EXPERIMENT NO:-1

- **AIM:** TO STUDY AND VERIFY TRUTH TABLES OF VARIOUS LOGIC GATES.

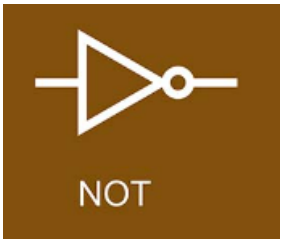
- **APPARATUS:** Logic trainer, IC 7402, IC 7400, IC 7408, IC 7432, IC 74266, IC 7486, IC 7404, Connecting wires, LEDs, Multimeter.

- **THEORY:**

A gate is a logic circuit that has one or more outputs. The outputs of the gate will depend upon the set of input conditions. The digital signal has two states LOW (0) and HIGH (1). Using gates we can implement variety of logic circuit that performs a particular task. For an example we can implement various arithmetic, logical and control units depending upon our requirement. Various types of gates are described below.

[1] NOT Gate: This gate has one input and one output. This gate inverts input at the output. When input is LOW output is HIGH and vice versa.

SYMBOL:



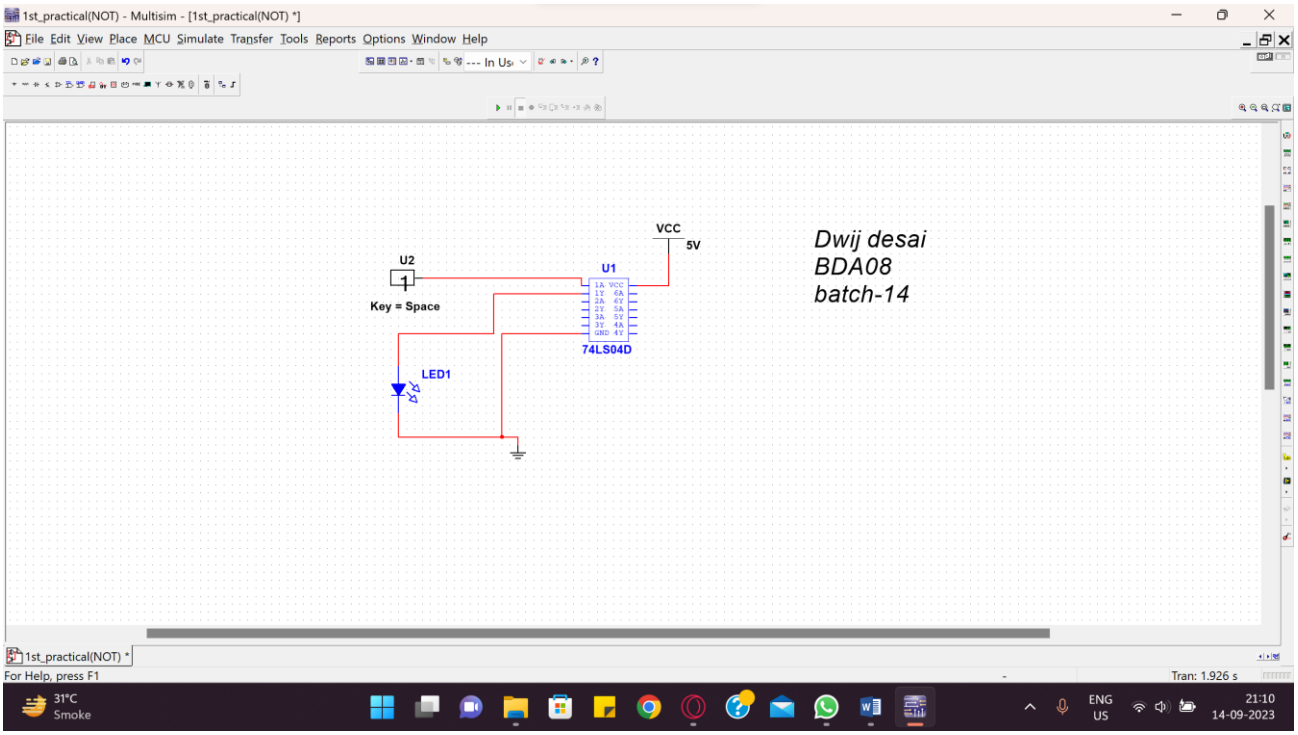
BOOLEAN EXPRESSION:

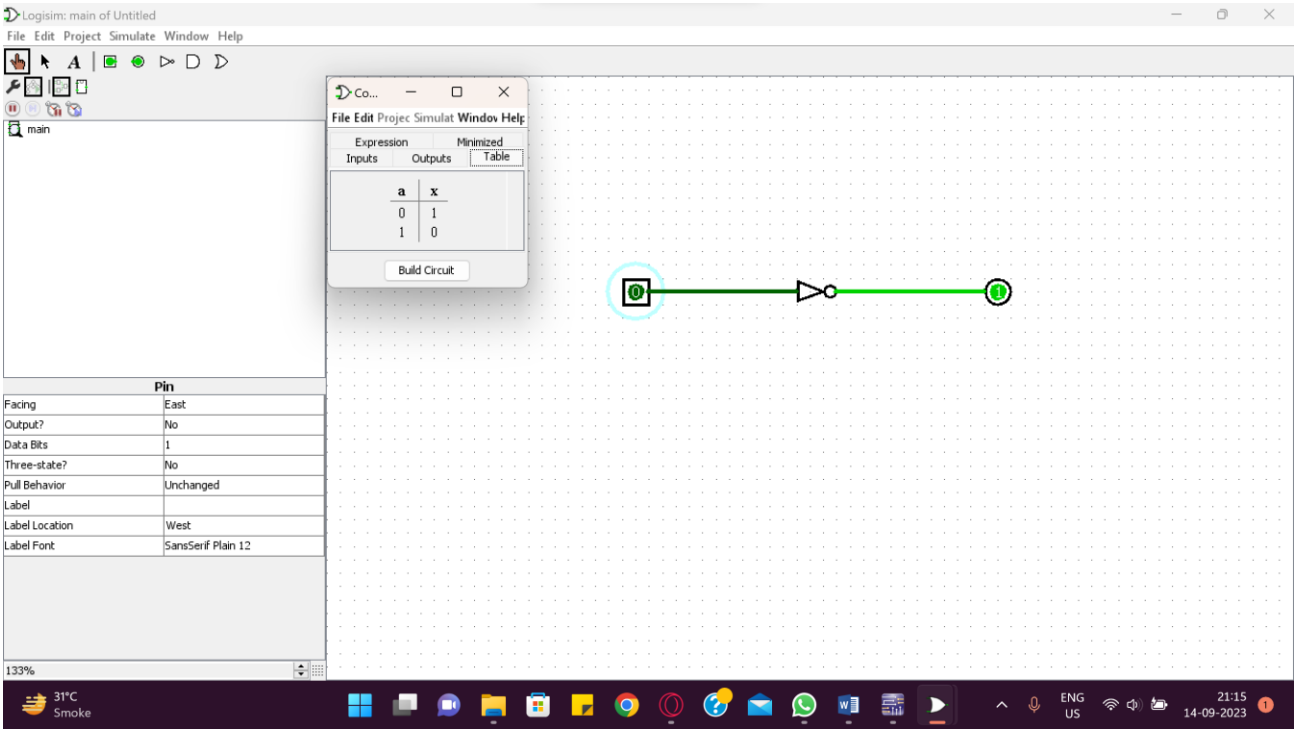
$$Y = \bar{A}$$

TRUTH TABLE:

Table 1: NOT GATE

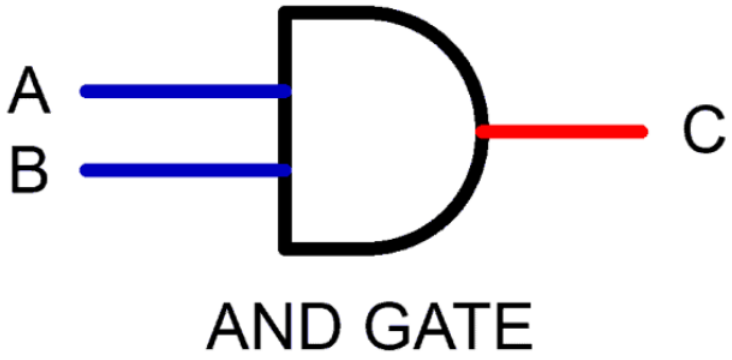
Input	Output	Voltages
A	Y =	(V)
0	1	
1	0	





[2]AND Gate:This gate has two or more inputs and one output. Output of AND gate will go HIGH when all inputs are HIGH, otherwise output will remain LOW.

SYMBOL



AND GATE

BOOLEAN EXPRESSION:

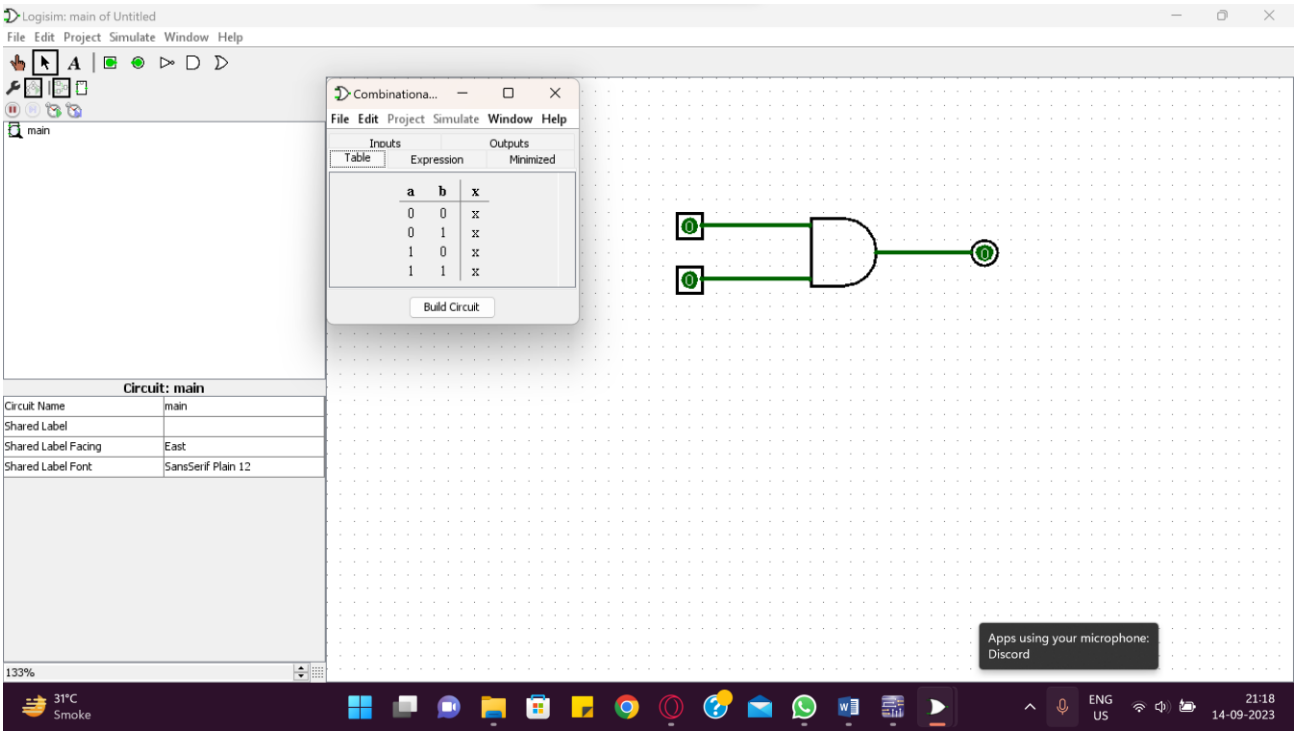
Y =

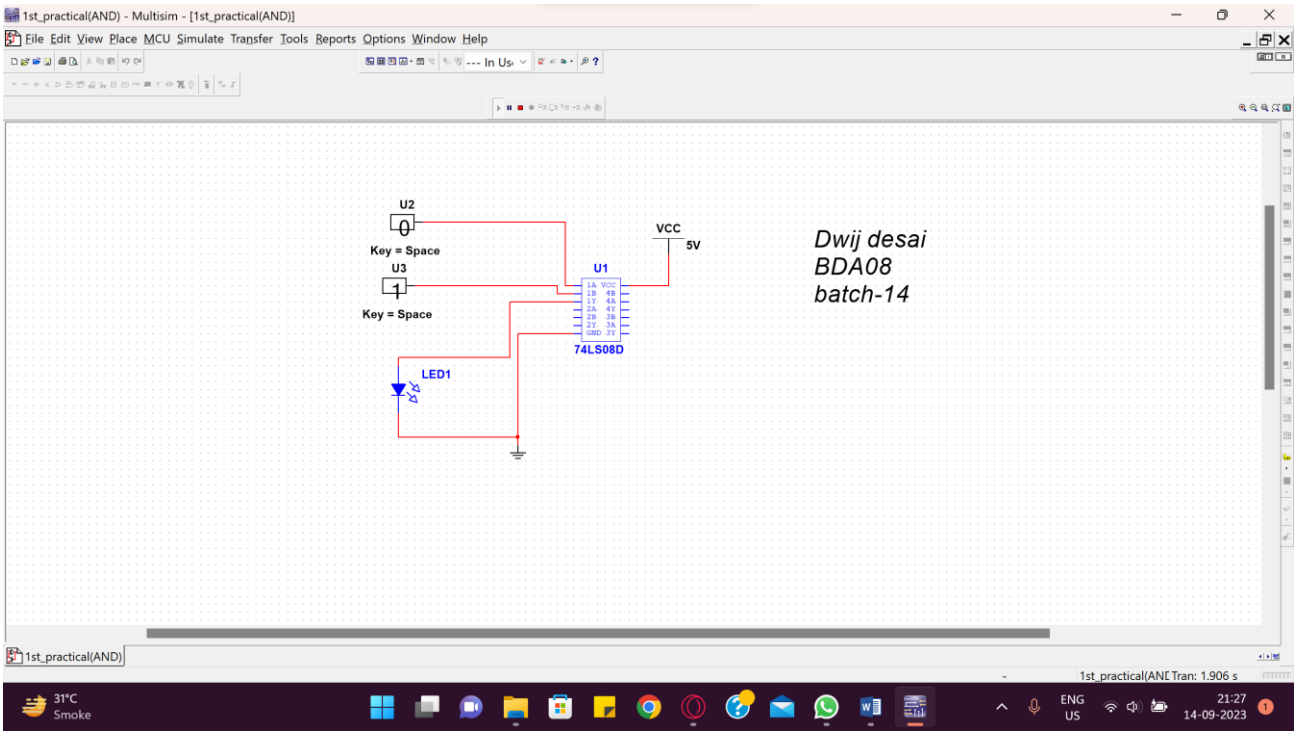
$A \cdot B$

TRUTH TABLE:

Table 2 : AND GATE

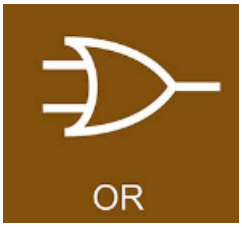
Inputs		Output	Voltages
A	B	Y=	(V)
0	0	0	
0	1	0	
1	0	0	
0	0	1	





[3] OR Gate: This gate has two or more inputs and one output. Output of OR gate will go HIGH when any of the input is HIGH. Output is LOW when all inputs are LOW.

SYMBOL:



BOOLEAN EXPRESSION:

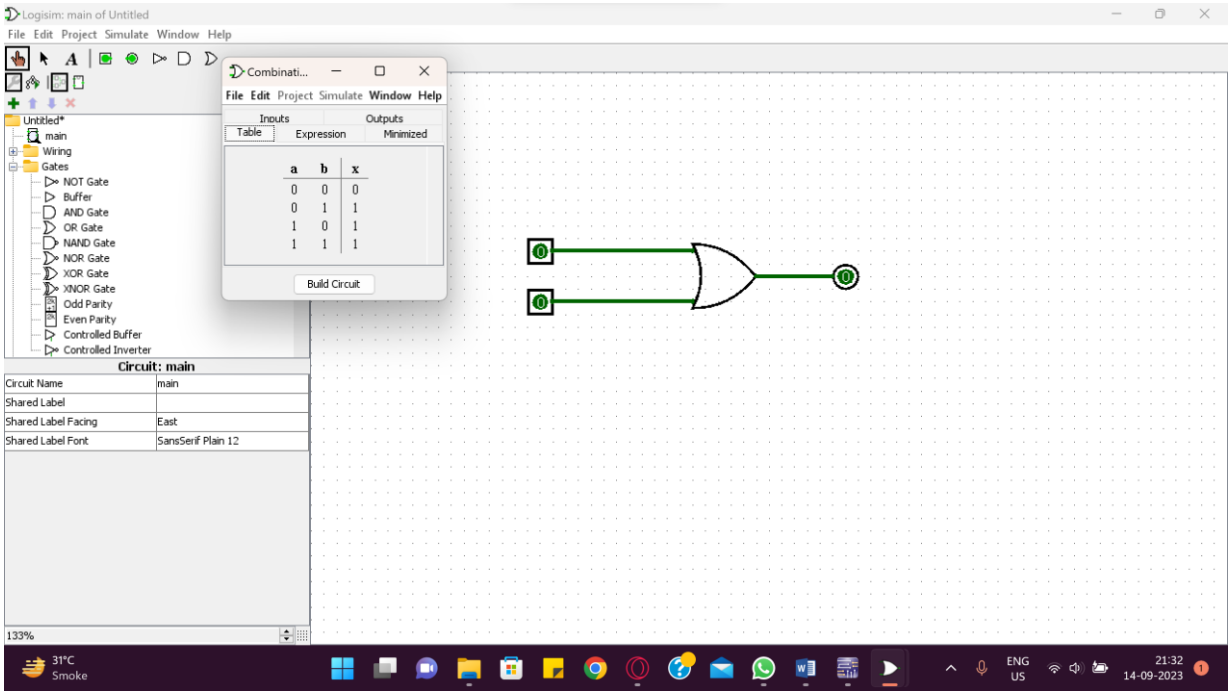
Y =

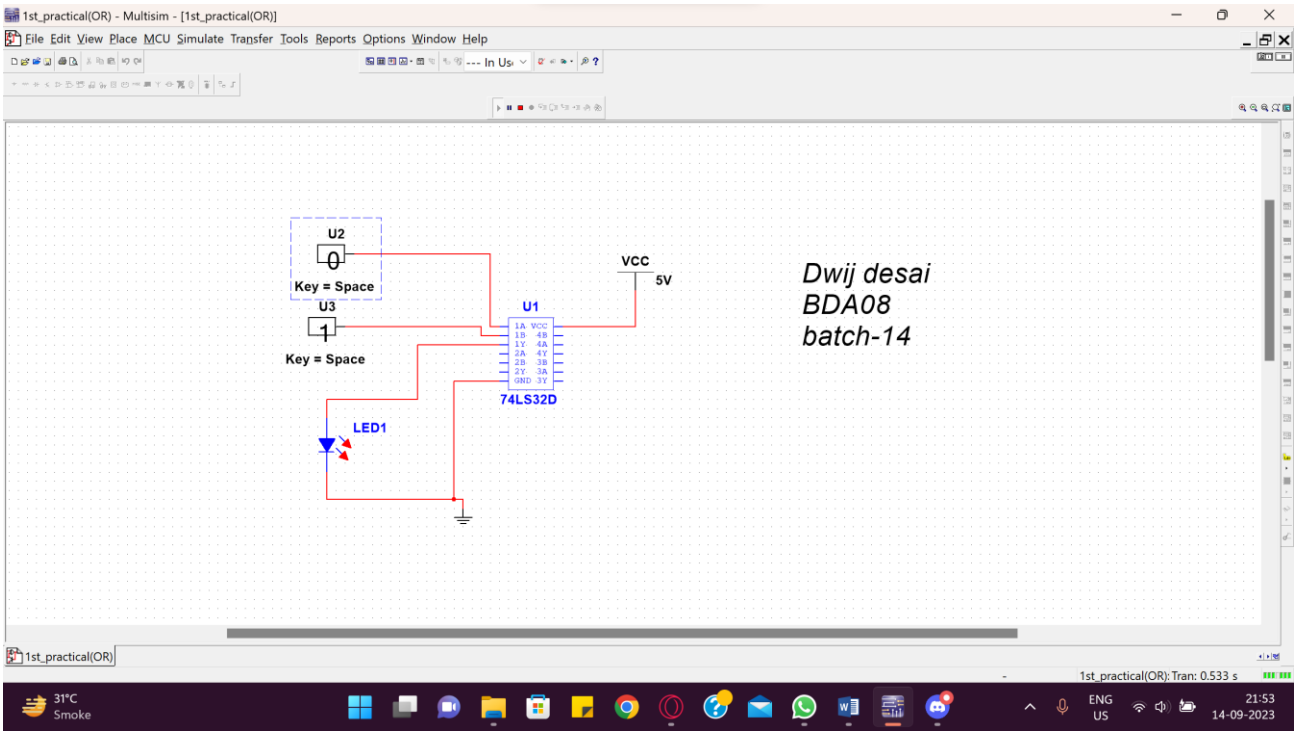
A + B

TRUTH TABLE:

Table 3: OR GATE

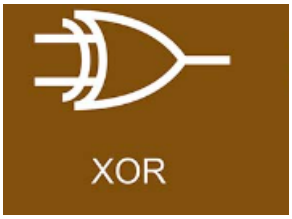
Inputs		Output	Voltages
A	B	Y=	(V)
0	0	0	
0	1	1	
1	0	1	
1	1	1	




1st_practical(OR) - Multisim - [1st_practical(OR)]
File Edit View Place MCU Simulate Transfer Tools Reports Options Window Help
U2
0
Key = Space
U3
1
Key = Space
U1
74LS32D
VCC
5V
LED1
Dwij desai
BDA08
batch-14
1st_practical(OR)
1st_practical(OR): Tran: 0.533 s
31°C
Smoke
ENG
US
21:53
14-09-2023

[4] XOR Gate: This gate has two inputs and one output. Output will go HIGH when all inputs are not of the same logic level (i.e. all inputs are not LOW or not HIGH at a time).

