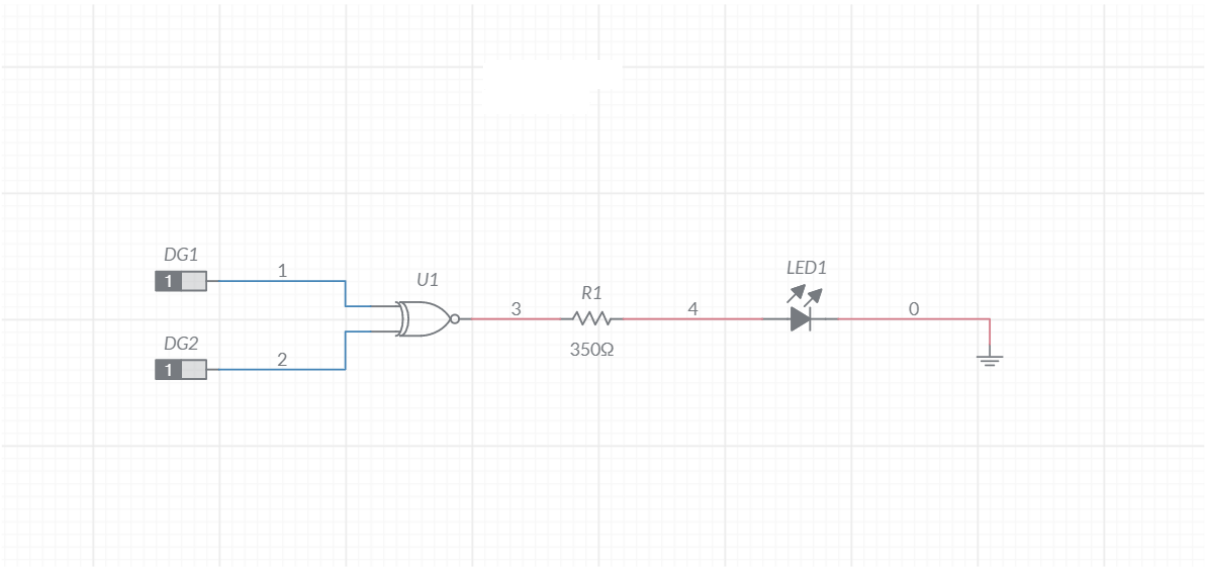


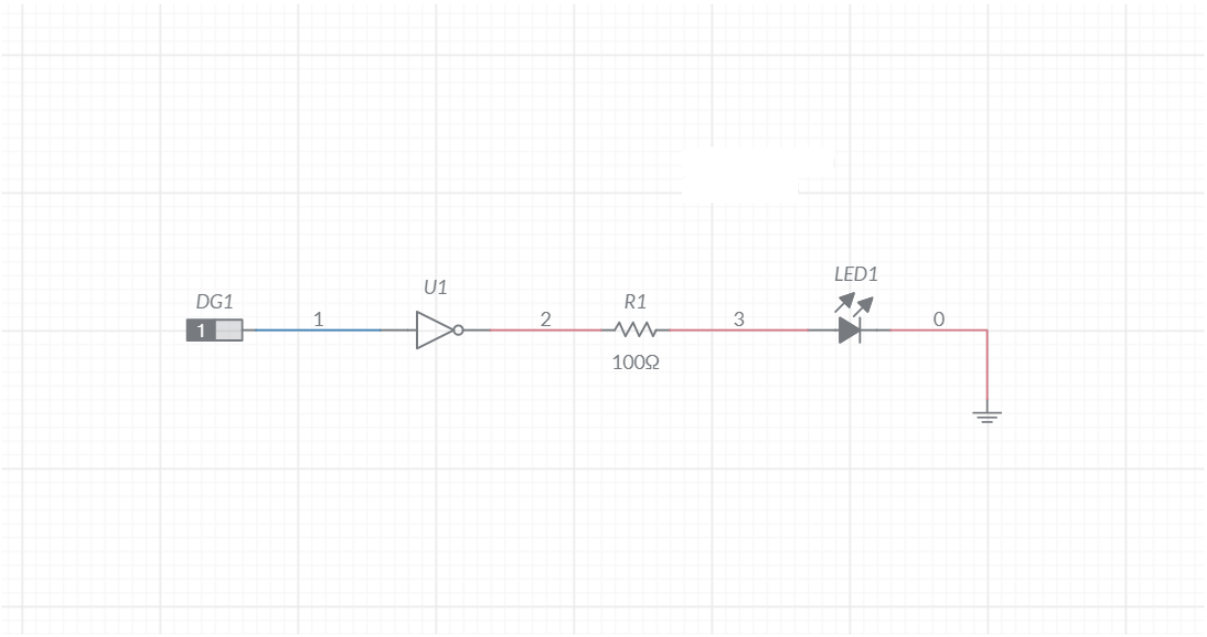
Figure: - 4. EX-OR gate



5. NOR gate



6. XNOR gate



7. NOT gate

AIM: Mr. Vivek wants to add two numbers in computer but computer only understands the binary numbers i.e. 0&1. So design a circuit that adds equivalent of two decimal numbers.

APPARATUS REQUIRED: Logic Trainer kit, connecting wires, bread board, IC `7432,7404,7486,7400, 7408.

HALF ADDER (BY USING FF)

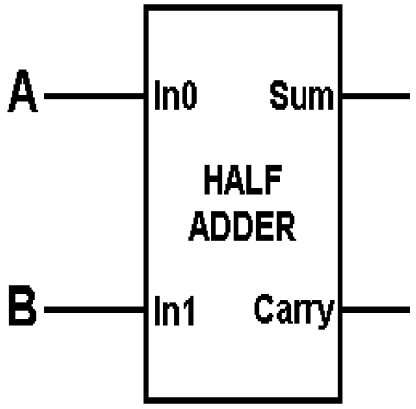


Fig:-2.1 Half adder Schematic diagram

(a) Logical Diagram:

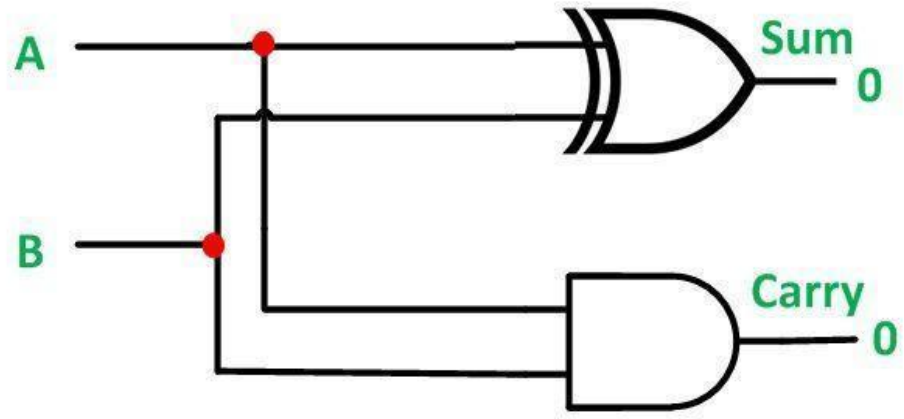


Figure: - 2.2 Half Adder Logical Diagram

(b) Truth Table:

Input		Output	
A	B	SUM	CARRY
0	0	**0**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**

(c) Logical Expression:

Sum:- A'.B + A.B'

Carry:- A.B

FULL ADDER (BY USING FF):

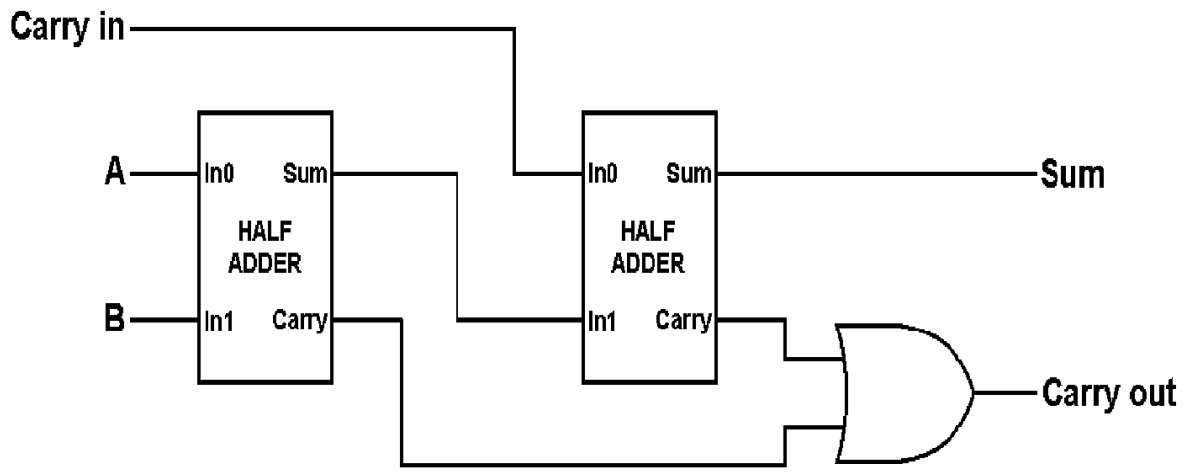


Figure: -2.3 Full Adders Schematic diagram

(a) Logical Diagram

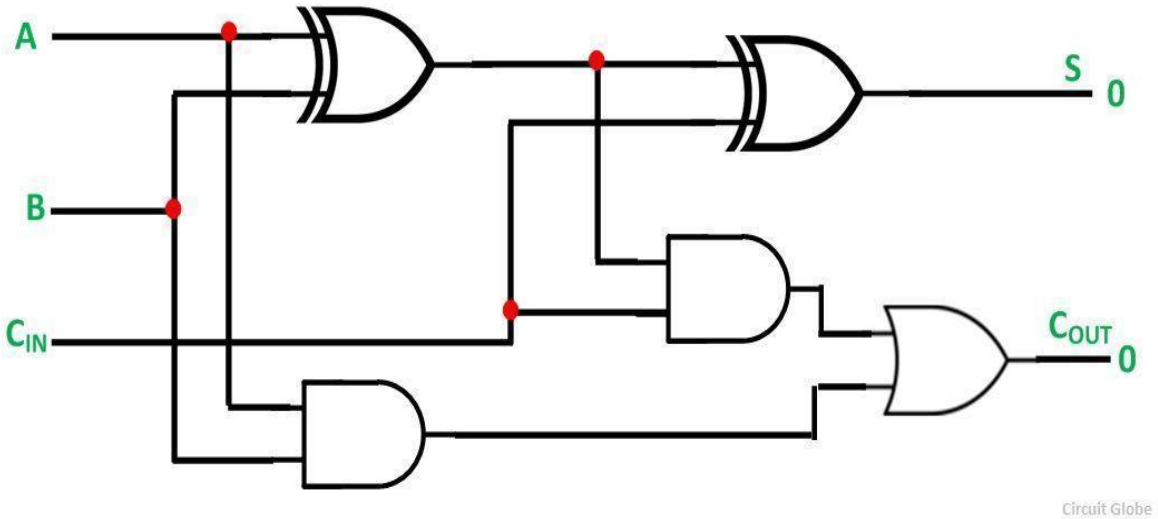


Figure: -2.4 Full Adder logical diagram

(b) Truth Table:

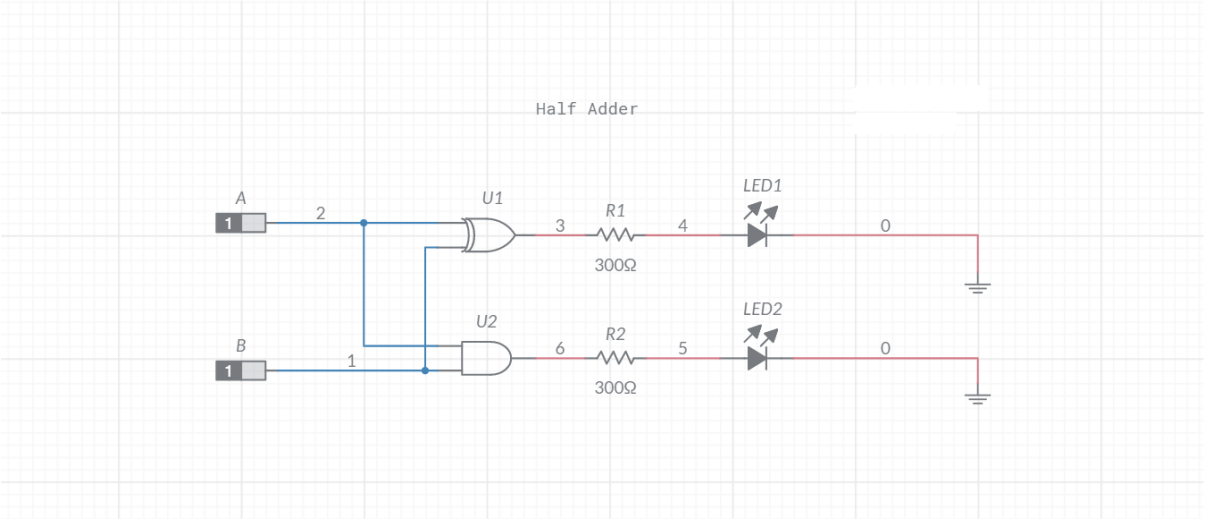
Input			Output	
A	**B**	**Cin**	**Sum**	**Cout**
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

(c) Logical Expression:

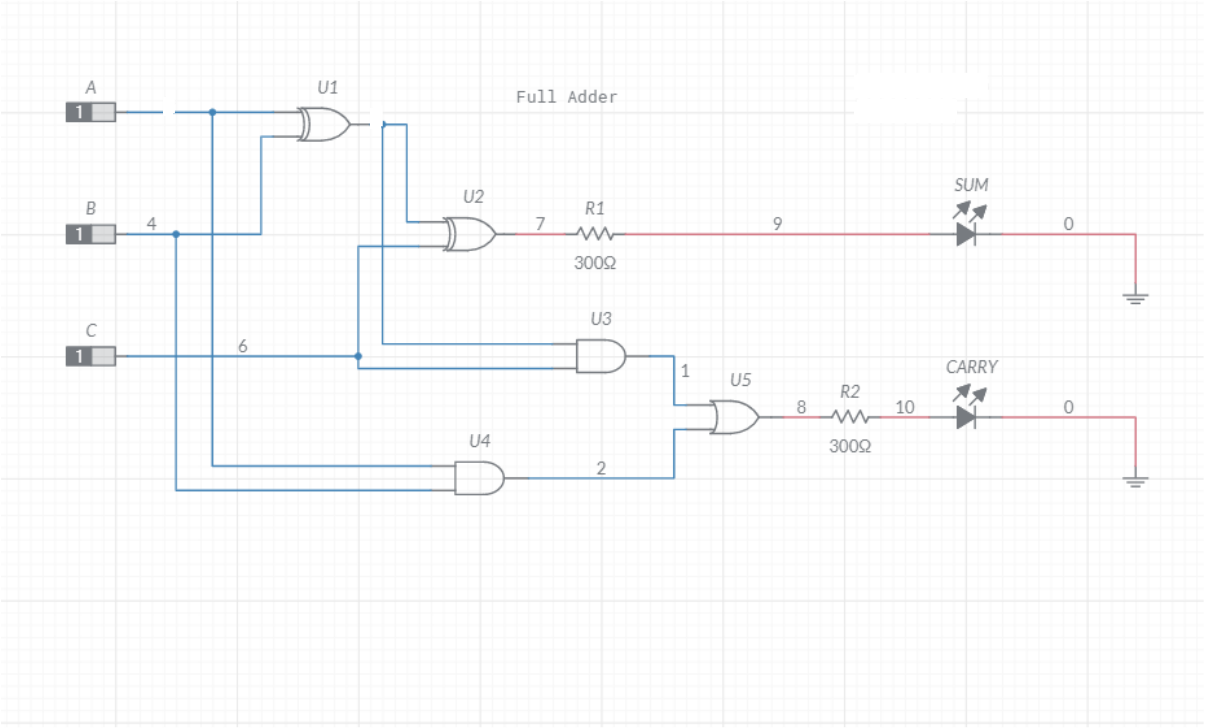
Sum:- A'(B.C' + B'.C) + A(B'.C' + B.C)

Carry:- AB + BC + AC

Half Adder Logical Diagram (Made On Multisim)



Full Adder Logical Diagram (Made on Multisim)



CONCLUSION:

. Hence we have designed the circuits of half adder and full adder circuit from the truth table and logical expression on multisim.

AIM:-Suppose there are two binary numbers as input and subtract one binary number input from other binary number input.

APPARATUS REQUIRED: -Logic trainer kit, bread board, IC 7408, 7404, 7432, 7486 , single stand connecting wire and 4mm connecting leads.

HALF SUBTRACTOR

(a) Logical Diagram:

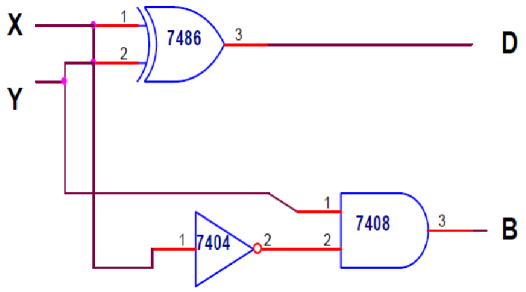


Figure 3.1 Half Subtractor

(b) Truth Table

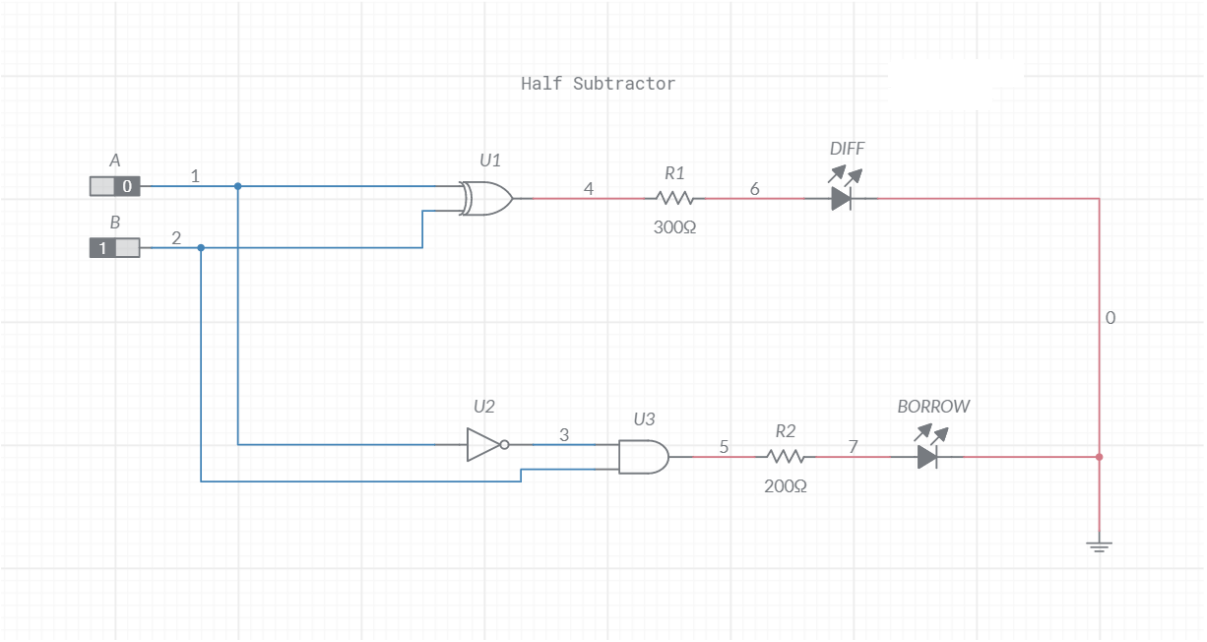
Input		Output	
A	**B**	**D (Difference)**	**B (Borrow)**
0	0	**0**	**0**
0	1	**1**	**1**
1	0	**1**	**0**
1	1	**0**	**0**

(c) Logical Expression

Difference:- A.B' + A'.B

Borrow:- A'.B

(d) Half Subtractor Logical Diagram (Made On Multisim)



FULL SUBTRACTOR

(a) Logical Diagram

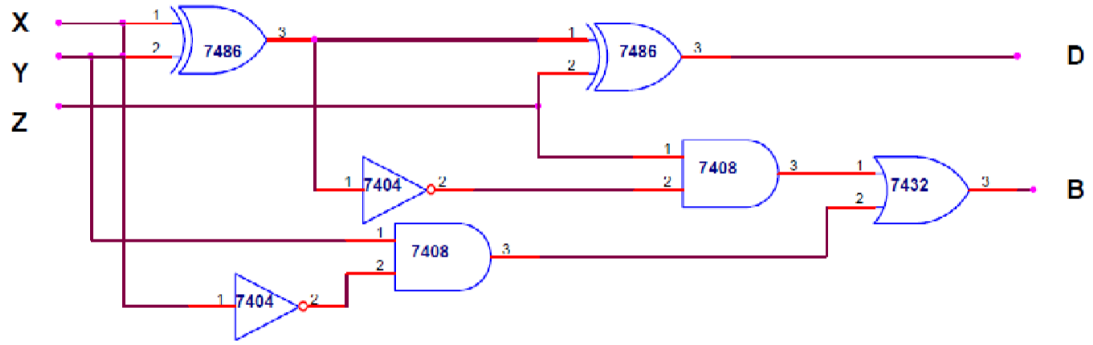


Figure 3.2 Full Subtractor

(b) Truth Table

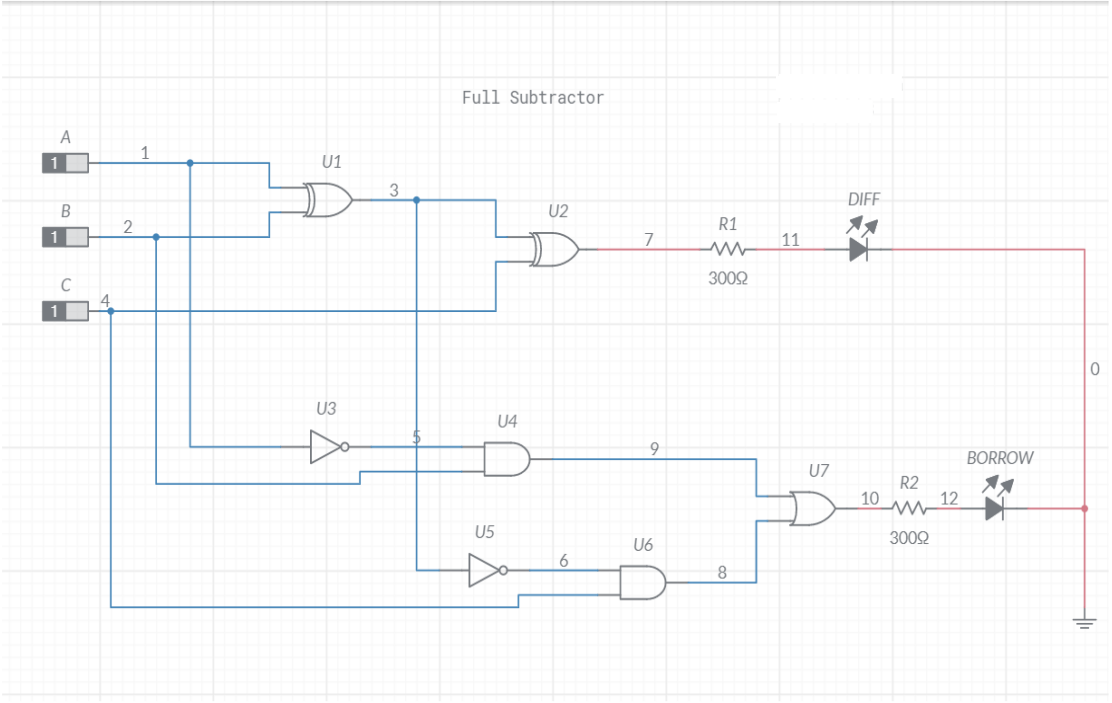
Input			Output	
A	**B**	**Cin**	**D**	**B**
0	0	0	**0**	**0**
0	0	1	**1**	**1**
0	1	0	**1**	**1**
0	1	1	**0**	**1**
1	0	0	**1**	**0**
1	0	1	**0**	**0**
1	1	0	**0**	**0**
1	1	1	**1**	**1**

(c) Logical Expression

Difference:- $A'(B'.B_{in} + B.B_{in}') + A(B'.B_{in}' + B.B_{in})$

Borrow-out:- $A'.B + A'B_{in} + B.B_{in}$

(d) Full Subtractor Logical Diagram (Made on Multisim)



PROCEDURE:

1. Verify the gates.
2. Make the connections as per the circuit diagram.
3. Switch on VCC and apply various combinations of input according to truth table.
4. Note down the output readings for half/full subtractor the borrow bit for different combinations of inputs verify their truth tables.

PRECAUTIONS:

1. All the connections should be made properly.
2. IC should not be reversed.

CONCLUSION:- Hence we have designed the circuits of half subtractor and full subtractor circuit from the truth table and logical expression on multisim.

AIM: Considering two numbers (each two bit), Design a circuit which produces the output that compares whether the number is greater than, less than or equal to the second number.

APPARATUS REQUIRED: Logic Trainer kit, connecting wires, bread board, IC 7432, 7404, 7408, 74266.

1- Bit Magnitude Comparator

Fig:-2.1 1- Bit Magnitude Comparator

(a) Logical Diagram:

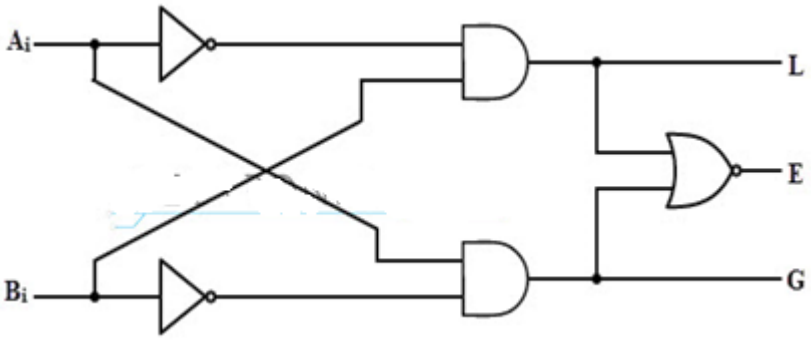


Figure: - 2.2 Half Adder Logical Diagram

(b) Truth Table:

Input		Output		
A	**B**	**L**	**E**	**G**
0	0	**0**	**1**	**0**
0	1	**1**	**0**	**0**
1	0	**0**	**0**	**1**
1	1	**0**	**1**	**0**

(c) Logical Expression:

A>B: AB'

