DIGITAL LOGIC AND CIRCUIT DESIGN LAB

LABORATORY ASSIGNMENTS

3rd SEMESTER COMPUTER SCIENCE & ENGINEERING

SUBMITTED BY

NAME : ADITYA KIRAN PAL
SECTION : A
ENROLLMENT NO : 20UCS119
REGISTRATION NO : 2012709
SUBJECT : DCLD LAB
SEMESTER & YEAR : 3rd SEM , 2nd YEAR B. Tech.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY, AGARTALA
Jirania PO, Agartala, Barjala, Tripura-799046

Acknowledgment

I, Aditya Kiran Pal 20UCS119, would like to express my special thanks of gratitude to my teacher (Sir Ardhendu Gupta) who gave me the golden opportunity to do this wonderful experiments of DCLD and his detailed explanation, which also helped me in doing a lot of Research and also came to know about the LOGIC CIRCUIT SIMULATOR PRO APP, which helps us understand the practical application of the components and also,so many new things.

Secondly i would also like to thank my parents and friends who helped me a lot in finishing this project within the limited time.

I am making this project not only for marks but to also increase my knowledge.

THANKS AGAIN TO ALL WHO HELPED ME.

INDEX

SL NO.	DATE	EXPERIMENT NO.	PAGE
1	27/08/2021	Exp 1 : Study of Logic Gates	
2	08/09/2021	Exp 2 : Study of Universal Gates	
3	14/09/2021	Exp 3 : Study of De-Morgan’s Theorem	
4	22/09/2021	Exp 4 : Study of Boolean Expression Simplification	
5	21/10/2021	Exp 5 : Study of Half Adder and Full Adder	
6	09/11/2021	Exp 6 : Study of Half Subtractor and Full Subtractor	
7	17/11/2021	Exp 7 : Study of Multiplexer	
8	17/11/2021	Exp 8 : Study of Demultiplexer	
9	25/11/2021	Exp 9 : Study of SR Flip Flop	
10	25/11/2021	Exp 10 : Study of JK Flip Flop	

EXPT NO.1 :
STUDY OF LOGIC GATES

Objective :
To study about the logic gates and verify their truth tables.

Equipments :
Logic Circuit Simulator Pro.

Theory :
Circuit that takes the logical decision and the process are called logic gates. Each gate has one or more input and only one output. OR, AND and NOT are basic gates. NAND, NOR and X-OR are known as universal gates. Basic gates form these gates.

AND GATE :
The AND gate performs a logical multiplication commonly known as AND function. The output is high when both inputs are high. The output is low when any one of the inputs is low.

OR GATE:
The OR gate performs a logical addition commonly known as OR function.The output is high when any one of the inputs is high. The output is low level when both inputs are low.

NOT GATE:
The NOT gate is called inverter. The output is high when input is low. The output is low when input is high.

NAND GATE:
The NAND gate is a contraction of AND-NOT. The output is high when both inputs are low and any one of the inputs is low. The output is low when both inputs as high.

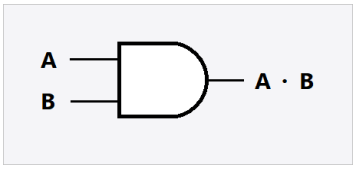
NOR GATE:
The NOR gate is a contraction of OR-NOT. The output is high when both inputs are low. The output is low when any one or both inputs are high.

X-OR GATE:
The output is high when any one of the inputs is high. The output is low when both inputs are high or low.

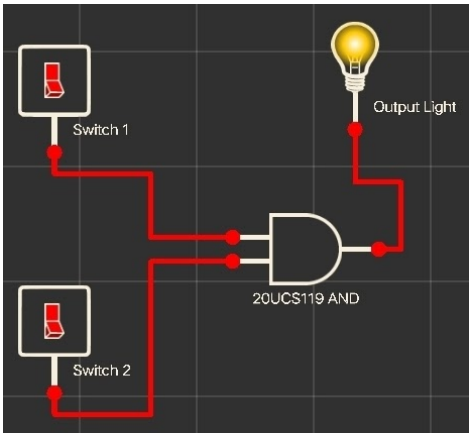
Procedure :

1. Firstly we have to install the LOGIC CIRCUIT SIMULATOR PRO APP and then after opening it, we have to make a new project.
2. Then we have to add the required elements to make the circuits i.e LOGIC gates, inputs, outputs etc.
3. Then connect the elements together with the help of a path to form a circuit and through the given gates we have to connect the input and output.
4. Lastly, we have to apply the TOUCH button to turn on/off the inputs i.e the switch by clicking on it, then we have to observe the output from the output section and verify the given truth tables.

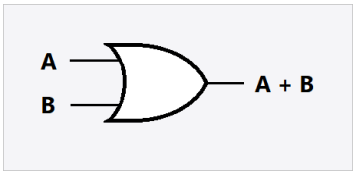
AND GATE :



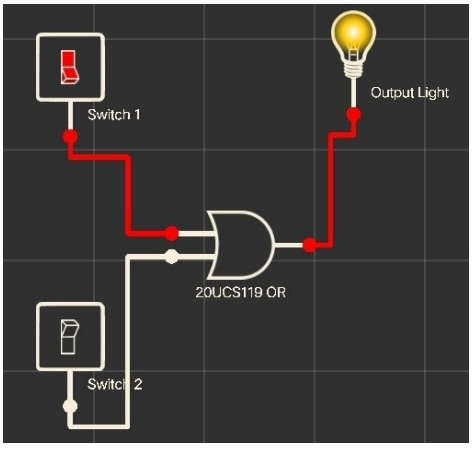
Truth Table		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1



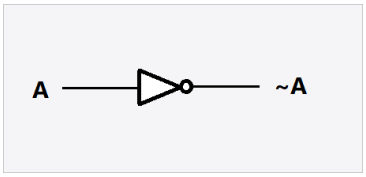
OR GATE:



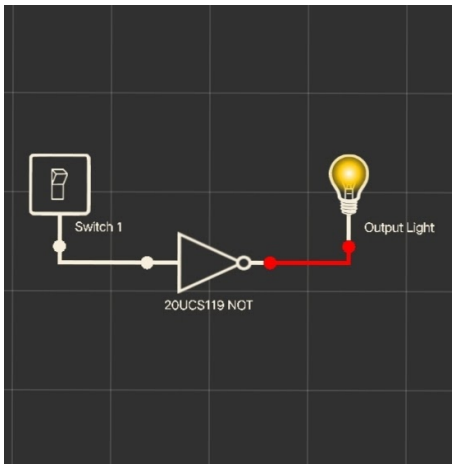
Truth Table		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1



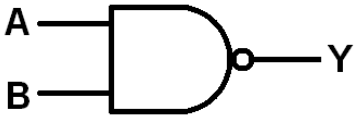
NOT GATE:



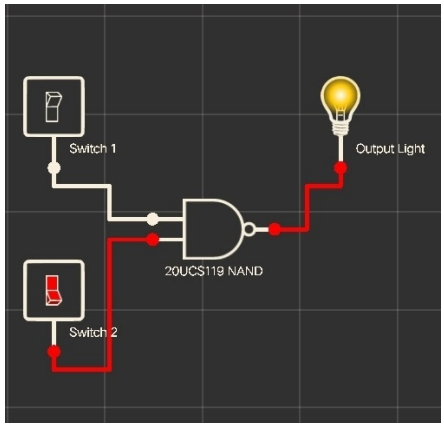
Truth Table	
A	A'
0	1
1	0



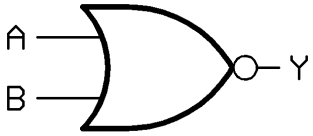
NAND GATE:



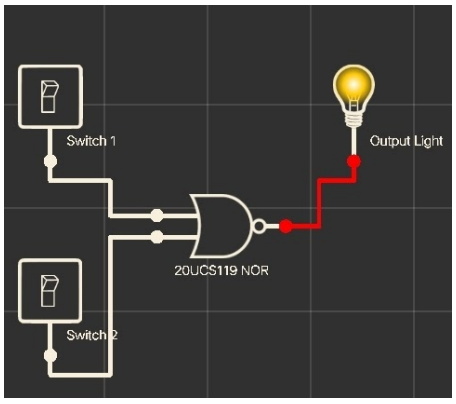
Truth Table		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0



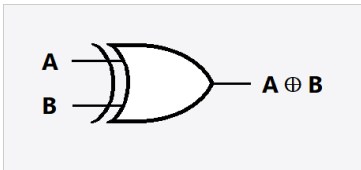
NOR GATE:



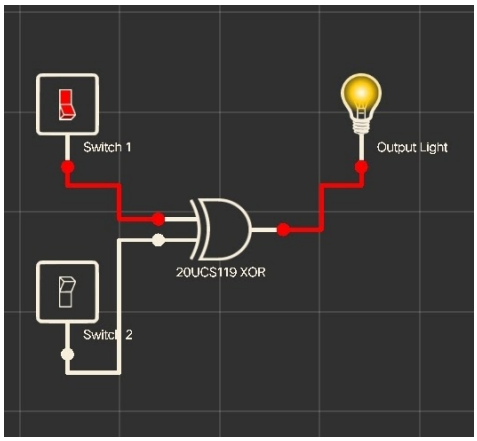
Truth Table		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



X-OR GATE:



Truth Table		
A	B	$\overline{A}B + A\overline{B}$
0	0	0
0	1	1
1	0	1
1	1	0



Conclusion :

The truth tables for various logic gates like AND, OR, NAND, NOT, X-OR, NOR are verified.

EXPT NO.2 :
STUDY OF UNIVERSAL GATES

Objective :
To study NAND and NOR gates as Universal Logic Gates.

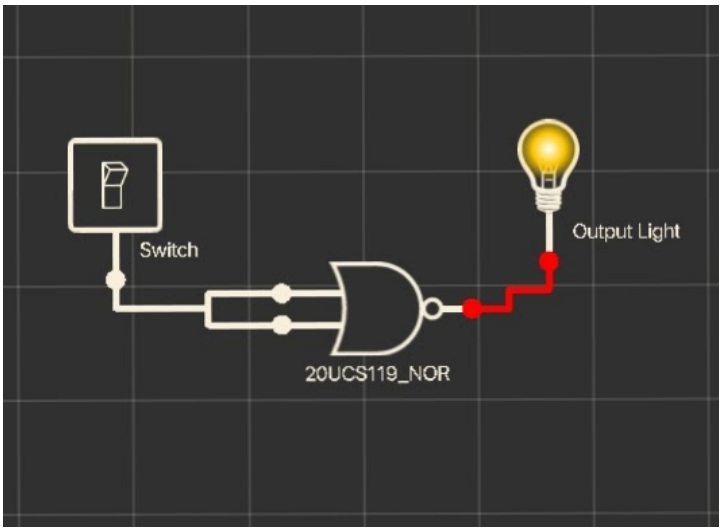
Equipments :
Logic Circuit Simulator Pro.

Procedure :

1. Firstly we have to install the LOGIC CIRCUIT SIMULATOR PRO APP and then after opening it, we have to make a new project.
2. Then we have to add the required elements to make the circuits i.e LOGIC gates, inputs, outputs etc.
3. Then connect the elements together with the help of a path to form a circuit and through the given gates we have to connect the input and output.
4. Lastly, we have to apply the TOUCH button to turn on/off the inputs i.e the switch by clicking on it, then we have to observe the output from the output section and verify the given truth tables.

1) Realization of all basic gates using NOR gate :

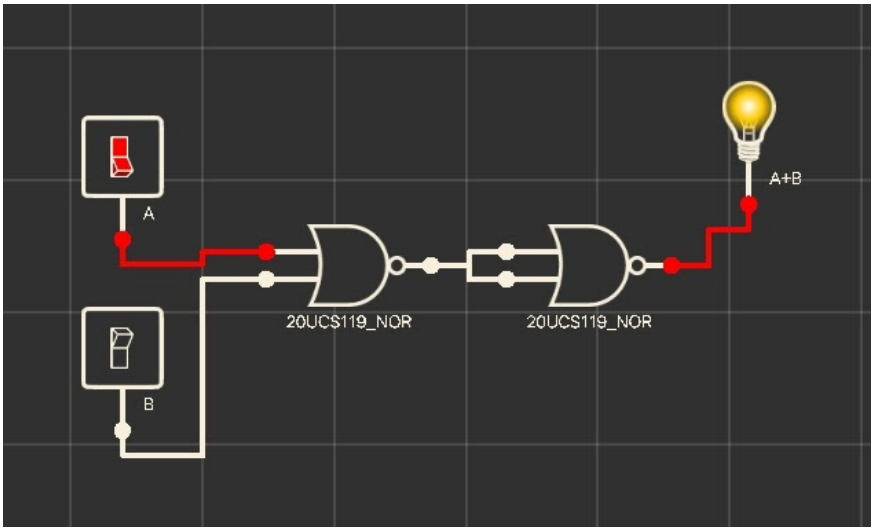
NOT GATE :



Truth Table	
A	A'
0	1
1	0

Figure : NOT gate

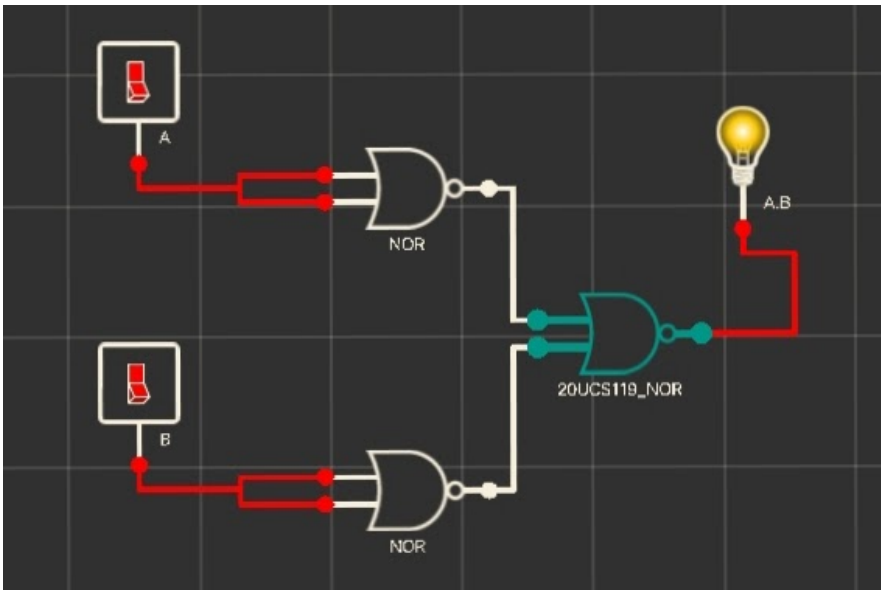
OR GATE:



Truth Table		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Figure : OR gate

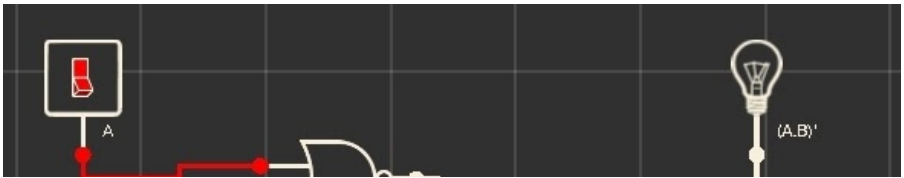
AND GATE:



Truth Table		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Figure : AND gate

NAND GATE:

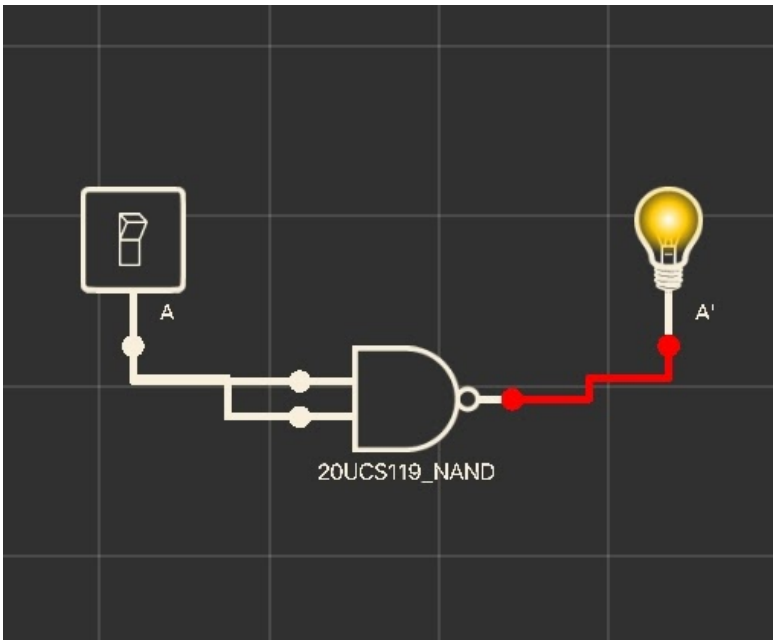


Truth Table		
A	B	$\overline{A.B}$

Figure : NAND gate

2) Realization of all basic gates using NAND gate :

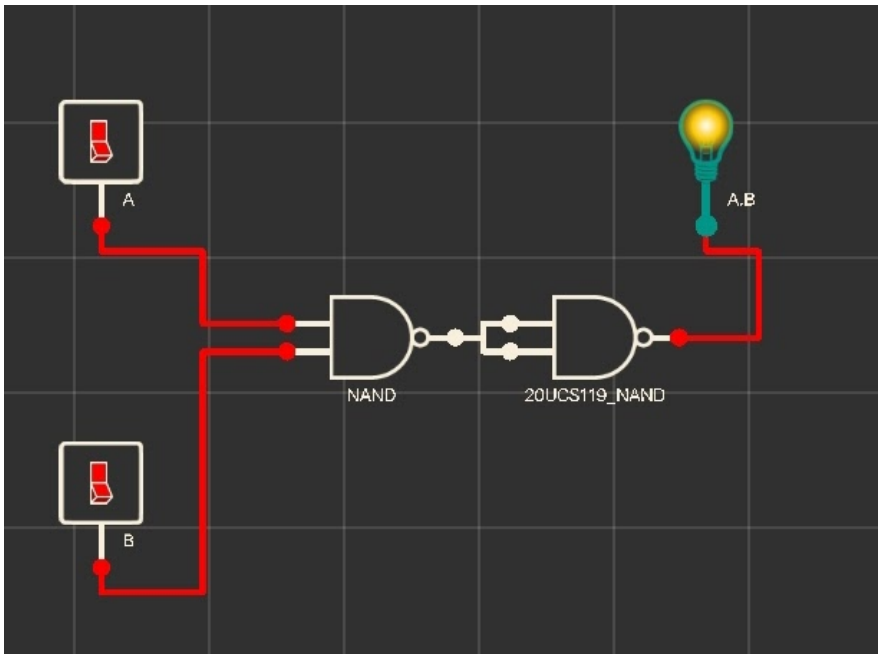
NOT GATE:



Truth Table	
A	A'
0	1
1	0

Figure : NOT gate

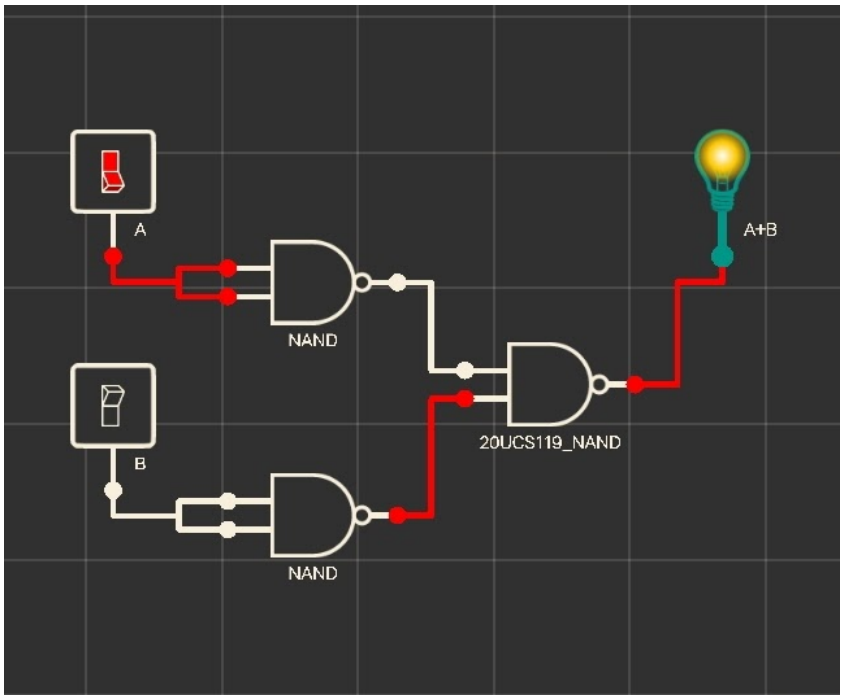
AND GATE:



Truth Table		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Figure : AND gate

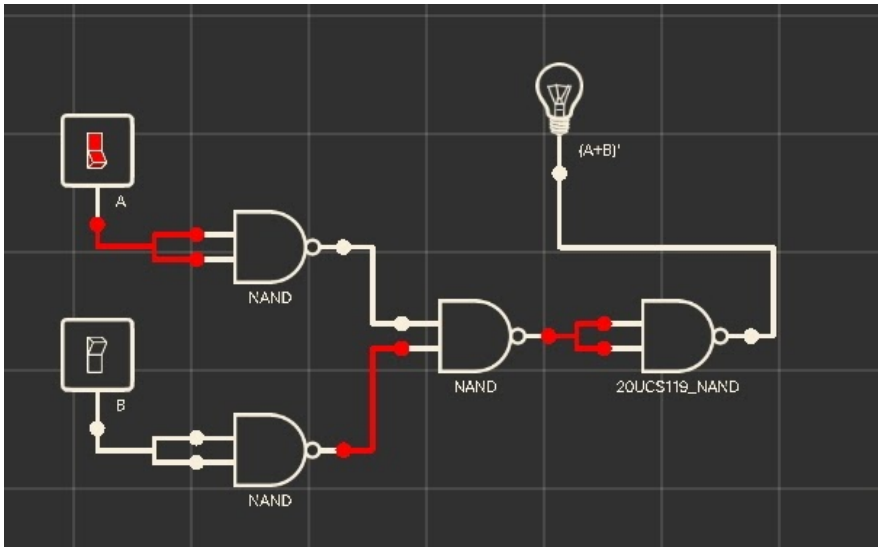
OR GATE:



Truth Table		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Figure : OR gate

NOR GATE:

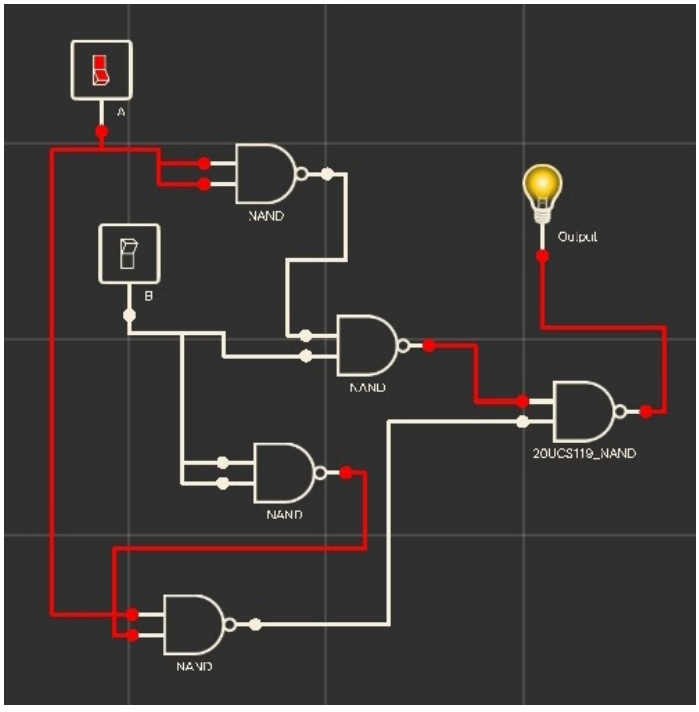


Truth Table		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Figure : NOR gate

3)Realization of XOR and XNOR gates using NAND and NOR gates :

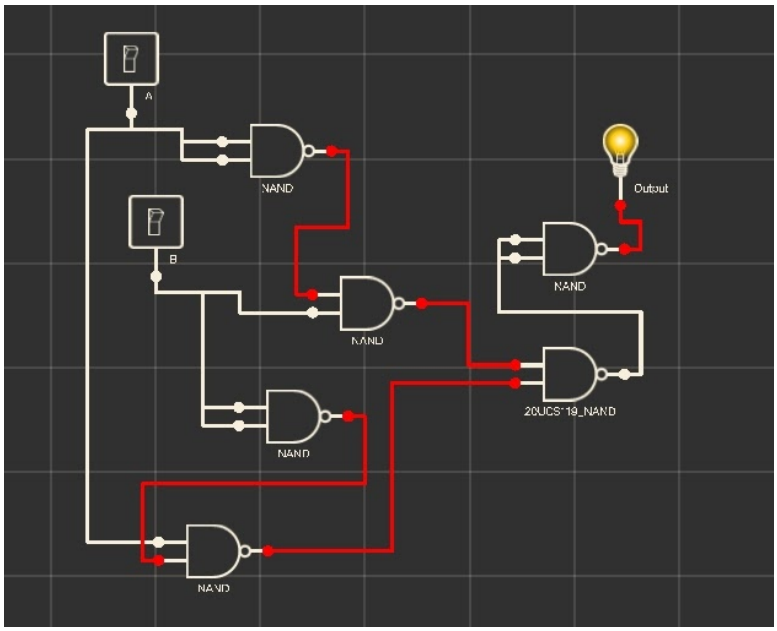
BASIC CONFIGURATION OF XOR GATE using NAND :



Truth Table		
A	B	XOR
0	0	0
1	0	1
0	1	1
1	1	0

Figure : XOR Gate using NAND

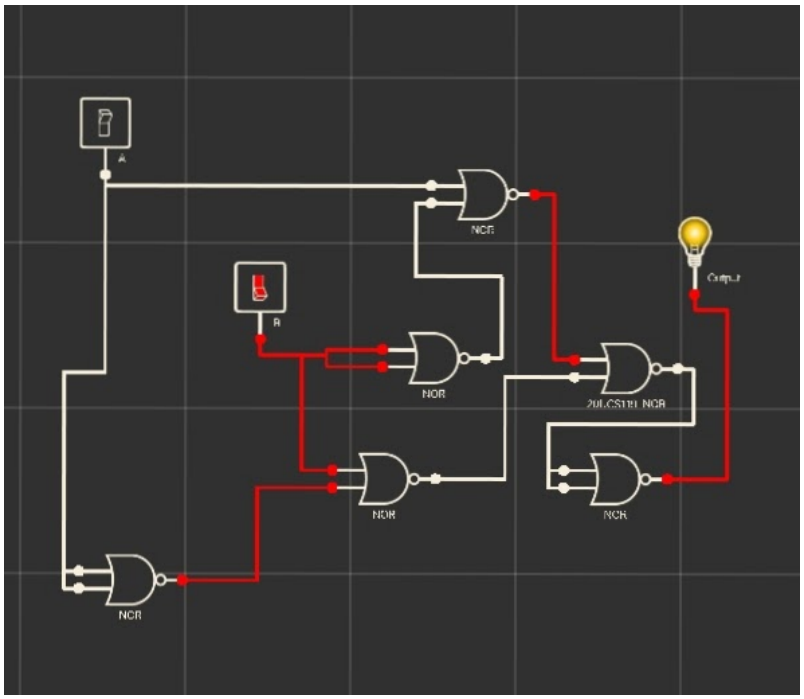
BASIC CONFIGURATION OF XNOR GATE using NAND :



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

Figure : XNOR Gate using NAND

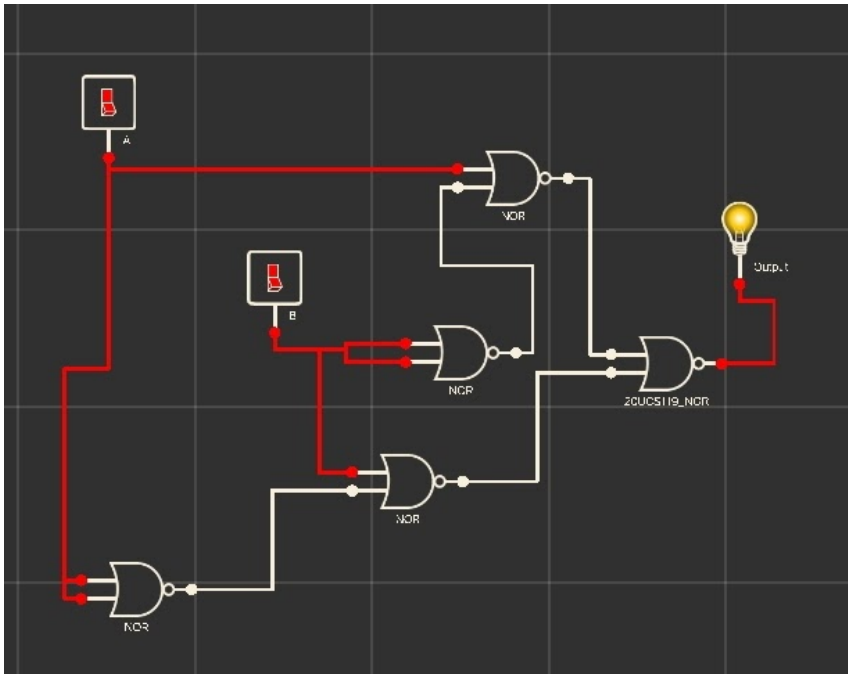
BASIC CONFIGURATION OF XOR GATE using NOR :



Truth Table		
A	B	XOR
0	0	0
1	0	1
0	1	1
1	1	0

Figure : XOR Gate using NOR

BASIC CONFIGURATION OF XNOR GATE using NOR :



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

Figure : XNOR Gate using NOR

Conclusion:

The truth tables for various digital gates like AND,OR ,NAND XNOR, and NOR are verified using the universal gates.

EXPT NO.3 :
STUDY OF DE-MORGAN'S THEOREM

Objective :

To study and verify de-morgan's theorem.

Equipments :

Logic Circuit Simulator Pro.

Theory :

A mathematician named DeMorgan developed a pair of rules regarding group complementation in Boolean algebra. By group complementation, represented by a long bar over more than one variable.

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. This theorems states the same equivalence in "backward" from: that inverting the output of any gate results in the same function as the opposite type of gate (AND vs OR) with inverted inputs :

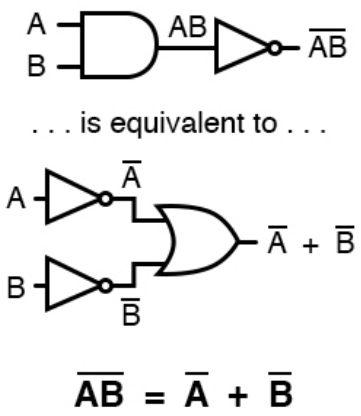
A long bar extending over the term AB acts as a grouping symbol, and as such is entirely different from the product of A and B independently inverted. In other words, (AB)' is not equal to A'B'. Because the "prime" symbol (') cannot be stretched over two variables like a bar can, we are forced to use parentheses to make it apply to the whole term AB in the previous sentence. A bar, however, acts as its own grouping symbol when stretched over more than one variable. This has a profound impact on how Boolean expressions are evaluated and reduced, as we shall see.

De Morgan's theorem may be thought of in terms of breaking a long bar symbol. When a long bar is broken, the operation directly underneath the break changes from addition to multiplication, or vice versa, and the broken bar pieces remain over the individual variables

Procedure :

THEOREM 1 : $\overline{AB} = \overline{A} + \overline{B}$

1. Do the connection as shown in the figure.
2. Connect A & B terminals to the logic inputs from input switches.
3. Connect both the outputs to led indicators in the Output section.
4. Provide different combinations of inputs A &B and observe the output on LEDs to verify the theorem.



Truth Table					
A	B	$\overline{AB}$	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
0	0	1	1	1	1
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	0	0	0

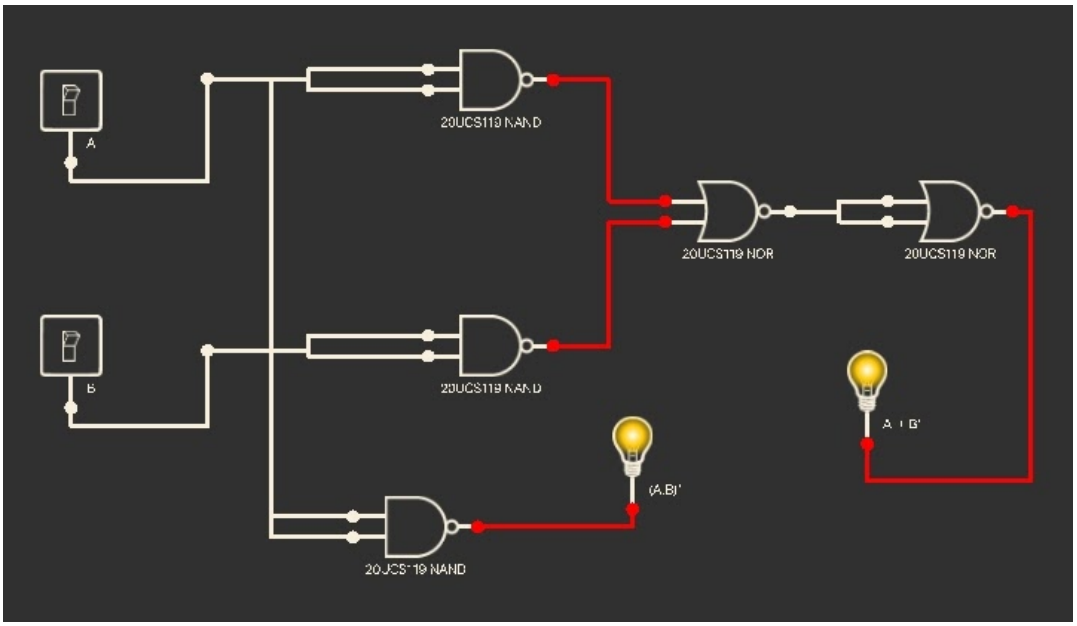


Figure : $\overline{AB} = \overline{A} + \overline{B}$

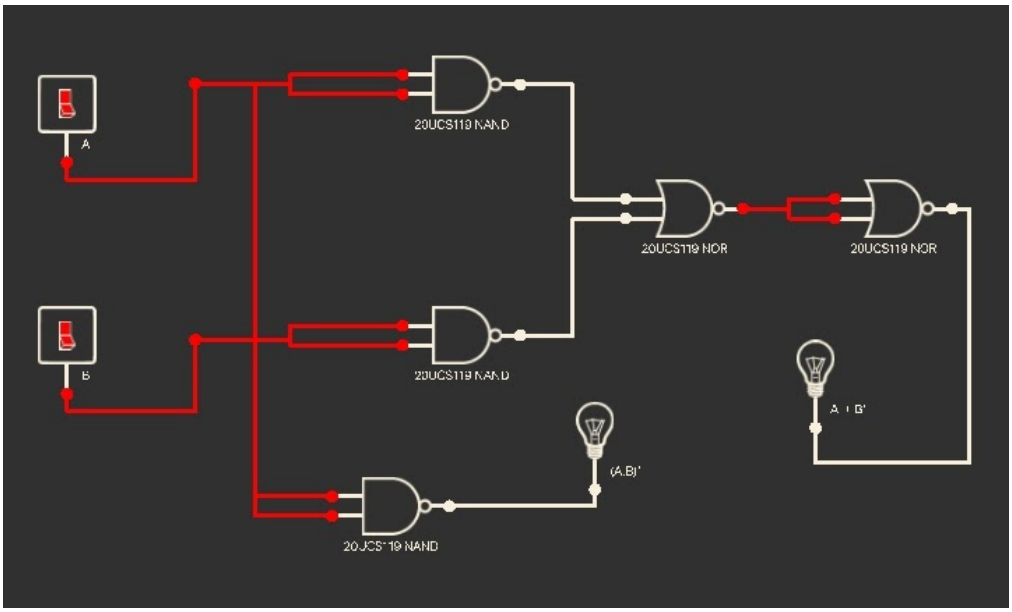


Figure : $\overline{AB} = \overline{A} + \overline{B}$

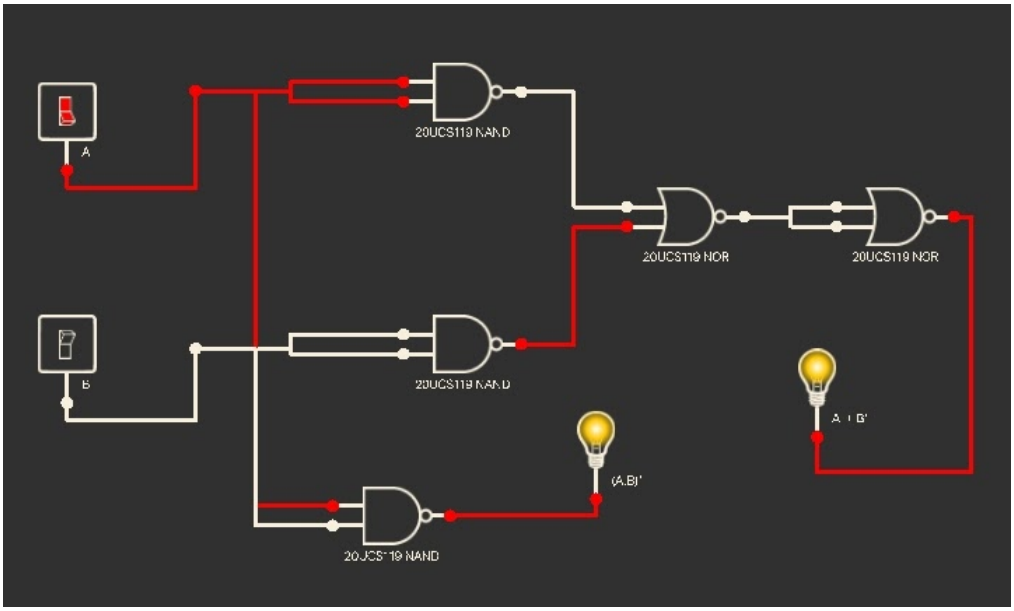
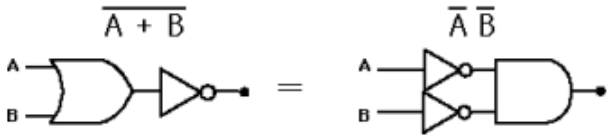


Figure : $\overline{AB} = \overline{A} + \overline{B}$

THEOREM 2 : $\overline{A+B} = \overline{A}.\overline{B}$

1. Do the connection as shown in the figure.
2. Connect A & B terminals to the logic inputs from input switches.
3. Connect both the outputs to led indicators in the Output section.
4. Provide different combinations of inputs A &B and observe the output on LEDs to verify the theorem.