SYMBOL:



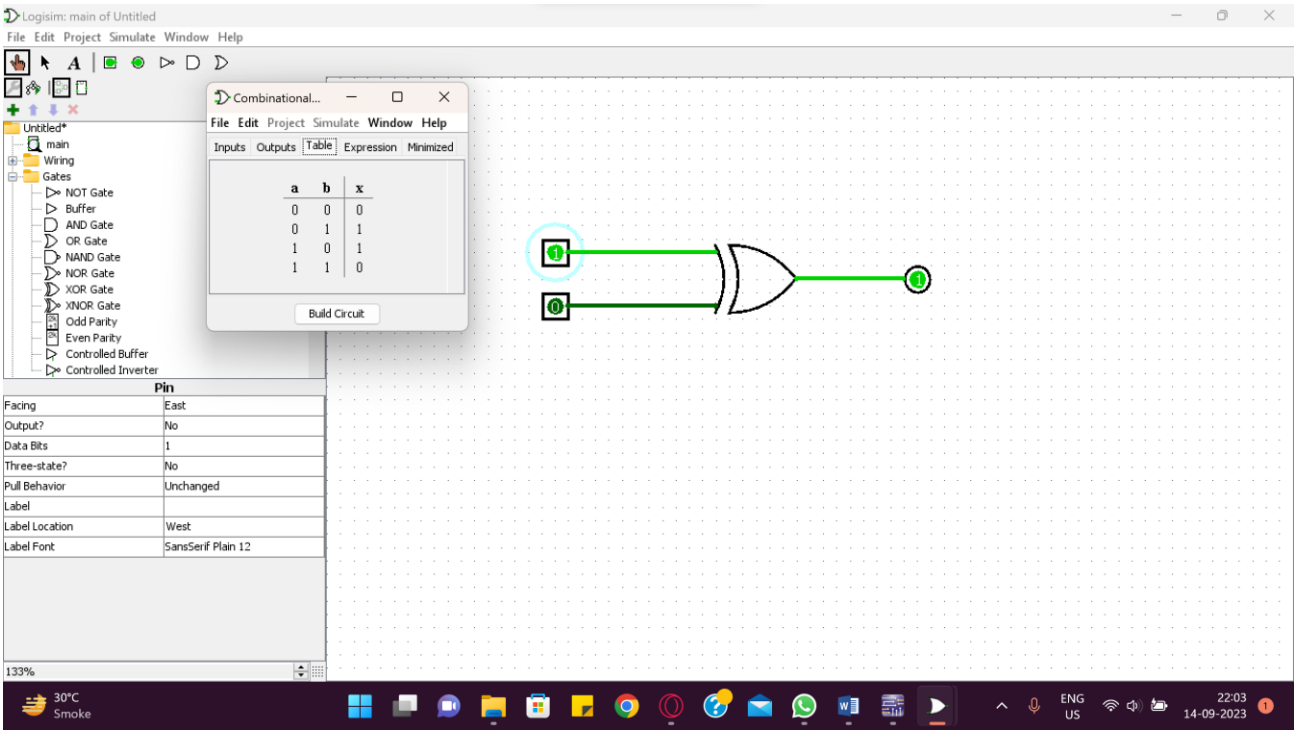
BOOLEAN EXPRESSION:

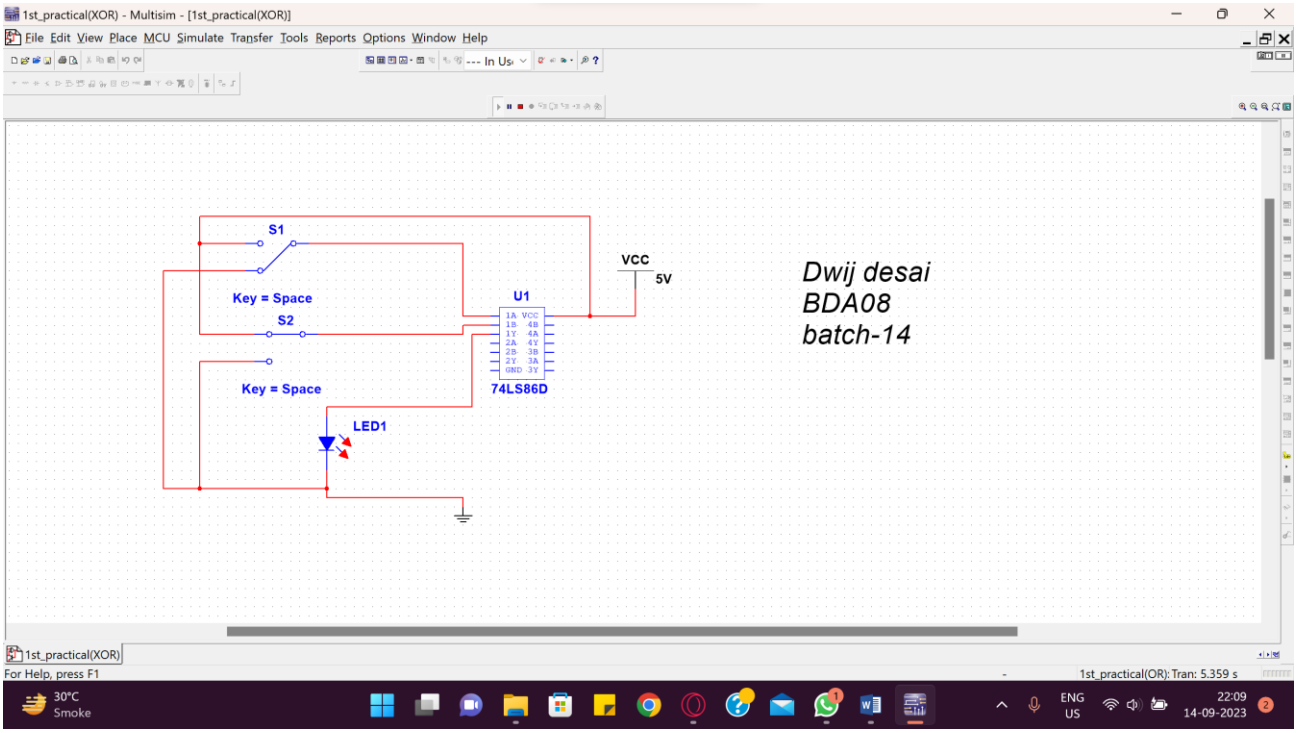
Y=

TRUTH TABLE:

Table 4: XOR GATE

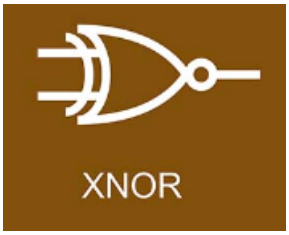
Inputs		Output	Voltages
A	B	Y=	(V)
0	0	0	
0	1	1	
1	0	1	
1	1	0	





[5] XNOR Gate: This gate has two inputs and one output. Output will go HIGH when all inputs are of same logic level (i.e. all inputs are LOW or HIGH at a time).

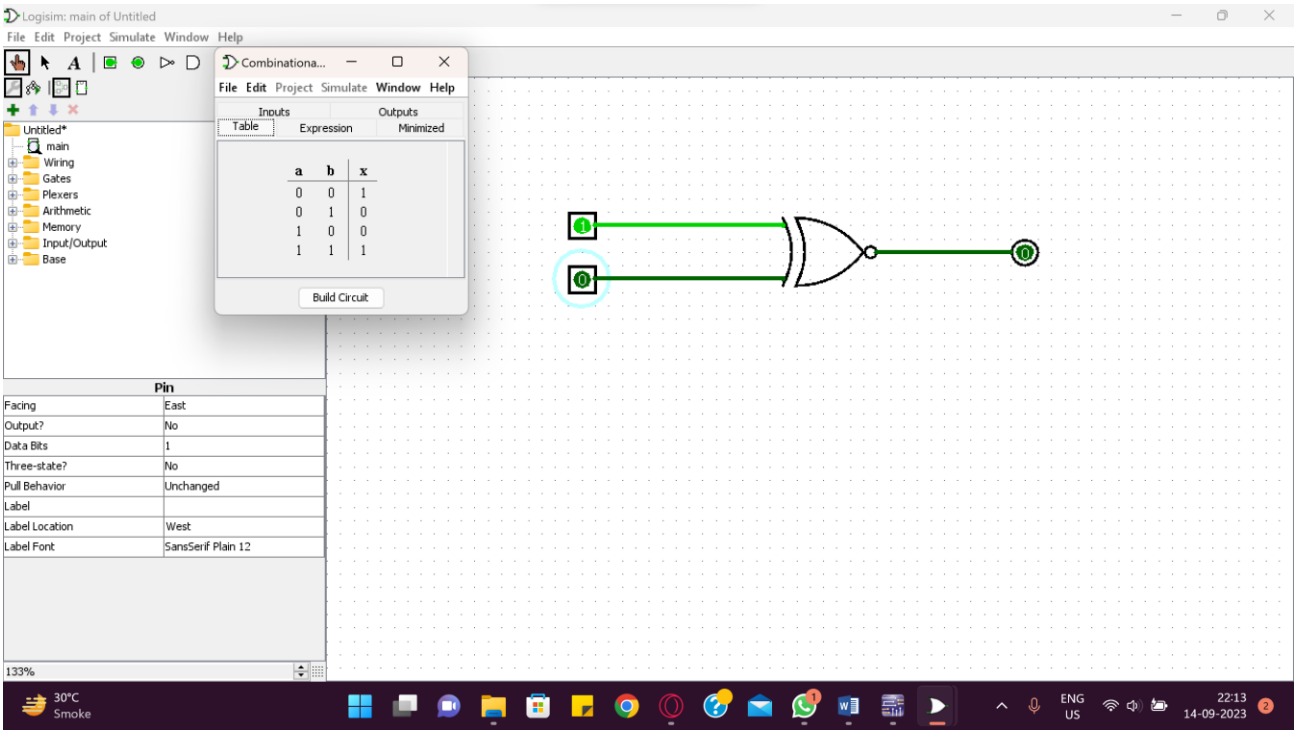
SYMBOL:

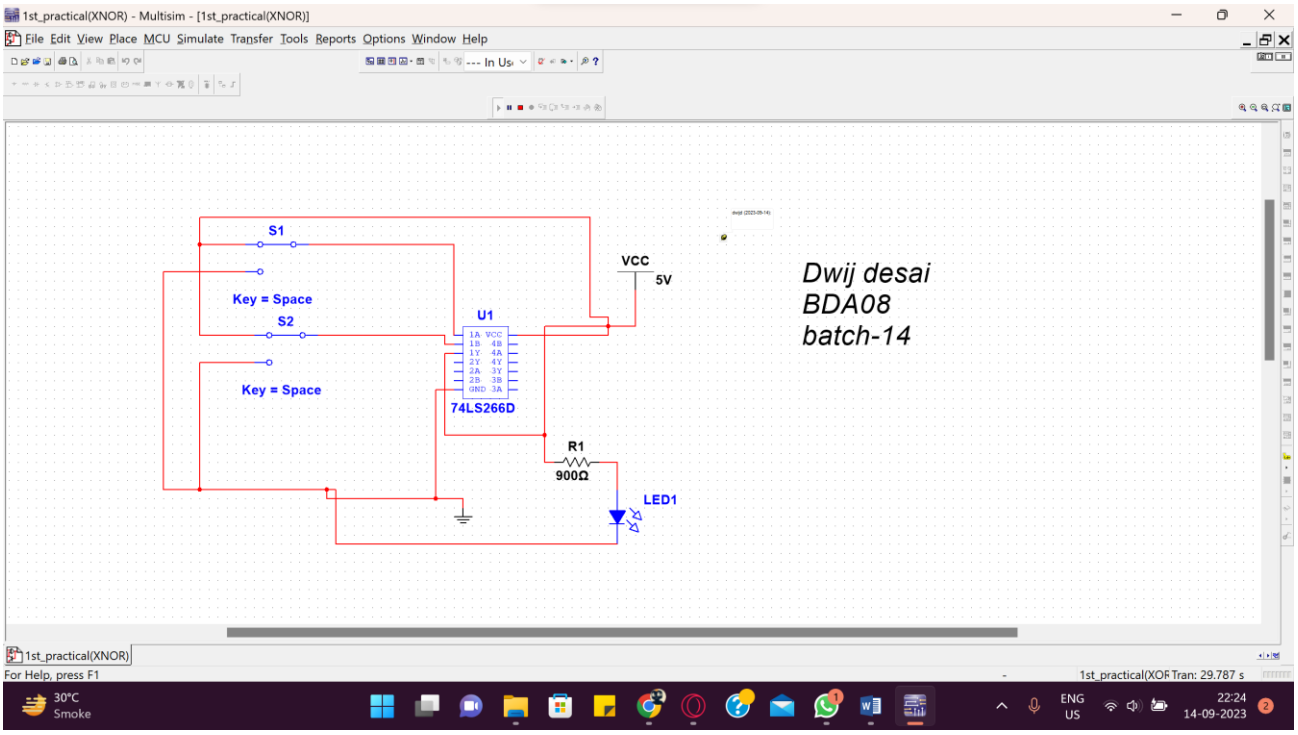


BOOLEAN EXPRESSION: Y =

TRUTH TABLE:

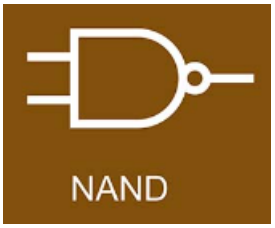
Inputs		Output	Voltages
A	B	Y=	(V)
0	0	1	
0	1	0	
1	0	0	
1	1	1	


Logisim: main of Untitled
File Edit Project Simulate Window Help
Combinationa...
File Edit Project Simulate Window Help
Inputs
Outputs
Table
Expression
Minimized
a b x
0 0 1
0 1 0
1 0 0
1 1 1
Build Circuit
Untitled*
main
Wiring
Gates
Plexers
Arithmetic
Memory
Input/Output
Base
Pin
Facing East
Output? No
Data Bits 1
Three-state? No
Pull Behavior Unchanged
Label
Label Location West
Label Font SansSerif Plain 12
133%
30°C
Smoke
ENG
US
22:13
14-09-2023


1st_practical(XNOR) - Multisim - [1st_practical(XNOR)]
File Edit View Place MCU Simulate Transfer Tools Reports Options Window Help
S1
Key = Space
S2
Key = Space
U1
74LS266D
VCC
5V
R1
900Ω
LED1
Dwij desai
BDA08
batch-14
1st_practical(XNOR)
For Help, press F1
1st_practical(XOR Tran: 29.787 s
30°C
Smoke
ENG
US
22:24
14-09-2023

[6] NAND Gate: If we put one inverter at the output of AND logic gate will be NAND gate.

SYMBOL:



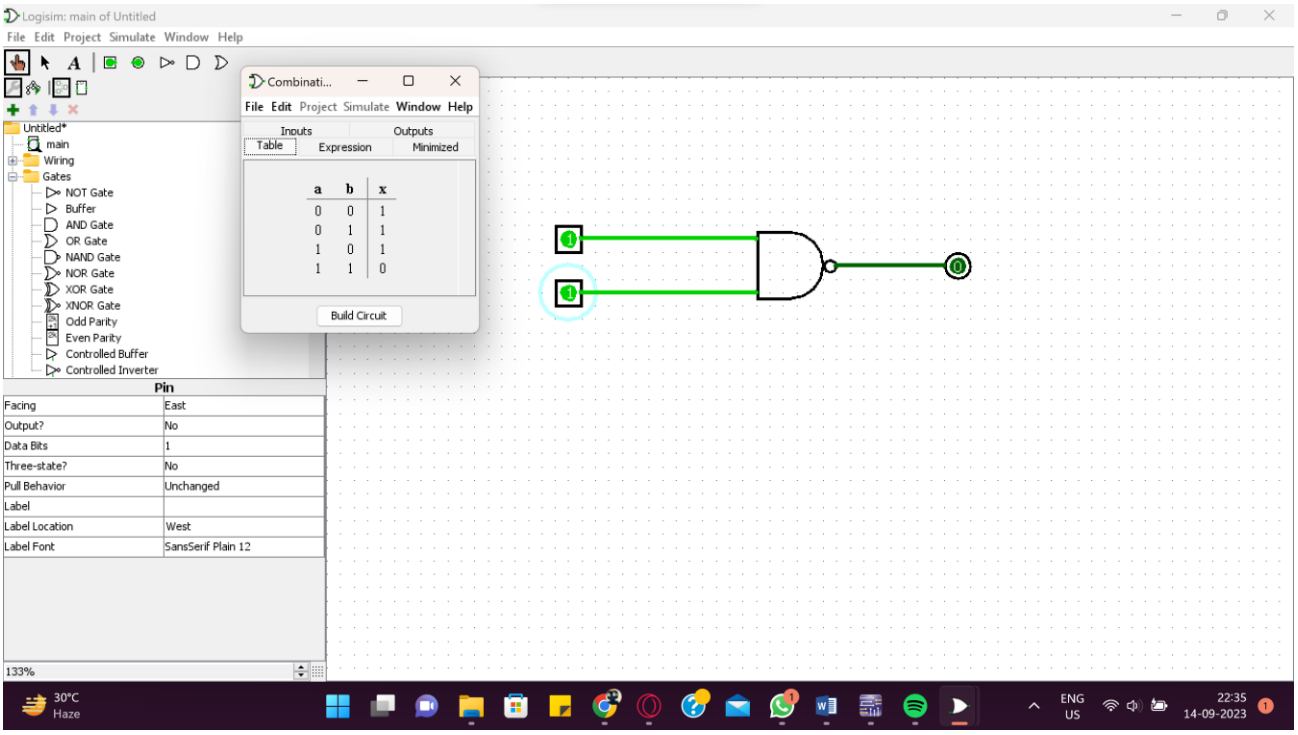
BOOLEAN EXPRESSION: Y =

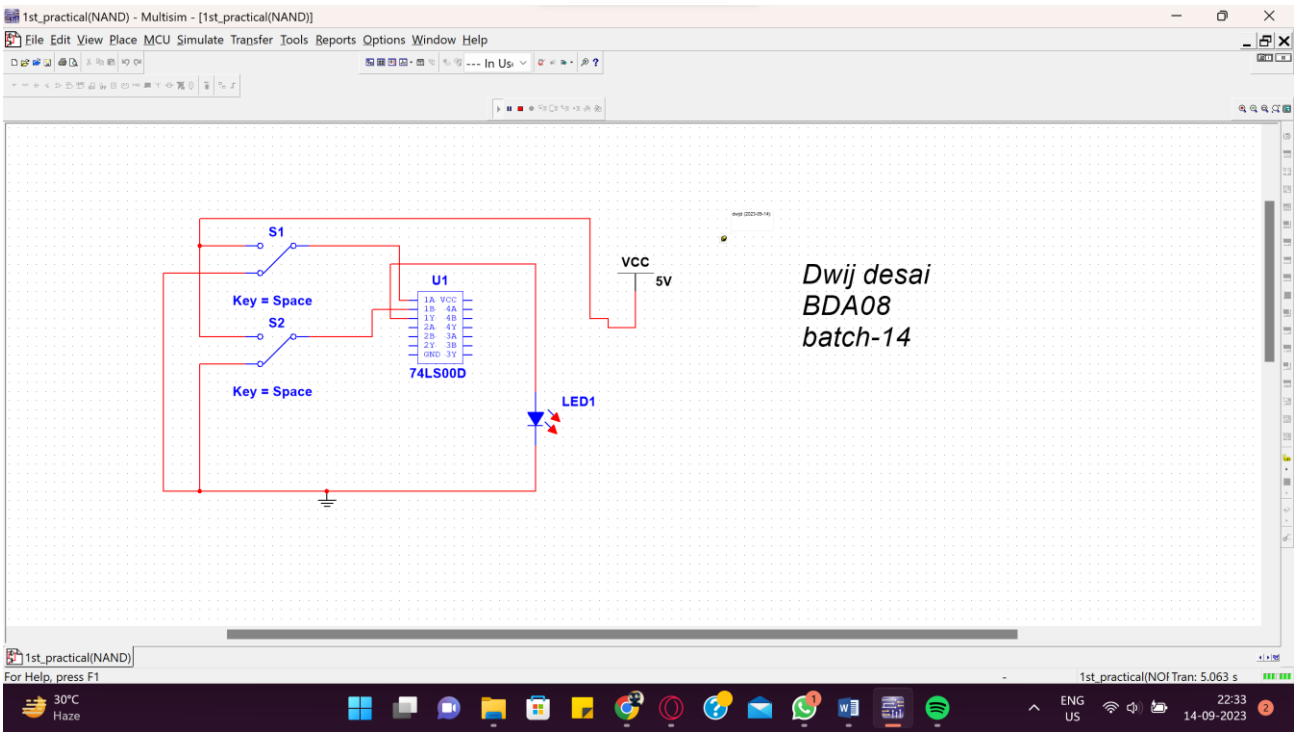
$A \cdot B$

TRUTH TABLE:

Table 6: NAND GATE

Inputs		Output	Voltages
A	B	Y=	(V)
0	0	1	
0	1	1	
1	0	1	
1	1	0	





➢ PROCEDURE:

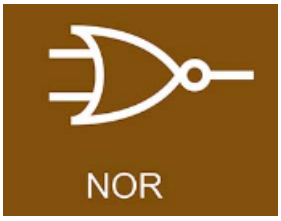
(1) Select appropriate IC for each logic gate

(2) Make the connections according to the requirements.

(3) Make sure the connections of Vcc and grounds are at their respective pins

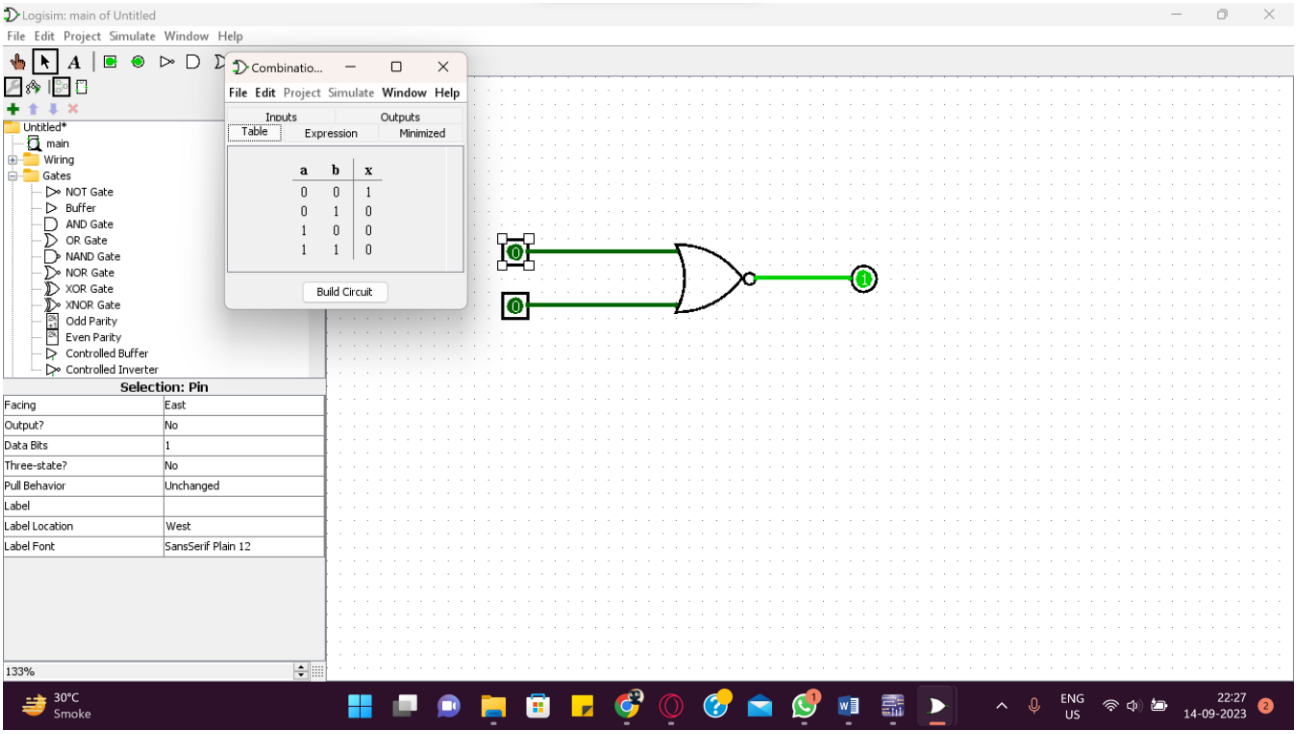
(4) Switch on the power and apply sequence of inputs and observe outputs.