A<B: A'B

A=B: A'B' + AB

2- Bit Magnitude Comparator

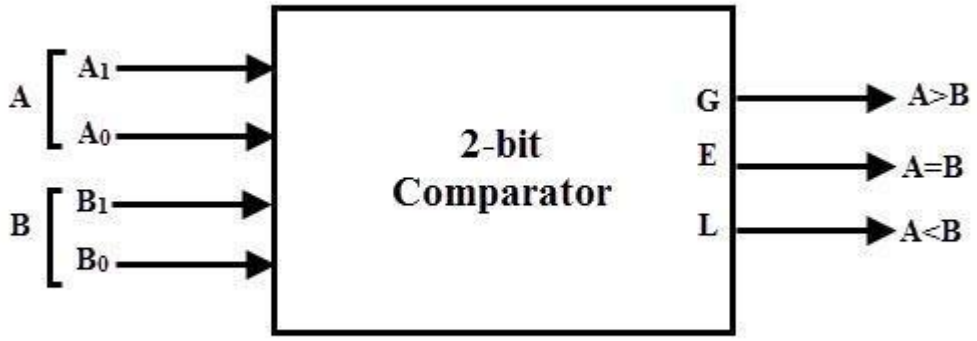


Figure: -2.3 2- Bit Magnitude Comparator Schematic diagram

(a) Logical Diagram

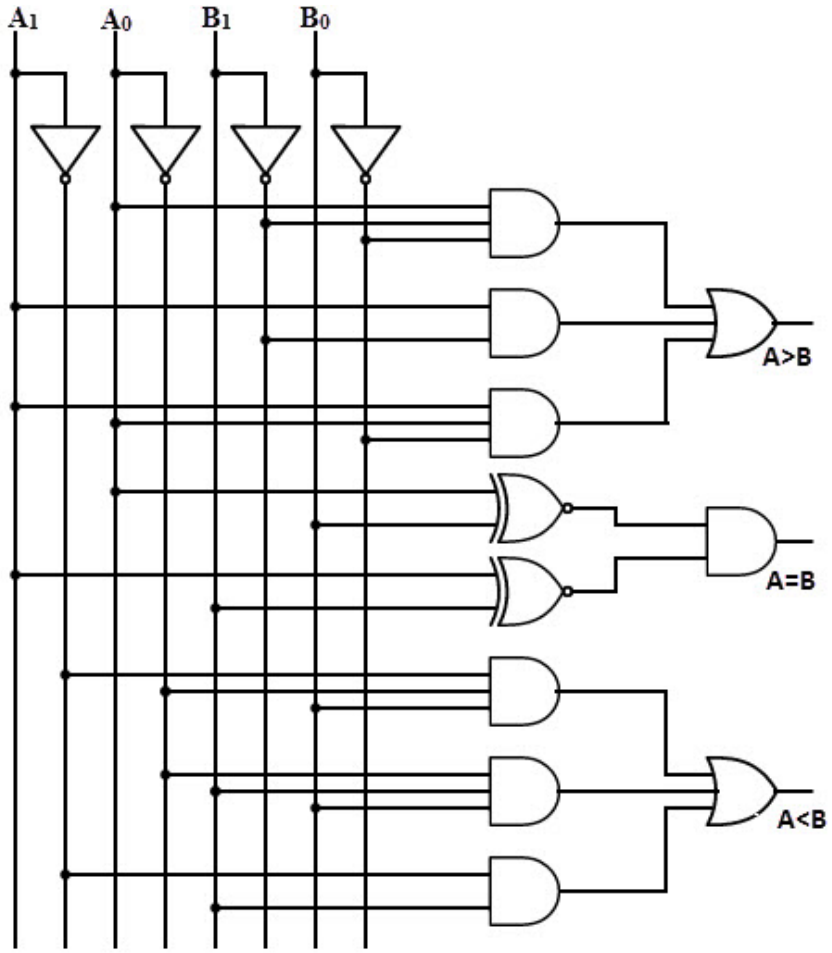


Figure: -2.4 2- Bit Magnitude Comparator logical diagram

(b) Truth Table:

Input				Output		
A1	**A0**	**B1**	**B0**	**L(A<B)**	**E(A=B)**	**G(A>B)**
0	0	0	0	**0**	**1**	**0**
0	0	0	1	**1**	**0**	**0**
0	0	1	0	**1**	**0**	**0**
0	0	1	1	**1**	**0**	**0**
0	1	0	0	**0**	**0**	**1**
0	1	0	1	**0**	**1**	**0**
0	1	1	0	**1**	**0**	**0**
0	1	1	1	**1**	**0**	**0**
1	0	0	0	**0**	**0**	**1**
1	0	0	1	**0**	**0**	**1**
1	0	1	0	**0**	**1**	**0**
1	0	1	1	**1**	**0**	**0**
1	1	0	0	**0**	**0**	**1**
1	1	0	1	**0**	**0**	**1**

1	1	1	0	**0**	**0**	**1**
1	1	1	1	**0**	**1**	**0**

(c) Logical Expression:

A>B: A1B1’ + A0B1’B0’ + A1A0B0’

A=B: A1’A0’B1’B0’ + A1’A0B1’B0 + A1A0B1B0 + A1A0’B1B0’

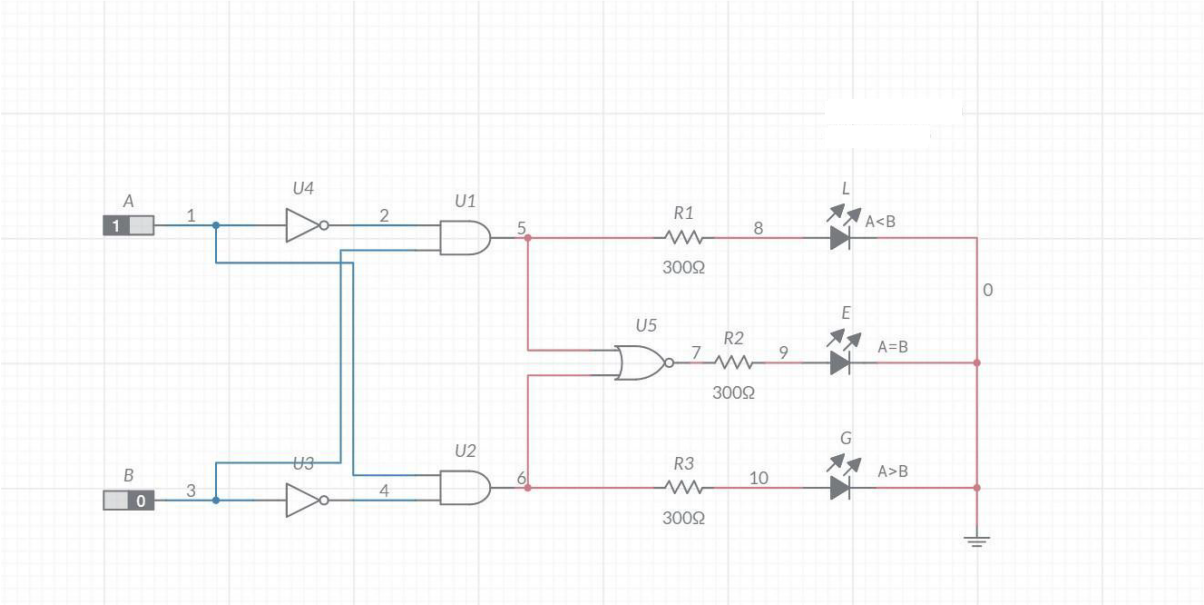
: A1’B1’ (A0’B0’ + A0B0) + A1B1 (A0B0 + A0’B0’)

: (A0B0 + A0’B0’) (A1B1 + A1’B1’)

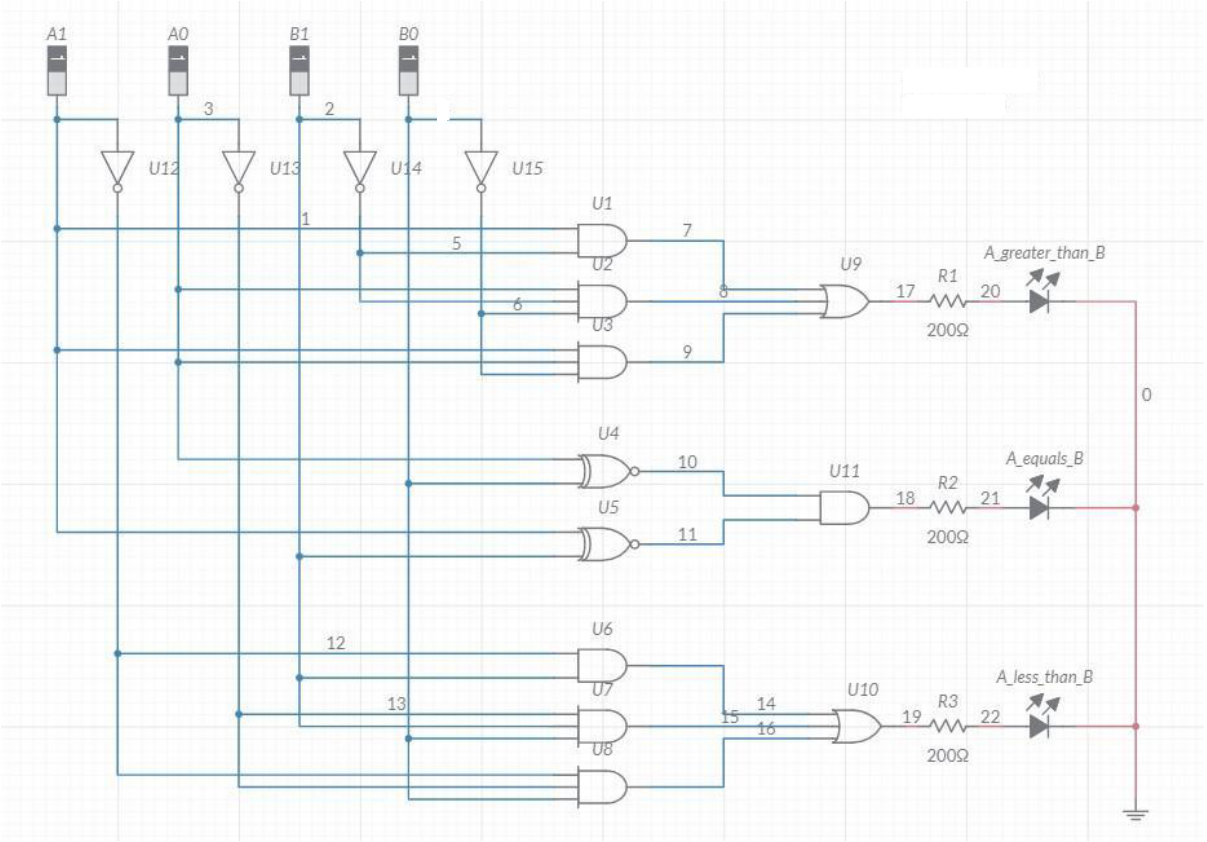
: (A0 Ex-Nor B0) (A1 Ex-Nor B1)

A<B: A1’B1 + A0’B1B0 + A1’A0’B0

1- Bit Magnitude Comparator Logical Diagram (Made On Multisim)



2- Bit Magnitude Comparator Logical Diagram (Made on Multisim)



CONCLUSION: So, considering two numbers (each two bit), we have designed a circuit which produces the output that compares whether the number is greater than, less than or equal to the second number.

Experiment 5

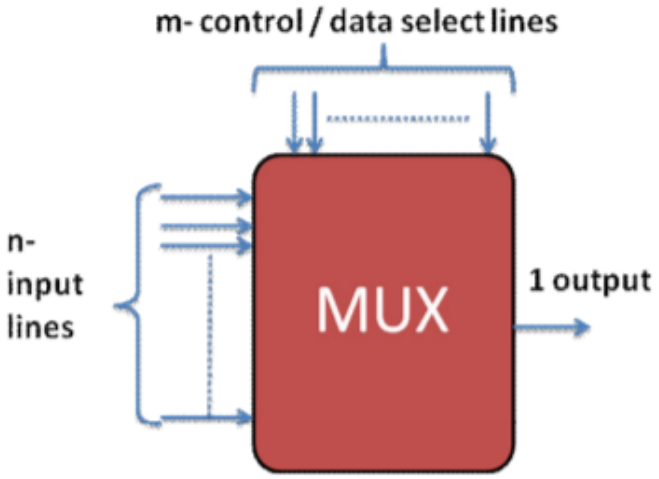
Aim: i) Suppose one input is to be selected from n inputs. Implement the circuit using IC 74150.

ii) A circuit distributes one input into n output lines. Design the circuit using IC 74154.

APPARATUS: MUX/ DEMUX Trainer kit, connecting leads.

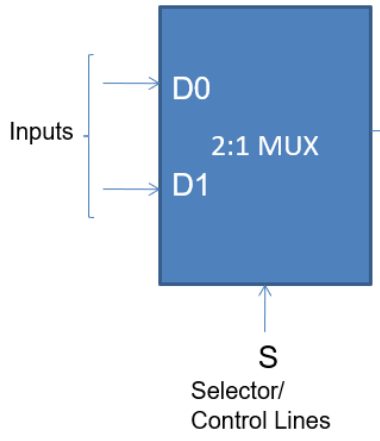
Theory:

Multiplexers (Many to 1)



- It is a combinational circuit
- It has many data inputs and single output
- For 2^n input lines, n selection lines are required.
- Multiplexers are also known as
  - **many to one circuit,**
  - **Data n selector,**
  - **parallel to serial convertor,**
  - **universal logic circuit.**
- Multiplexers are mainly used to increase amount of the data that can be sent over the network within certain amount of time and bandwidth.
- Multiplexer can act as universal combinational circuit. All the standard logic gates can be implemented with multiplexers.

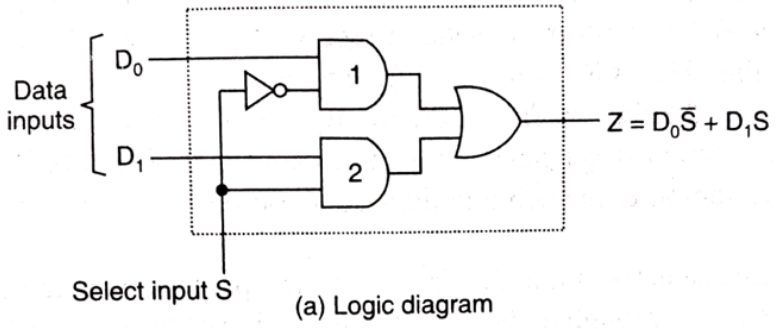
2:1 MUX:



When S=0,
AND gate 1 is enabled and AND gate 2 is disabled,
So Z=D0.