Truth Table

A	B	$\overline{A}$	$\overline{B}$	A+B	$\overline{A+B}$	$\overline{A}.\overline{B}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

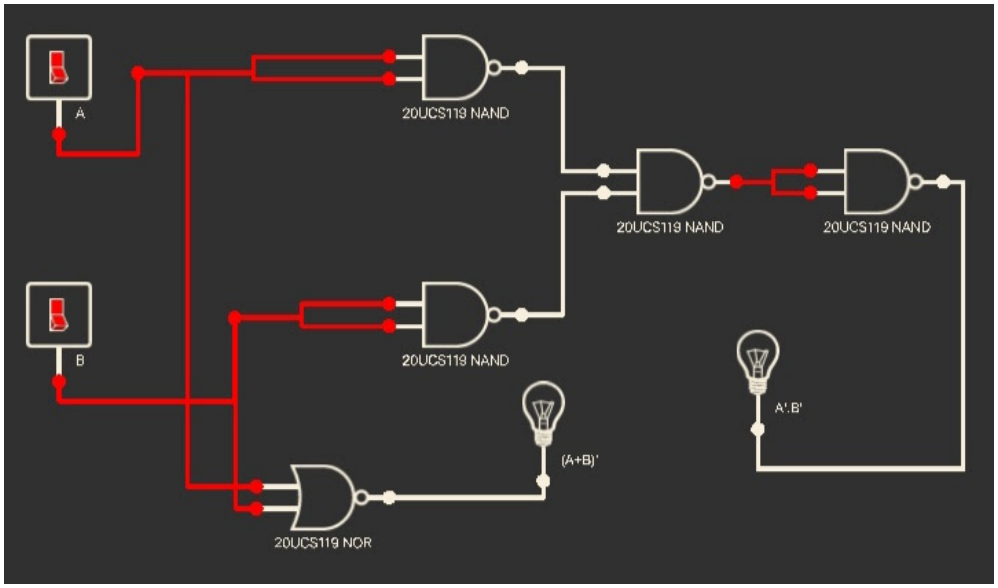


Figure : $\overline{A+B} = \overline{A}.\overline{B}$

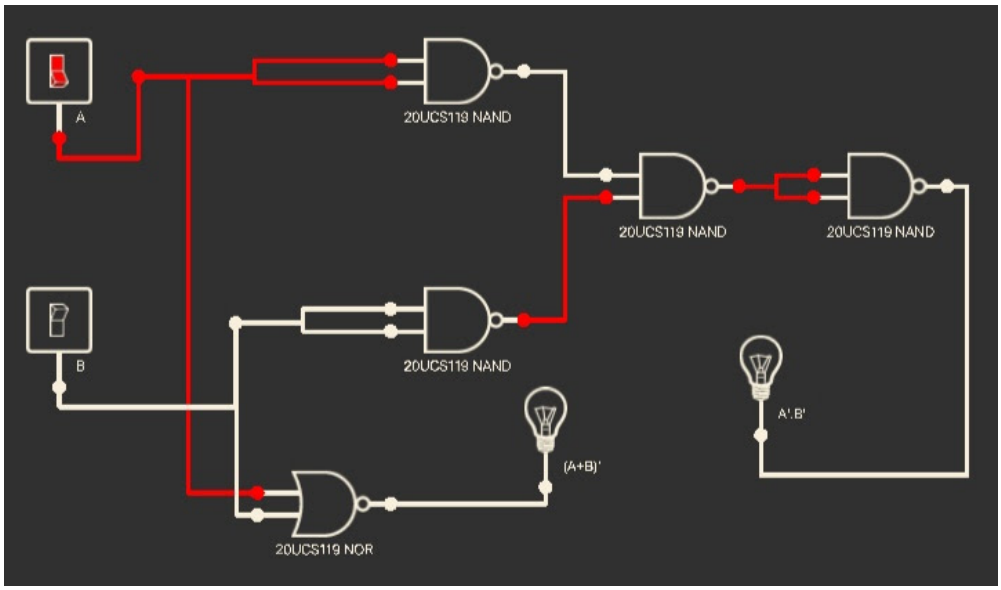


Figure : $\overline{A+B} = \overline{A}.\overline{B}$

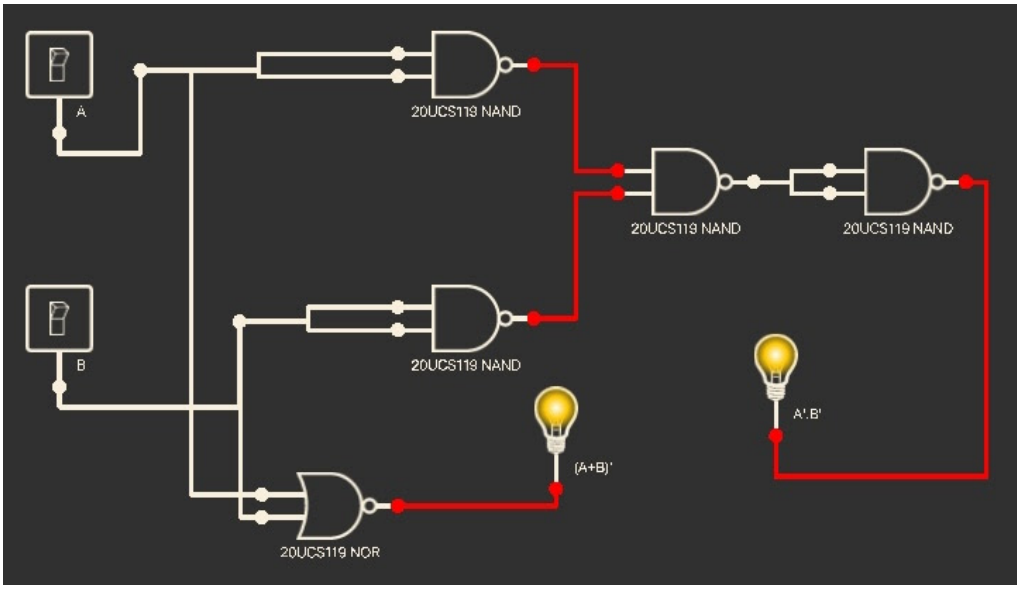


Figure : $\overline{A+B} = \overline{A}.\overline{B}$

Conclusion :

Hence, De-Morgan's theorem is verified.

EXPT NO.4 :
STUDY OF BOOLEAN EXPRESSION SIMPLIFICATION

Objective :

To study the Boolean rules and Boolean Expression simplification.

Equipments :

Logic Circuit Simulator Pro.

Theory :

A set of rules or Laws of Boolean Algebra expressions have been invented to help reduce the number of logic gates needed to perform a particular logic operation resulting in a list of functions or theorems known commonly as the Laws of Boolean Algebra. Boolean Algebra is the mathematics we use to analyse digital gates and circuits. Boolean Algebra is therefore a system of mathematics based on logic that has its own set of rules or laws which are used to define and reduce Boolean expressions.

List of some of the Boolean Laws are given below :-

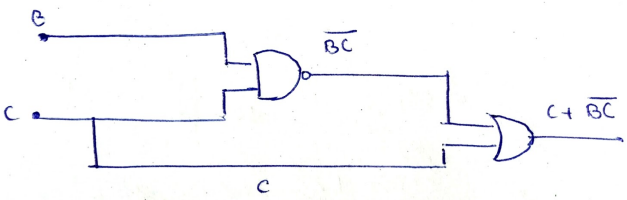
1. **The Idempotent Laws :**
   i) A.A=A ii) A+A=A
2. **The Associative Laws :**
   i) (AB)C = A(BC) ii) (A+B)+C = A+(B+C)
3. **The Commutative Laws :**
   i) AB = BA ii) A+B = B+A
4. **The Distributive Laws :**
   i) A(B+C) = AB + AC ii) A+BC = (A+B)(A+C)
5. **The Identity Laws :**
   i) AF = F, AT = A ii) A+F = A, A+T = T
6. **The Complement Laws :**
   i) A.$\overline{A}$ = 0 ii) A+$\overline{A}$ = 1
7. **The involution Law :**
   i) $\overline{\overline{A}}$ = A
8. **The De-Morgan's Laws :**
   i) $\overline{A+B} = \overline{A} + \overline{B}$ ii) $\overline{(A.B)} = \overline{A} + \overline{B}$

Simplification of some Boolean expressions and their verification :-

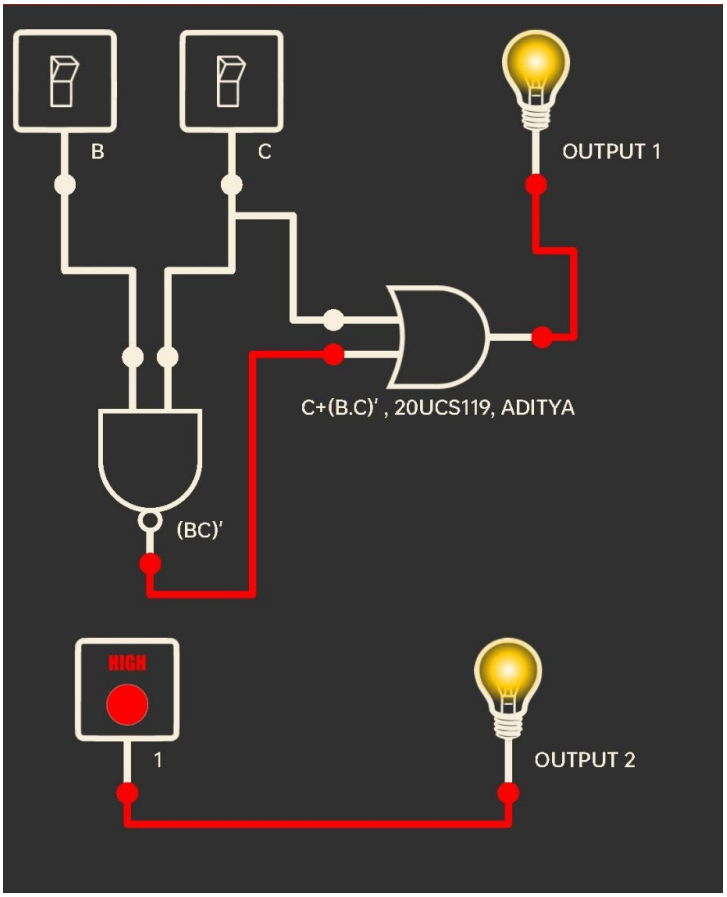
1. Simplify: C + $(BC)'$:

Expression	Rule(s) Used
C + (BC)′	Original Expression.
C+ (B' + C')	DeMorgan's Law.
(C+C') + B'	Commutative, Associative Laws.
T+ B'	Complement Law.
T	Identity Law.

Circuit Diagram :-



Logic diagram and simplified :-

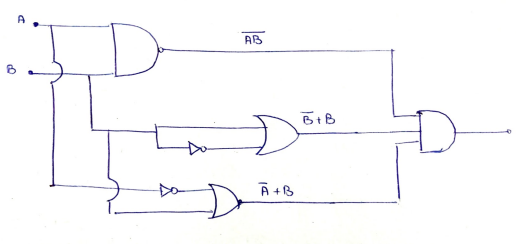


TRUTH TABLE		
B	**C**	**C+ (BC)'**
0	**0**	**1**
1	**0**	**1**
0	**1**	**1**
1	**1**	**1**

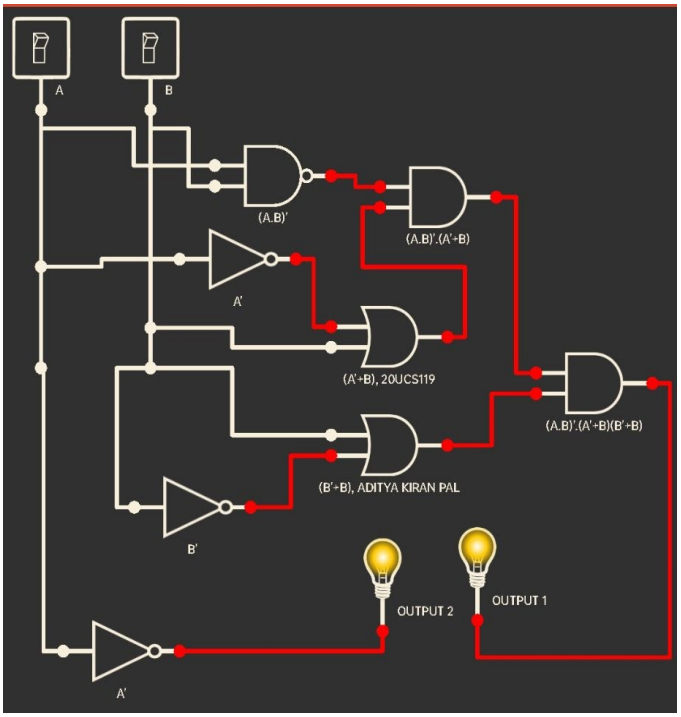
2. Simplify: (*AB*)' + (*A*' + B) (*B*' + B)

Expression	Rules used
(*AB*)'+ (*A*' + B) (*B*' + B)	Original Expression
A'*B*'(*A*' + B)	Complement law, Identity law
(*A*' + *B*')(*A*' + B)	DeMorgan's Law
A' + *B*'B	Distributive law.
This step uses the fact that or distributes over and. It can look a bit strange since addition does not distribute over multiplication.	
A	Complement. Identity.

Circuit Diagram :-



Logic diagram and simplified :-



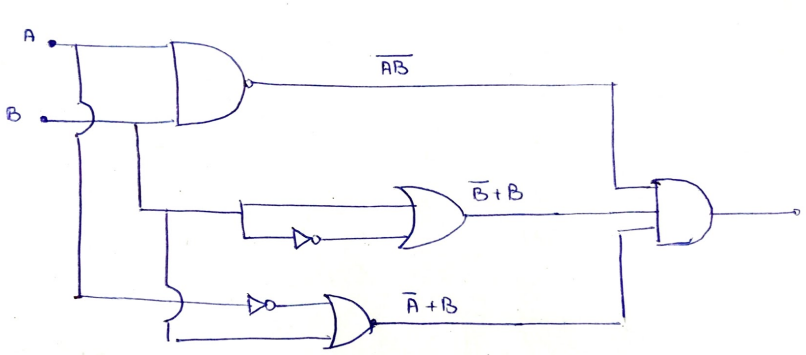
TRUTH TABLE			
A	B	A'	(AB)'(A' + B)(B'+B)
0	0	1	1
1	0	0	0
0	1	1	1
1	1	0	0

3. Simplify: (A+C)(AD +A*D*′) + AC + C:

Expression	Rule(s) Used
(A+C)(AD +A*D*′) + AC + C:	Original Expression
(A+C)A(D +*D*′) + AC + C:	Distributive.
(A + C)A + AC + C	Complement, Identity.
A((A+C) + C) +C	Commutative. Distributive.
A(A+C) + C	Associative. Idempotent
AA+AC+C	Distributive
A+ (A + T)C	Idempotent, Identity, Distributive.
A+C	Identity, twice.

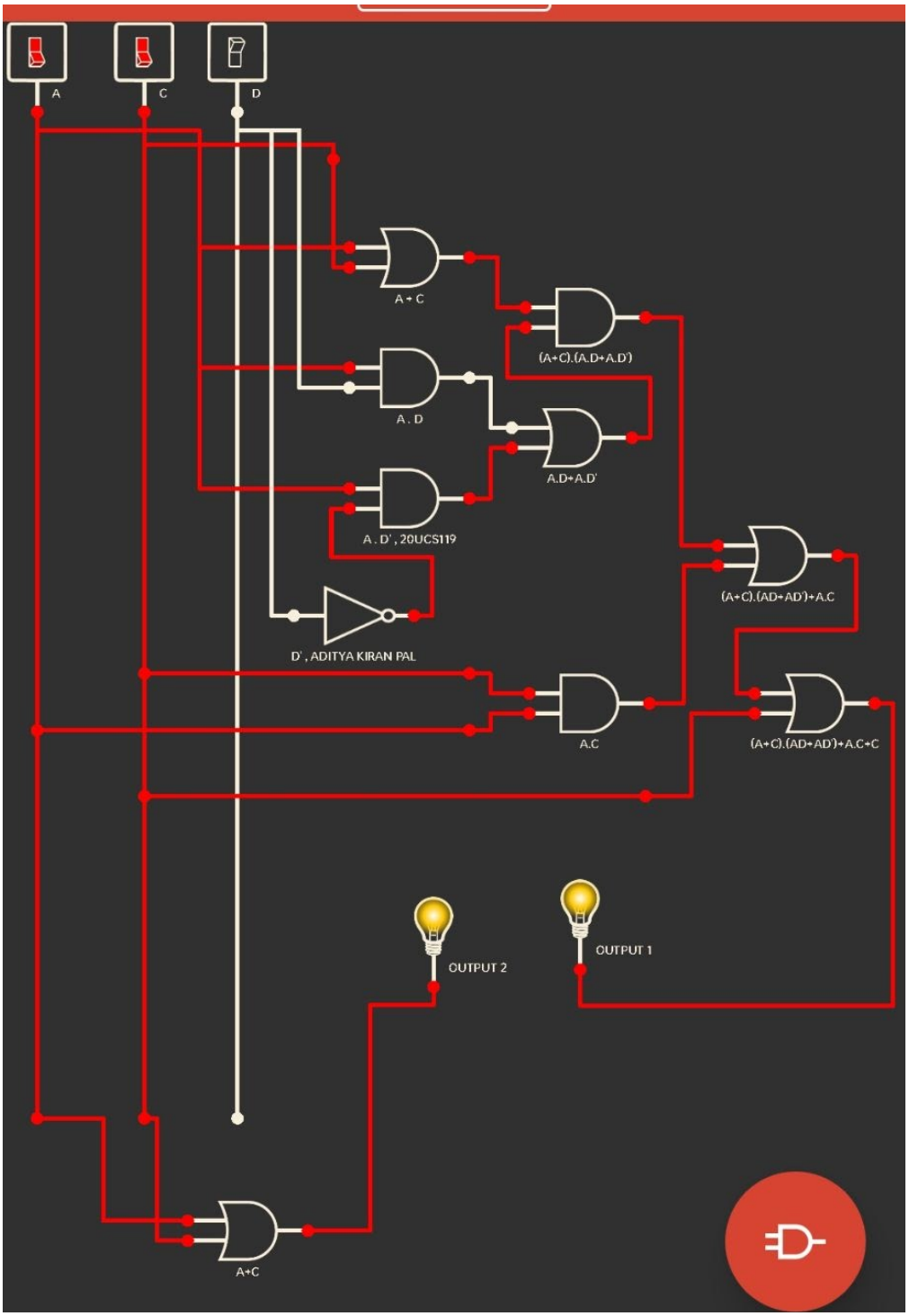
You can also use distribution of or over and starting from A(A+C) +C to reach the same result by another route.