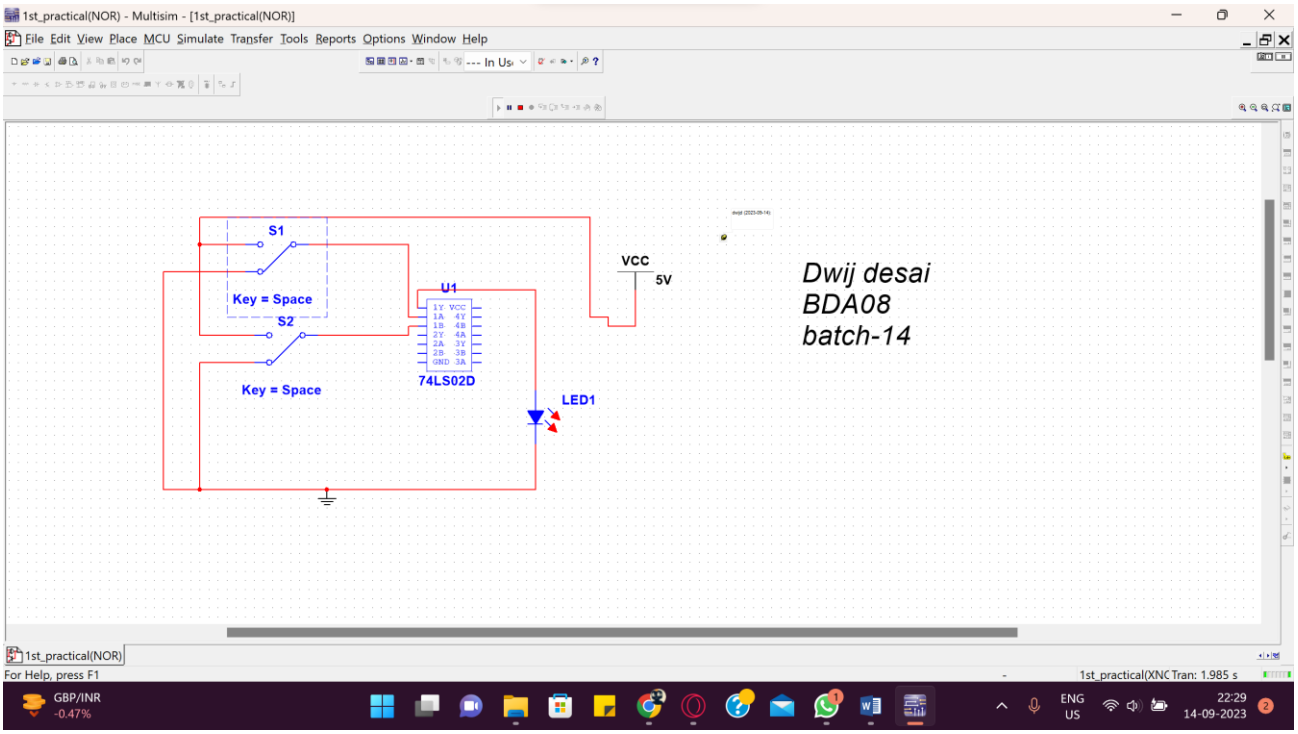
Table 7: NOR GATE



$$Y = \overline{A + B}$$

Inputs		Output	Voltages
A	B	Y=	(V)
0	0	1	
0	1	0	
1	0	0	
1	1	0	





 PROCEDURE:

(1) Select appropriate IC for each logic gate

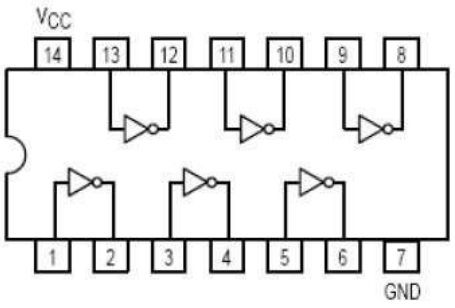
(2) Make the connections according to the requirements.

(3) Make sure the connections of Vcc and grounds are at their respective pins

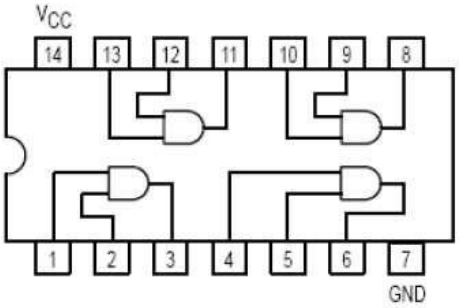
(4) Switch on the power and apply sequence of inputs and observe outputs.

➢ CONCLUSION:

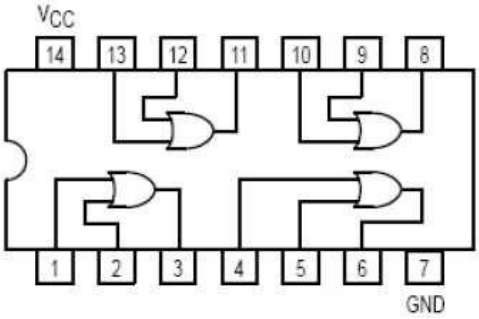
Inverter Gate (NOT Gate) → 7404



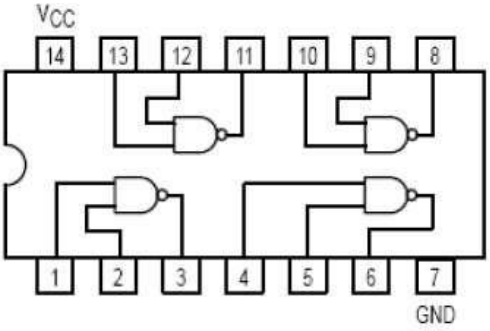
2-Input AND Gate → 7408



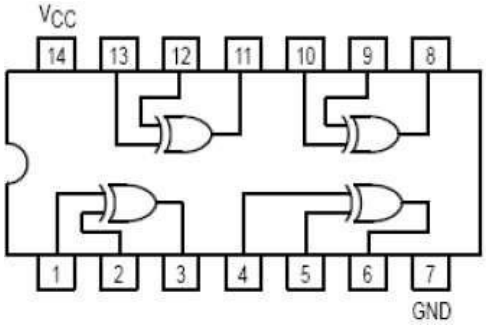
2-Input OR Gate → 7432



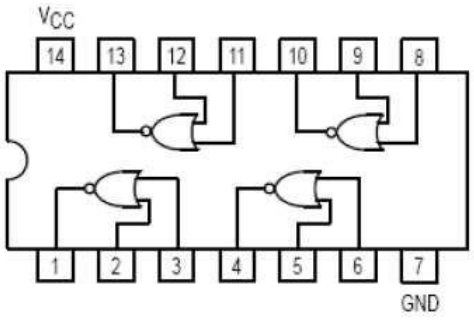
2-Input NAND Gate → 7400



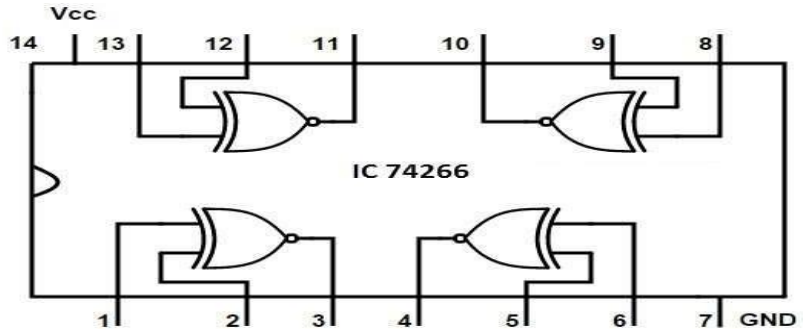
2-Input EX-OR Gate → 7486



2-Ir

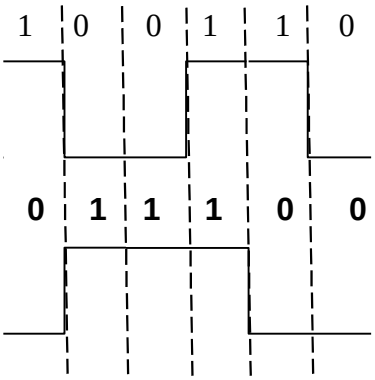


2- Input EX-NOR Ic: 74266



- **EXERCISE :**

Draw output waveform of each gate for the given input signals.



EXPERIMENT NO:-2

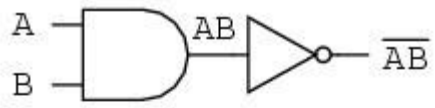
- **AIM*:*** To verify the De'Morgan's Theorems.

- **APPARATUS :**

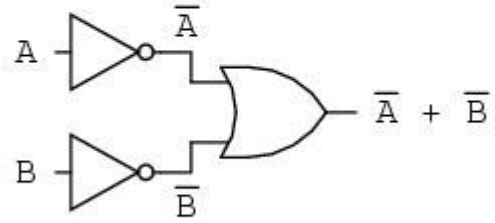
  - IC 7400 : Quad - Dual input NAND Gate
  - IC 7402 : Quad – Dual input NOR Gate
  - IC 7408 : Quad – Dual input AND Gate
  - IC 7432 : Quad – Dual input OR gate
  - IC 7404 : Hex – Not gate

- **THEORY:**

  Inverting all inputs to a gate reverses that gate's essential function from AND to OR, or vice versa, and also inverts the output. So, an OR gate with all inputs inverted (a Negative-OR gate) behaves the same as a NAND gate, and an AND gate with all inputs inverted (a Negative-AND gate) behaves the same as a NOR gate. DeMorgan's theorems state the same equivalence in "backward" form: that inverting the output of any gate results in the same function as the opposite type of gate (AND vs. OR) with inverted inputs:

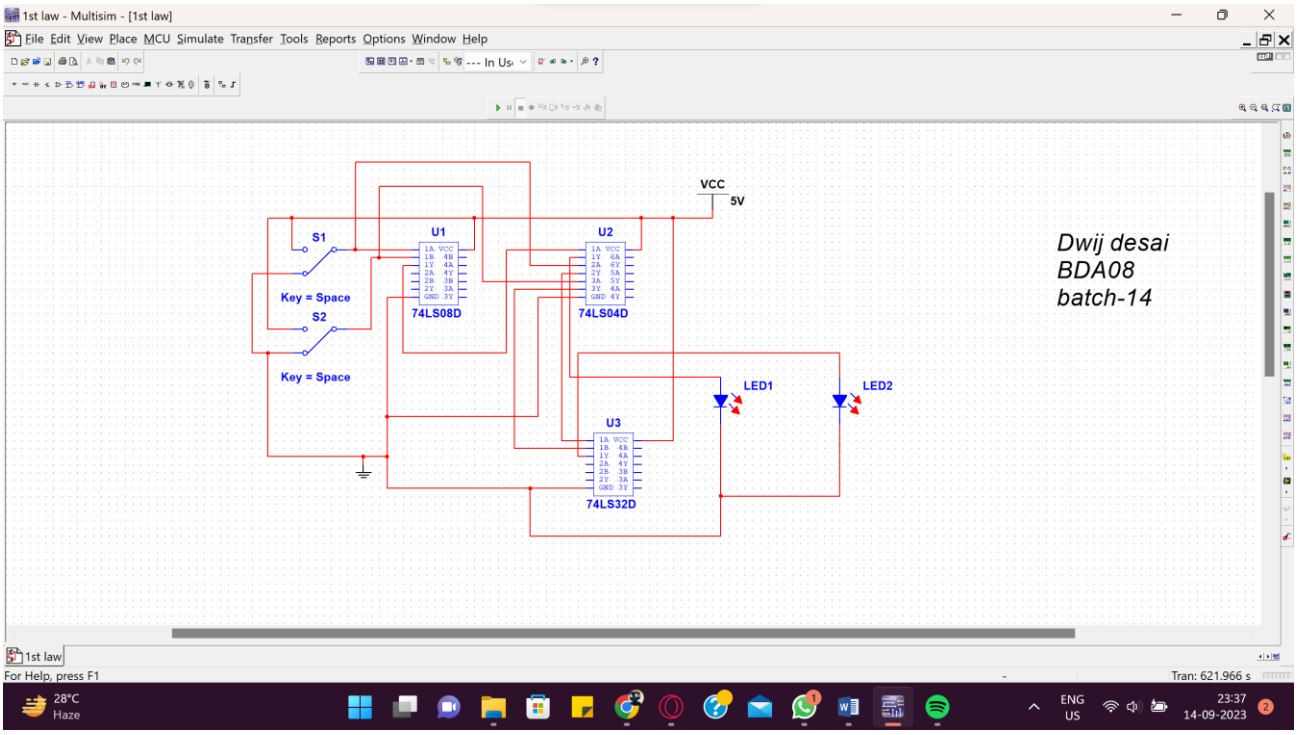


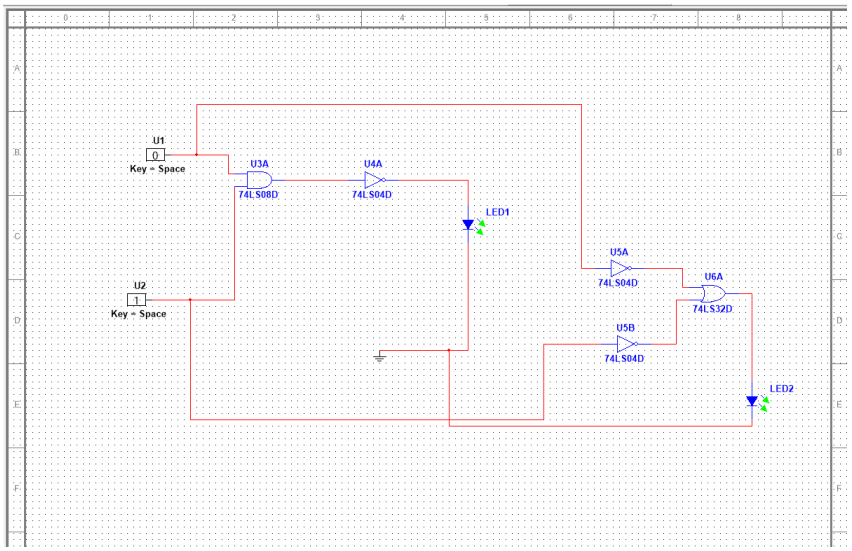
. . . is equivalent to . . .

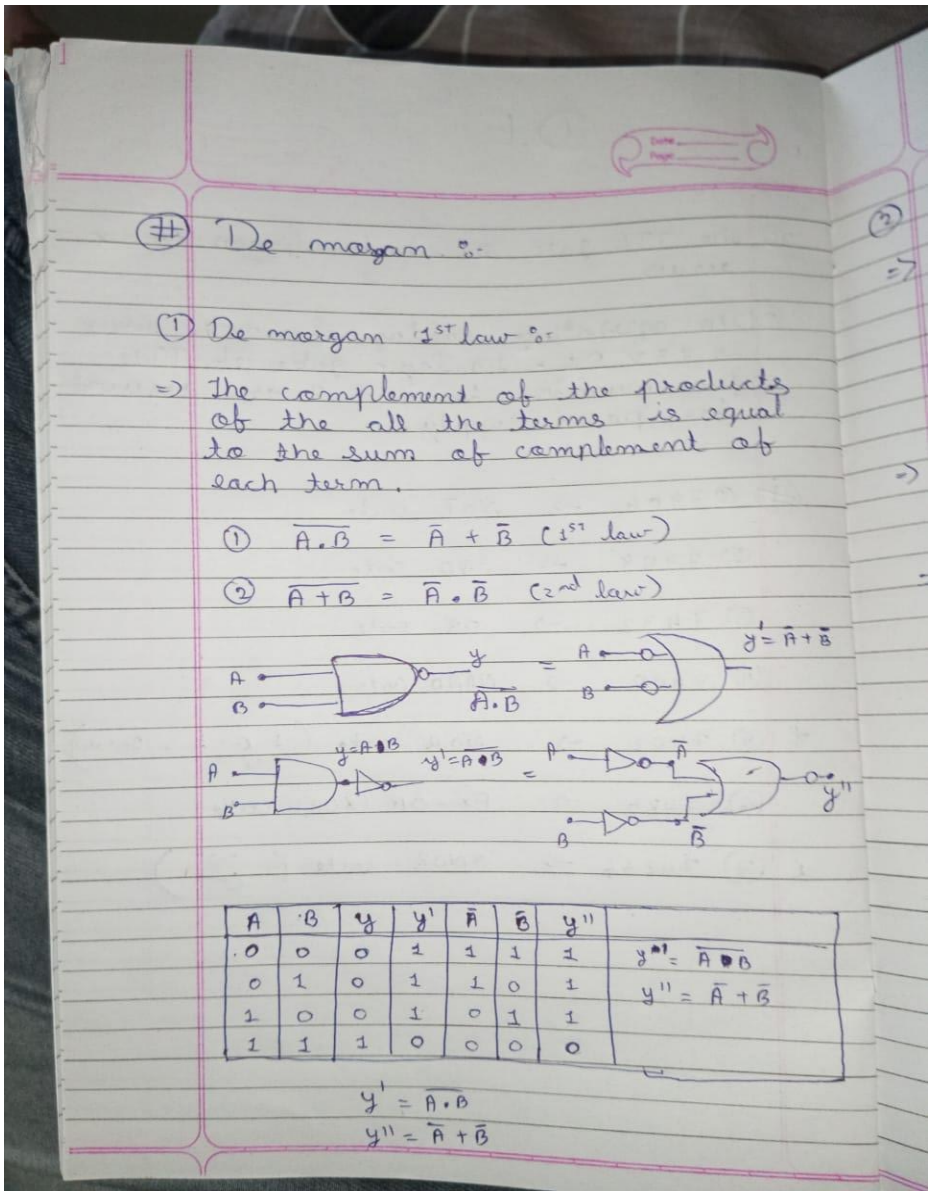


$$\overline{AB} = \overline{A} + \overline{B}$$

[A] Statement and Proof of De'Morgan's First Law**:**







De morgan :-

(1) De morgan 1st law :-

=> The complement of the products of the all the terms is equal to the sum of complement of each term.

(1) $\overline{A \cdot B} = \bar{A} + \bar{B}$ (1st law)

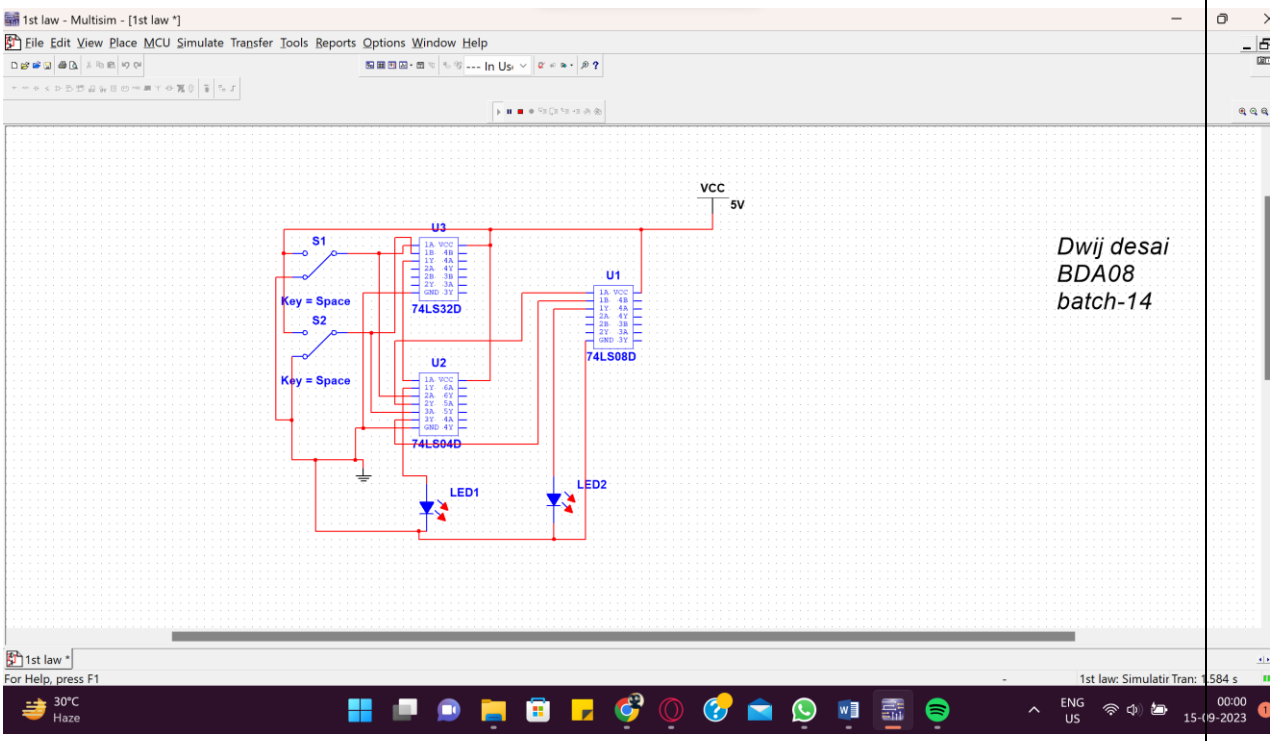
(2) $\overline{A + B} = \bar{A} \cdot \bar{B}$ (2nd law)