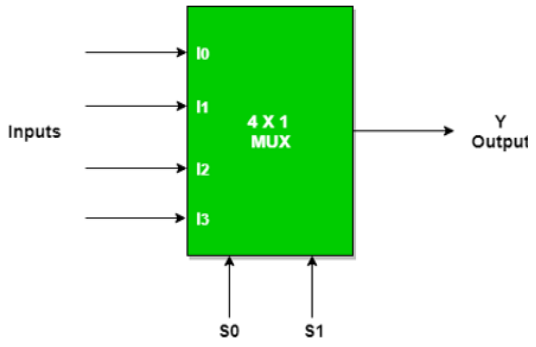
When S=1,
AND gate 1 is diables and AND gate 2 is enabled,
So Z=D1.

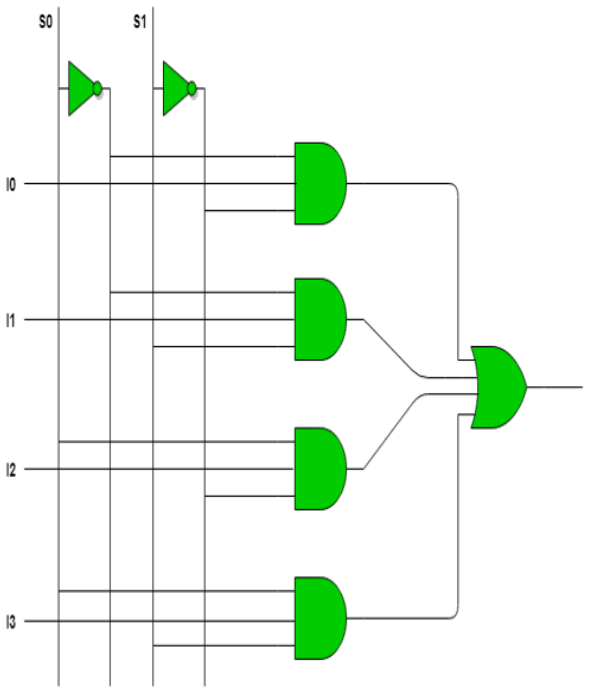
S	Y
0	D0
1	D1



$$Y = \overline{S}.D0 + S.D1$$

4:1 MUX:





Truth Table

S0	S1	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

So, final equation,

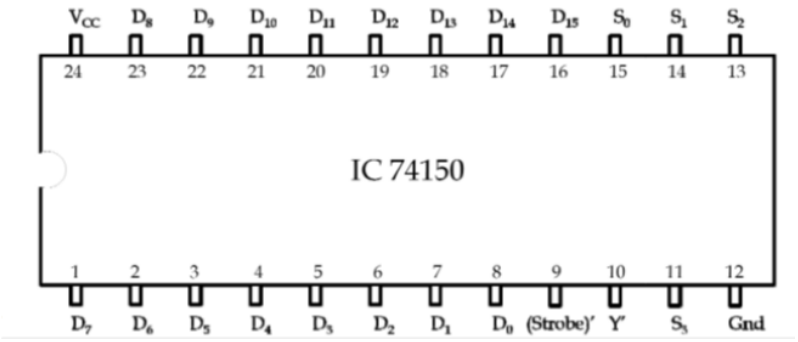
Y = S0'.S1'.I0 + S0'.S1.I1 + S0.S1'.I2 + S0.S1.I3

TRUTH TABLE 16:1 MUX

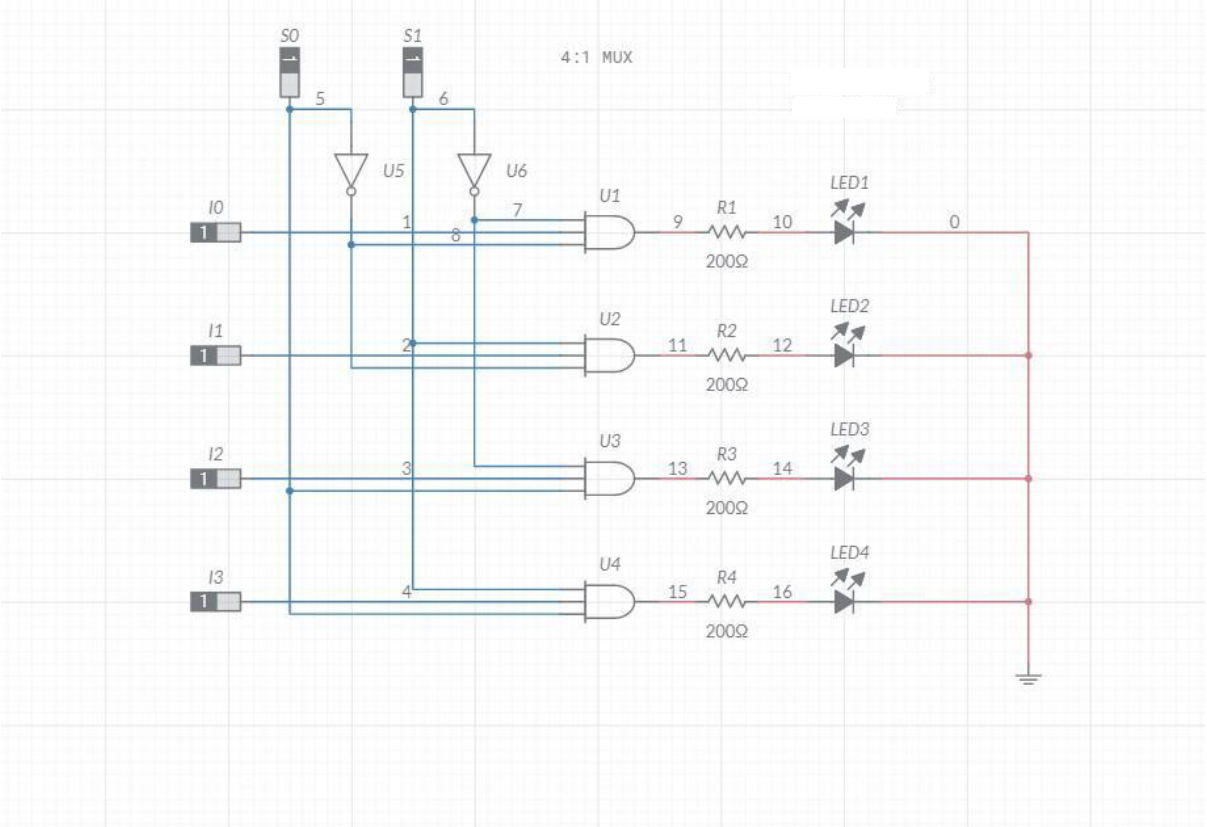
Strobe input	Selection Lines				Output
(Strobe)′	S_3	S_2	S_1	S_0	Y
0	0	0	0	0	D_0'
0	0	0	0	1	D_1'
0	0	0	1	0	D_2'
0	0	0	1	1	D_3'
0	0	1	0	0	D_4'
0	0	1	0	1	D_5'
0	0	1	1	0	D_6'
0	0	1	1	1	D_7'
0	1	0	0	0	D_8'
0	1	0	0	1	D_9'
0	1	0	1	0	D_{10}'
0	1	0	1	1	D_{11}'
0	1	1	0	0	D_{12}'
0	1	1	0	1	D_{13}'
0	1	1	1	0	D_{14}'
0	1	1	1	1	D_{15}'
1	x	x	x	x	1

Mux IC: 74150:

PIN DIAGRAM FOR IC 74150:

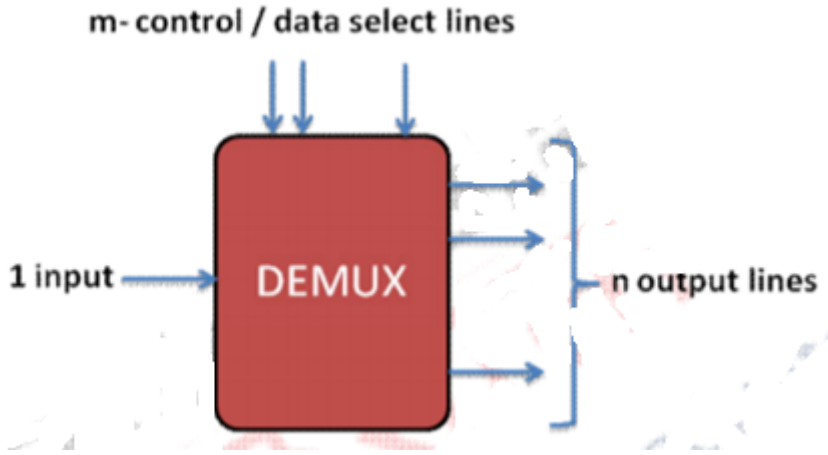


Mux Multisim Circuit:

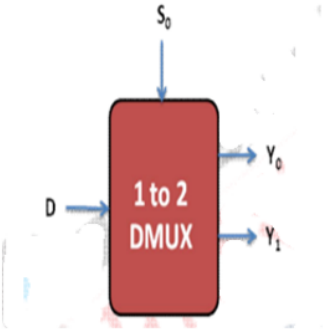


DEMULTIPLEXER:

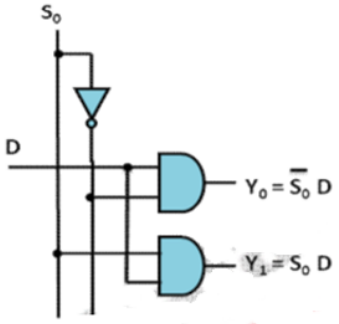
- Demultiplex means one into many.
- A demultiplexer reverses the multiplexing operation.
- It takes one data input source and selectively distributes it to 1 of N output channels

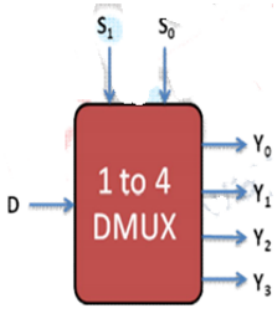


1:2 DEMUX and 1:4 DEMUX:

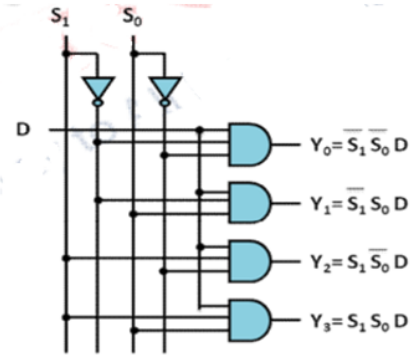


Select input	Outputs	
S0	Y0	Y1
0	D	0
1	0	D





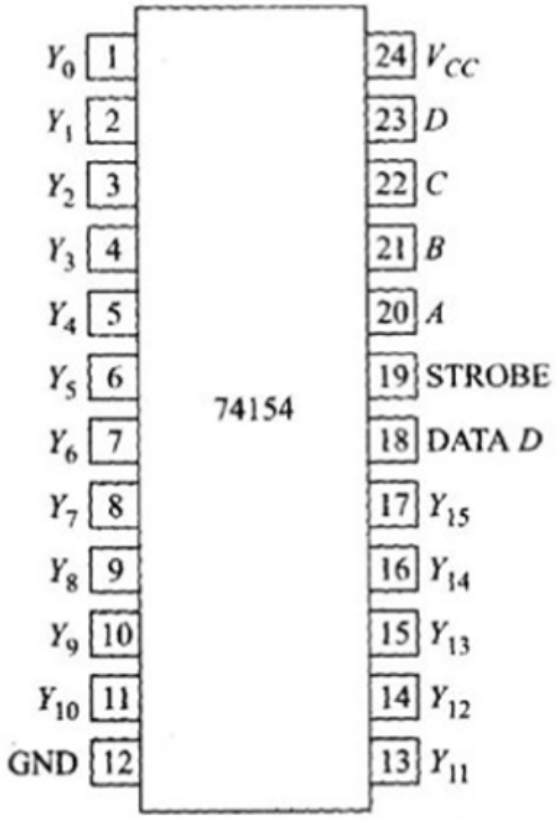
Inputs		Outputs			
S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D



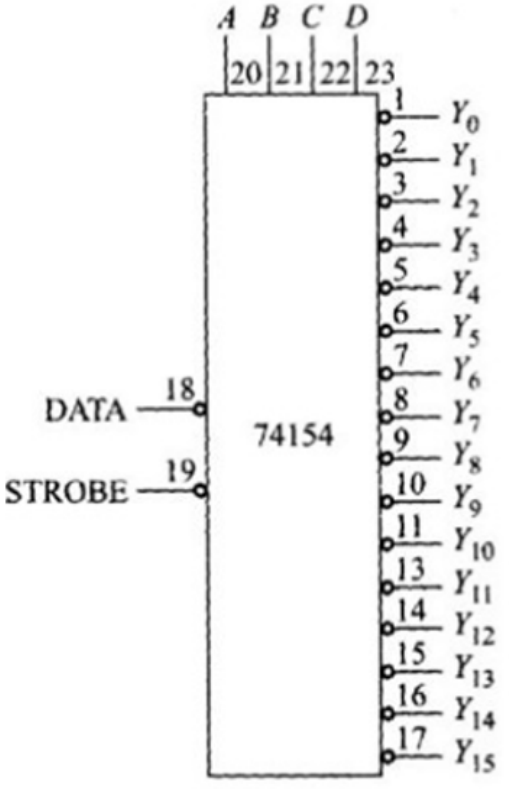
TRUTH TABLE 1:16 DEMUX

INPUTS					OUTPUTS															
$\overline{E}$	A3	A2	A1	A0	O0	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15
1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