Circuit Diagram :-



Truth table				
A	C	D	A+C	(A + C)(AD + AD') + AC + C
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
1	0	0	1	1
1	1	0	1	1
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

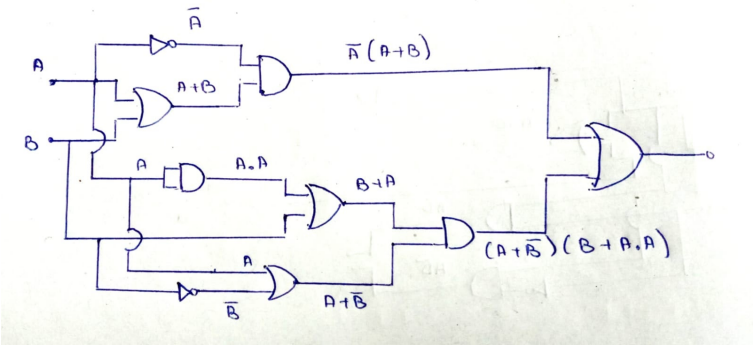
Logic diagram and simplified :-



4. Simplify: A' (A+B) + (B + AA) (A + B'):

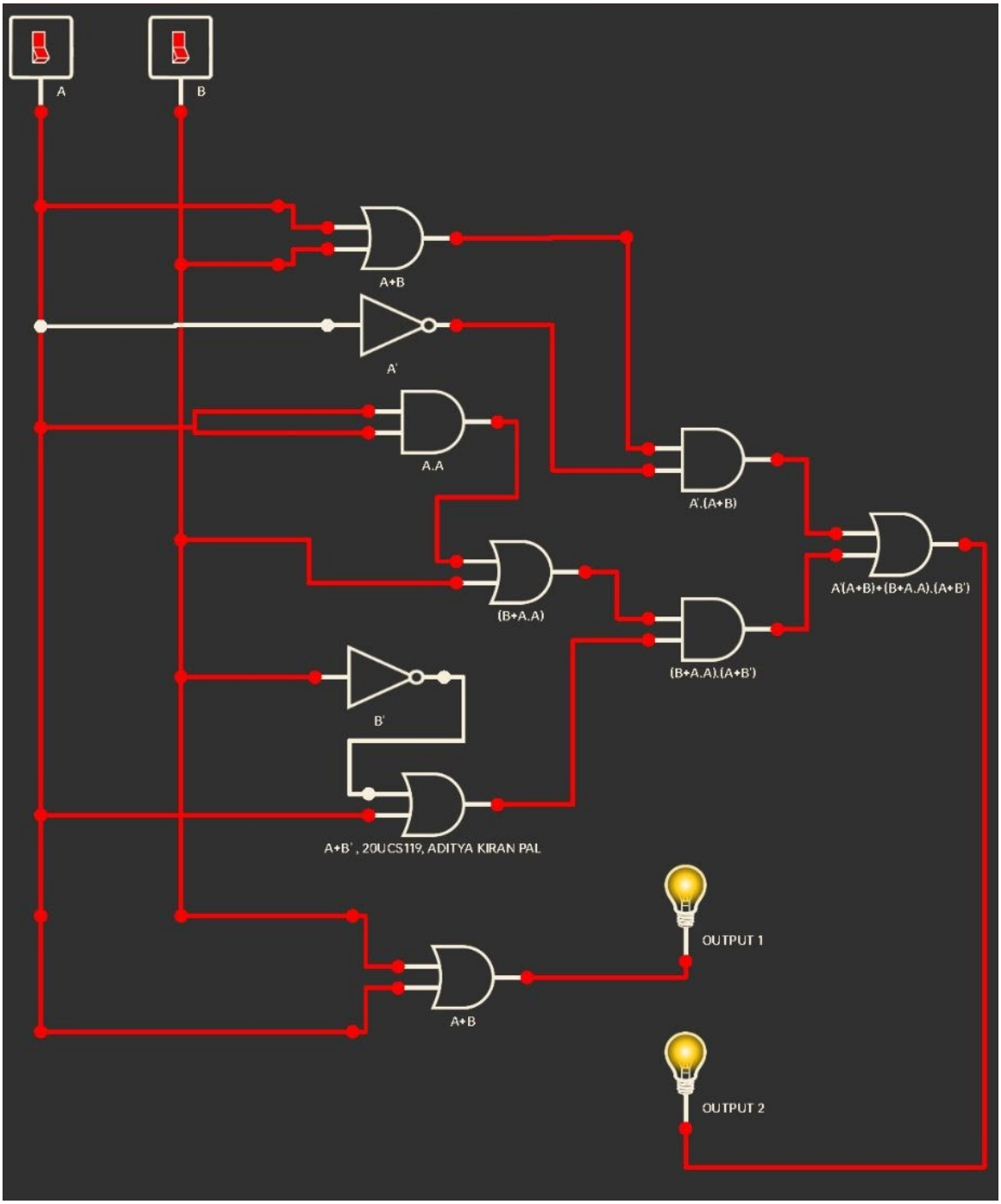
Expression	Rule(s) Used
A'(A+B) + (B + AA) (A + B'):	Original Expression
A'A+A'B+ (B +A) A+(B+A) B'	Idempotent (AA to A), then Distributive, used twice.
AB + (B + A) A+ (B + A) B	Complement, then Identity. (Strictly speaking, we also used the Commutative Law for each of these applications.)
A'B+BA+AA + B B'+ AB'	Distributive, two places AB+BA+A+AB Idempotent (for the A's), then Complement and Identity to remove BB.
A'B + AB+AT+AB'	Commutative, Identity; setting up for the next
A'B + A(B+T+B')	Distributive.
A'B+A	Identity, twice (depending how you count it).
A+ A'B	Commutative.
(A+ A')(A+B)	Distributive.
A+B	Complement, Identity

Circuit Diagram :-



TRUTH TABLE			
A	B	A+B	A' (A + B) + (B +AA)(A+B')
0	0	0	0
1	0	1	1
0	1	1	1
1	1	1	1

Logic diagram and simplified :-

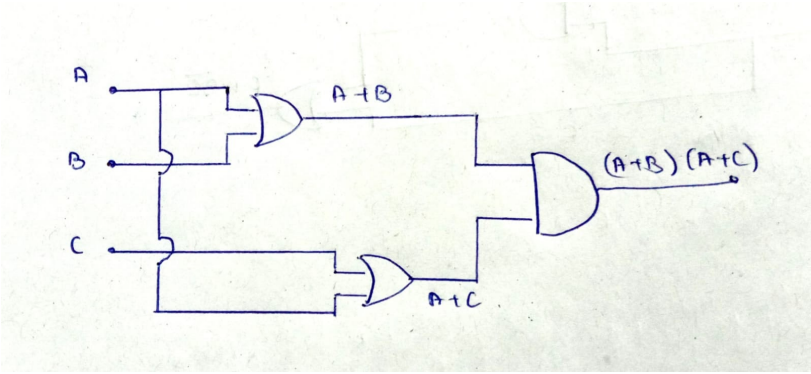


5. Simplify: (A+B). (A+C)

Expression	Rules Used
A. A+A.C + A.B + B.C	Distributive law
A+A.C + A.B +B.C	Idempotent AND law
(A.A = A) A(1+C) + A. B+B.C	Distributive law
A.1 + A. B+B.C	Identity OR law
(1+C = 1) A(1+B) + B.C	Distributive law
A.1 +B.C	Identity OR law
(1+ B = 1) Q= A + (B.C)	Identity AND law
(A.1 = A)	

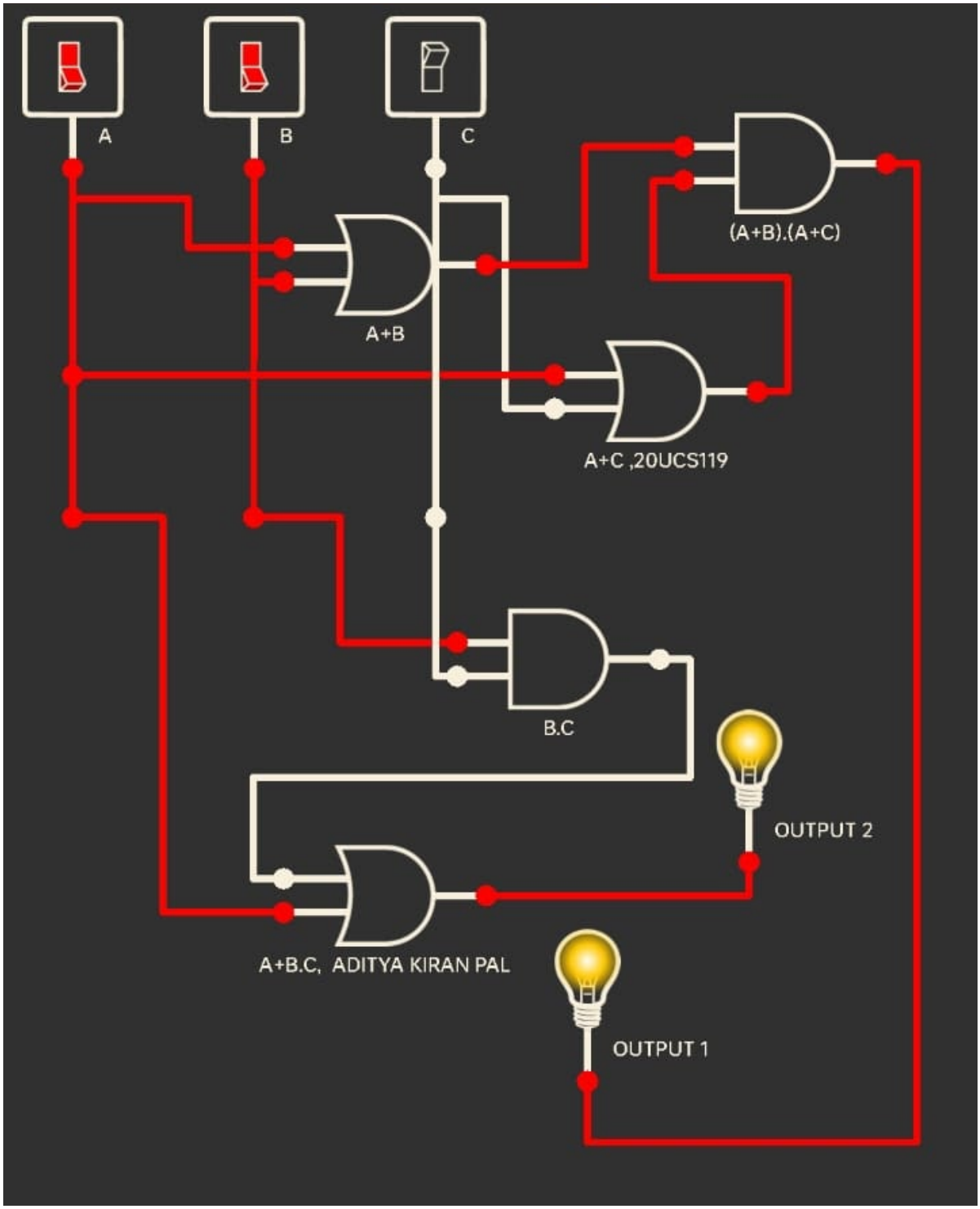
Expression: (A + B) (A+C) can be simplified to A+ (B.C) as in the Distributive law.

Circuit Diagram :-



TRUTH TABLE				
A	B	C	A + BC	(A+ B)(A+C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
1	0	0	1	1
1	1	0	1	1
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

Logic diagram and simplified :-



6. Simplify: AB+A(B+C) +B(B+C)

Step 1: Apply the distributive law

AB + AB + AC + BB + BC {Distributive law; A (B+C) = AB+AC, B (B+C) = BB+BC}

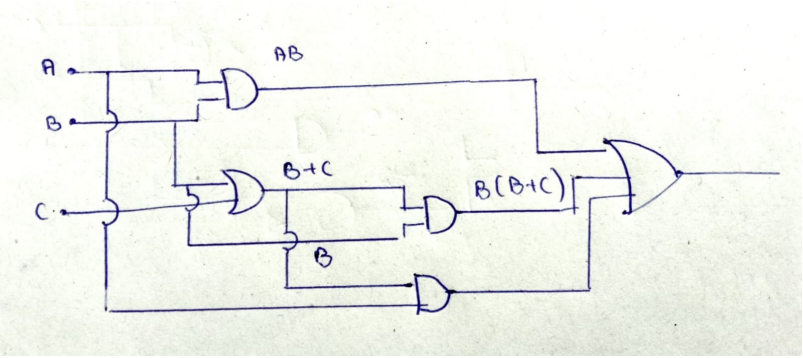
Step 2: Apply the idempotent law AB + AB + AC + B + BC {Idempotent law; BB = B}

Step 3: Apply the idempotent law AB + AC + B + BC {Idempotent law; AB+AB = AB}

Step 4: Apply the absorption law AB + AC +B {Absorption law; B+BC = B}

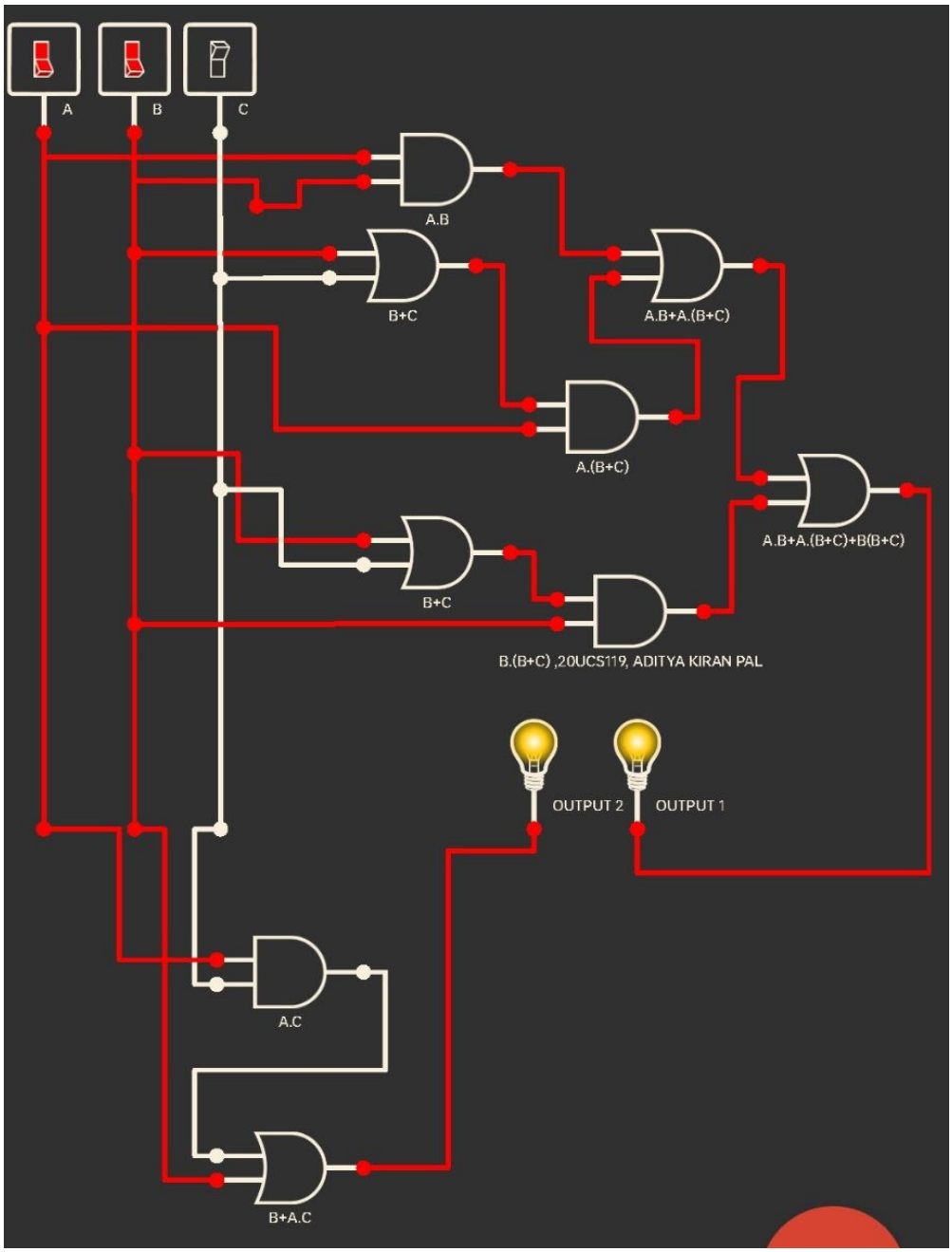
Step 5: Apply the absorption law B + AC {Absorption law; AB+B = B} Hence, the simplified Boolean function will be B + AC.

Circuit Diagram :-



TRUTH TABLE				
A	B	C	B+AC	AB + A(B + C) + B(B+C)
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
1	0	0	0	0
1	1	0	1	1
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

Logic diagram and simplified :-



Conclusion :

All the basic rules of boolean algebra are verified.

EXPT NO.5 :
TO STUDY HALF ADDER AND FULL ADDER

Objective :

To study the half adder, full adder, half subtractor and full subtractor.

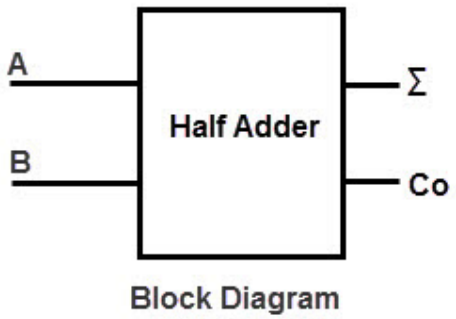
Equipments :

Logic Circuit Simulator Pro.