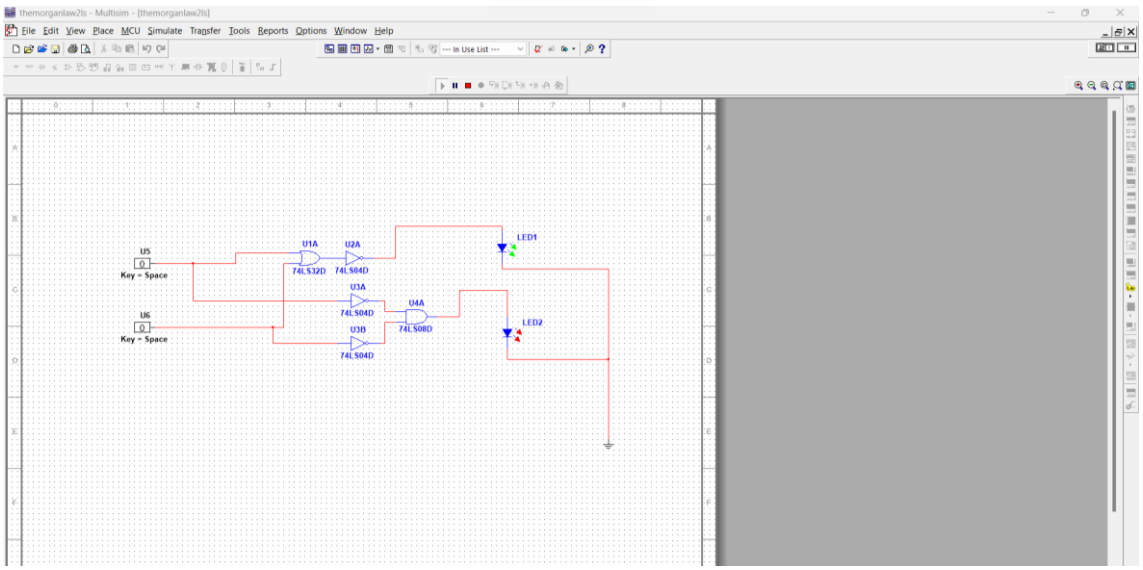
A	B	y	y'	$\bar{A}$	$\bar{B}$	y''
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

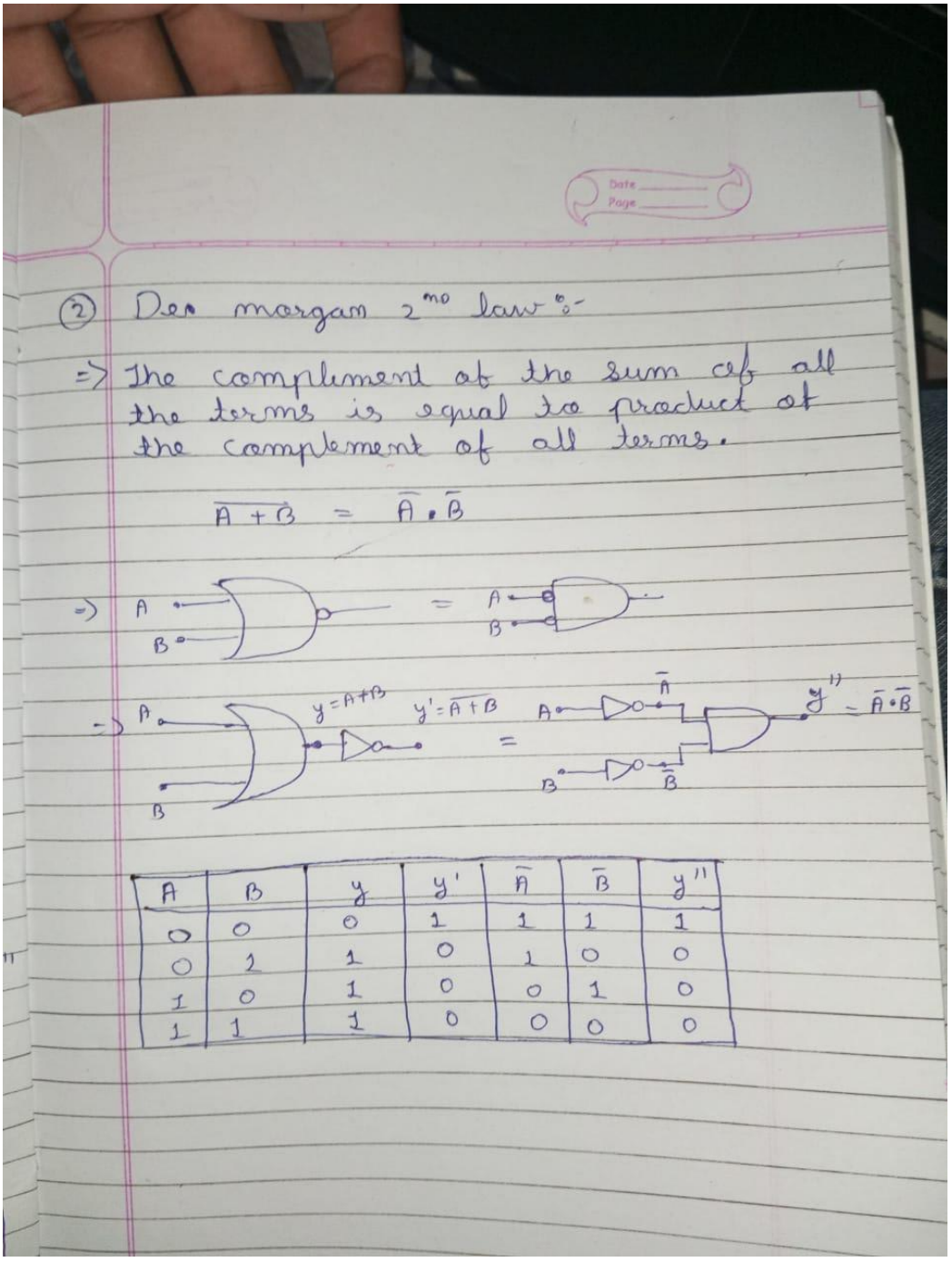
$y' = \overline{A \cdot B}$

$y'' = \bar{A} + \bar{B}$

- **[B]** Statement and proof of De'Morgan's second Law :


1st law - Multisim - [1st law *]
File Edit View Place MCU Simulate Transfer Tools Reports Options Window Help
VCC
5V
U3
S1
Key = Space
S2
Key = Space
74LS32D
U1
74LS08D
U2
74LS04D
LED1
LED2
Dwij desai
BDA08
batch-14
1st law *
For Help, press F1
1st law: Simulatir Tran: 1.584 s
30°C
Haze
ENG
US
00:00
15-09-2023


themorganlaw2ls - Multisim - [themorganlaw2ls]
File Edit View Place MCU Simulate Transfer Tools Reports Options Window Help
U5
Key = Space
U6
Key = Space
U1A
U2A
74LS32D
74LS04D
U3A
74LS04D
U3B
74LS04D
U4A
74LS08D
LED1
LED2



(2) Dem morgam 2nd law :-

=> The complement of the sum of all the terms is equal to product of the complement of all terms.

$$\overline{A+B} = \overline{A} \cdot \overline{B}$$

=>

=>

A	B	y	y'	Ā	B̄	y''
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

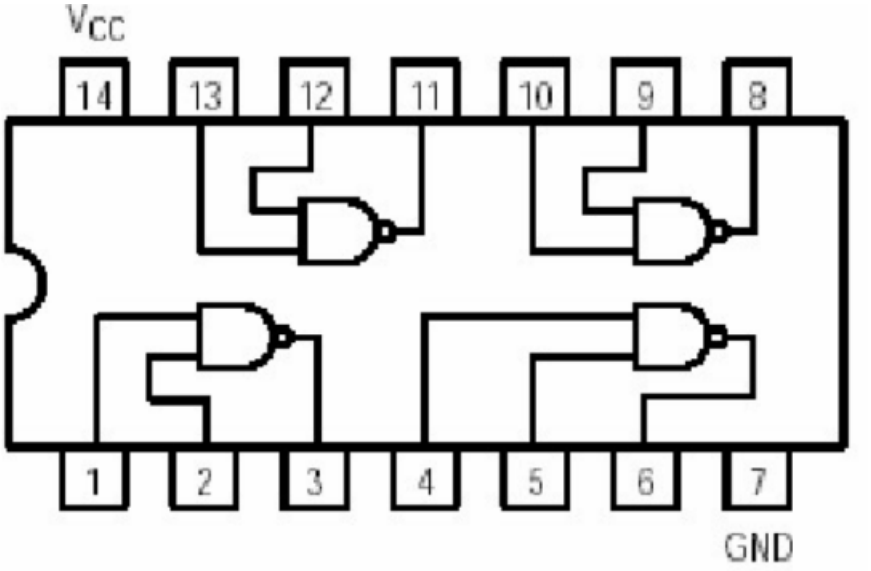
- **PROCEDURE:**

1. Connect the circuit on the bread board using ICs.
2. Switch ON the power supply.
3. Test the truth table of different gates by changing the input levels (i.e. '1' means HIGH & '0' means LOW) and check the level of output voltage.(if LED glows it is at level '1' and if LED doesn't glow output is at level '0').
4. Verify that the De'Morgan's laws are proved.

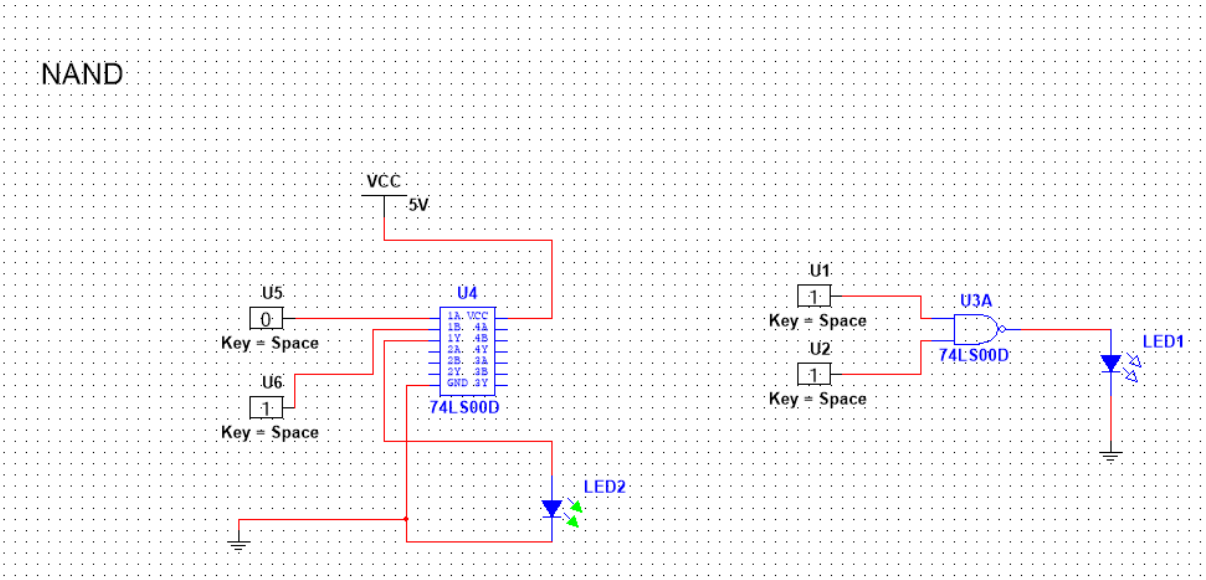
EXPERIMENT NO:-3

 AIM: TO STUDY AND TEST NAND & NOR GATE AS A UNIVERSAL GATE.
 APPARATUS: Bread Board, Connecting wires, LEDs, Resistor
IC's (1) 74LS00 (Quad two I/P NAND gate)
(2) 74LS02 (Quad two I/P NOR gate)
 THEORY:
We know that AND, OR and NOT gates are the basic building blocks of digital computer. They are called the basic gates. Any digital circuit of any complexity can be build using only these three gates. A universal gate is a gate, which alone can be used to build any logic circuit. So to show that the NAND gate and the NOR gate are universal gates. We have to show that the all three basic gates can be realized using only NAND gates or using only NOR gates.
As AOI gates can be implemented by using NAND & NOR gates as shown in circuit diagram, NAND & NOR gates are called as universal gates.
There are number of logic families. The logic gates so far and many other logic circuits are available in one of those logic families. Each logic family has its own merits and demerits.
TTL (Transistor- Transistor-Logic) is most popular of logic families. TTL or TTL family is named because of its dependence on transistor alone to perform basic logic operation. It is most widely used bipolar digital IC family. The TTL uses transistors operating in saturated logic families. The basic TTL logic circuit is NAND gate. Good speed, low manufacturing cost, wide range of circuit availability is SSI and MSI are its merits.

THEORY:
[A] NAND GATE as UNIVERSAL GATE:
Pin configuration of the 74LS00 NAND gate IC:



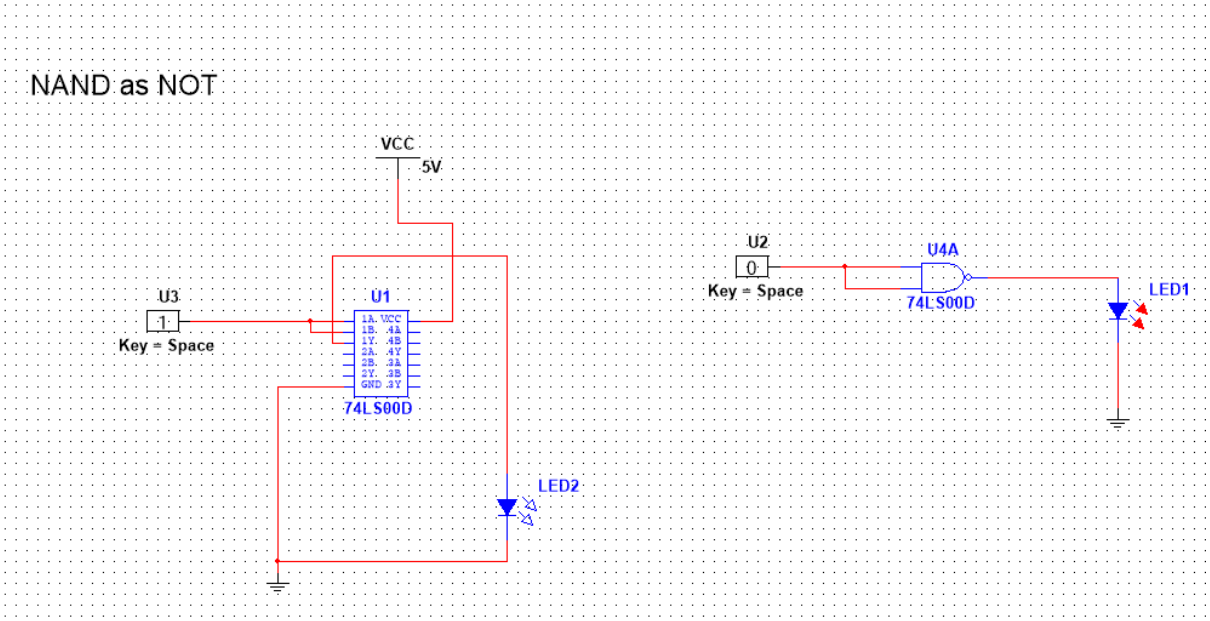
Draw figure of NAND Gate here



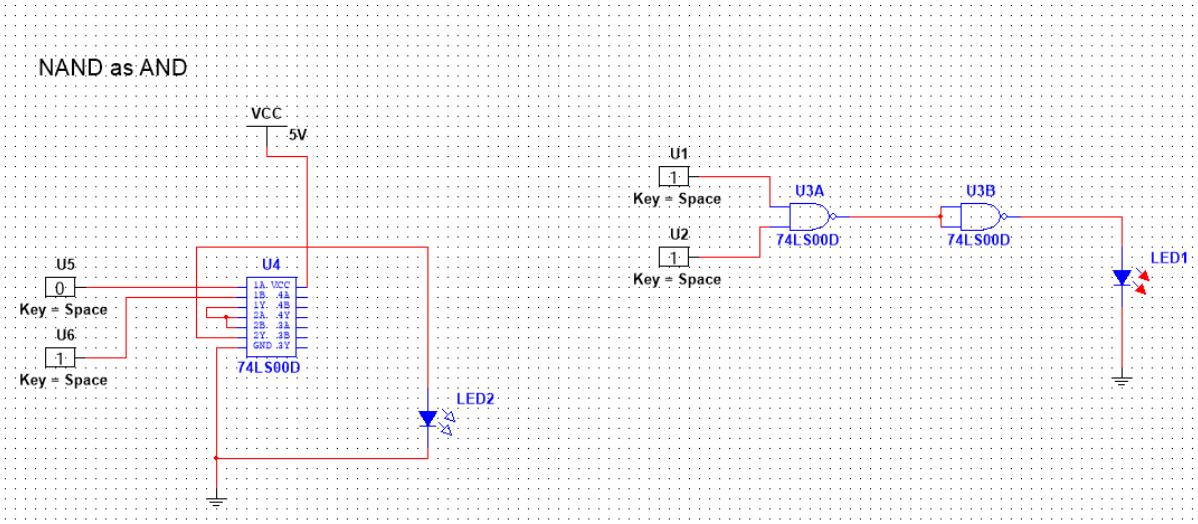
TRUTH TABLE:

A	B	OUTPUT
0	0	1
0	1	1
1	0	1
1	1	0

NAND AS NOT

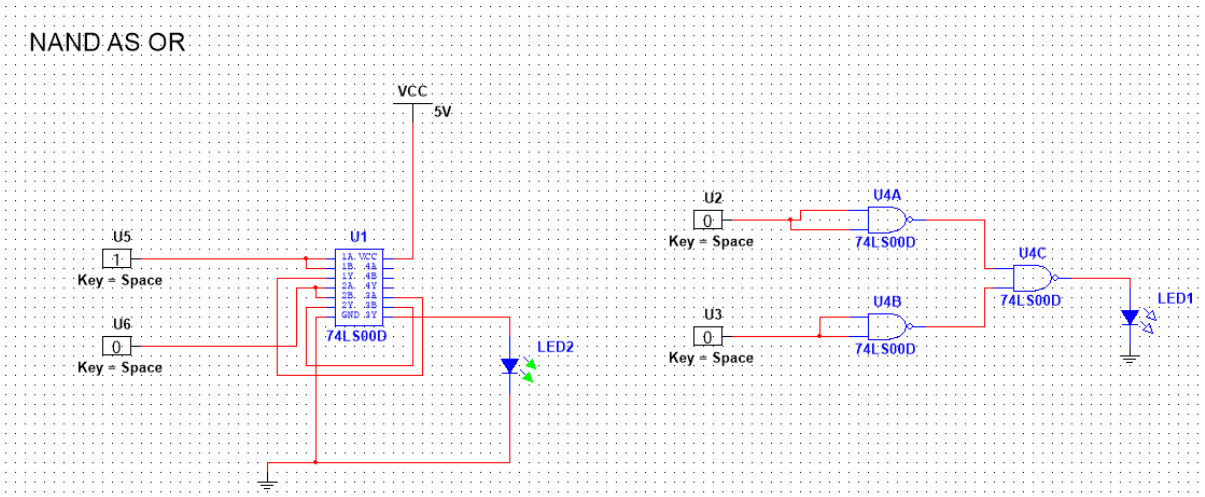


NAND AS AND

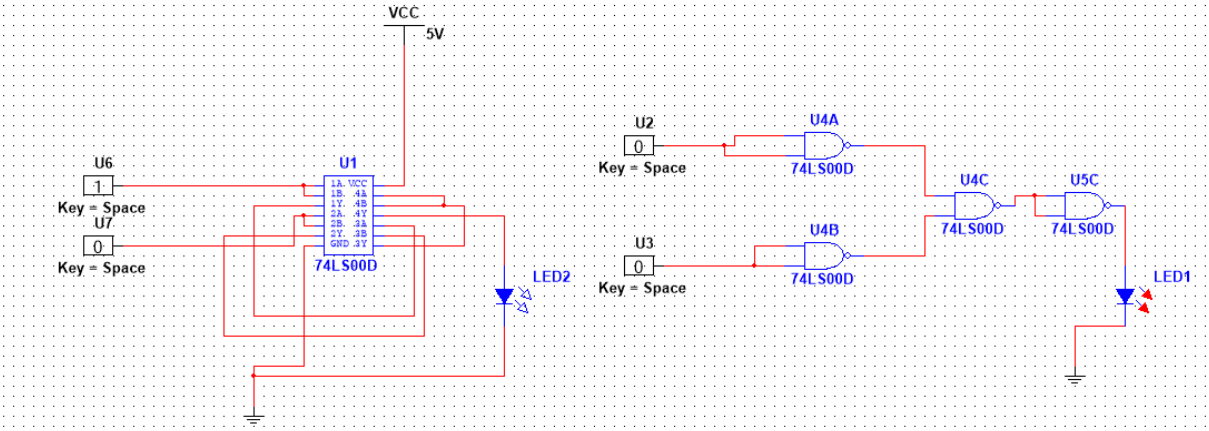


NAND AS OR

NAND AS OR

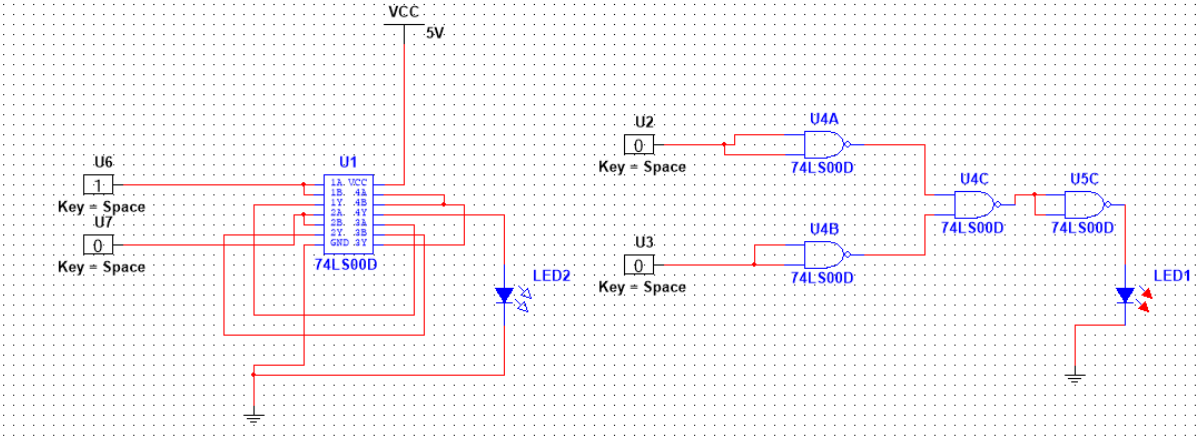


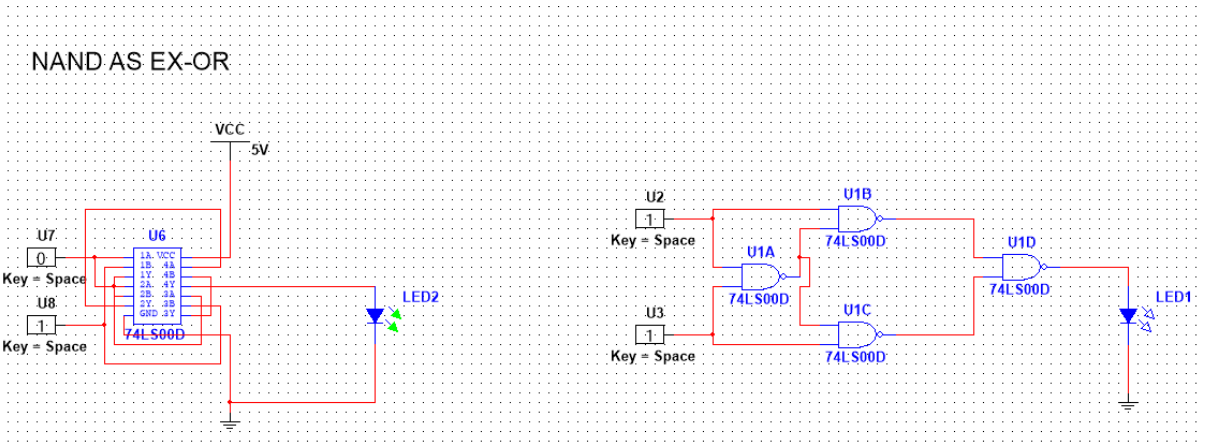
NAND AS NOR



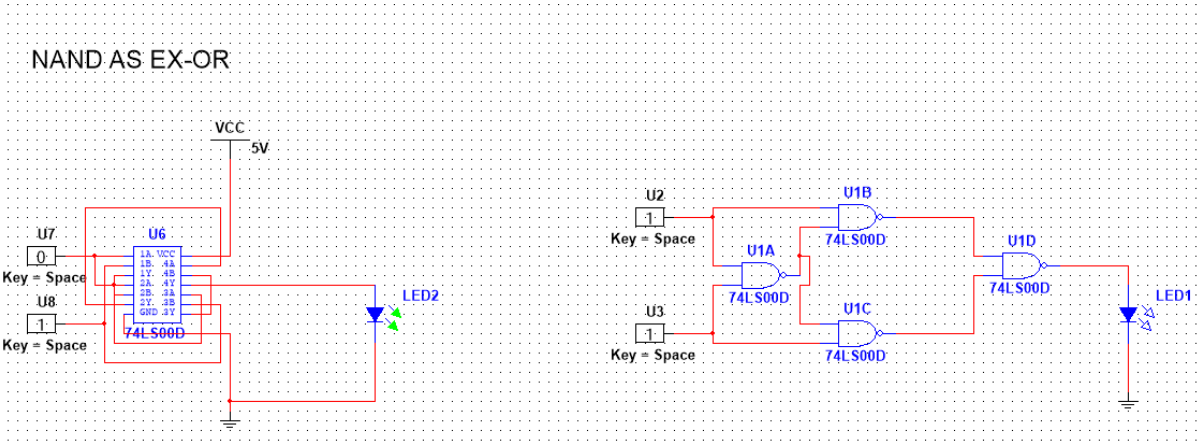
NAND AS NOR

NAND AS NOR

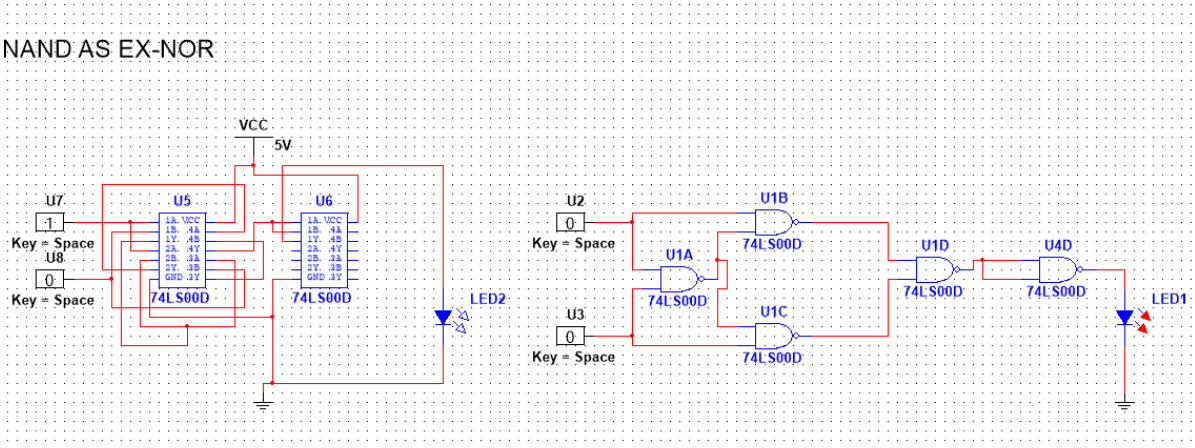




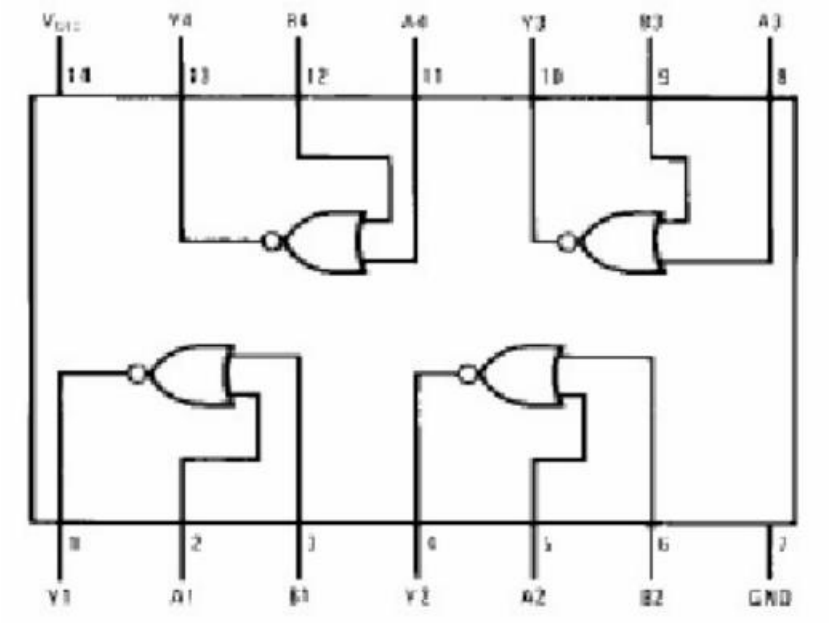
NAND AS XOR



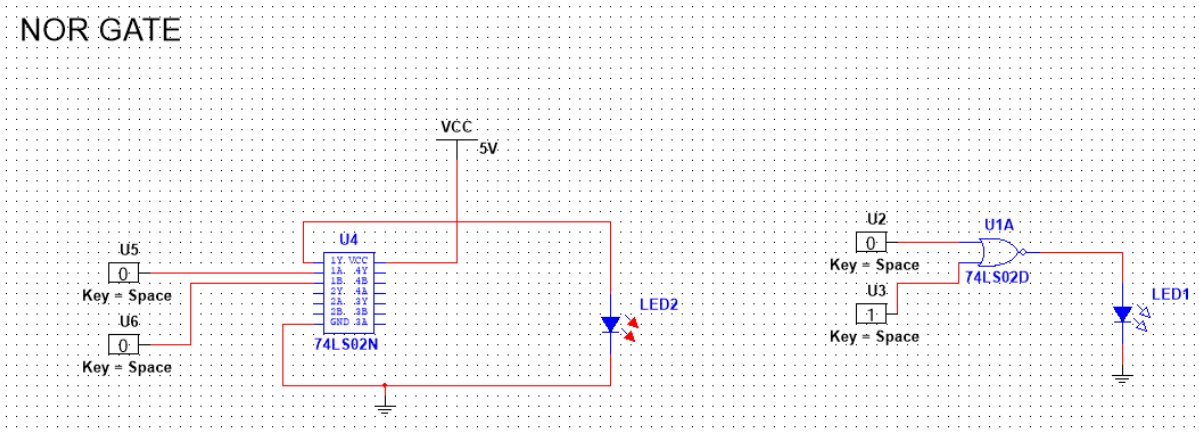
NAND AS XNOR



[B] NOR GATE as UNIVERSAL GATE:
Pin configuration of the 74LS02 NOR gate IC:



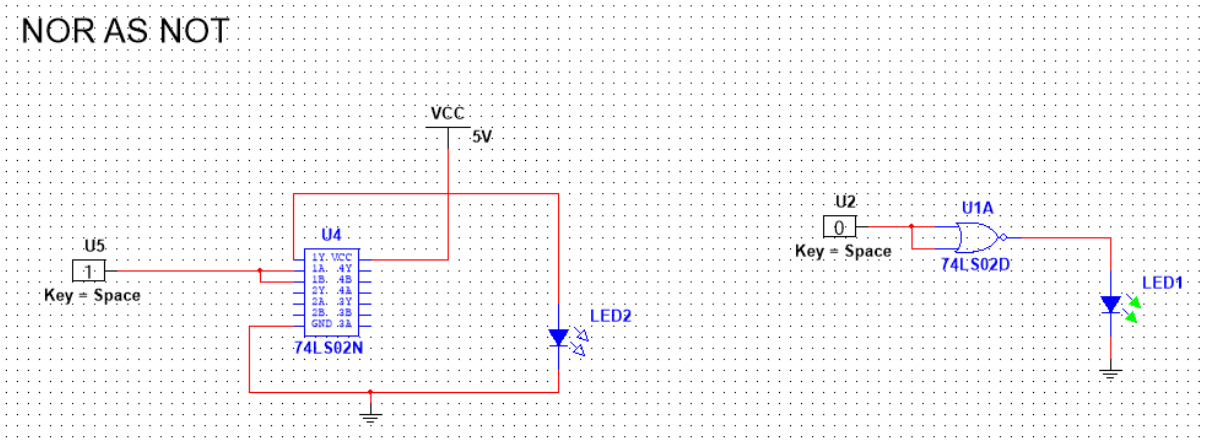
Draw figure of NOR Gate here



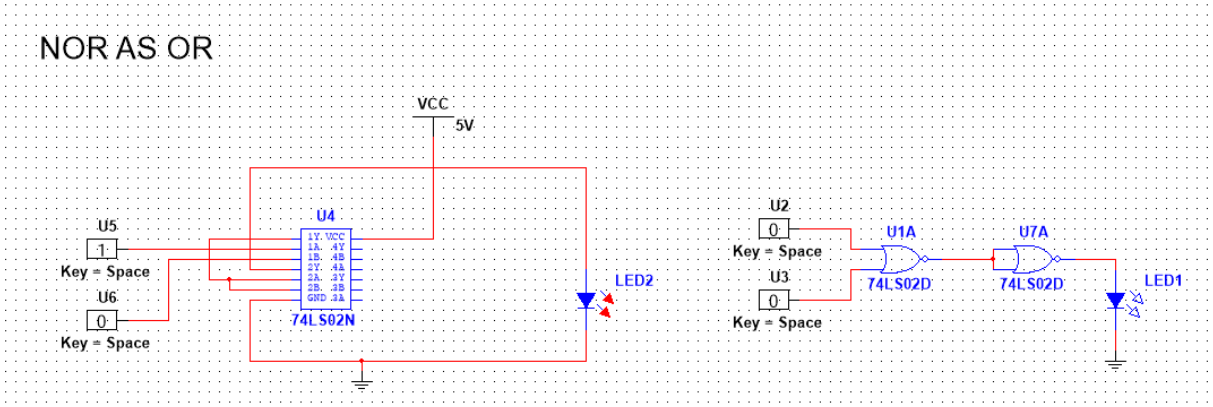
TRUTH TABLE:

A	B	OUTPUT	Output voltage level
0	0	1	
0	1	0	
1	0	0	
1	1	1	

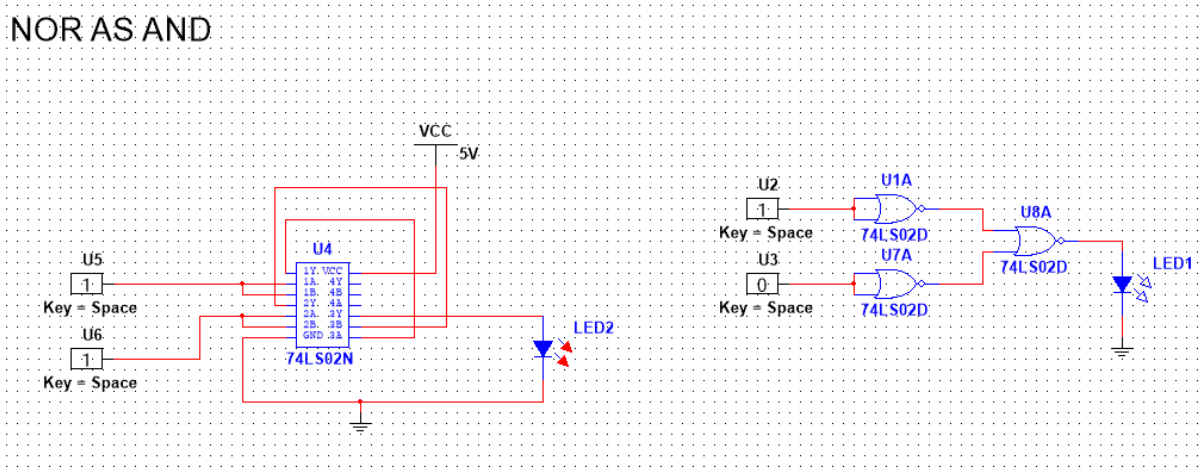
NOR AS NOT



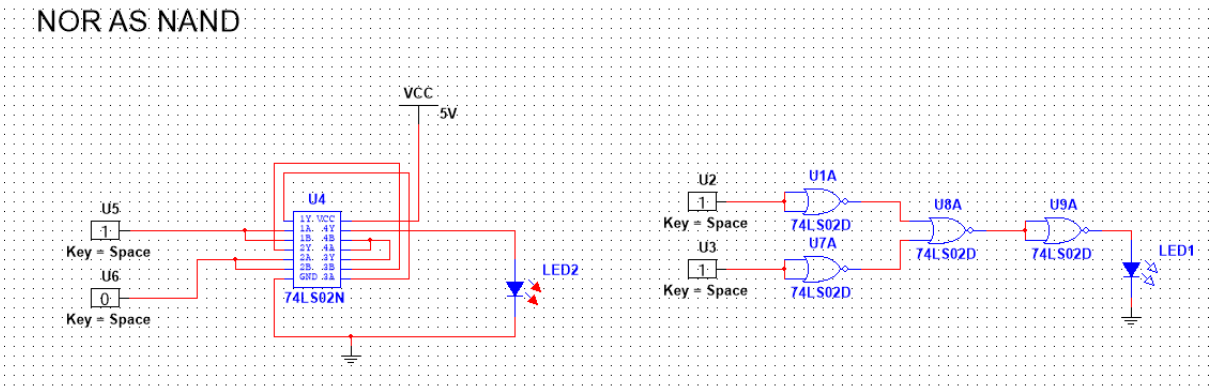
NOR AS OR



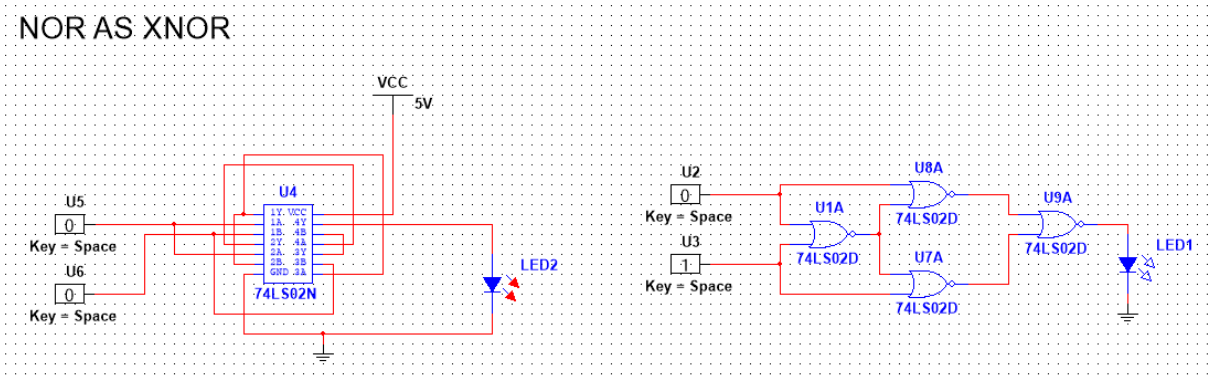
NOR AS AND



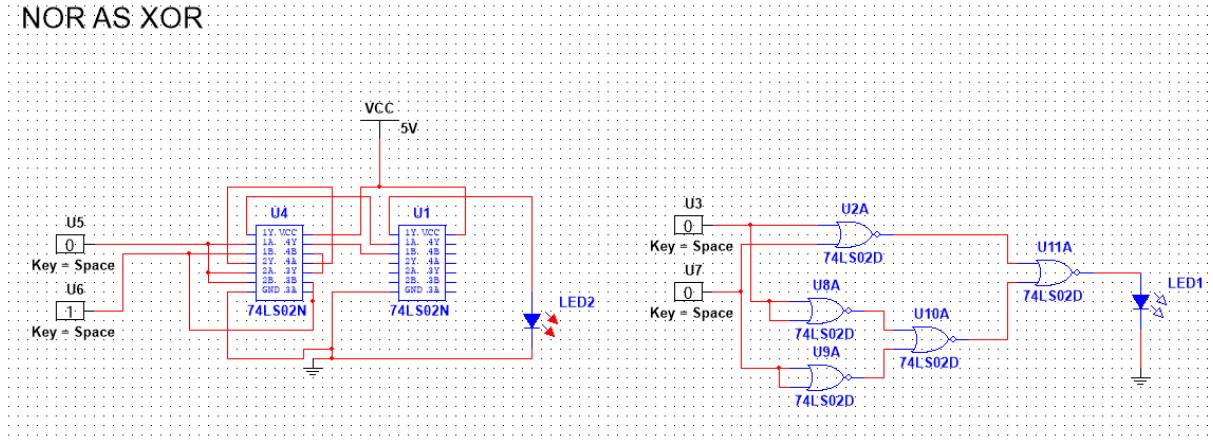
NOR AS NAND



NOR AS XNOR



NOR AS XOR



PROCEDURE:

(1) Install ICS 7400 & 7402 on the bread board.
(2) Connect pin number 7 & 14 of all ICS to ground & +5v supply respectively.
(3) Make the connection as shown in the logic diagram for AND, OR, NOT, and EX- OR gate using NAND gate only.
(4) Verify the truth tables of AND, OR, NOT and EX-OR gate.

EXPERIMENT NO:-4

➢ **AIM:** To design and test Half / Full adder and Subtractor circuits.

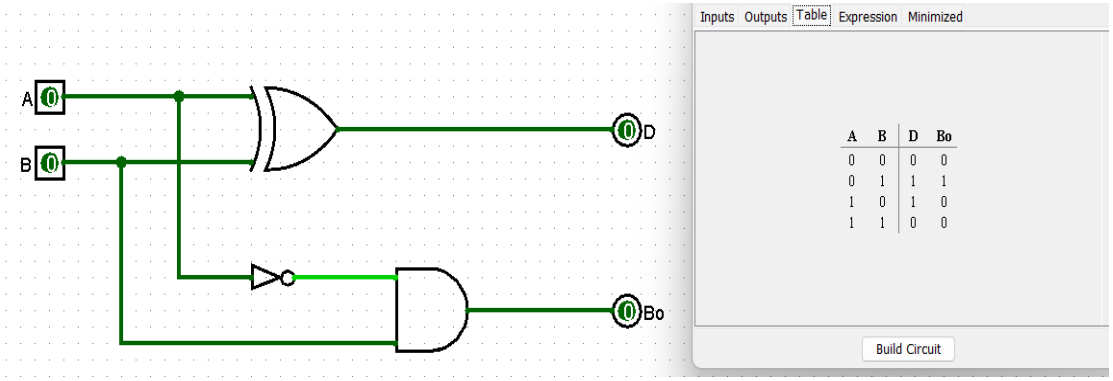
➢ **APPARATUS**: Logisim simulator.

➢ **THEORY:**

Digital computers perform variety of information processing task. Among the basic functions encountered are the various types of arithmetic operations. Here we will see how these operations can be performed using digital hardware.

➢ **(1)Half subtractor:** This Subtractor subtracts one bit from another but ignores any borrow from the previous stage. The outputs of the half adder are DIFFERENCE and BORROW. Truth table and expression for difference and borrow are given below.

CIRCUIT DIAGRAM FOR HALF SUBSTRACTOR:



TRUTH TABLE:

HALF SUBSTRACTOR

A	B	D	Bo
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

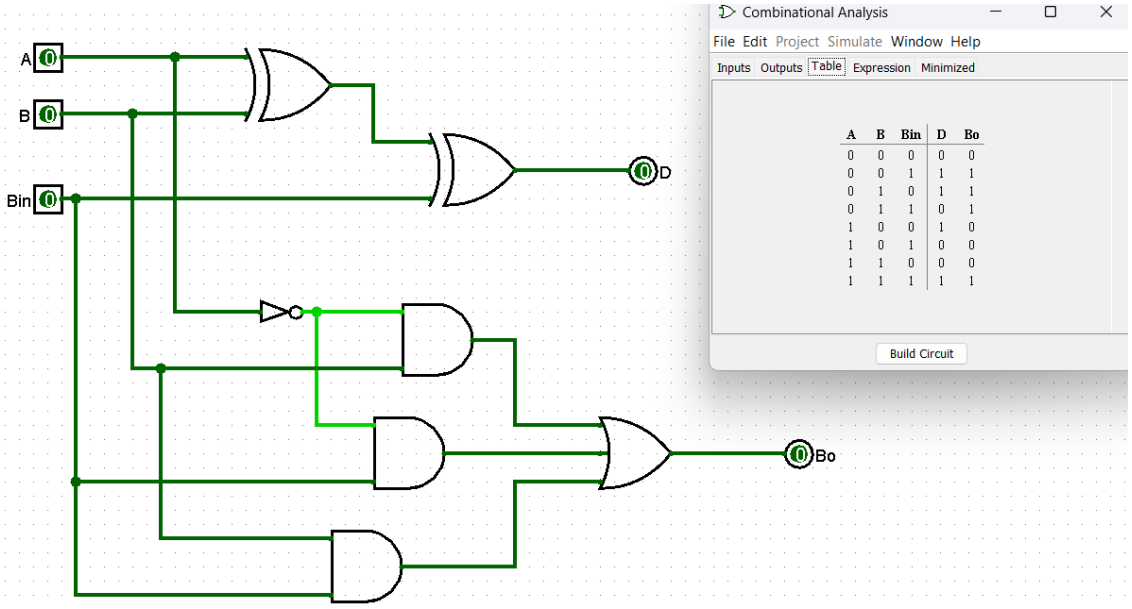
Expression for Difference and Borrow Out:

D = A(+)B

Bo = A`.B

➢ **(2)Full Subtractor:** This Subtractor subtracts binary digits along with borrow from the previous stage. The outputs of the Subtractor are difference and borrow out.