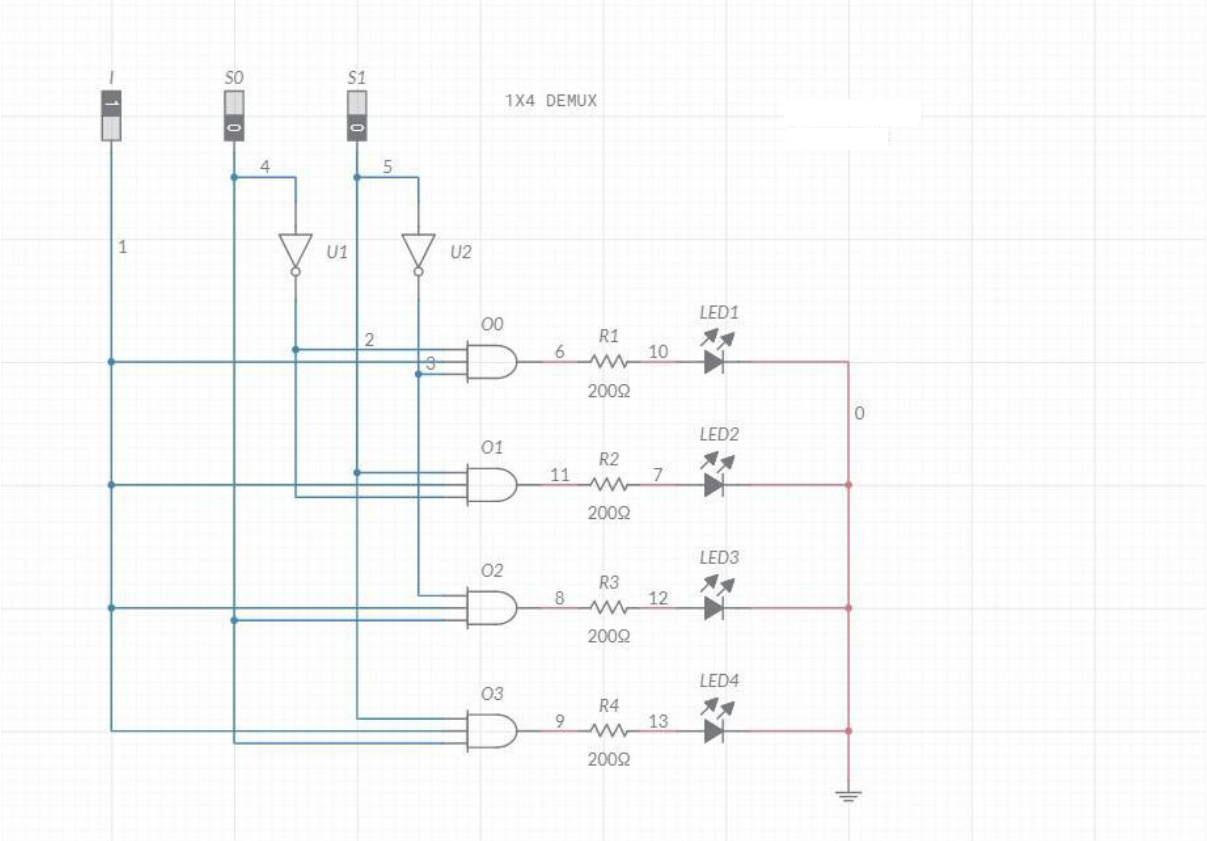
Pinout diagram



Logic diagram



DEMUX Multisim Circuit:



CONCLUSION: So, we have successfully designed MUX and DEMUX on multisim using the truth table and logical expression.

AIM: A code represents each number in the sequence of integers {0...2^N-1} as a binary string of length N in an order such that adjacent integers have code representations that differ in only one bit position. Design a convertor that has above property.

APPARATUS REQUIRED: Logic Trainer kit, connecting wires, bread board, IC 7486.

Fig:-5.1 4-bit Binary to Gray code converter

(a) Logical Diagram:

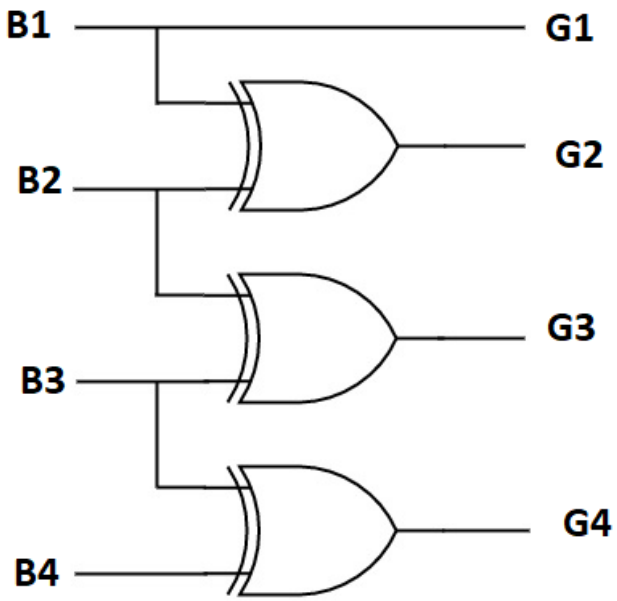


Figure: - 5.2

(b) Truth Table:

Input				Output			
B1	**B2**	**B3**	**B4**	**G1**	**G2**	**G3**	**G4**
0	0	0	0	**0**	**0**	**0**	**0**
0	0	0	1	**0**	**0**	**0**	**1**
0	0	1	0	**0**	**0**	**1**	**1**
0	0	1	1	**0**	**0**	**1**	**0**
0	1	0	0	**0**	**1**	**1**	**0**
0	1	0	1	**0**	**1**	**1**	**1**
0	1	1	0	**0**	**1**	**0**	**1**
0	1	1	1	**0**	**1**	**0**	**0**
1	0	0	0	**1**	**1**	**0**	**0**
1	0	0	1	**1**	**1**	**0**	**1**
1	0	1	0	**1**	**1**	**1**	**1**
1	0	1	1	**1**	**1**	**1**	**0**
1	1	0	0	**1**	**0**	**1**	**0**
1	1	0	1	**1**	**0**	**1**	**1**
1	1	1	0	**1**	**0**	**0**	**1**
1	1	1	1	**1**	**0**	**0**	**0**

$$\mathbf{G1} = \sum m\,(8, 9, 10, 11, 12, 13, 14, 15)$$

$$\mathbf{G2} = \sum m\,(4, 5, 6, 7, 8, 9, 10, 11)$$

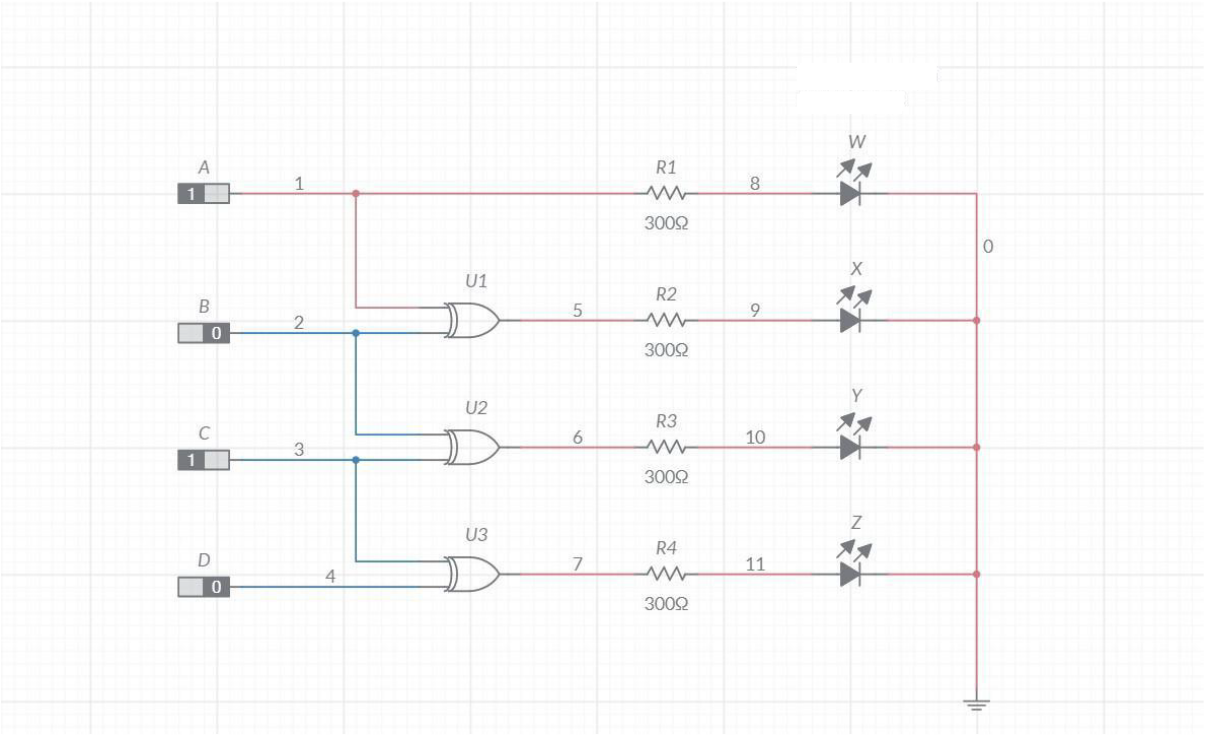
$$\mathbf{G3} = \sum m\,(2, 3, 4, 5, 10, 11, 12, 13)$$

$$\mathbf{G4} = \sum m\,(1, 2, 5, 6, 9, 10, 13, 14)$$

(c) Logical Expression:

(i) G1 = B1
(ii) G2 = B1.B2' + B1'.B2 = B1 X-OR B2
(iii) G3 = B2.B3' + B2'.B3 = B2 X-OR B3
(iv) G4 = B3.B4' + B3'.B4 = B3 X-OR B4

4-bit Binary to Gray code converter Logical Diagram (Made On Multisim)



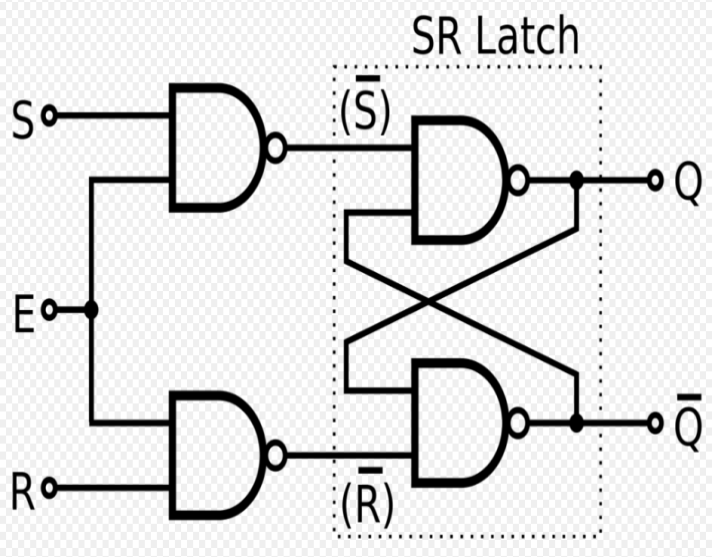
CONCLUSION: So, we have designed a convertor which has the property of a code that represents each number in the sequence of integers {0...2^N-1} as a binary string of length N in an order such that adjacent integers have code representations that differ in only one bit position.

EXPERIMENT 7

AIM: In How many ways one bit of information can be stored in computers. Design and verify at least three different methods using sequential logic circuits.

APPARATUS REQUIRED: -connecting wires, breadboard trainer kit, IC 7486**,** 7476.

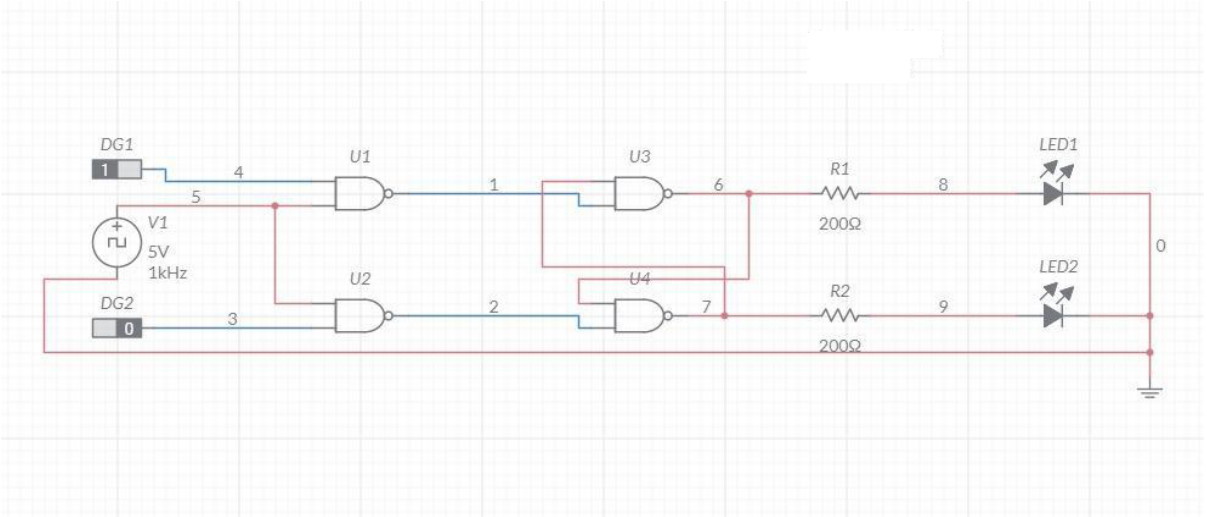
The S R Flip Flop



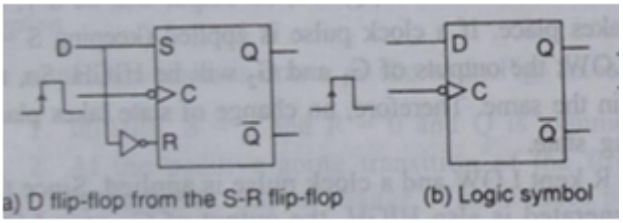
Truth Table

Inputs		Outputs		State
S	R	Q	Q'	
0	0	0	1	No change
0	1	0	1	Reset
1	0	1	0	Set
1	1	X	X	Invalid

Multisim Circuit:

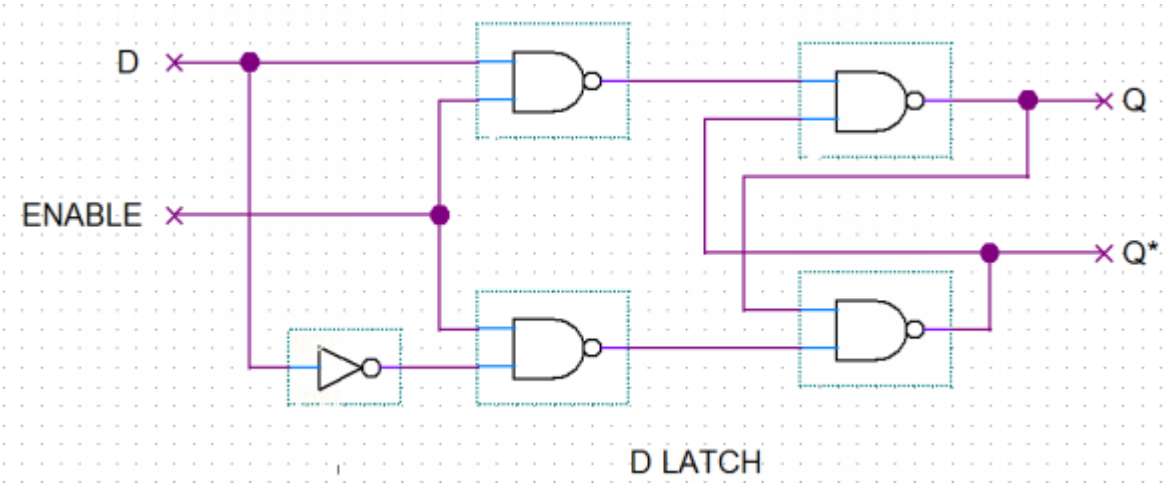


D Flip Flop



Truth Table

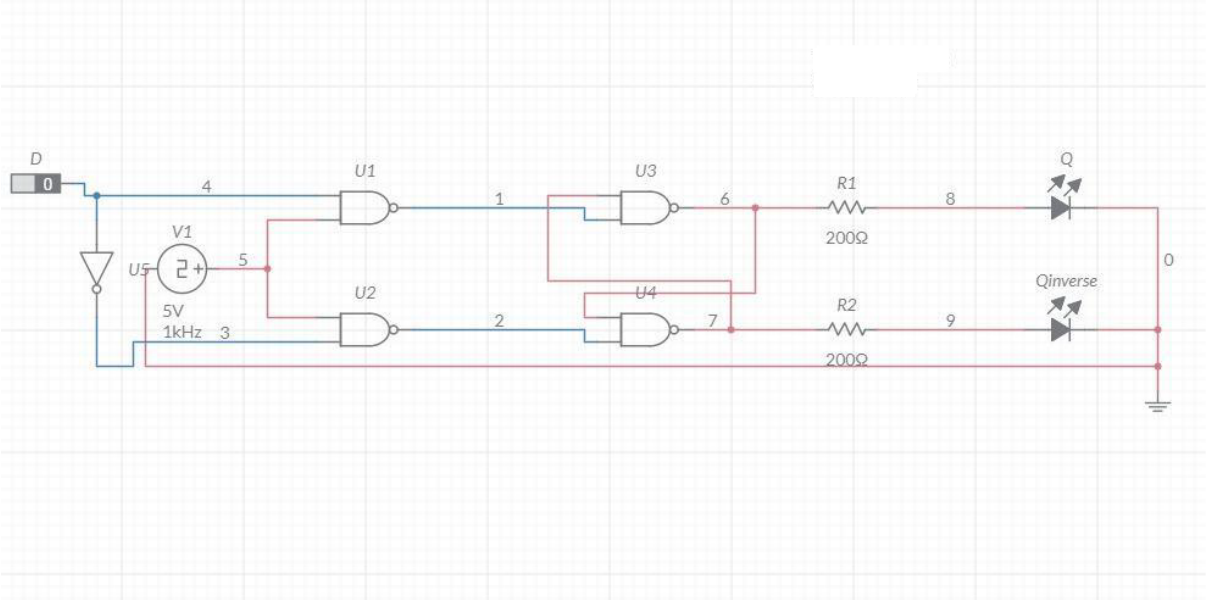
D	Clock	Q	State
0	↓	0	Reset
1	↓	1	Set



Truth Table

Inputs		Outputs		
Clock	D	Q	Q’	State
0	0	Q	Q’	No change
0	1	Q	Q’	No change
1	0	0	1	Reset
1	1	1	0	Set

Multisim Circuit:

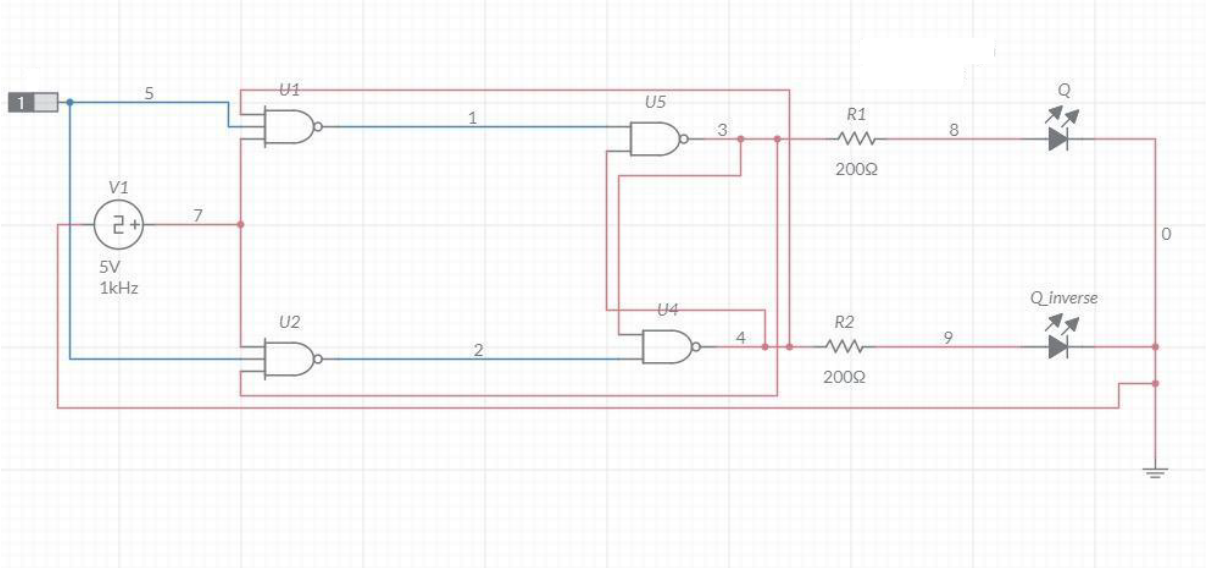


T Flip Flop

Truth Table

T	Clock	Q	State
0	1	0	No change
1	1	1	Toggle

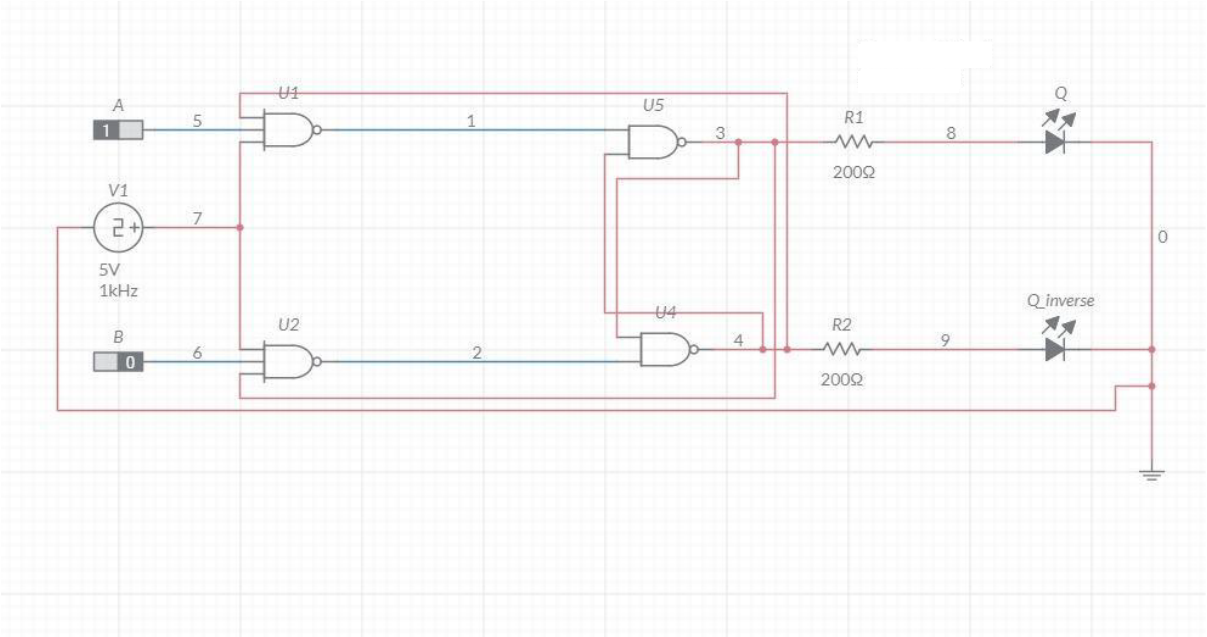
Multisim Circuit



JK Flip Flop

Inputs		Outputs		State
J	K	Q	Q’	
0	0	0	1	No change
0	1	0	1	Reset
1	0	1	0	Set
1	1	0	1	Toggle

Multisim Circuit:



AIM: In digital logic and computing, a counter is a device which stores the number of times a particular event or process has occurred in relationship to a clock signal.

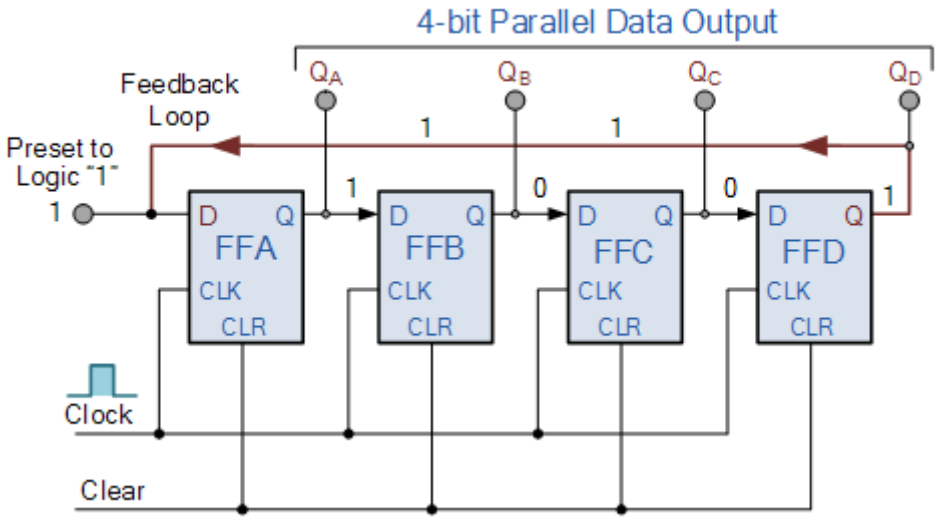
i) Design such a counter which uses a circulating shift register in which last flip flop shifts its value into the first flip flop.
ii) Also design a counter in which the inverted output of the last flip flop is connected to the input of first flip flop.

APPARATUS REQUIRED: - connecting wires, breadboard trainer kit, IC 7476

Ring Counter

Then by looping the output back to the input, (feedback) we can convert a standard shift register circuit into a ring counter. Consider the circuit below.

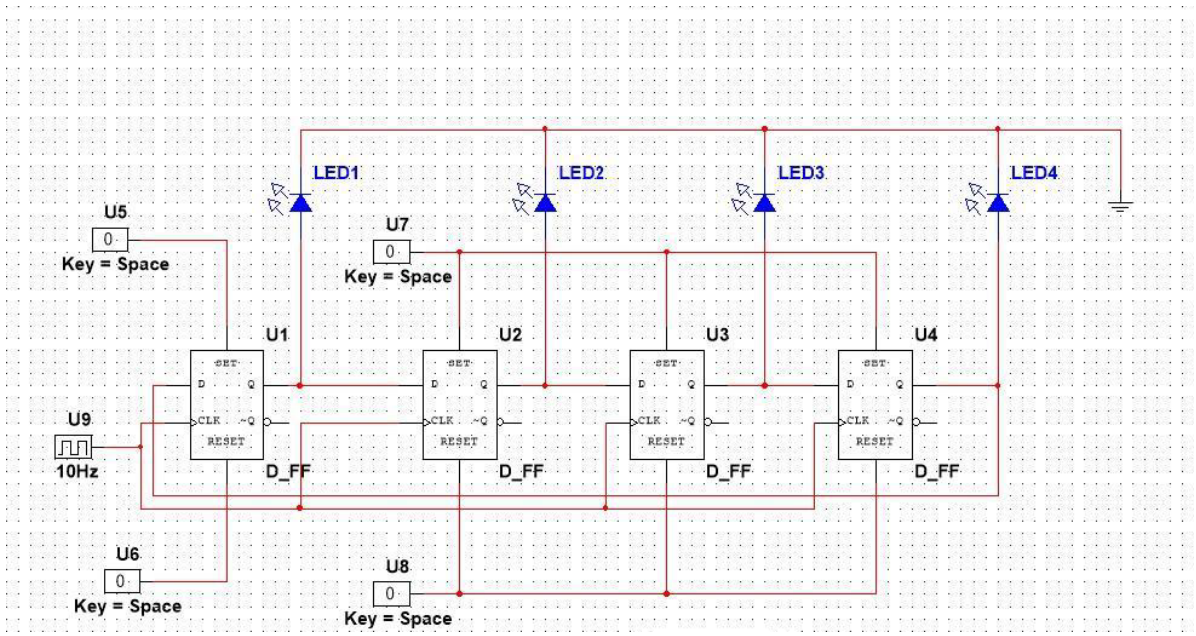
4-bit Ring Counter



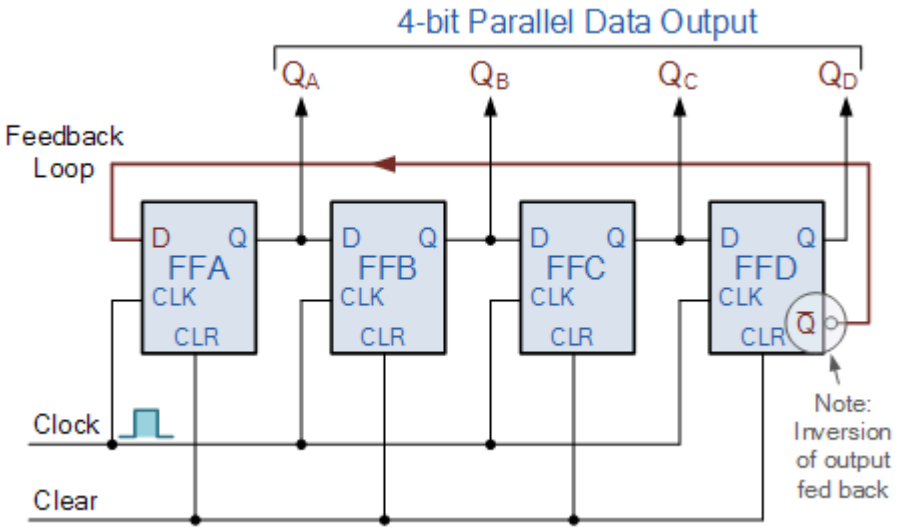
Truth Table

Clock	FFA	FFB	FFC	FFD
0	-	-	-	-
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0
5	0	0	0	1
6	0	0	1	0
7	0	1	0	0
8	1	0	0	0