Theory :

Half Adder :

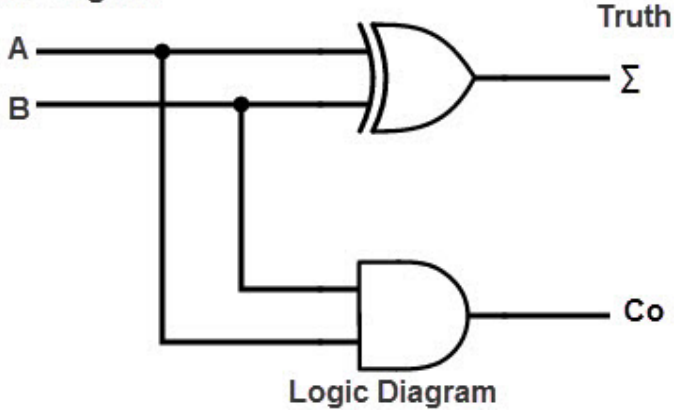
A half adder has two inputs for the two bits to be added and two outputs one from the sum 's' and other from the carry 'c' into the higher adder position. Above circuit is called as a carry signal from the addition of less significant bits sum from the X-OR gate and the carry out from the AND gate.



Block Diagram

A	B	Sum (∑)	Carry Out (Co)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth Table

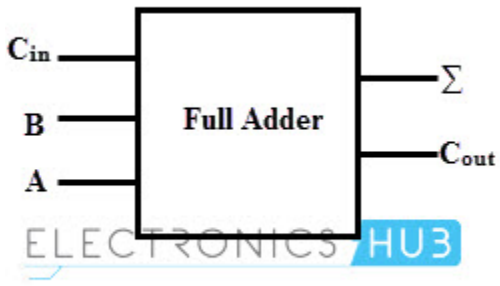


Logic Diagram

Sum (s) = $\bar{A}B + A\bar{B}$, Carry (c) = A.B

Full Adder :

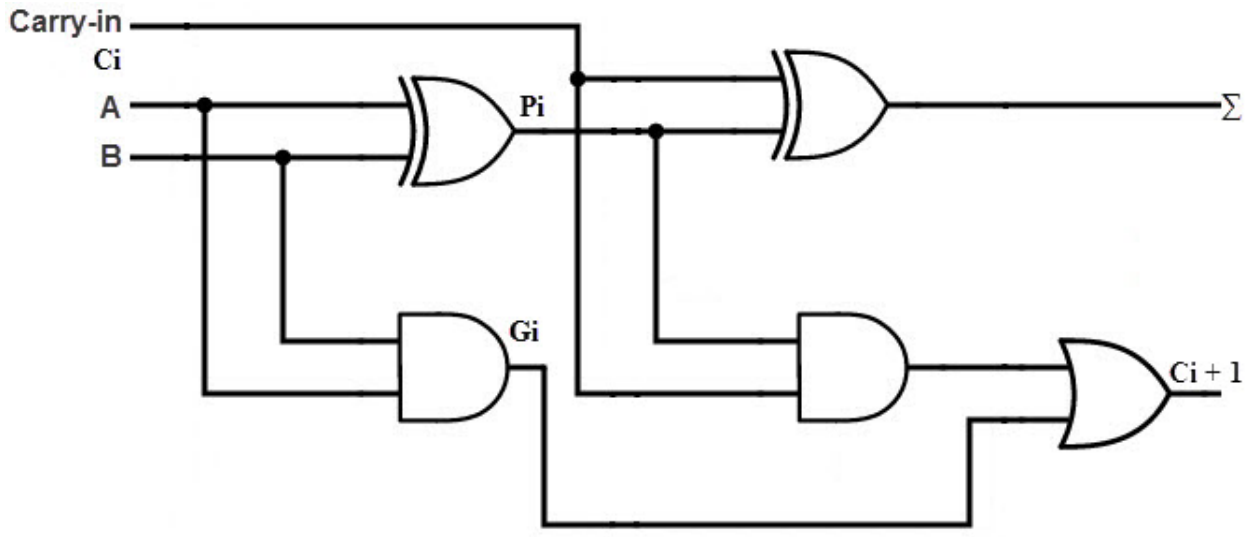
A full adder is a combined circuit that forms the arithmetic sum of input; it consists of three inputs and two outputs. A full ladder is used to add two three bits at a time but a half Adder cannot do so. In full adder sum output will be taken from X-OR gate, carry output will be taken from OR gate.



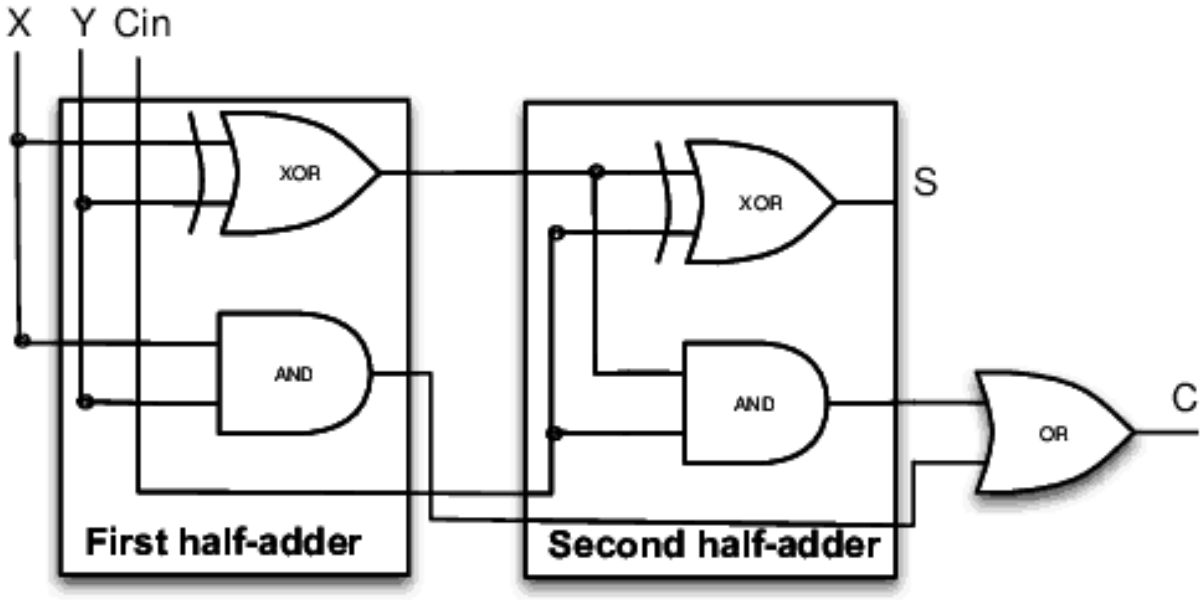
Block Diagram of Full Adder

C_{in}	B	A	∑	C_{out}
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**

Truth Table



Full Adder using two Half Adder :

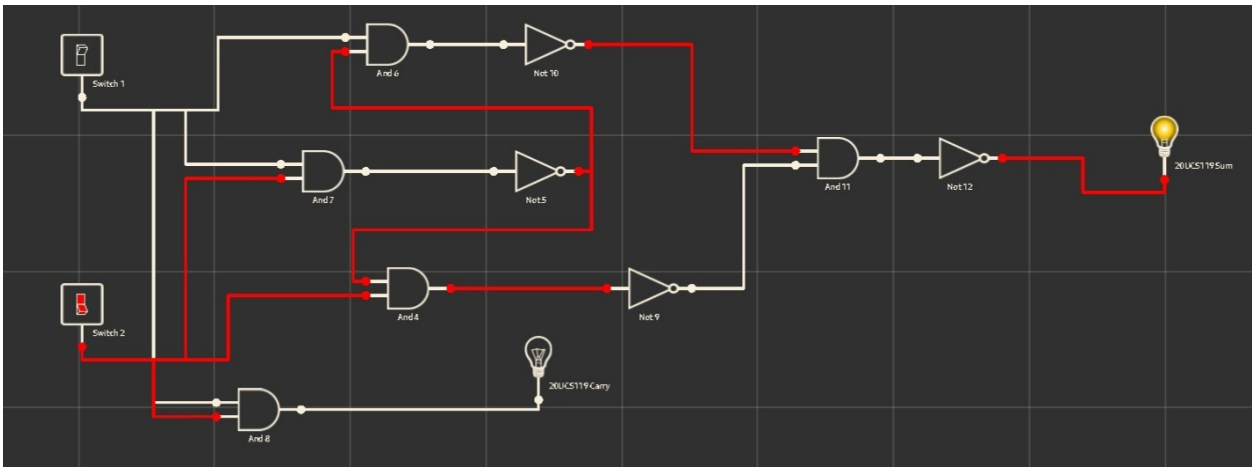


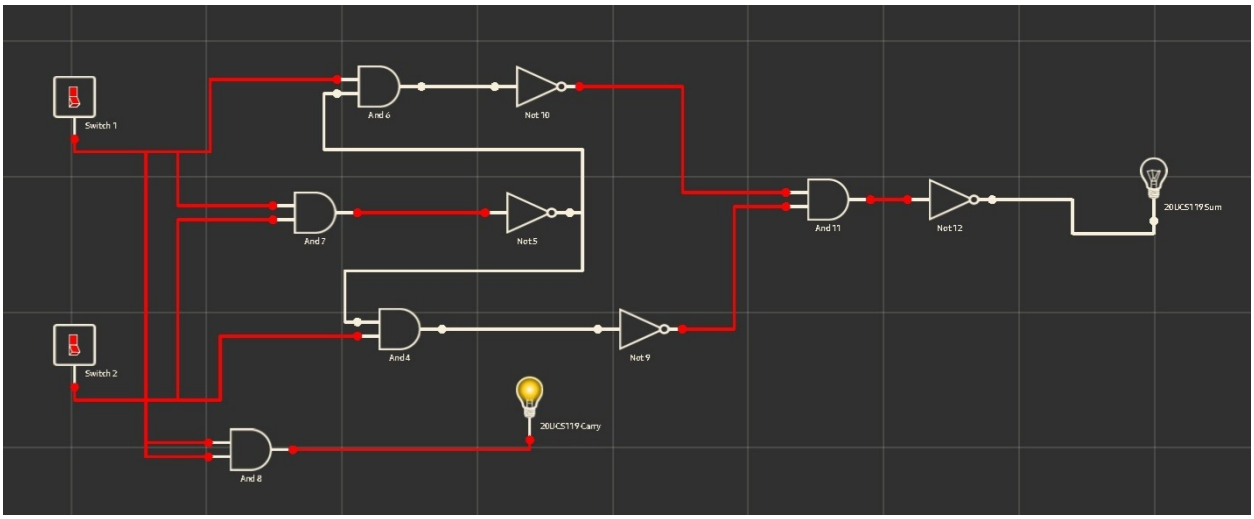
The sum in case of full adder is: **A ⊕ B ⊕ C**
And, the carry can be **"AB+BC+AC "or "(A⊕ B)C +AB"**

Procedure :

1. Half Adder :

- Connect A and B input of half adder to switches from the input switches section.
- Connect the sum output of half adder to L1 and carry output of half adder to L2 in the output section.
- Switch on the power supply of the kit and provide proper inputs to half adder using switches as per truth table shown above.
- Observe the output of half adder and verify the functionality of half adder as per the truth table.

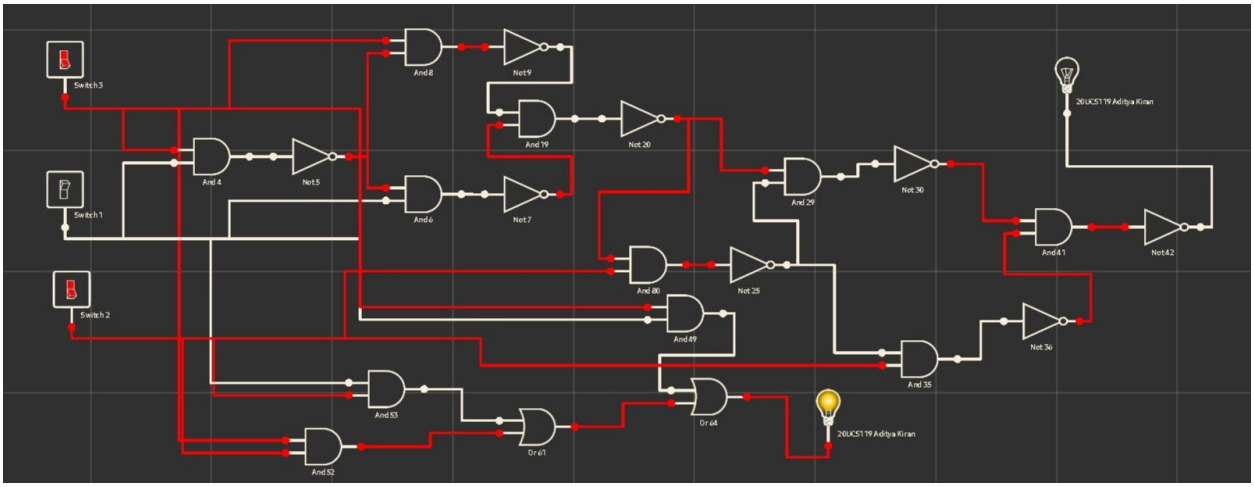


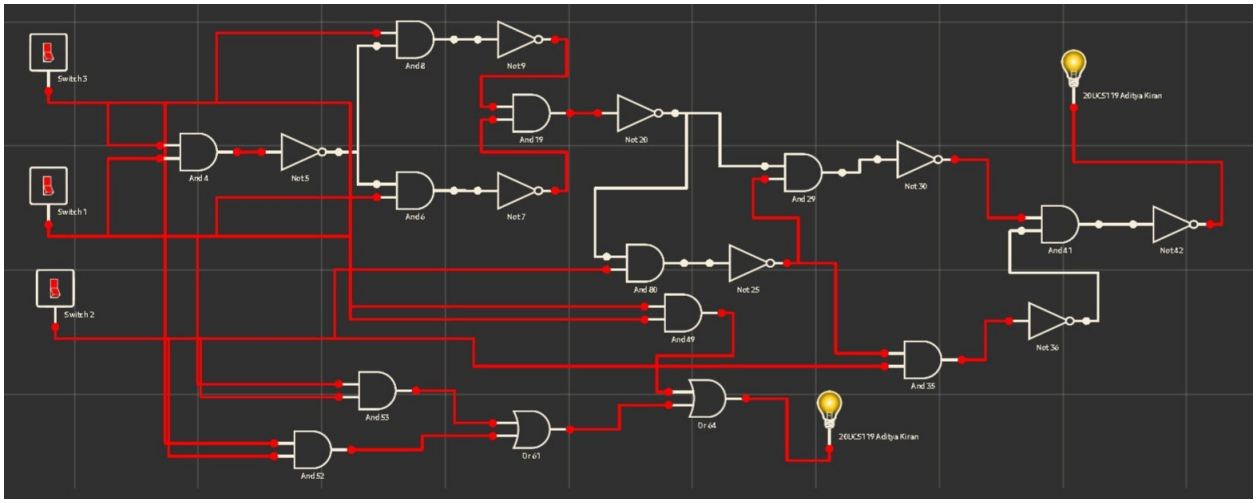


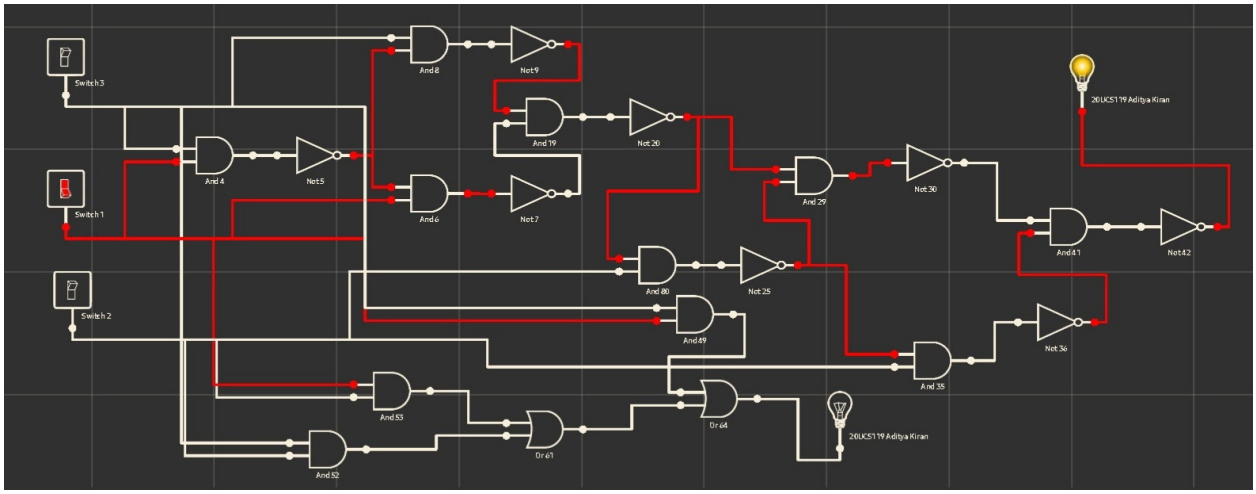
2. FULL ADDER :

- Connect A, B and C input of full adder to switches from the input switches section.
- Connect the sum output of full adder to L1 and carry output of half adder to L2 in the output section.
- Switch on power supply of the kit and provide proper inputs to full adder using switches as per truth table shown above.
- Observe the output of full adder and verify the functionality of half adder as per truth table.