CRCUIT DIAGRAM OF FULL SUBTRACTOR:



A	B	Bin	D	Bo
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

Expression for Difference and Borrow Out:
D = [A(+)B](+)Bin
Bo = A`.Bin + B.Bin + A`.B

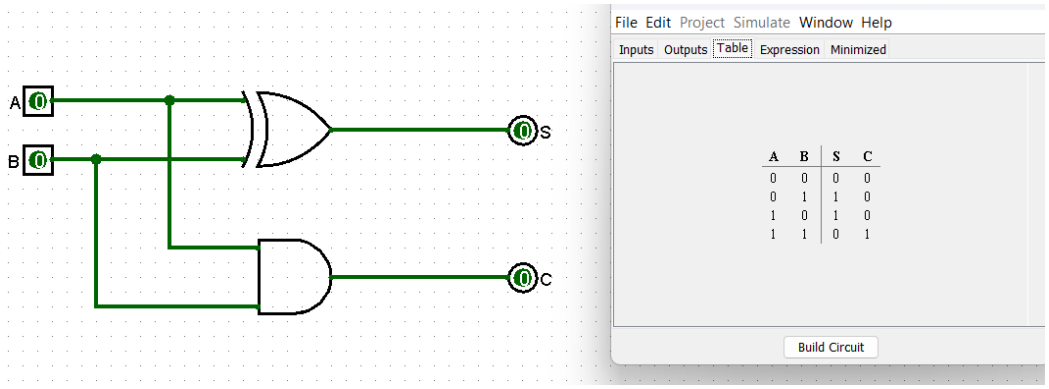
TRUTH TABLE:

FULL SUTRACTOR

A	B	Bin	D	Bo
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

➢ **(3)Half adder:** The adder adds only two bits and carries from the previous stage will not be added. The outputs of the adder and are SUM and CARRY. Truth table of half adder is given below.

CIRCUIT DIAGRAM OF HALF ADDER:



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

TRUTH TABLE:

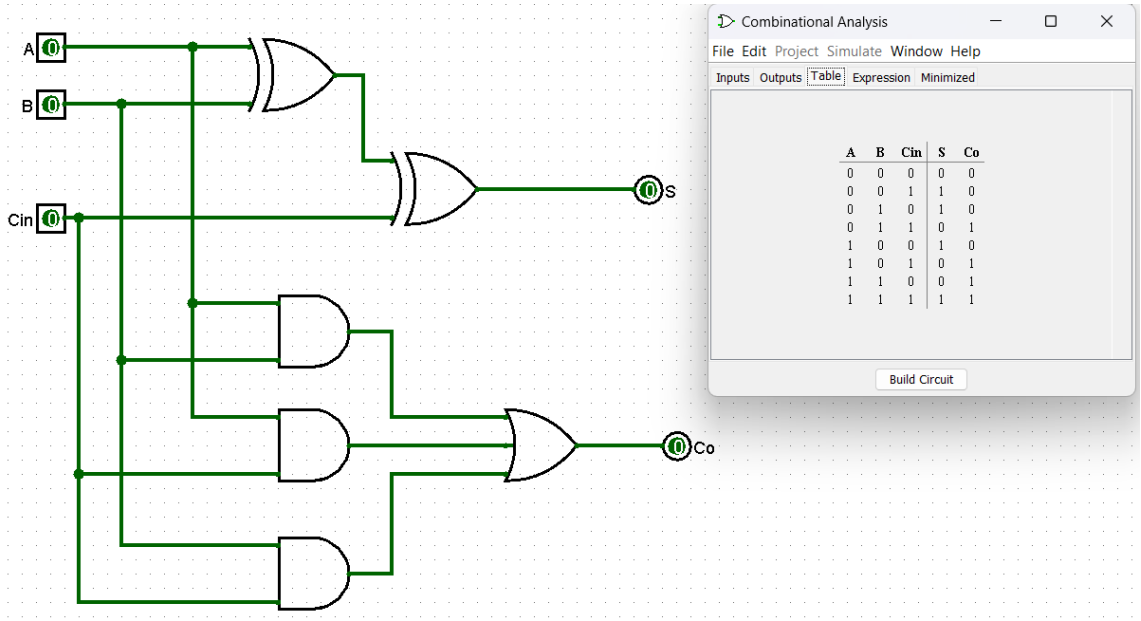
HALF ADDER

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Expression for Sum and Carry Out:
S = A(+)B
C = AB

➢ **(4)Full adder:** This adder adds two bits and carries from the previous stage. The outputs of the adder are SUM and CARRY. Truth table and simplified expression for sum and carry are given below.

CIRCUIT DIAGRAM OF FULL ADDER:



Expression for Sum and Carry Out:
S = [A(+)B](+)Cin
Cout = A.B + A.Cin + B.Cin

TRUTH TABLE:

FULL ADDER

A	B	Cin	S	Co
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

EXPERIMENT NO:-5

➢ **AIM:** To design and test 4-bit Binary to Gray and Gray to Binary Converter circuits.

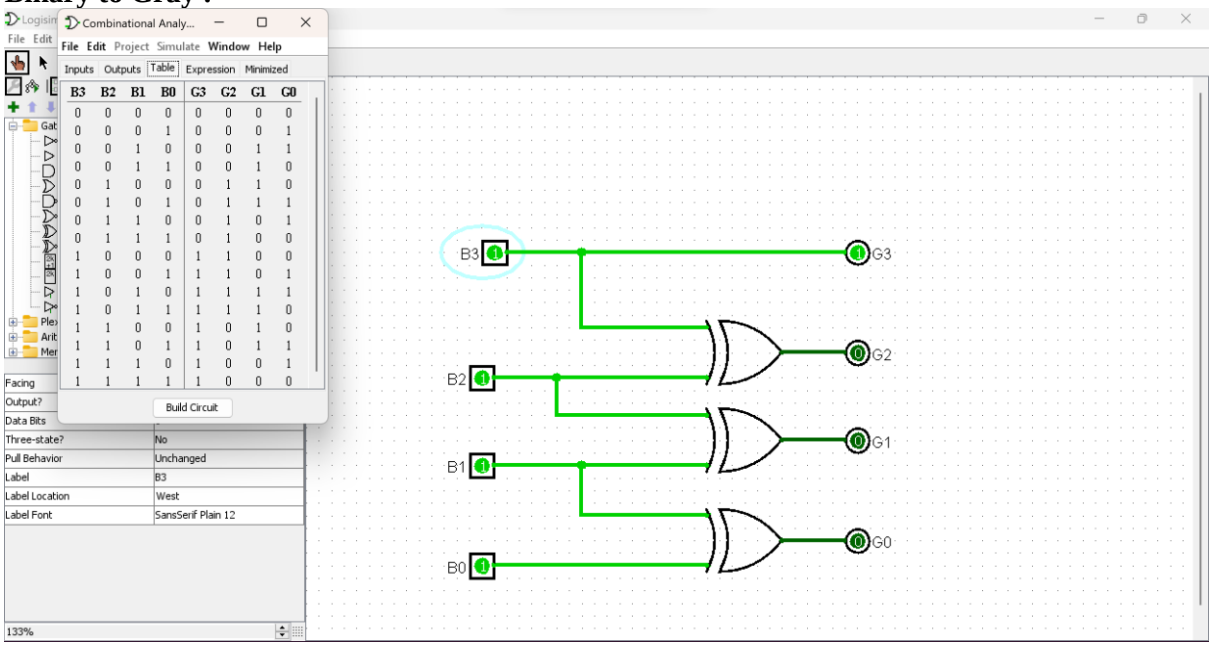
➢ **APPARATUS:** Logic trainer kit, Logisim simulator.

➢ **THEORY:**

Computers and other digital circuits are required to handle data that may be numerical alphabet or special character. Since digital circuit in binary fashion, the numerical, alphabets and other special characters are required to be converted into binary format. There are various possible ways of doing this, which is called encoding. Some commonly used binary codes are BCD, Excess-3 and Gray etc.

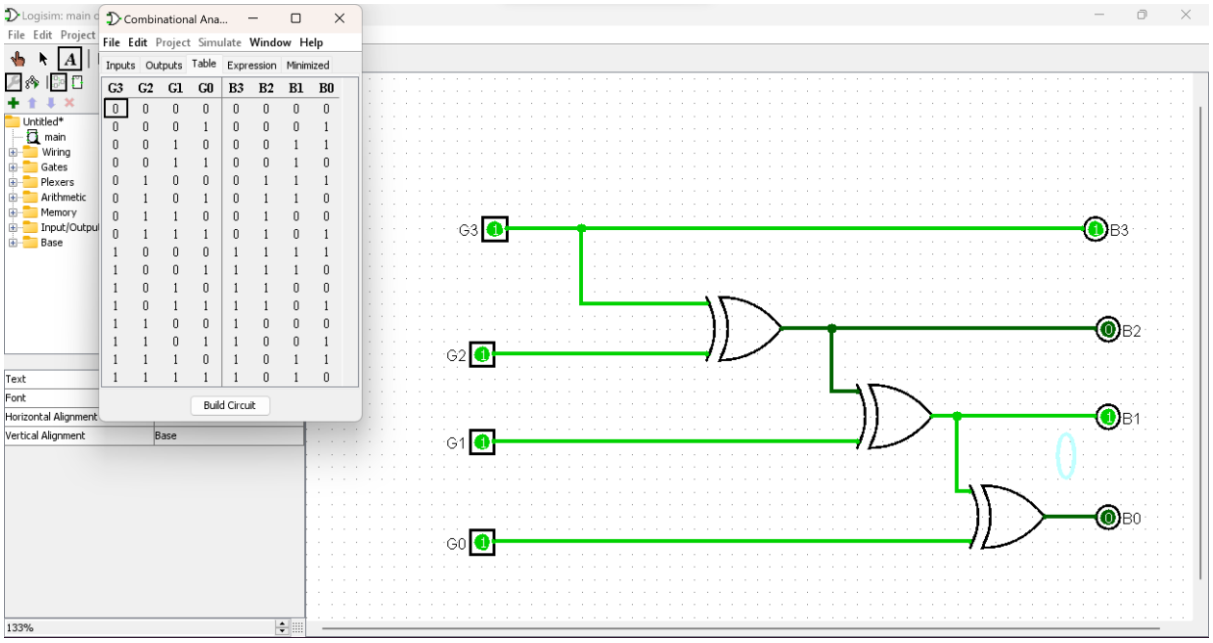
Many physical systems provide continuous data at their output. This data must be converted in to digital form before they are applied to a digital system. Continuous analog information is converted to digital form by means of analog to digital converter. Here it is useful to use the reflected (or gray) code to represent digital data converted from analog data. The advantage of reflected code over pure binary number is that the reflected code changes only be one bit as it proceeds from one number to the next

Binary to Gray :



G3 = B3
G2 = B3 (+) B2
G1 = B2 (+)B1
G0 = B1 (+) B0

Gray to Binary :



B3 = G3
B2 = G3 (+) G2
B1 = G3 (+) [G2 (+) G1]
B0 = [G3 (+) (G2 (+) G1)] (+) G0

➢ CODE CONVERSION TABLE:

BINARY				GRAY			
B3	**B2**	**B1**	**B0**	**G3**	**G2**	**G1**	**G0**
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

CONCLUSION - The design and testing of 4-bit Binary to Gray and Gray to Binary Converter circuits using Logisim successfully demonstrated the functionality of these converters. They efficiently converted between binary and Gray code representations, confirming their suitability for integration into digital systems.

EXPERIMENT NO:-6

➢ **AIM:** To Design and test decoder circuit.

➢ **APPARATUS:** Logisim simulator

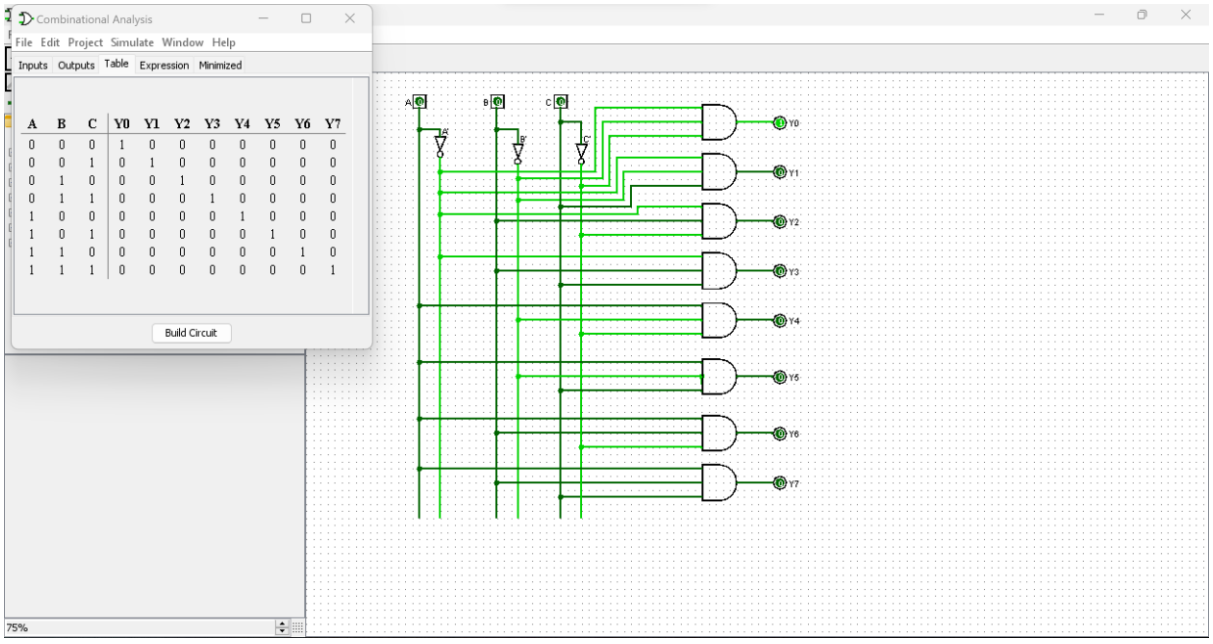
➢ **THEORY:**

Discrete quantities of information are represented in digital systems with binary codes. A binary code of n bits is capable of representing up to 2^n distinct elements of the coded information. A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. If the n-bit decoded information has unused or don't-care combinations, the decoder output will have less than 2^n outputs.

The decoders presented here are called n-to-m line decoders where m<=2^n. Their purpose is to generate the 2^n (or less) minterms of n input variables. The name decoder is also used in conjunction with some code such as BCD-to seven -segment decoder.

Consider the 3 to 8 line decoder circuit. The three inputs are decoded into eight outputs. Each output representing one of the minterms of the 3-input variables. The three inverters provide the complement of the outputs, and each one of eight AND gates generate one of the minterms. A particular application of this decoder would be a binary to octal conversion. The input variables may represent a binary number, and the outputs will then represent the eight digits in the octal number system. However a 3-to-8-line decoder can be used for decoding and 3-bit code to provide eight outputs, one for each element of the code.

CIRCUIT DIAGRAM OF 3 TO 8 BIT DECODER:



TRUTH TABLE OF 3 TO 8 BIT DECODER:

INPUTS				OUTPUTS							
ENABLE	ADDRESS LINES										
EN	A	B	C	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0

1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

CONCLUSION - Designing and testing a decoder circuit (using AND - NOT) is a fundamental exercise in digital electronics, providing practical experience and reinforcing key concepts in binary decoding and logic circuits.

EXPERIMENT NO:-7

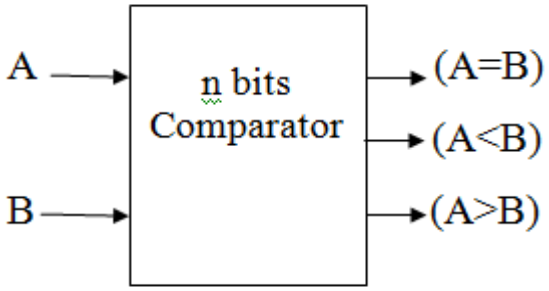
➢ **AIM:** To design and test 1-bit Magnitude comparator.

➢ **APPARATUS:** Breadboard, jumpers wires, IC's, LED's, power supply.

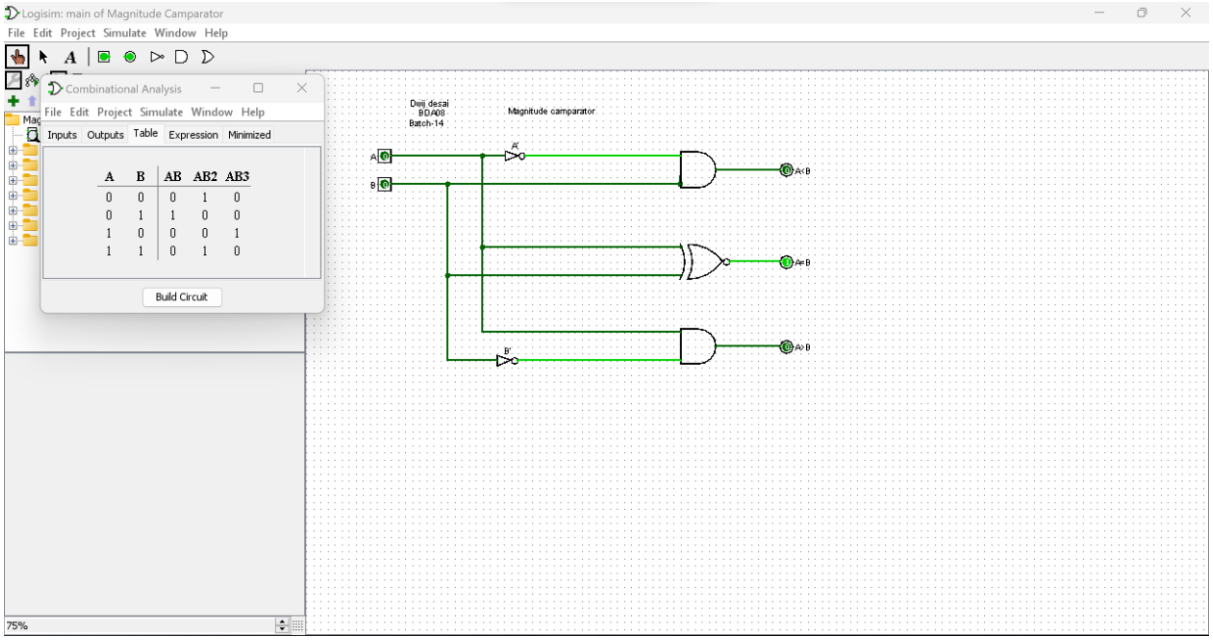
➢ **THEORY:**

The 1 bit magnitude comparator is a combinational circuit that compares magnitude of two 4 bit numbers to make either of its O/P (A>B, A=B, A<B) at logic high level.
Let A=A0 & B= B0 are 1-bit number respectively. The 1-bit magnitude comparator compares magnitudes as per following expressions for outputs.
Let xi will be at logic high level when Ai & Bi are at equal level. (i= 0, 1)

BLOCK DIAGRAM OF 1-BIT MAGNITUDE COMPARATOR:



CIRCUIT DIAGRAM OF 1-BIT MAGNITUDE COMPARATOR:



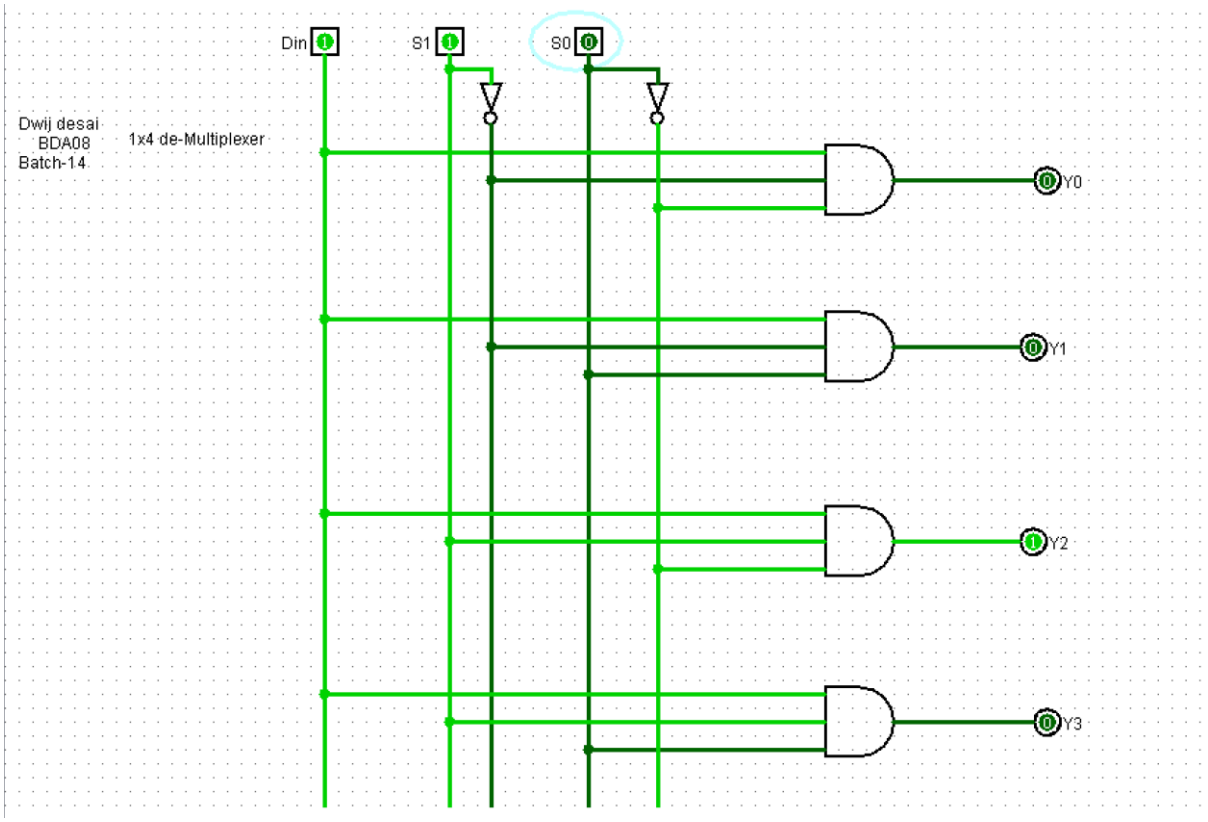
TRUTH TABLE

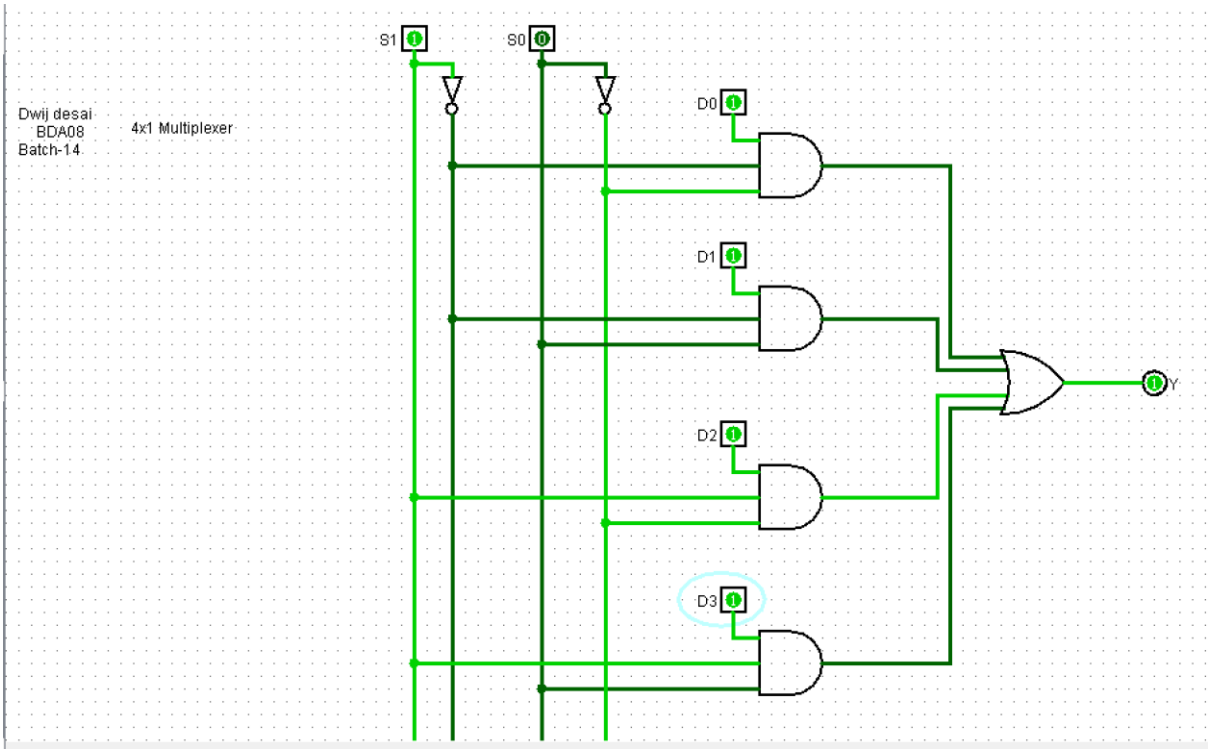
A	*B*	*A<B*	*A=B*	*A>B*
0	*0*	*0*	*1*	*0*
0	*1*	*1*	*0*	*0*
1	*0*	*0*	*0*	*1*
1	*1*	*0*	*1*	*0*

EXPERIMENT NO:-8

➢ **AIM: To design and test multiplexer / de-multiplexer circuit**.
➢ **APPARATUS** : Logisim simulator.