Multisim Circuit



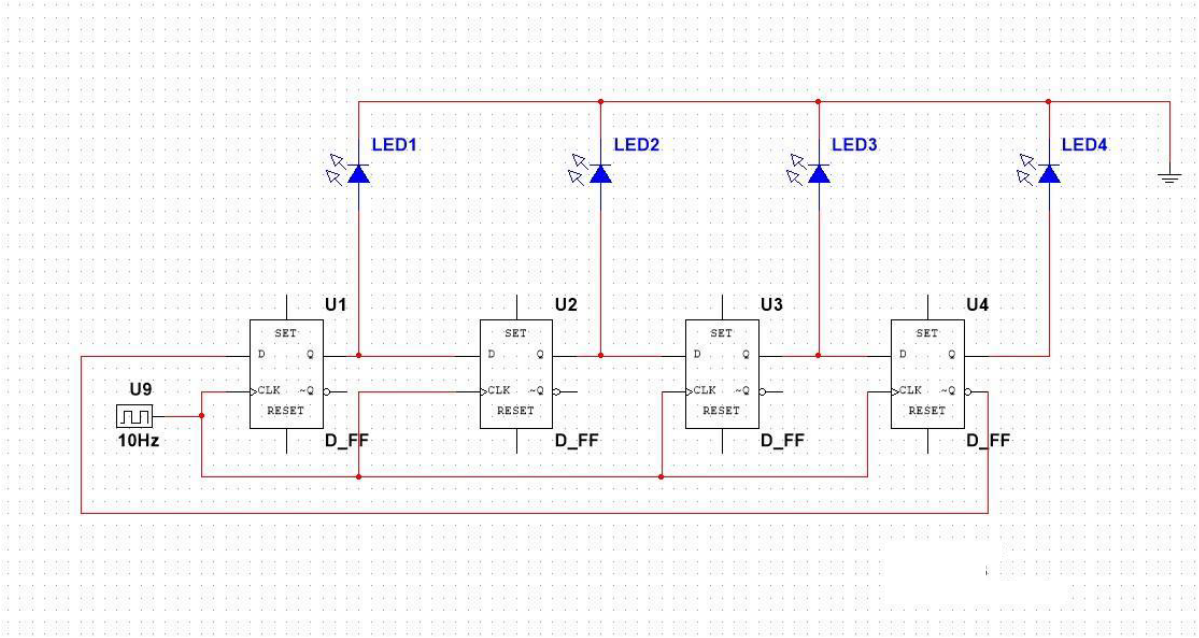
4-bit Johnson/ Twisted Ring Counter



Truth Table

Clock	FFA	FFB	FFC	FFD
0	-	-	-	-
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	1	1	1	0
5	1	1	1	1
6	0	1	1	1
7	0	0	1	1
8	0	0	0	1

Multisim circuit:



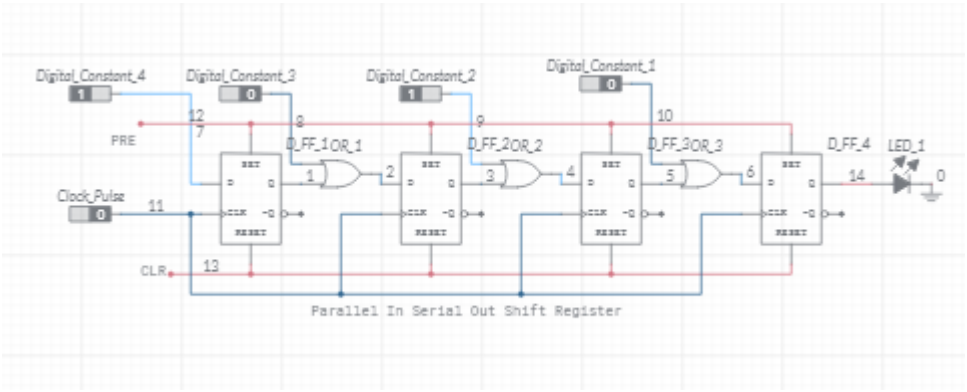
Conclusion:

So, with the help of the truth tables and multisim(app), we designed such a counter which uses a circulating shift register in which last flip flop shifts its value into the first flip flop and a counter in which the inverted output of the last flip flop is connected to the input of first flip flop.

AIM: Suppose there is a need to store 4 bit of data. Which device is required for this purpose also show the transfer of data in SISO, SIPO, PISO and PIPO forms.

APPARATUS REQUIRED: - connecting wires, breadboard trainer kit, IC 7495

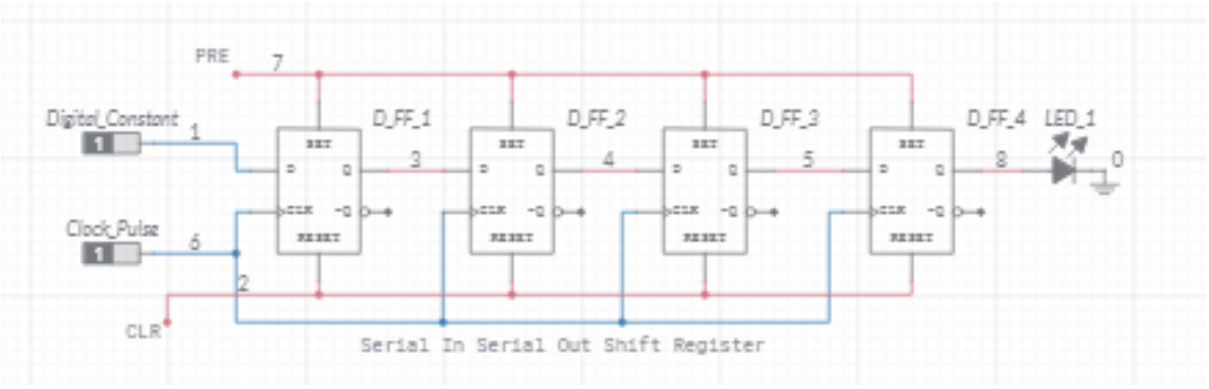
1. **4-bit Parallel-in to Serial-out Shift Register**



Truth Table

Clk	D0	D1	D2	D3	Q3
1	1	0	1	0	0
2	0	1	0	1	1
3	0	0	1	0	0
4	0	0	0	1	1

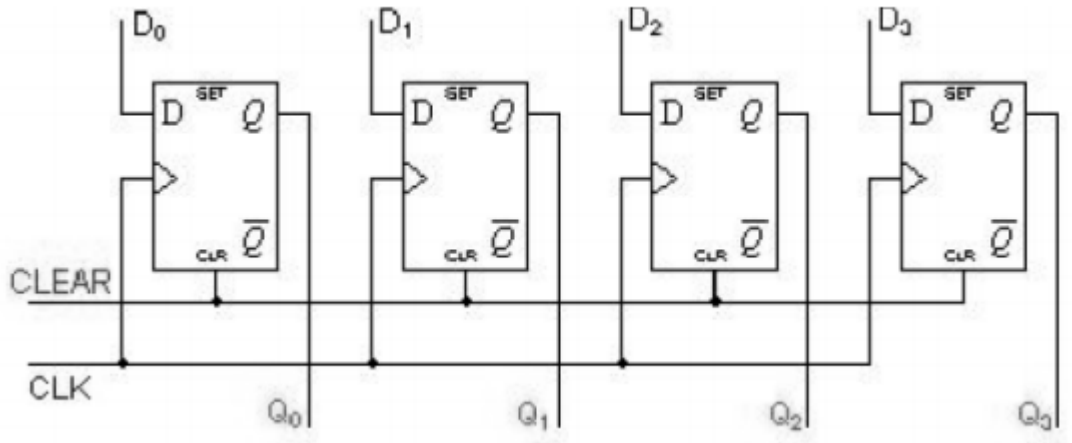
2. **4-bit Serial-in to Serial-out (SISO)**



Truth Table

Clk	D Input	QA	QB	QC	QD
0	0	0	0	0	0
1	1	1	0	0	0
2	0	0	1	0	0
3	1	1	0	1	0
4	0	0	1	0	1
5	-	0	0	1	0
6	-	0	0	0	1
7	-	0	0	0	0
8	-	0	0	0	0

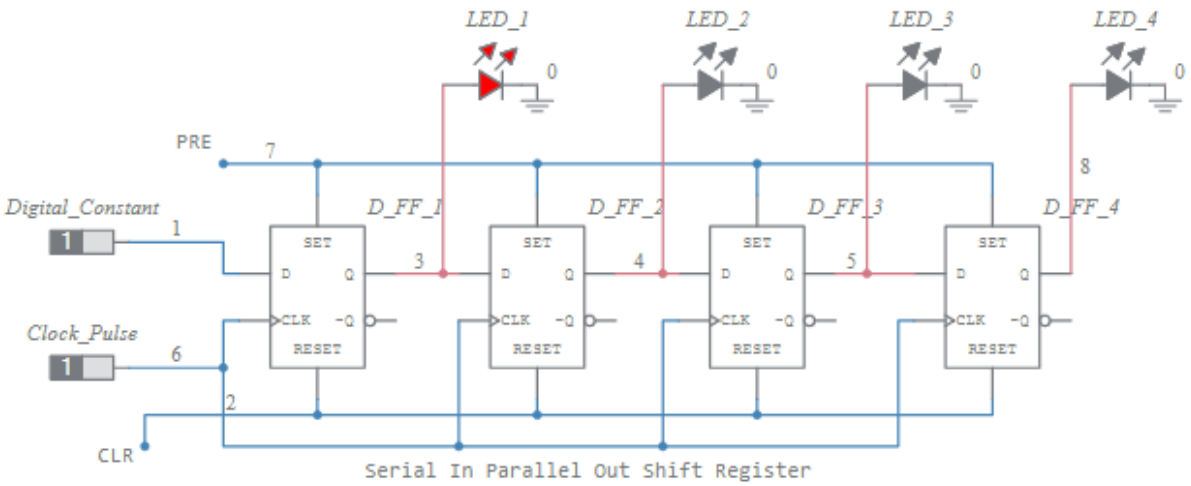
3. 4-bit Parallel-in to Parallel-out Shift Register