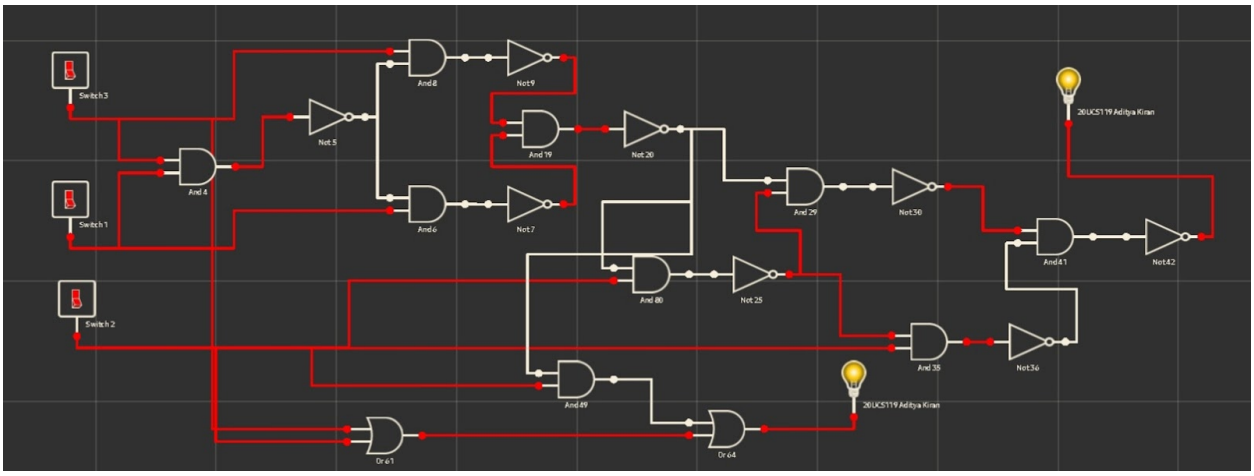
CASE 1: When C-out = AB+BC+AC:

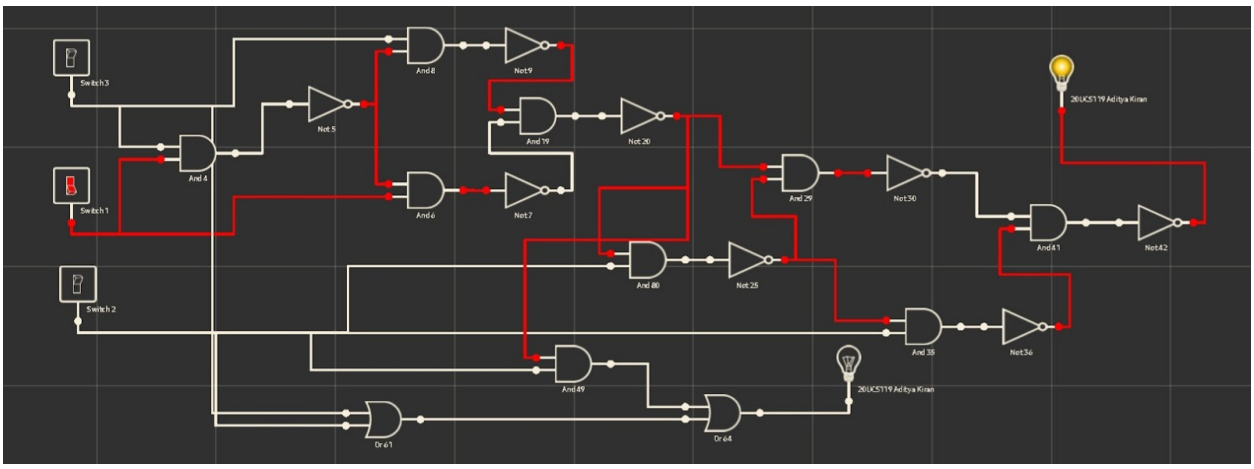


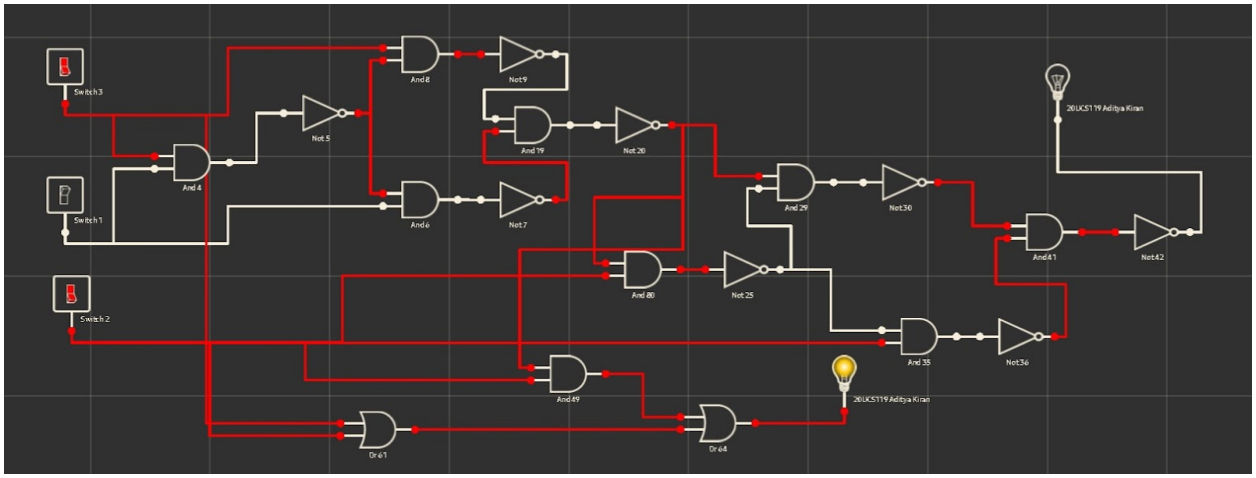




CASE 2: When C-out = (A ⊕ B)C + AB:







Truth Table :

HALF ADDER:

A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

FULL ADDER:

A	B	Cin	CARRY	SUM
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

Conclusion :

Hence the functionality of half adder and full adder are verified.

EXPT NO.6 :
STUDY OF HALF SUBTRACTOR AND FULL SUBTRACTOR

Objective :

To study the half subtractor and full subtractor.

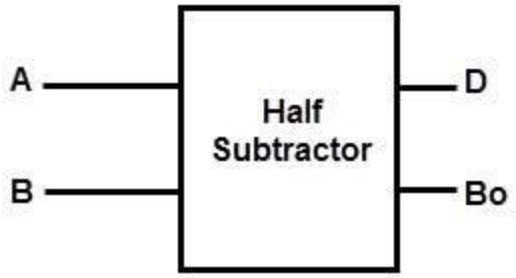
Equipments :

Logic Circuit Simulator Pro.

Theory :

Half Subtractor :

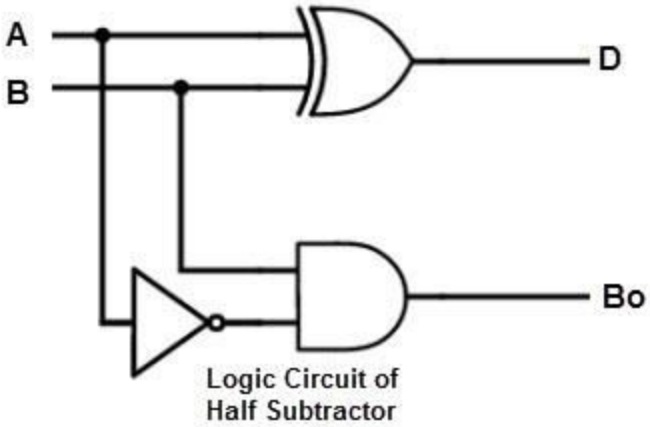
The half subtractor is constructed using X-OR and AND. The half subtractor has two inputs and outputs. The outputs are borrow and difference. The difference can be applied using X-OR gate, borrow out can be implemented using AND gate and an inverter.



Block Diagram

A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Truth Table

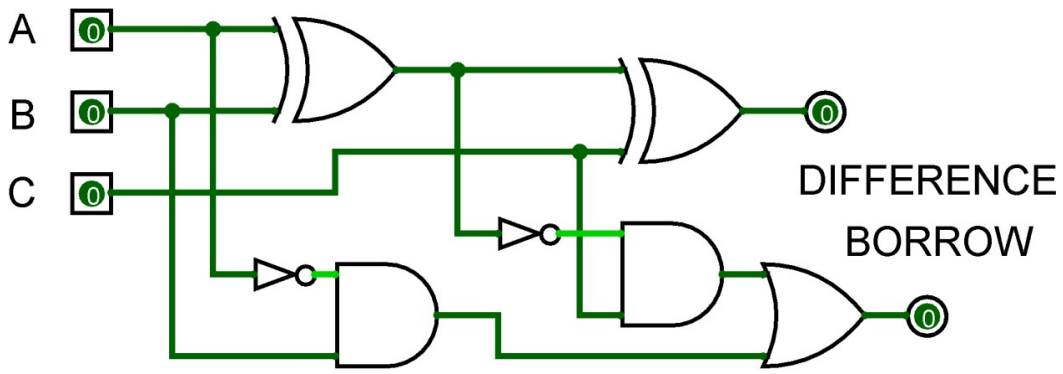


Logic Circuit of Half Subtractor

Here D = Difference and Bo = Borrow
Difference (D) = $\bar{A}B + A\bar{B}$, Borrow (B) = A.B

Full Subtractor :

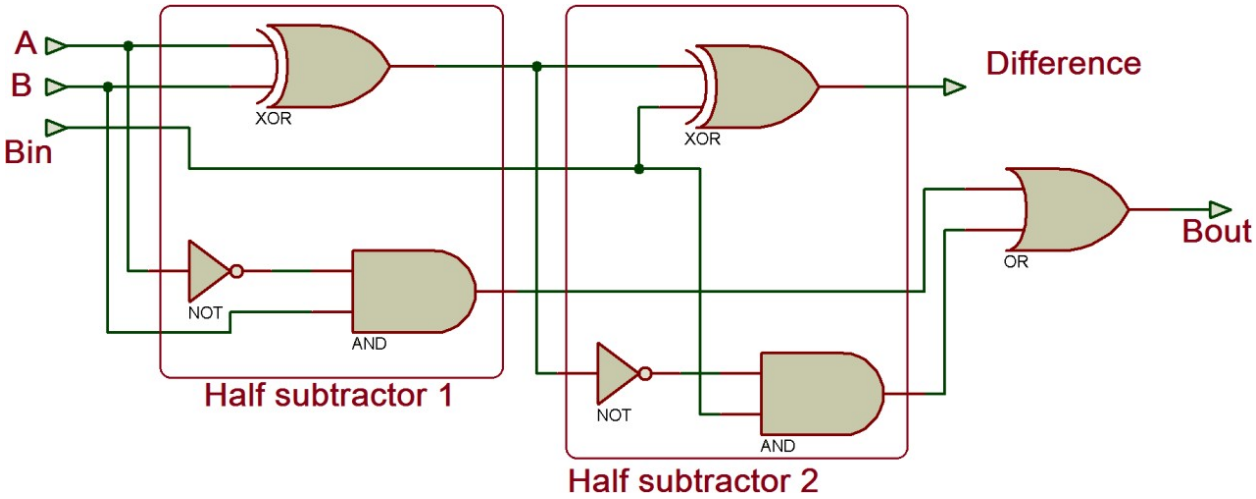
The full subtractor is a combination of X-OR, AND, OR, NOT gates. In a full subtractor the logic circuit has three inputs and two outputs. The two half subtractors put together gives a full subtractor.



TRUTH TABLE:

Full Subtractor-Truth Table				
Input			Output	
A	B	C	Difference	Borrow
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
www.flintgroups.com				

Full Adder using two Half Subtractor :

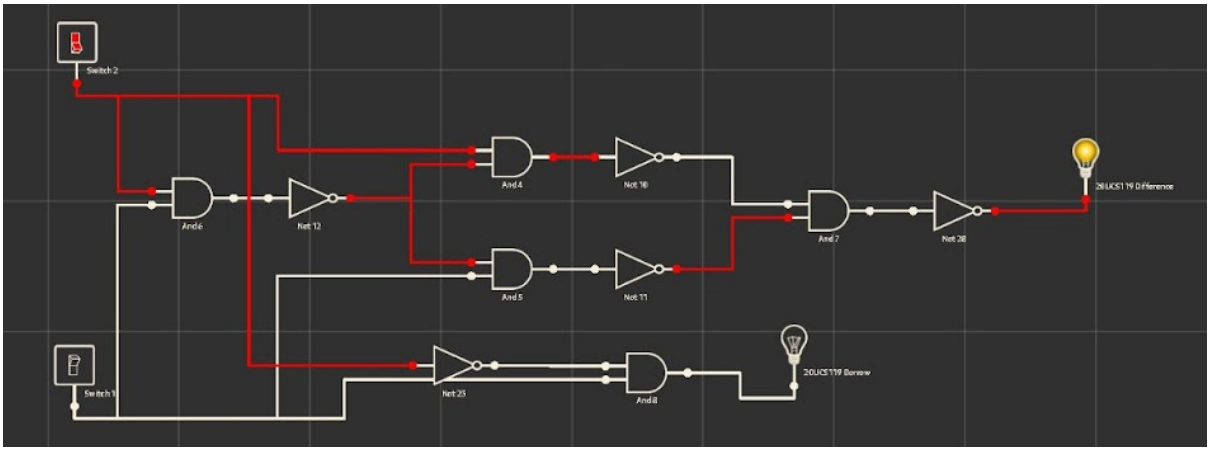


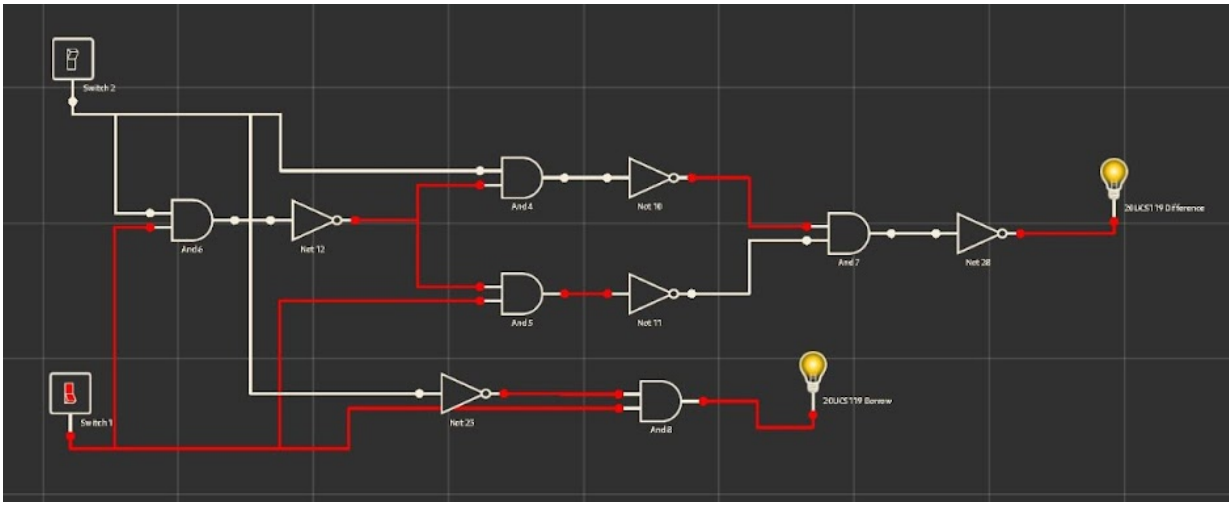
The difference in case of full adder is: A ⊕ B ⊕ B_{in} And the carry can be "$\bar{A}$B+BB_{in}+$\bar{A}B_{in}$"or "(A⊕ B)B_{in}+$\bar{A}$B

Procedure :

1. **Half Subtractor :**

- Connect A and B input of half subtractor to switches from the input switches section.
- Connect difference output of half subtractor to L1 and borrow output of half subtractor to L2 in output section
- Switch on the power supply of the kit and provide proper inputs to half subtractor using switches as per truth table shown above.
- Observe the output of half subtractor and verify the functionality of half subtractor as per the truth table.





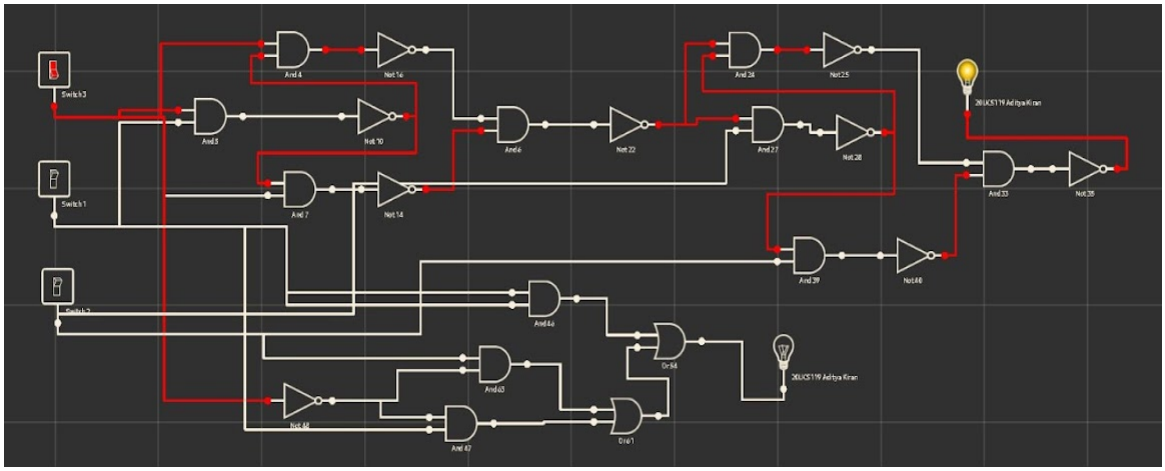
TRUTH TABLE:

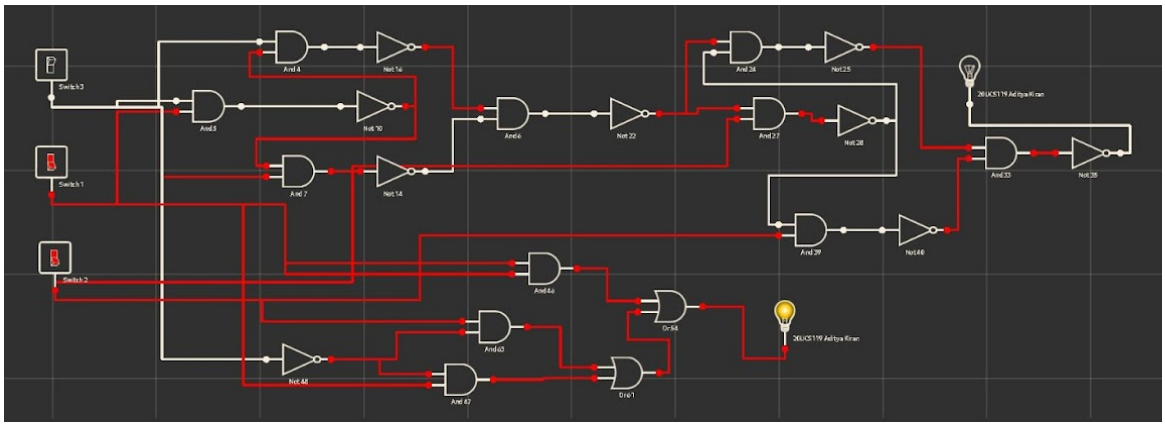
Inputs		Outputs	
A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

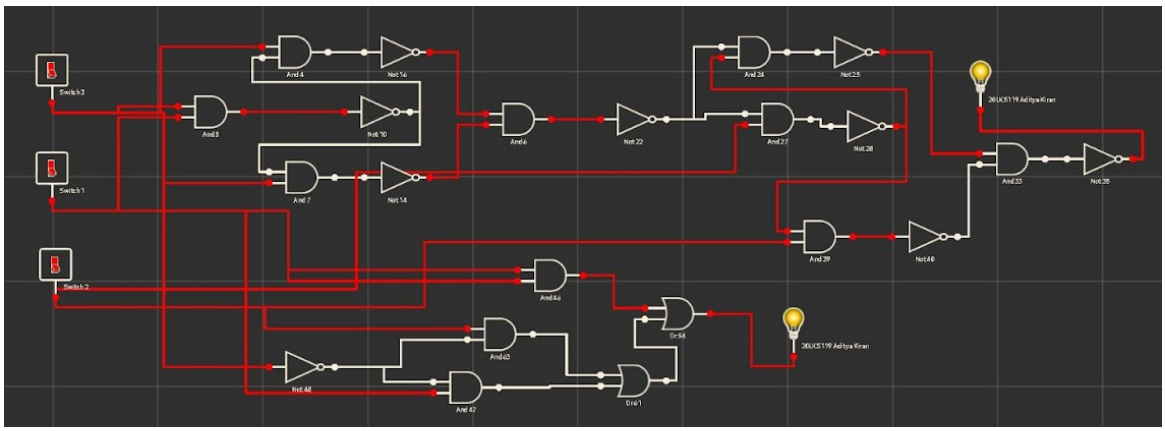
FULL Subtractor :

- Connect A, B and C input of full subtractor to switches from the input switches section.
- Connect the sum output of full subtractor to L1 and borrow output of full subtractor to L2 in the output section.
- Switch on power supply of the kit and provide proper inputs to full subtractor using switches as per truth table shown above.
- Observe the output of full adder and verify the functionality of full subtractor as per truth table.