➢ **THEORY:**
(It includes circuit description, truth table, circuit diagram and logical explanation.)





➢ PROCEDURE:

-Design the circuit using Truth Table and K-Map.
-Use Logisim simulator
-Take appropriate components/gates from part selection
- Place it on mains, define the respective values
-connect appropriate with inputs and output bits with the components/gates.
-Run the simulator and observe the results.
-Test the truth table of each converter circuit.

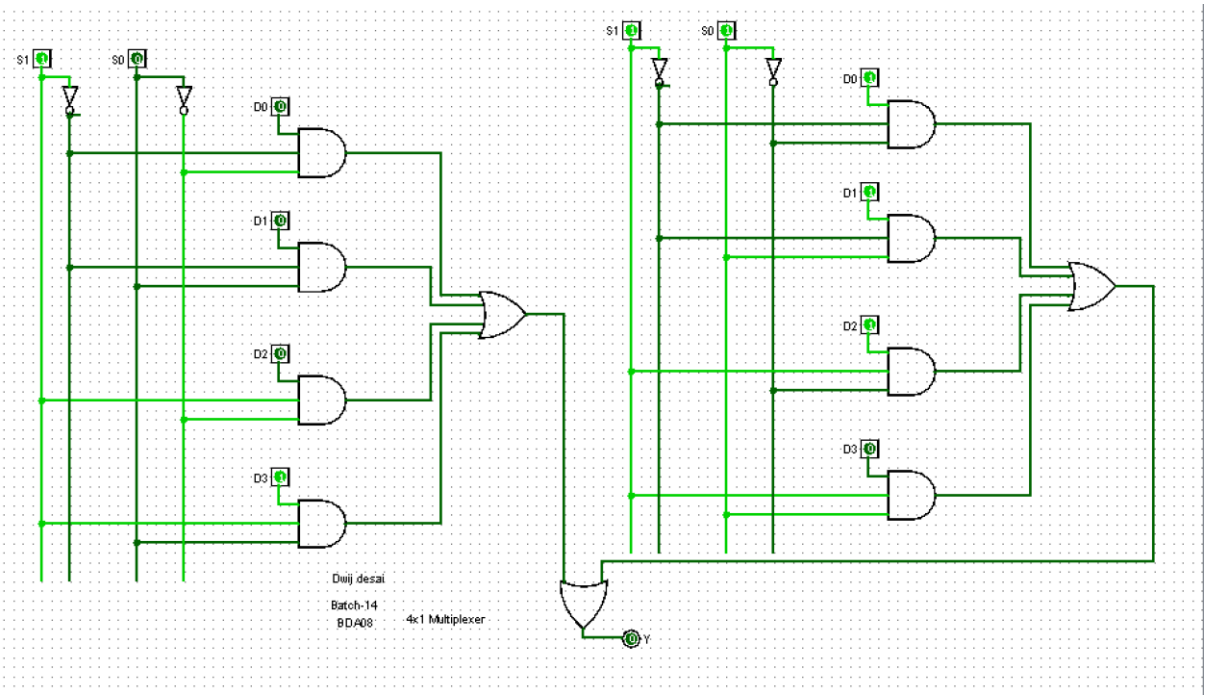
➢ CONCLUSION:

- Multiplexer takes 4 input and gives and gives 1 output so it's name 4x1 Multiplexer.

- De-multiplexer takes 1 input and give 4 output hence the name 1x4 De-multiplexer, it helps user to control the output of certain component.

➢ Exercise:

Implement an 8x1 Mux by using 4x1 Mux

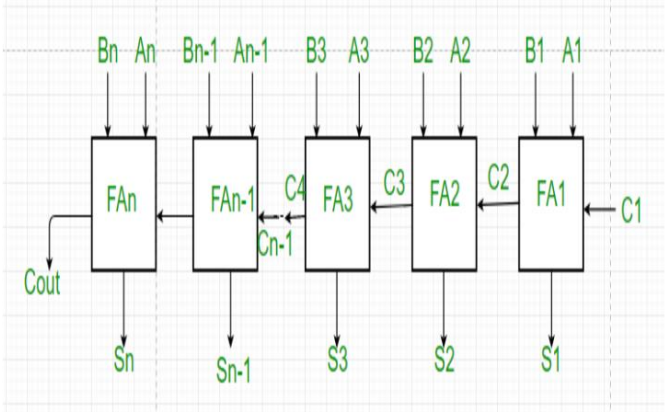


Practical-9

AIM: To design and test 4-bit parallel adder / Subtractor.
APPARATUS : Logisim simulator.

THEORY:

Parallel Adder – A single full adder performs the addition of two one bit numbers and an input carry. But a Parallel Adder is a digital circuit capable of finding the arithmetic sum of two binary numbers that is greater than one bit in length by operating on corresponding pairs of bits in parallel. It consists of full adders connected in a chain where the output carry from each full adder is connected to the carry input of the next higher order full adder in the chain. A n bit parallel adder requires n full adders to perform the operation. So for the two-bit number, two adders are needed while for four bit number, four adders are needed and so on. Parallel adders normally incorporate carry look ahead logic to ensure that carry propagation between subsequent stages of addition does not limit addition speed.



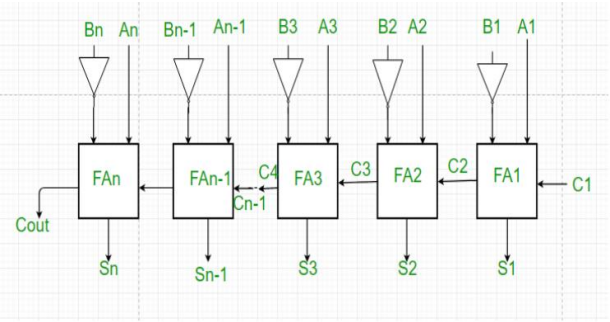
Working of parallel Adder –

1. As shown in the figure, firstly the full adder FA1 adds A1 and B1 along with the carry C1 to generate the sum S1 (the first bit of the output sum) and the carry C2 which is connected to the next adder in chain.

2. Next, the full adder FA2 uses this carry bit C2 to add with the input bits A2 and B2 to generate the sum S2(the second bit of the output sum) and the carry C3 which is again further connected to the next adder in chain and so on. 3. The process continues till the last full adder FAn uses the carry bit Cn to add with its input An and Bn to generate the last bit of the output along last carry bit Cout.

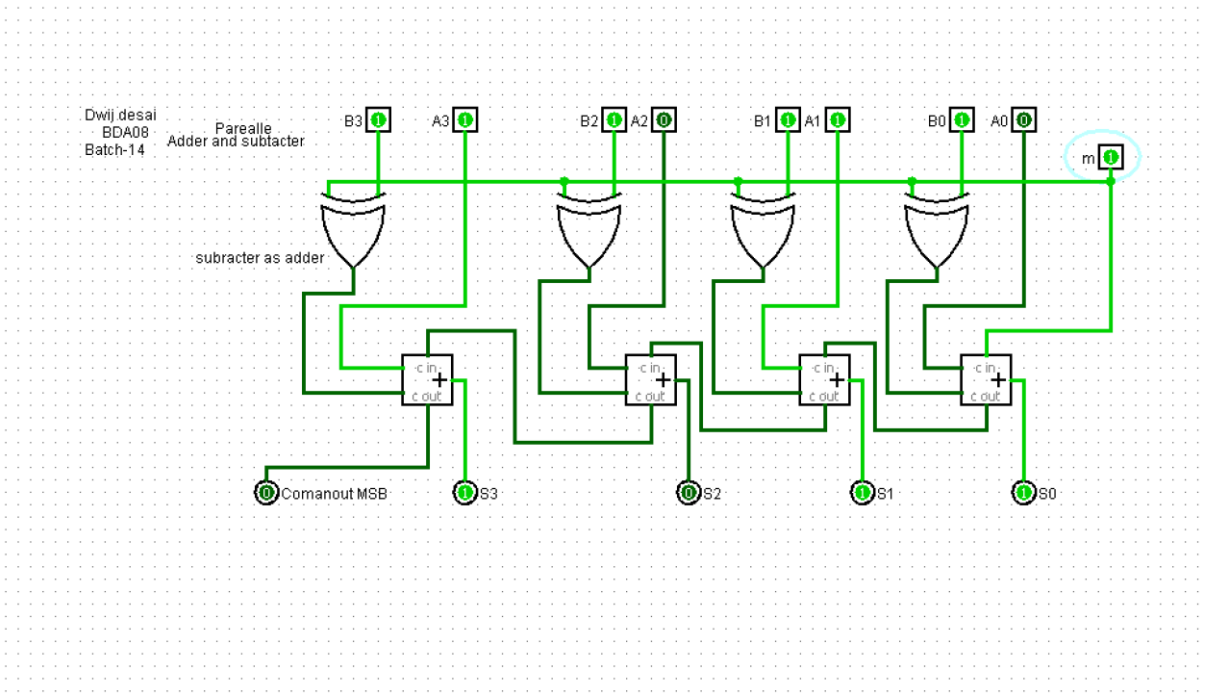
Parallel Subtractor –

A Parallel Subtractor is a digital circuit capable of finding the arithmetic difference of two binary numbers that is greater than one bit in length by operating on corresponding pairs of bits in parallel. The parallel subtractor can be designed in several ways including combination of half and full subtractors, all full subtractors or all full adders with subtrahend complement input.



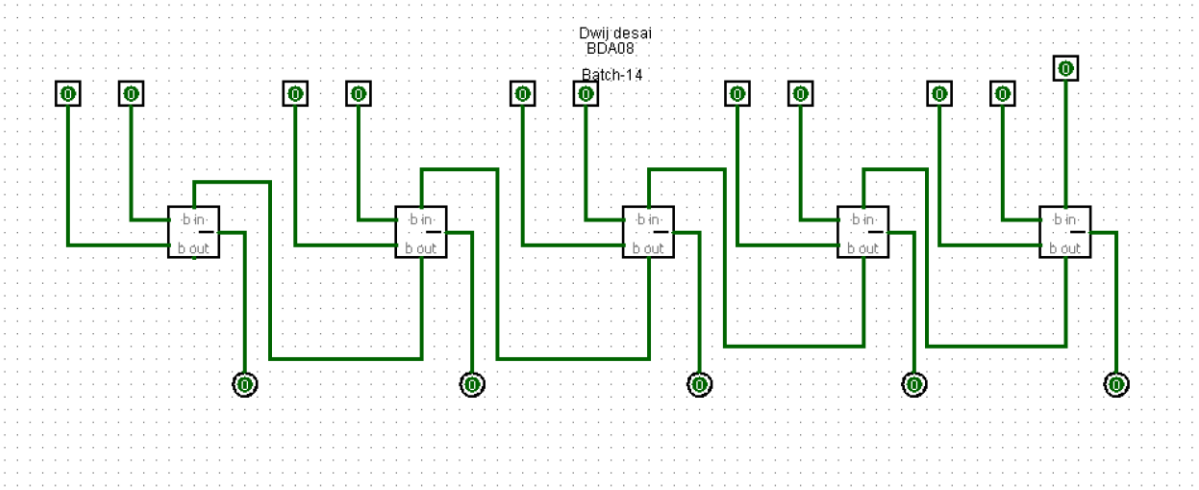
Working of Parallel Subtractor –

1. As shown in the figure, the parallel binary subtractor is formed by combination of all full adders with subtrahend complement input.
2. This operation considers that the addition of minuend along with the 2's complement of the subtrahend is equal to their subtraction. 31
3. Firstly the 1's complement of B is obtained by the NOT gate and 1 can be added through the carry to find out the 2's complement of B. This is further added to A to carry out the arithmetic subtraction.
4. The process continues till the last full adder FAn uses the carry bit Cn to add with its input An and 2's complement of Bn to generate the last bit of the output along last carry bit Cout



Exercise:

Make circuit for 5- bit Subtractor.



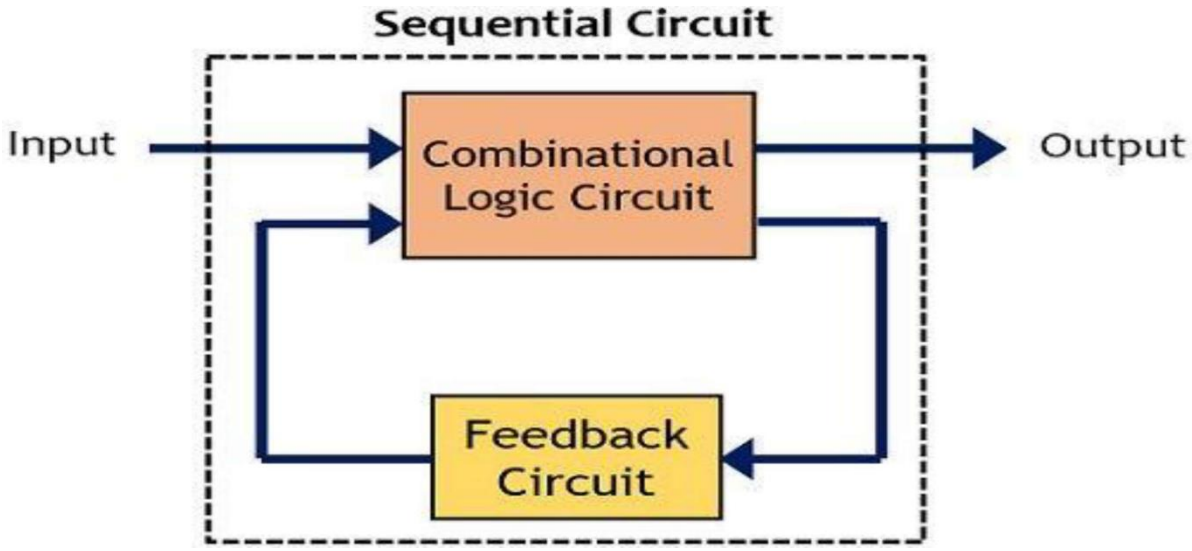
Conclusion –

The design and testing of the 4-bit parallel adder/subtractor confirmed its ability to perform both addition and subtraction operations efficiently, making it a crucial component in digital arithmetic circuits.

EXPERIMENT NO:-10

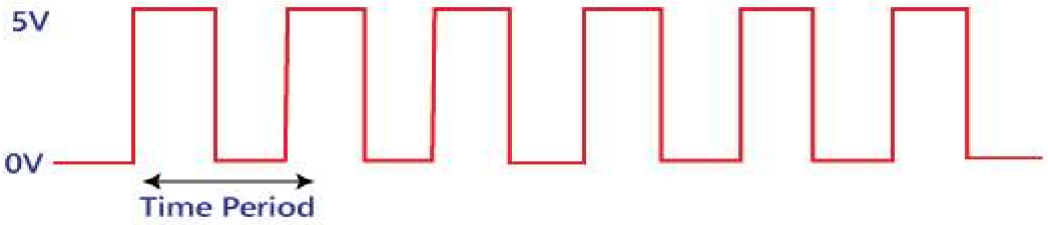
- **AIM*:*** To study and test Flip-Flop circuits.(SR, D, T , JK FFs)
- **APPARATUS :** Multisim/ Logisim software, Trainer kits

The sequential circuit is a special type of circuit that has a series of inputs and outputs. The outputs of the sequential circuits depend on both the combination of present inputs and previous outputs. The previous output is treated as the present state. So, the sequential circuit contains the combinational circuit and its memory storage elements.



Clock signal

A clock signal is a periodic signal in which ON time and OFF time need not be the same. When ON time and OFF time of the clock signal are the same, a square wave is used to represent the clock signal. Below is a diagram which represents the clock signal:

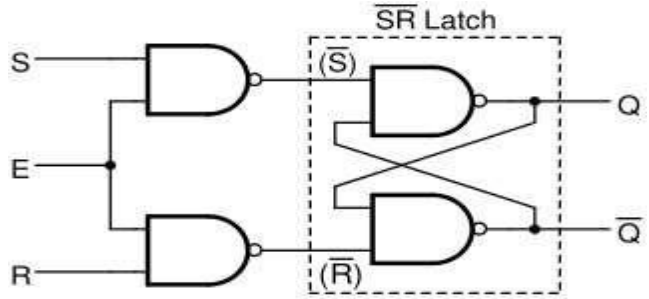


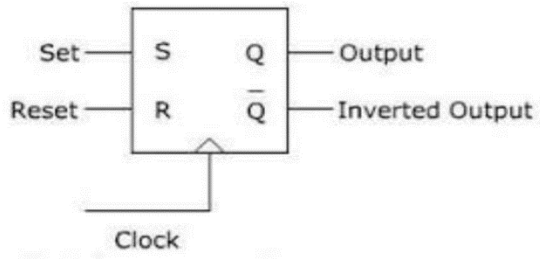
Basics of Flip Flop

A circuit that has two stable states is treated as a flip flop. These stable states are used to store binary data that can be changed by applying varying inputs. The flip flops are the fundamental building blocks of the digital system. Flip flops and latches are examples of data storage elements. In the sequential logical circuit, the flip flop is the basic storage element. The latches and flip flops are the basic storage elements but different in working.

SR flip flop:

The SR flip flop is a 1-bit memory bistable device having two inputs, i.e., SET and RESET. The SET input 'S' set the device or produce the output 1, and the RESET input 'R' reset the device or produce the output 0. The SET and RESET inputs are labeled as **S** and **R**, respectively. The SR flip flop stands for "Set-Reset" flip flop. The reset input is used to get back the flip flop to its original state from the current state with an output 'Q'. This output depends on the set and reset conditions, which is either at the logic level "0" or "1".

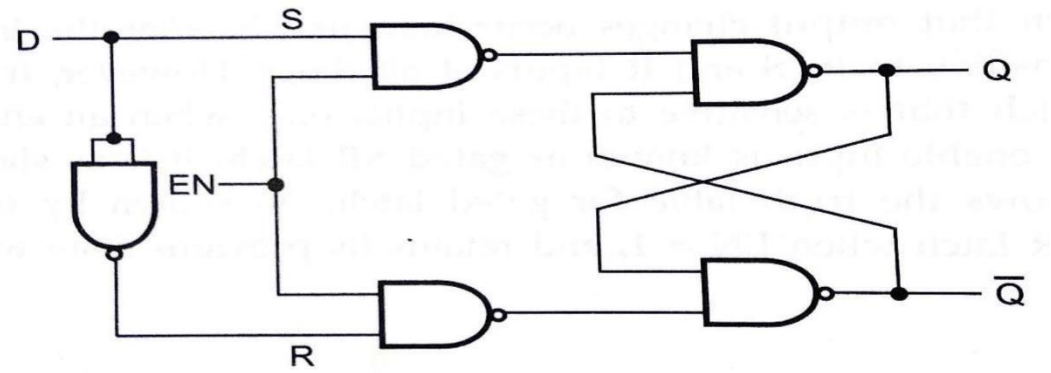




Serial No.	clock	S	R	Q(n-1)	$\bar{Q}$(n-1)	Q	$\bar{Q}$	Remark
1	0	0	0	X	X	0	1	No Change
2	1	0	0	0	1	0	1	No change
3	1	1	0	0	1	1	0	set
4	1	0	1	1	0	0	1	Reset
5	1	1	1	0	1	0	0	INVALID

D- FF:

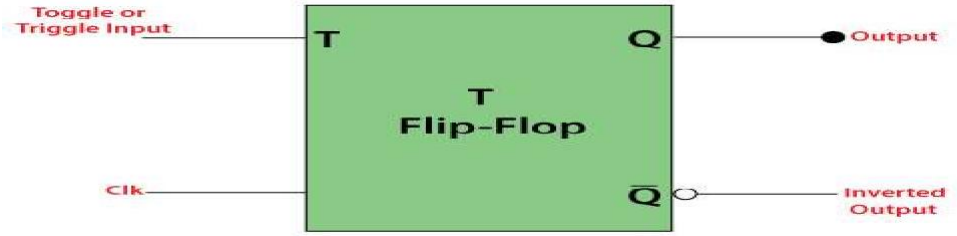
D flip flop is an electronic devices that is known as "delay flip flop" or "data flip flop" which is used to store single bit of data. D flip flops are synchronous or asynchronous. The clock single required for the synchronous version of D flip flops but not for the asynchronous one. The D flip flop has two inputs, data and clock input which controls the flip flop. When clock input is high, the data is transferred to the output of the flip flop and when the clock input is low, the output of the flip flop is held in its previous state.