Truth Table

Clk	D0	D1	D2	D3	Q1	Q2	Q3	Q4
	0	0	0	0	0	0	0	0
↑	1	0	1	0	1	0	1	0
↑	0	0	0	0	0	0	0	0

4. 4-bit Serial-in to Parallel-out Shift Register

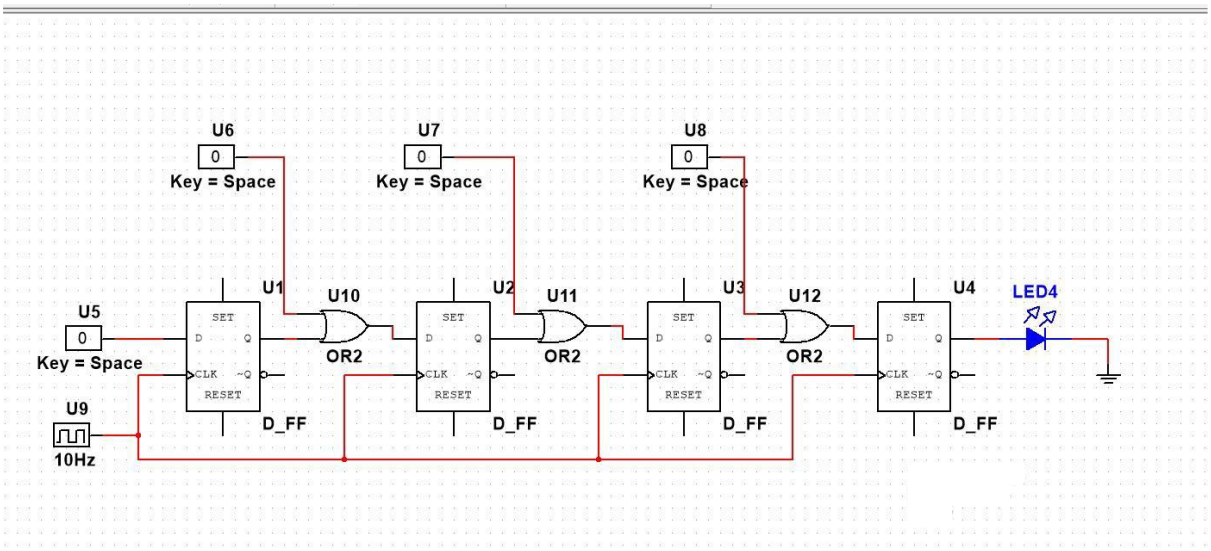


Truth Table

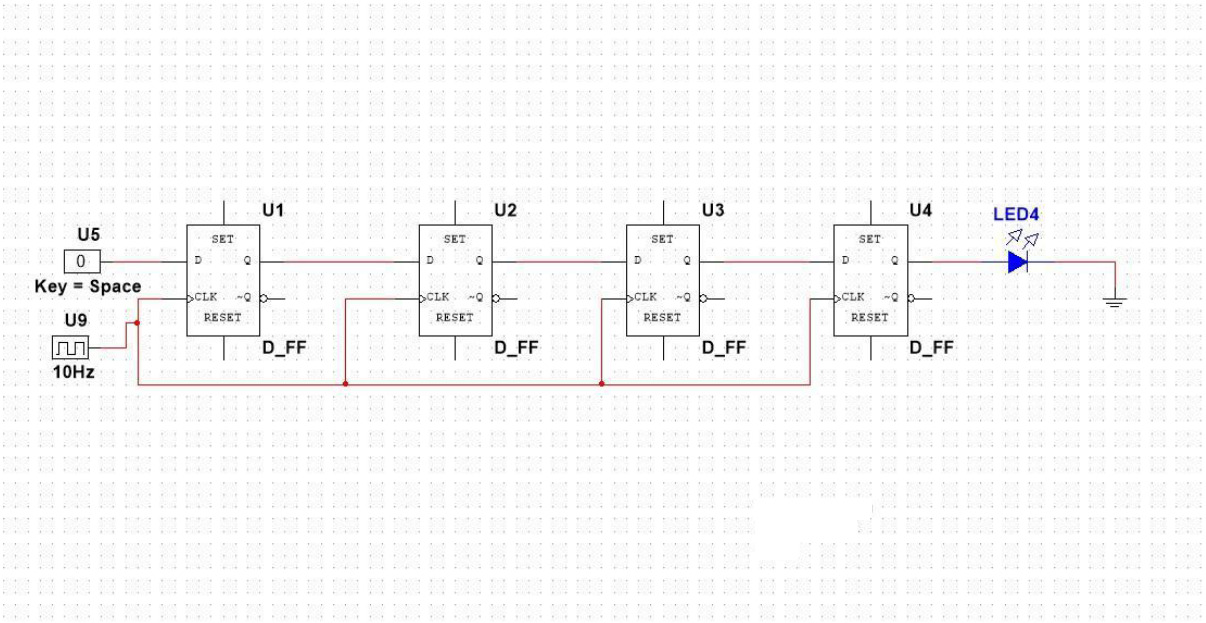
CLK ↑	D INPUT	Q0	Q1	Q2	Q3
↑	1	1	0	0	0
	0	0	1	0	0
↑	1	1	0	1	0
↑	0	0	1	0	1

Multisim Circuits:

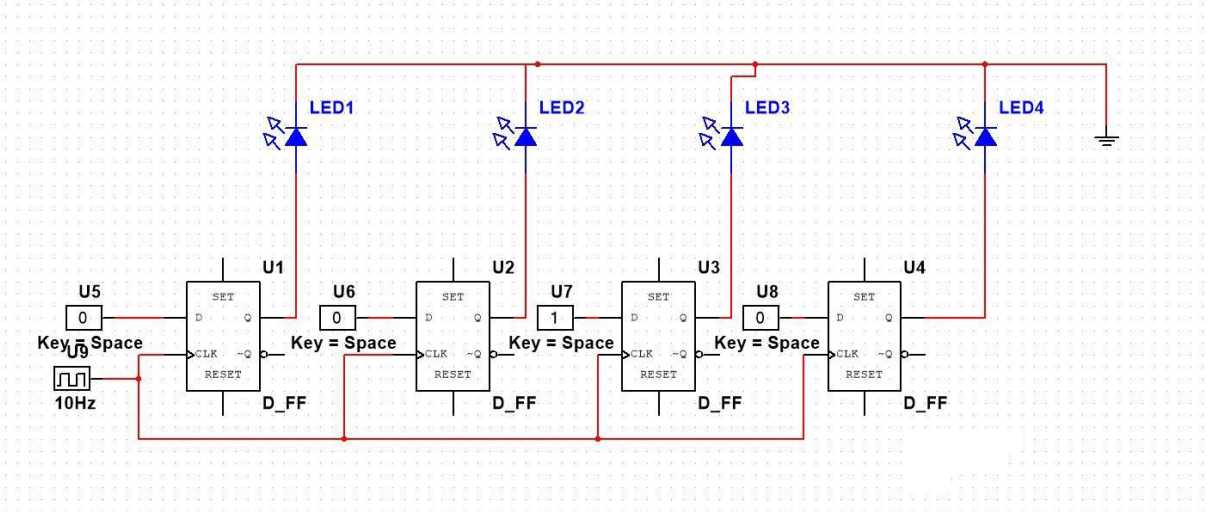
PISO:


U6
0
Key = Space
U7
0
Key = Space
U8
0
Key = Space
U5
0
Key = Space
U9
10Hz
U1
U2
U3
U4
U10
U11
U12
OR2
OR2
OR2
SET
D
Q
CLK
~Q
RESET
D_FF
LED4

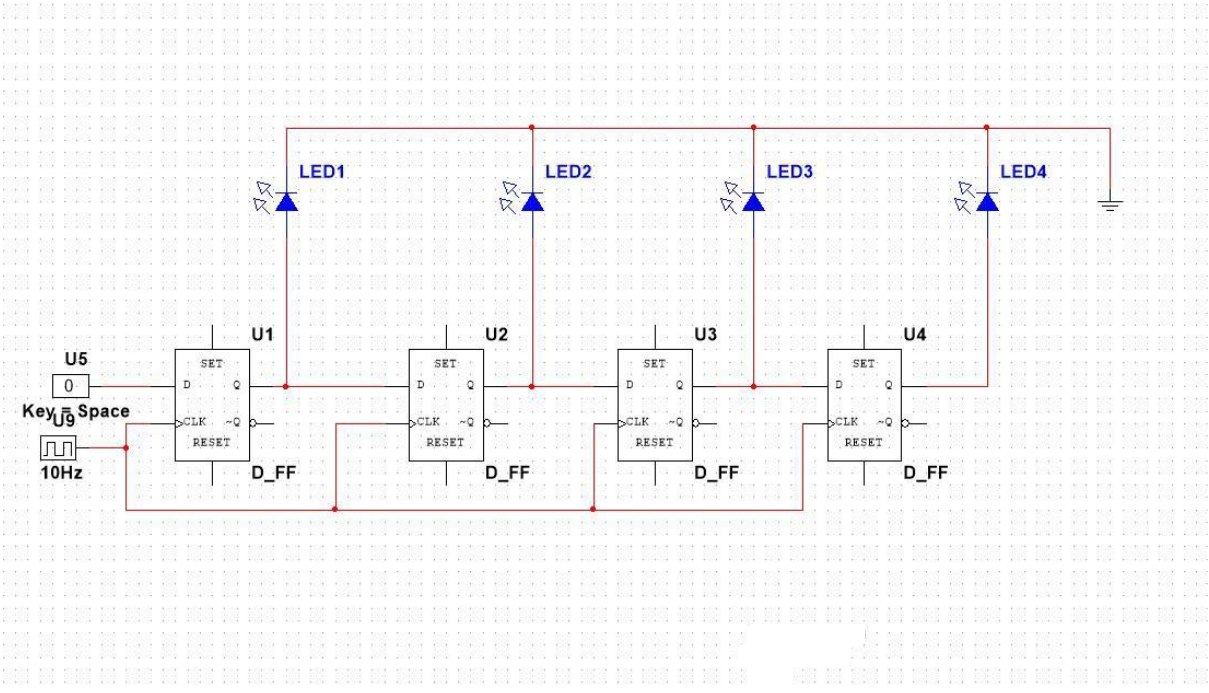
SISO:


U5
0
Key = Space
U9
10Hz
U1
U2
U3
U4
SET
D
Q
CLK
~Q
RESET
D_FF
LED4

PIPO:



SIPO:



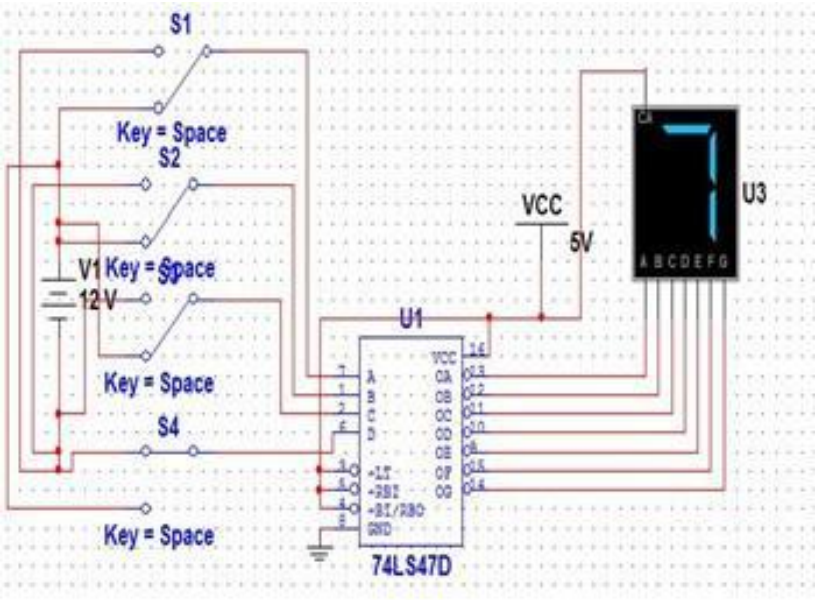
Conclusion:

Successfully built Parallel-in-Serial-out(PISO), Parallel-in-Parallel-out(PIPO), Serial-in-Serial-out(SISO), Serial-in-Parallel-out(SIPO) with the help of the truth tables and multisim(app).

Experiment No. - 10

AIM: Verify the truth table of decoder driver 7447.Hence operates a 7 segment LED display Implement a circuit and verify its operation that requires power-supply, inputs and outputs.

APPARATUS REQUIRED: - Logic Trainer kit, connecting wires, bread board, IC 7447.



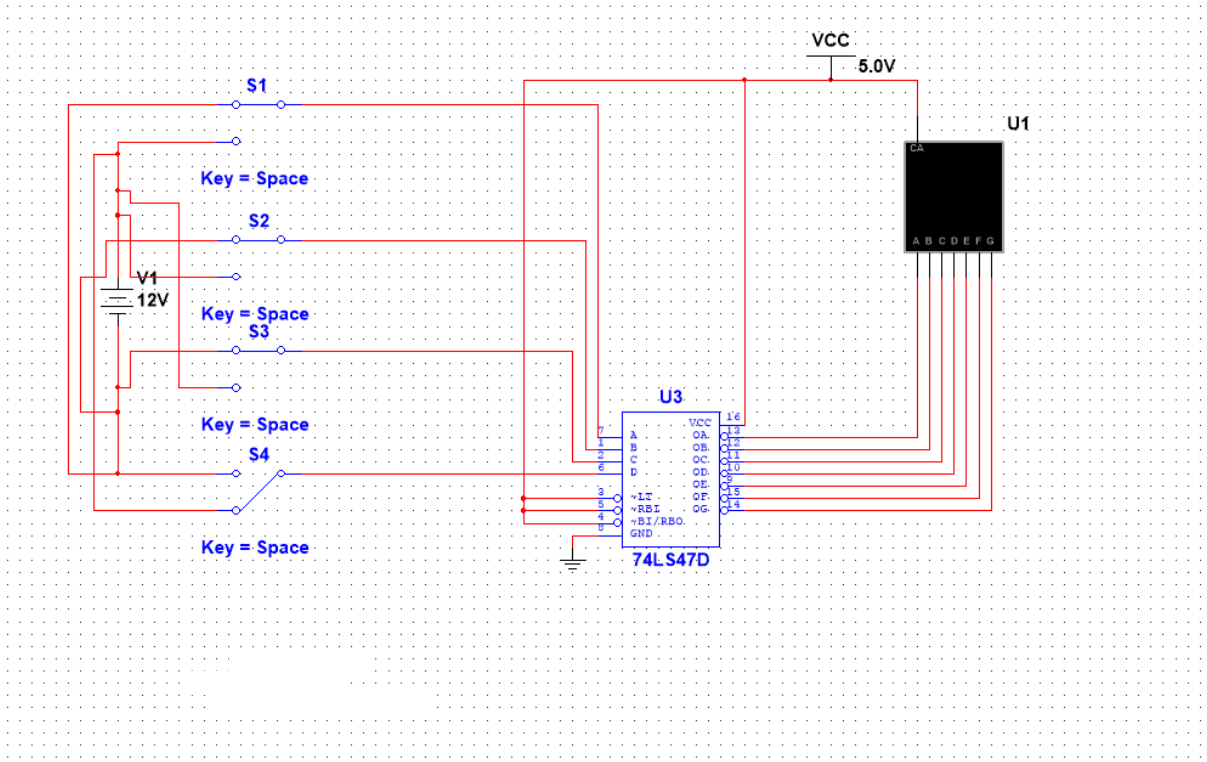
Truth Table

Output observed on seven segment display

Input	Output	Display Pattern
0000	1111110	0
0001	0110000	1
0010	1101101	2
0011	1111001	3
0100	0110011	4
0101	1011011	5
0110	1011111	6
0111	1110000	7
1000	1111111	8
1001	1111011	9

Experiment No. - 10

Multisim Circuit ->>



Conclusion ->> Hence we have Successfully designed the circuit of 7 segment LED Display from truth table on Ni Multisim.