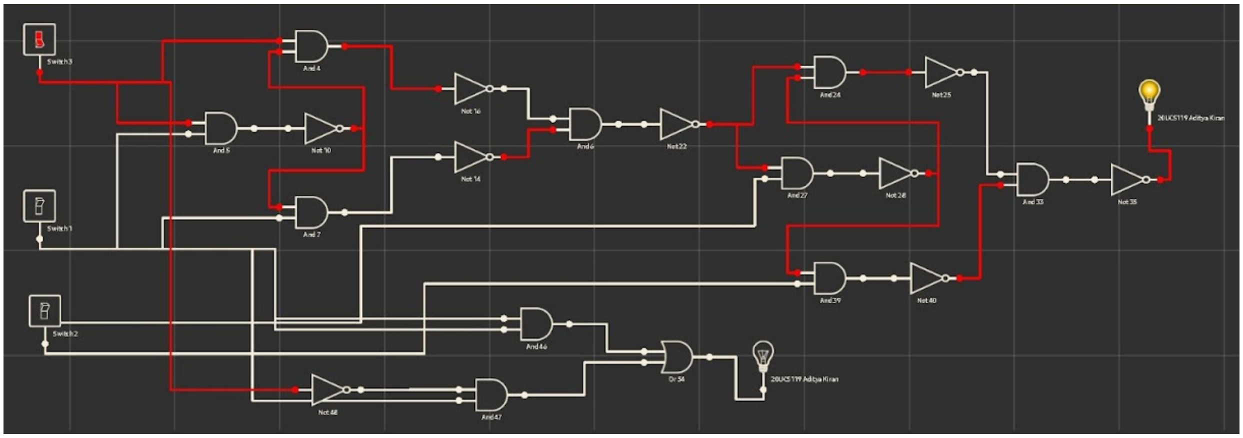
CASE 1: When B-out = $\bar{A}B+BB_{in}+\bar{A}B_{in}$

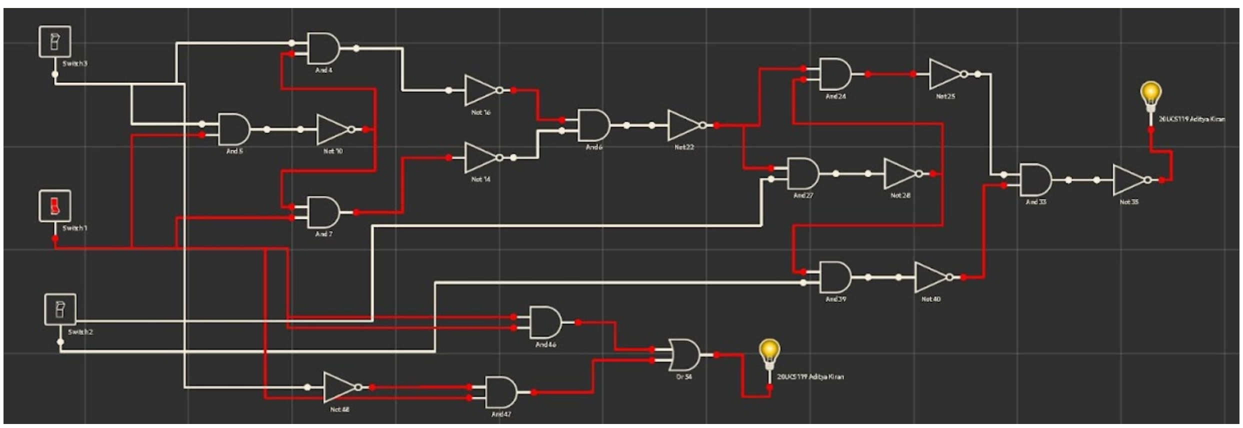


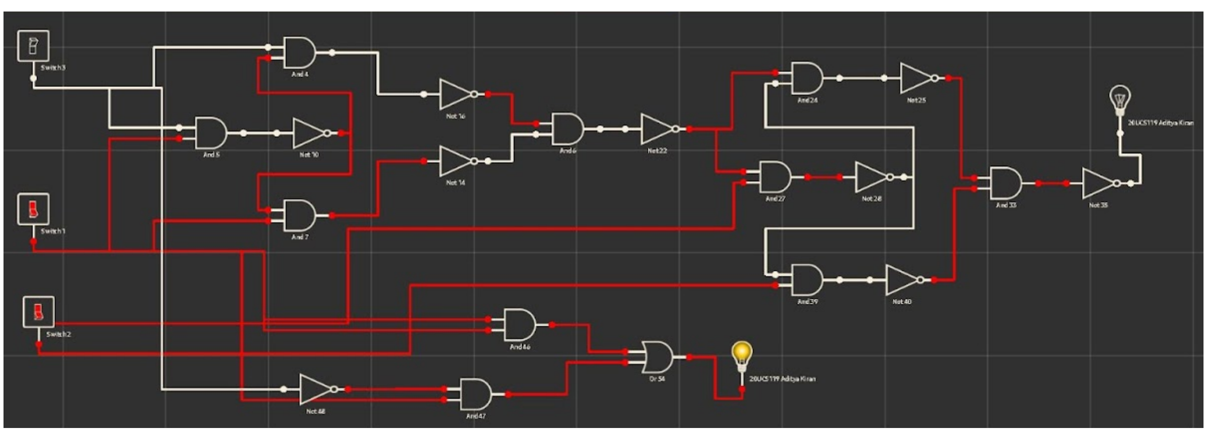




CASE 2: When C-out = $(A \oplus B)B_{in} + \bar{A}B$:







TRUTH TABLE:

A	B	Bin	Difference(D)	Borrow(Bout) $\bar{A}B+BB_{in}+\bar{A}B_{in}$	Borrow(Bout) $(A\oplus B)B_{in}+\bar{A}B$
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	1	1

Conclusion : Hence the functionality of half subtractor and full subtractor are verified.

EXPT NO.7 :
STUDY OF MULTIPLEXER

Objective :

To study the multiplexer.

Equipments :

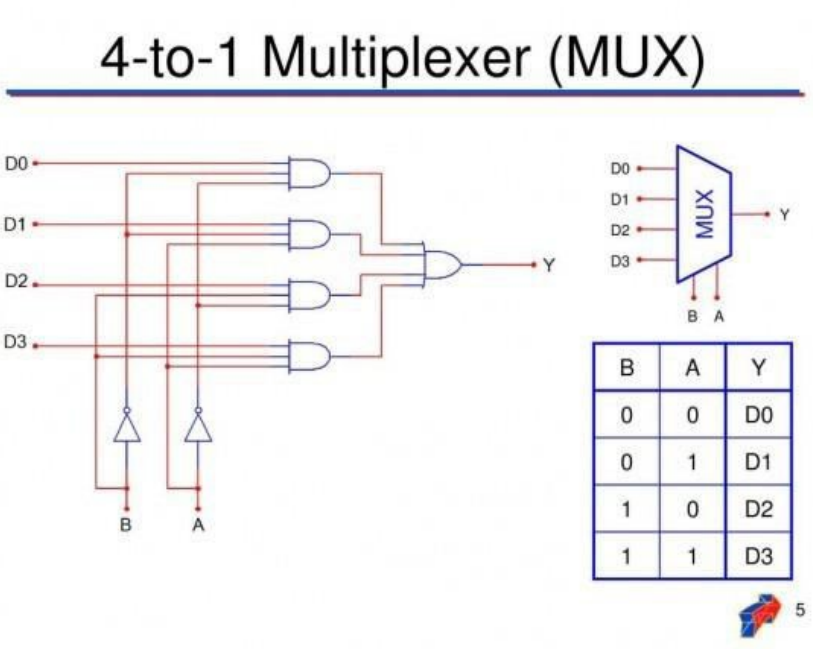
Logic Circuit Simulator Pro.

Theory :

Multiplexer :

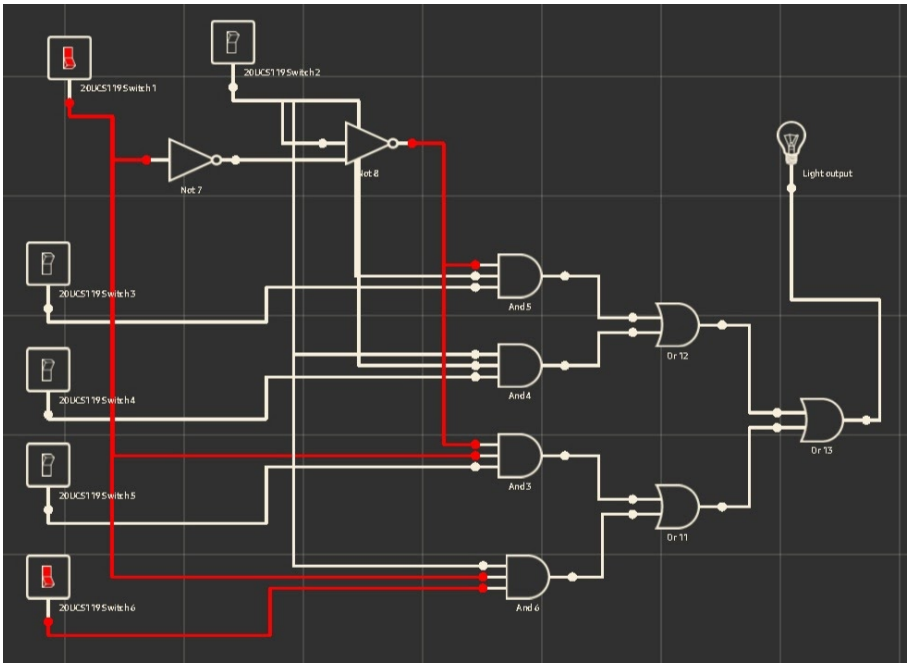
Multiplexer means transmitting a large number of information units over a smaller number of channels or lines.

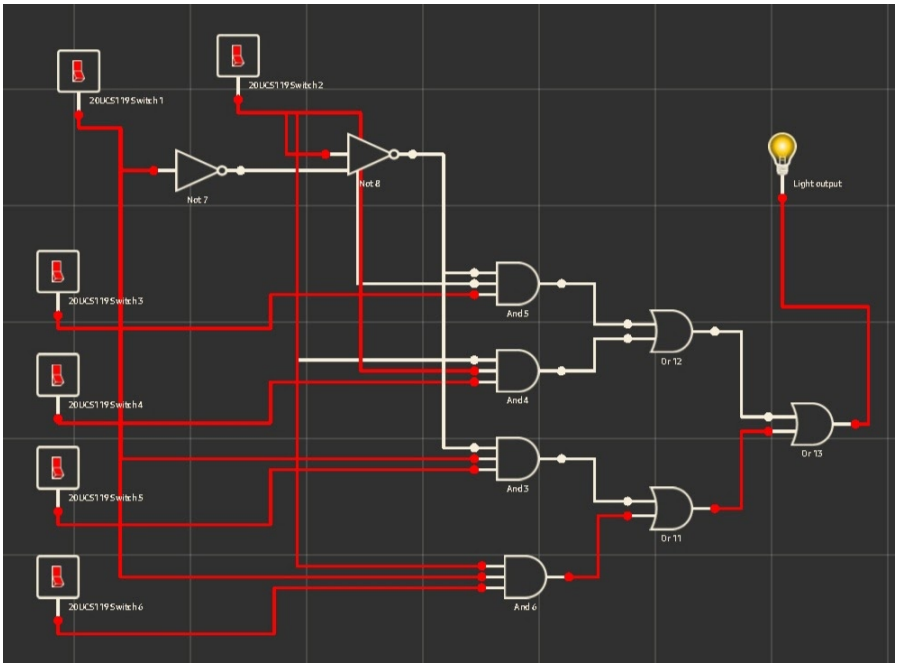
A multiplexer is a combinational logic circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines, whose bit combination determines which input is selected.

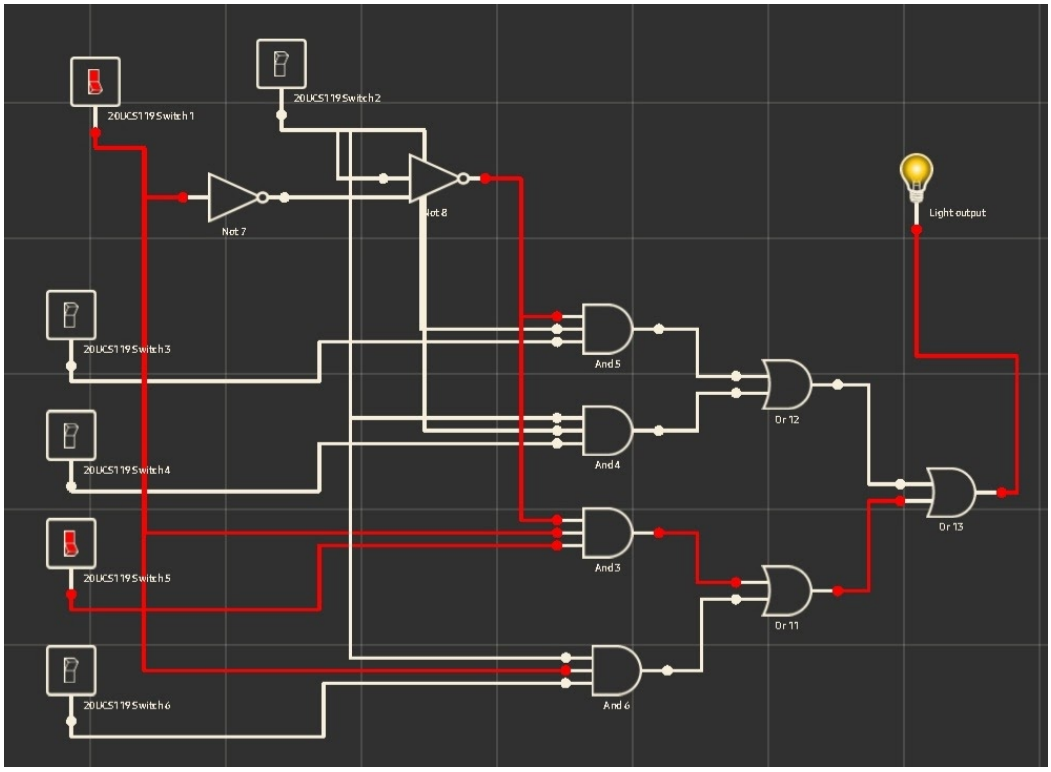


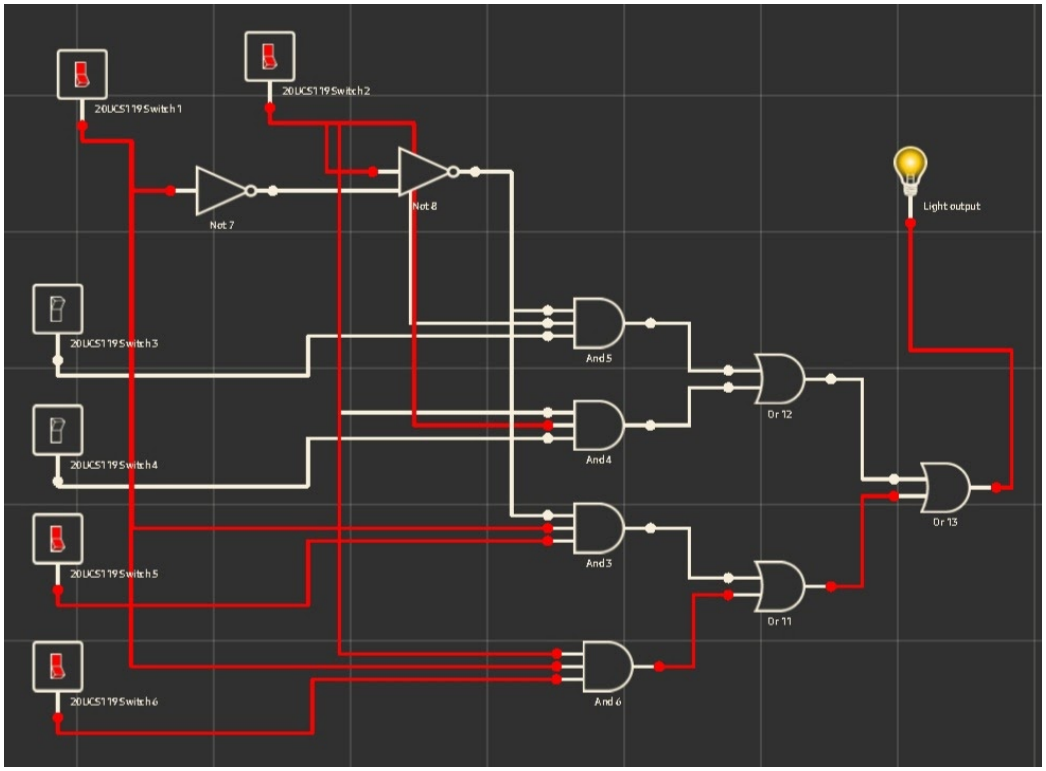
Procedure :

- Do the connections as shown and switch on the power supply.
- Apply proper logic inputs on the multiplexer and observe the output on LEDs.
- Verify the function table of the multiplexer in both the cases.