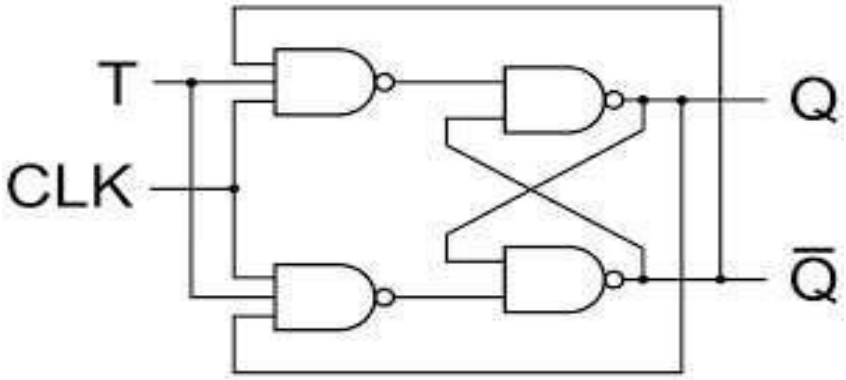


Serial No.	clock	D	Q(n-1)	Q'(n-1)	Q	Q'	Remark
1	0	0	X	X	0	1	No Change
2	0	1	0	1	0	1	No change
3	1	0	0	1	0	1	Reset
4	1	1	0	1	1	0	set

T-Flip Flop:

In T flip flop, "T" defines the term "Toggle". In SR Flip Flop, we provide only a single input called "Toggle" or "Trigger" input to avoid an intermediate state occurrence. Now, this flip-flop work as a Toggle switch. The next output state is changed with the complement of the present state output. This process is known as "Toggling"'. We can construct the "T Flip Flop" by making changes in the "JK Flip Flop". The "T Flip Flop" has only one input, which is constructed by connecting the input of JK flip flop. This single input is called T. In simple words, we can construct the "T Flip Flop" by converting a "JK Flip Flop". Sometimes the "T Flip Flop" is referred to as single input "JK Flip Flop".





Serial No.	Clock	T	Q_{n-1}	$\bar{Q}_{n-1}$	Q	$\bar{Q}$	Remarks
1	0	0	X	X	0	1	No Change
2	1	0	0	1	0	1	No change
3	0	1	0	1	0	1	No Change
4	1	1	0	1	1	0	Toggle

J-K Flip Flop:

The SR Flip Flop or Set-Reset flip flop has lots of advantages. But, it has the following switching problems:

When Set 'S' and Reset 'R' inputs are set to 0, this condition is always avoided.

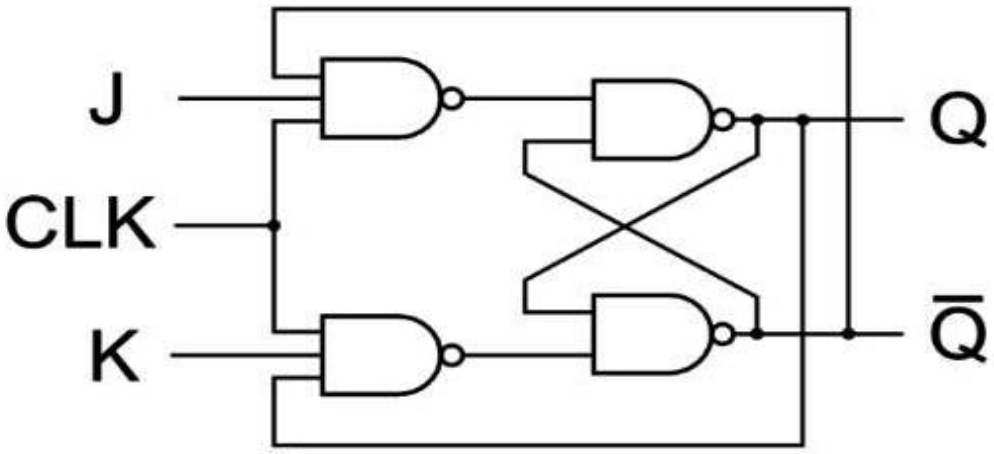
When the Set or Reset input changes their state while the enable input is 1, the incorrect latching action occurs.

The JK Flip Flop removes these two drawbacks of SR Flip Flop.

The JK flip flop is one of the most used flip flops in digital circuits. The JK flip flop is a universal flip flop having two inputs 'J' and 'K'. In SR flip flop, the 'S' and 'R' are the shortened abbreviated letters for Set and Reset, but J and K are not. The J and K are themselves autonomous letters which are chosen to distinguish the flip flop design from other types.

The JK flip flop work in the same way as the SR flip flop work. The JK flip flop has 'J' and 'K' flip flop instead of 'S' and 'R'. The only difference between JK flip flop and SR flip flop is that when both inputs of SR flip flop is set to 1, the circuit produces the invalid states as outputs, but in case of JK flip flop, there are no invalid states even if both 'J' and 'K' flip flops are set to 1.

J-K Type



Serial No.	clock	J	K	Q(n-1)	$\bar{Q}$(n-1)	Q	$\bar{Q}$	Remark
1	0	0	0	X	X	0	1	No Change
2	1	0	0	0	1	0	1	No change
3	1	0	1	0	1	0	1	Reset
4	1	1	0	0	1	1	0	set
5	1	1	1	1	0	0	1	toggle

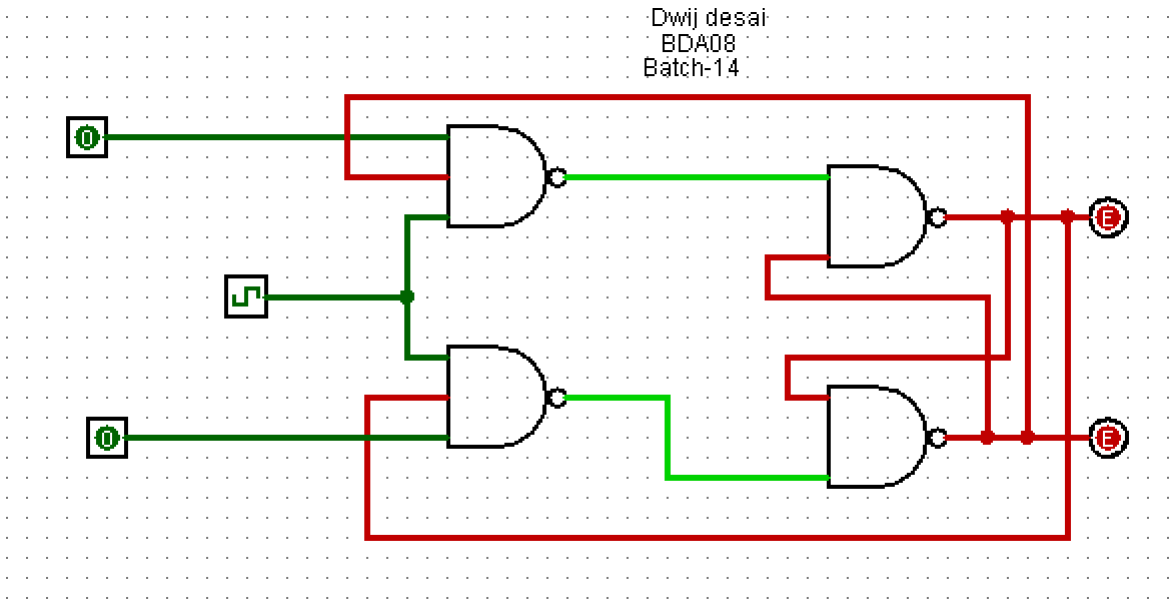
➢ PROCEDURE:

1. Connect the circuit on the bread board using ICs or if there is a kit ready, then connect the circuit as per the given instructions in kit's manual.
2. Switch ON the power supply.
3. Test the truth table of each flip-flop circuit and verify it.

➢ CONCLUSION: Designing for flip flop circuit is crusial part of DE to understand Clock ‘s And RAM { How to store information }

➢ Exercise:

Q. Design and show function of J-K master slave FF using NAND only gates.

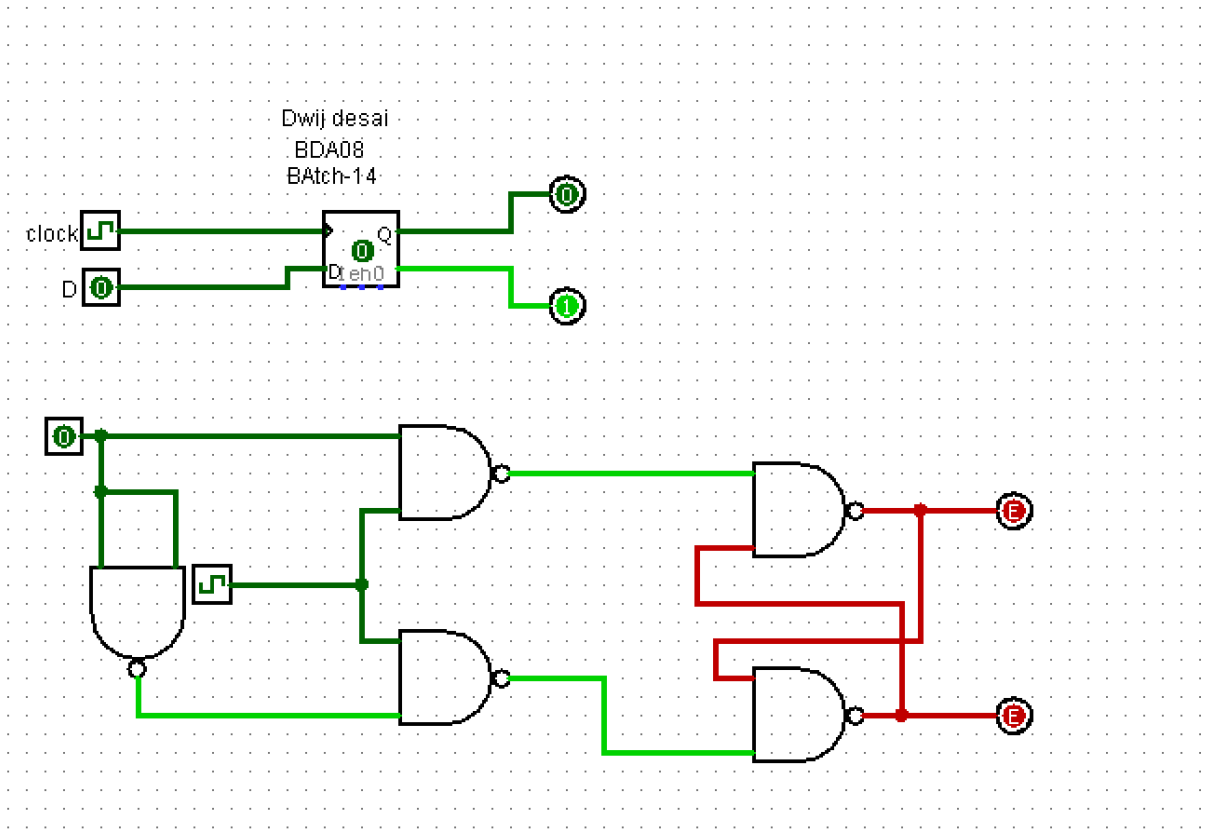


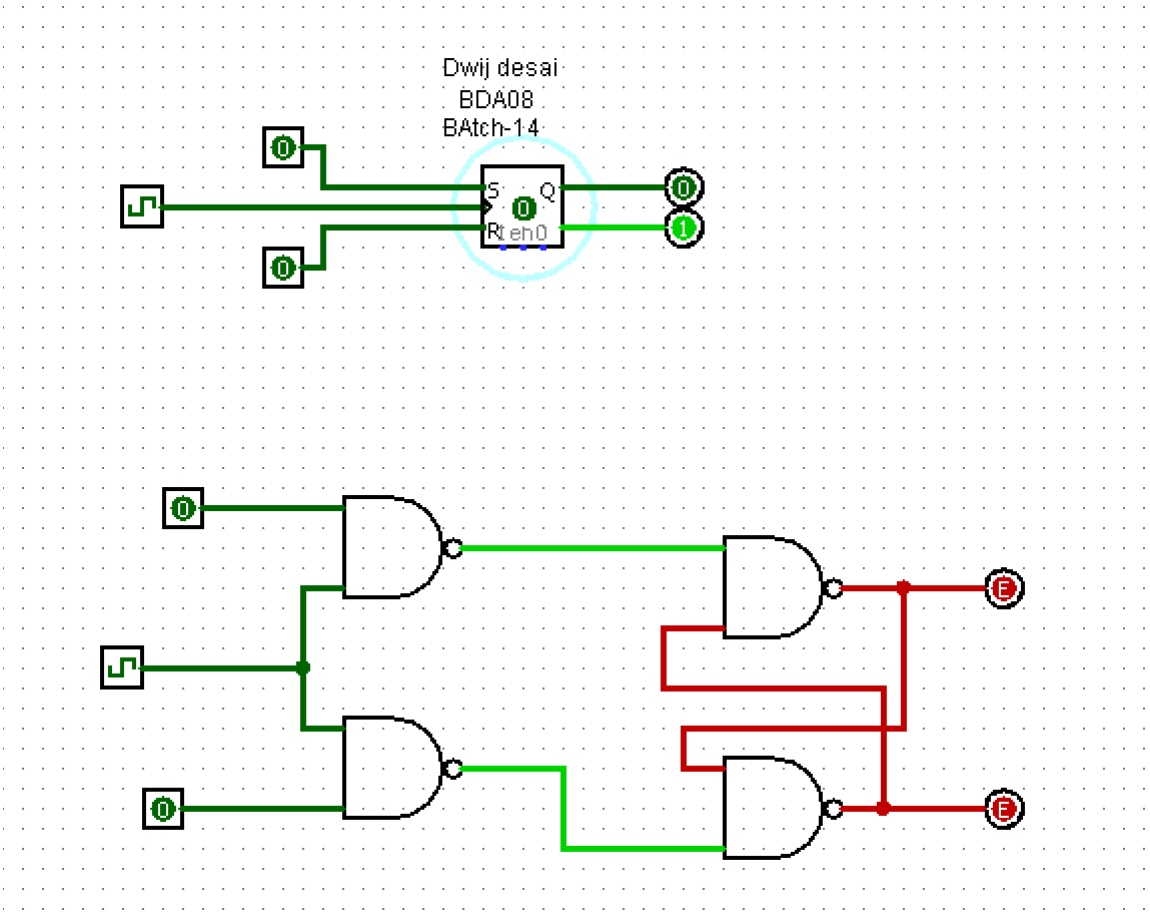
Q. Why we use J-K master-slave?

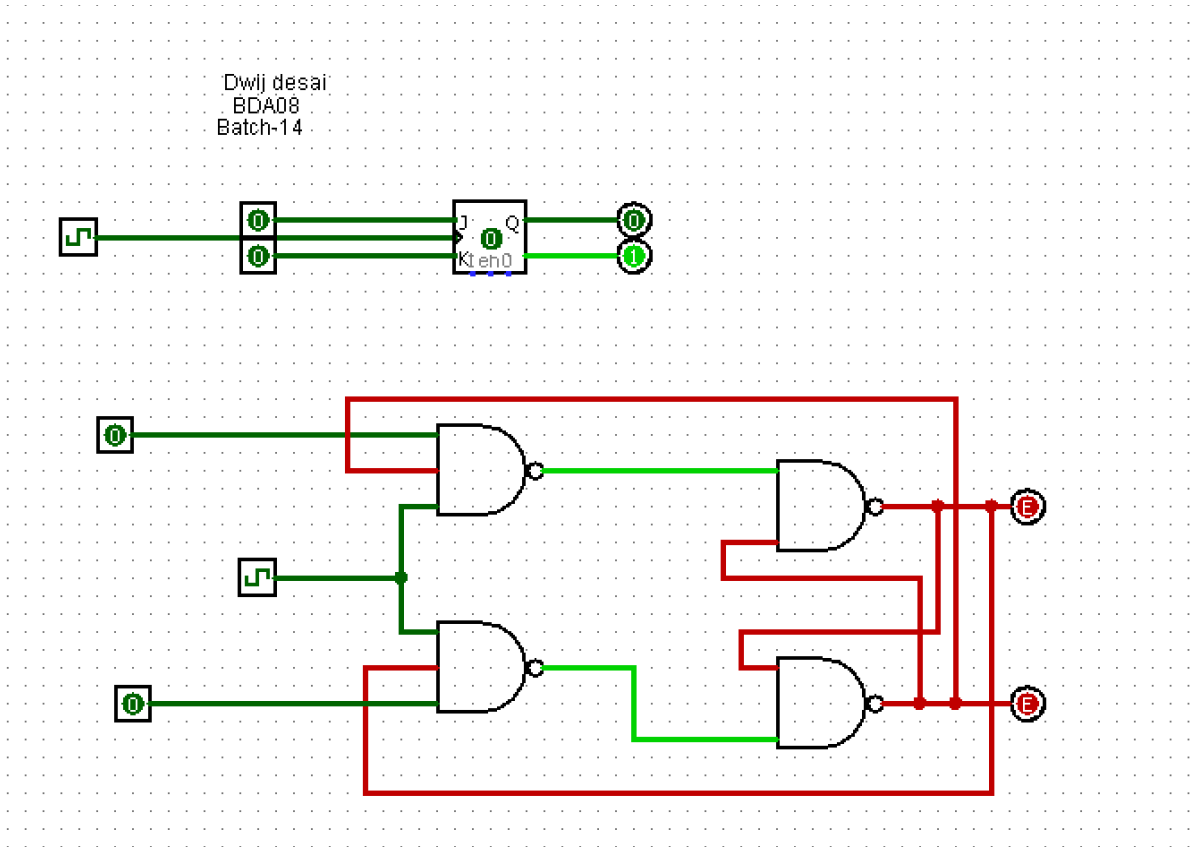
To avoid the problem of race around condition in JK flip flop, we use the JK flip flop in the Master and Slave Mode. Hence, the JK flip flop is called Master-Slave Flip Flop.

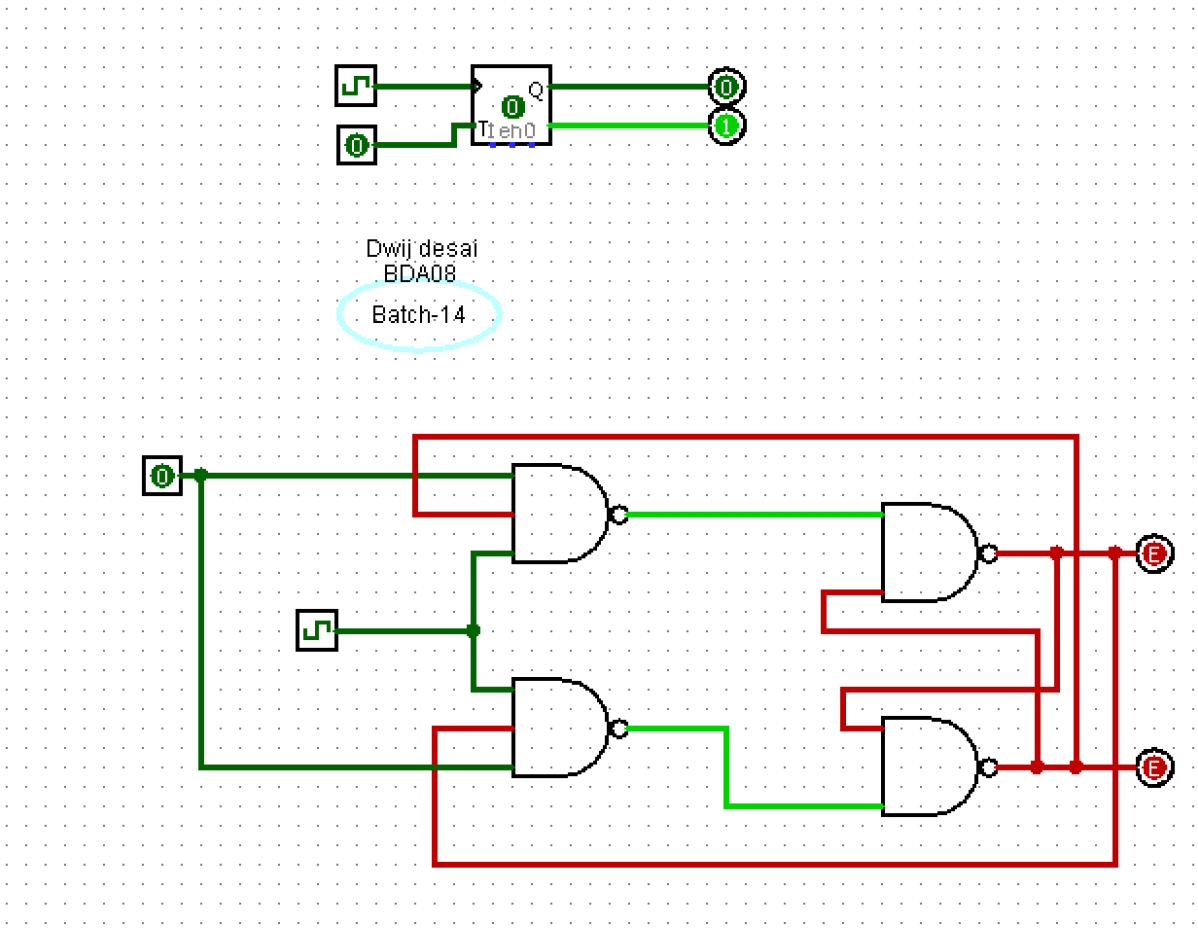
Race Around Condition In JK Flip-flop – For J-K flip-flop, if J=K=1, and if clk=1 for a long period of time, then Q output will toggle as long as CLK is high, which makes the output of the flip-flop unstable or uncertain. This problem is called race around condition in J-K flip-flop.

Q. Make D FF using SR FF.


Dwij desai
BDA08
BAtch-14
clock
D
Q
Reh0


Dwij desai
BDA08
BAtch-14
S
Q
Reh0


Dwij desai
BDA08
Batch-14
J
Q
K
clr


Q
Tteh0
Dwij desai
BDA08
Batch-14