20UCS119 Switch 1
20UCS119 Switch 2
Not 7
Not 8
Light output
20UCS119 Switch 3
20UCS119 Switch 4
20UCS119 Switch 5
20UCS119 Switch 6
And 5
And 4
And 3
And 6
Or 12
Or 11
Or 13


20UCS119 Switch 1
20UCS119 Switch 2
Not 7
Not 8
Light output
20UCS119 Switch 3
20UCS119 Switch 4
20UCS119 Switch 5
20UCS119 Switch 6
And 5
And 4
And 3
And 6
Or 12
Or 11
Or 13

TRUTH TABLE:

Select Data Inputs		Output
S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Conclusion :

Hence the functionality of Multiplexer is verified.

EXPT NO. 8 :
STUDY OF DEMULTIPLEXER

Objective :

To study the Demultiplexer.

Equipments :

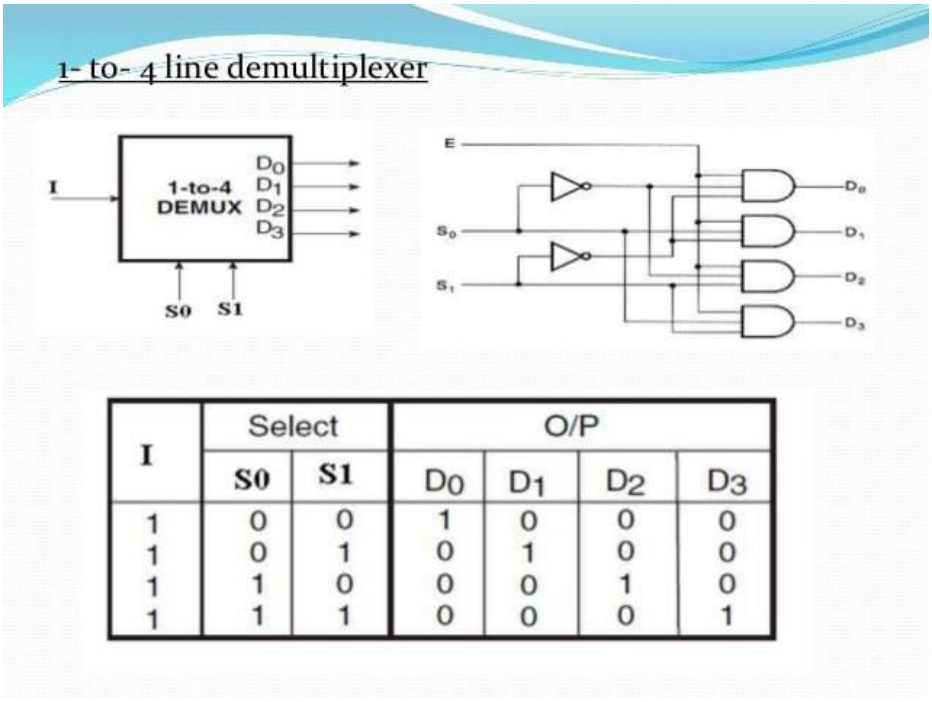
Logic Circuit Simulator Pro.

Theory :

Demultiplexer :

The demultiplexer is a combinational logic circuit having a single input and many outputs. It performs the reverse operation of a multiplexer. It accepts a single input and distributes it to output lines, according to the select lines. For this reason it is also known as a data distributor.

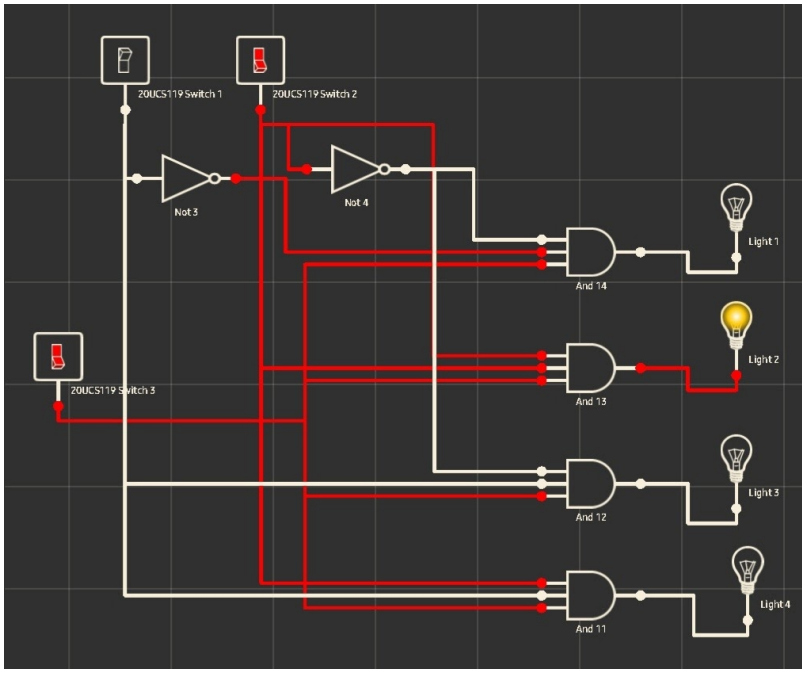
Decoder can also be used as a demultiplexer.

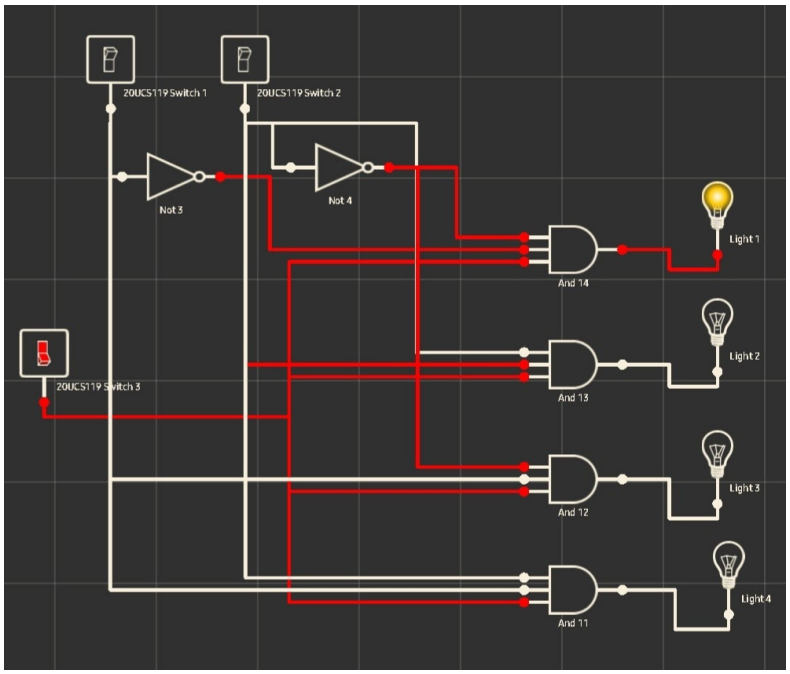


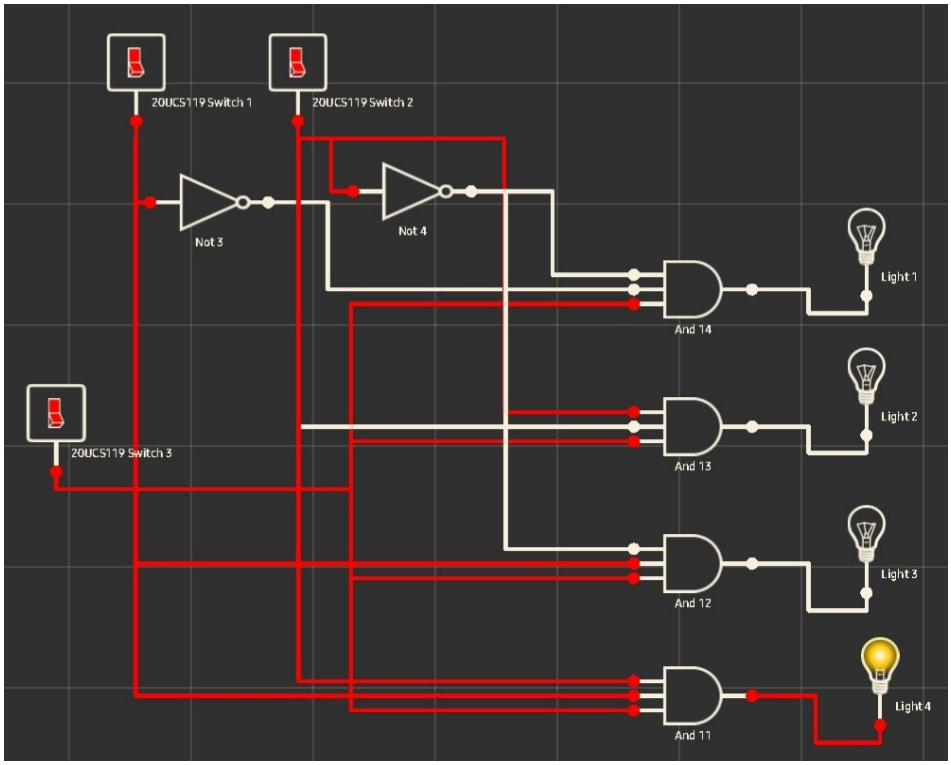
I	Select		O/P			
	S0	S1	D0	D1	D2	D3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

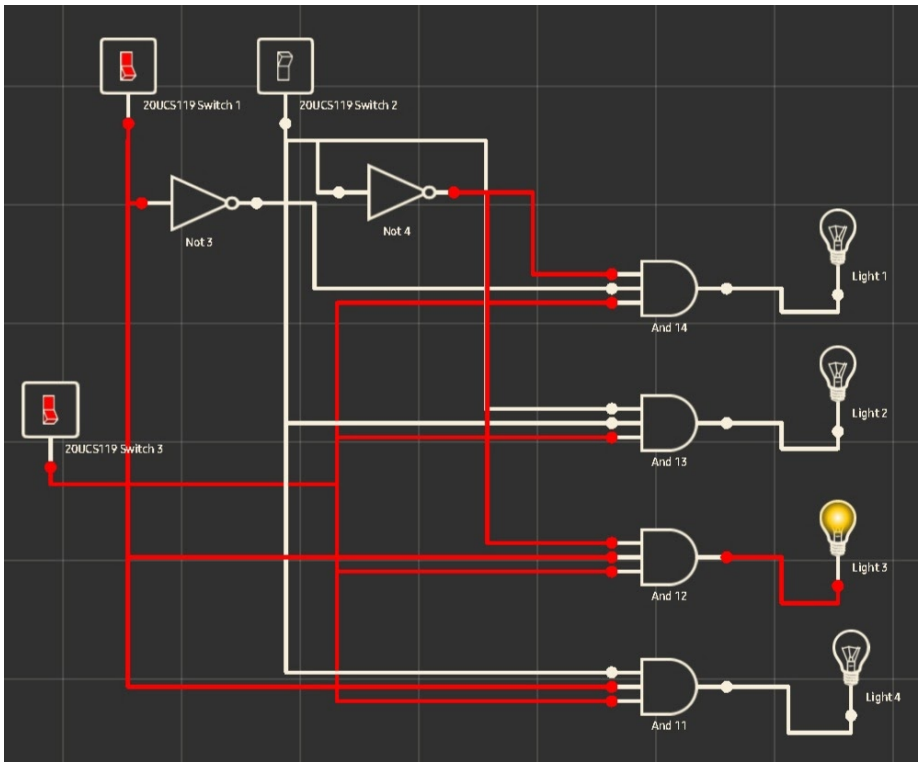
Procedure :

- Do the connections as shown and switch on the power supply.
- Apply proper logic inputs on the demultiplexer and observe the output on LEDs.
- Verify the function table of demultiplexer in both the cases.






20UCS119 Switch 1
20UCS119 Switch 2
Not 3
Not 4
Light 1
And 14
20UCS119 Switch 3
Light 2
And 13
Light 3
And 12
Light 4
And 11


20UCS119 Switch 1
20UCS119 Switch 2
Not 3
Not 4
Light 1
And 14
20UCS119 Switch 3
Light 2
And 13
Light 3
And 12
Light 4
And 11

TRUTH TABLE :

Data Input	Select Inputs		Outputs			
D	S_1	S_0	Y_3	Y_2	Y_1	Y_0
D	0	0	0	0	0	D
D	0	1	0	0	D	0
D	1	0	0	D	0	0
D	1	1	D	0	0	0

Conclusion :

Hence the functionality of Demultiplexer is verified.

EXPT NO. 9 :
STUDY OF S-R FLIP FLOP

Objective :

To study the S-R flip flop.

Equipments :

Logic Circuit Simulator Pro.

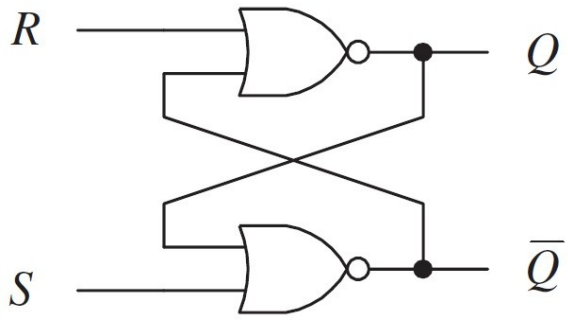
Theory :

S-R Flip Flop :

A flip-flop circuit can be constructed from two NAND gates or two NOR gates. These flip-flops are shown in Figure 2 and Figure 3. Each flip-flop has two outputs, Q and Q' and two inputs, set and reset. This type of flip-flop is referred to as an SR flip-flop or SR latch. The Flip-flop in Figure 2 has two useful states. When Q=1 and Q'=0, it is in the set state (or 1-state). When Q=0 and Q'=1, it is in the clear state (or 0-state). The outputs Q and Q' are complements of each other and are referred to as the normal and complement outputs respectively. The binary state of the flip-flop is taken to be the value of the normal output When a 1 is applied to both the set and reset inputs of the flip-flop in Figure 2, both Q and Q' outputs go to O. This condition violates the fact that both outputs are complements of each other. In normal operation this condition must be avoided by making sure that 1's are not applied to both inputs simultaneously.

TRUTH TABLE

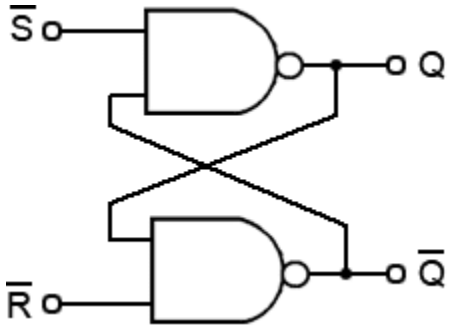
S	R	Q	Q'
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0



The NAND basic flip-flop circuit in Figure 3(a) operates with inputs normally at 1 unless the state of the flip-flop has to be changed. A 0 applied momentarily to the set input causes Q to go to 1 and Q' to go to 0. putting the flip-flop in the set state. When both inputs go to both outputs go to 1. This condition should be avoided in normal operation.

TRUTH TABLE

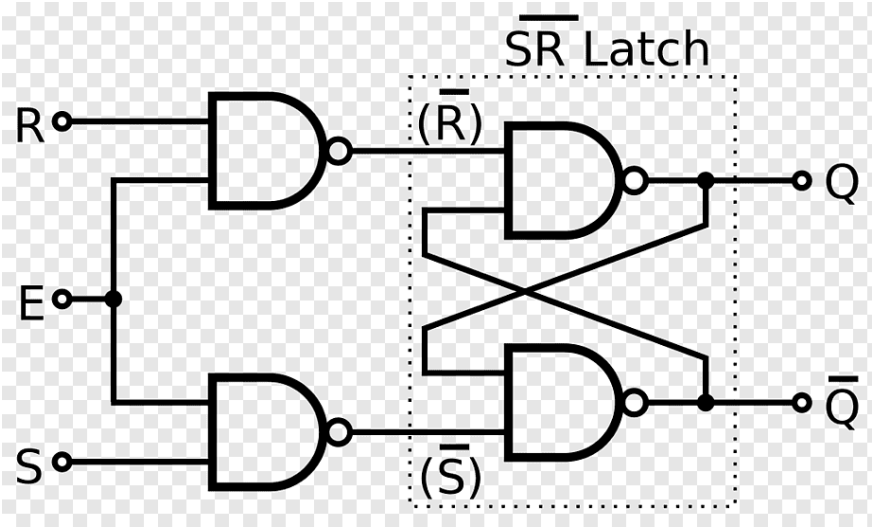
S	R	Q	Q’
1	0	0	0
1	1	0	1
0	1	1	0
1	1	1	0
0	0	1	1



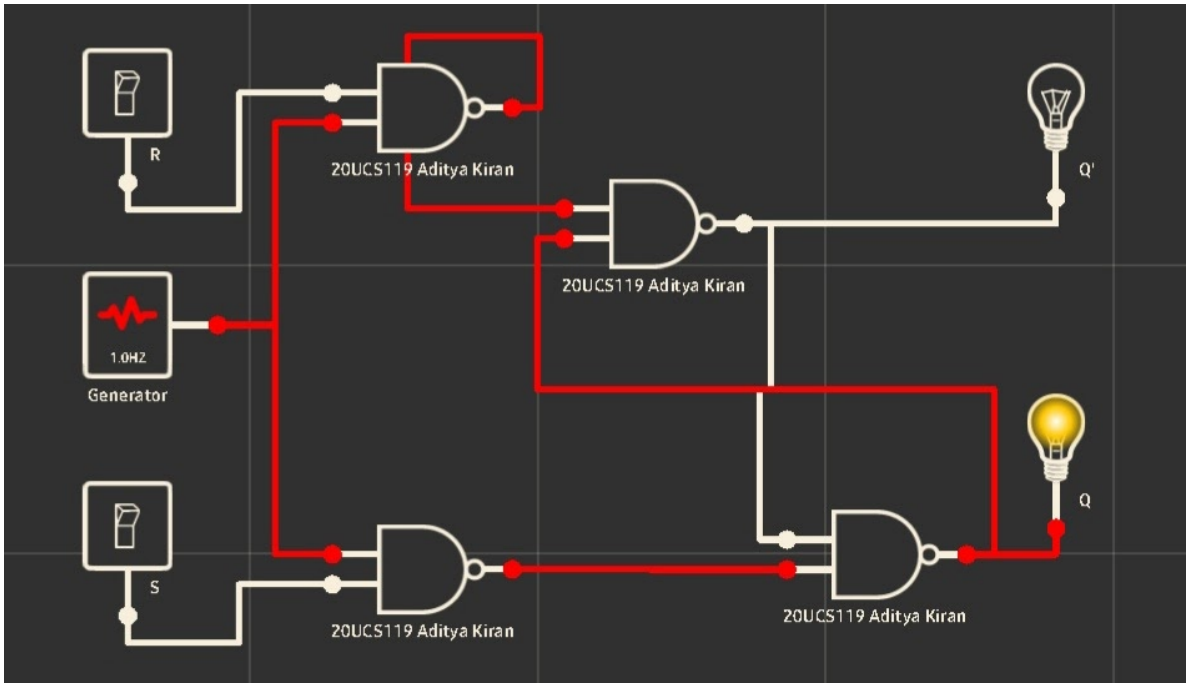
Procedure :

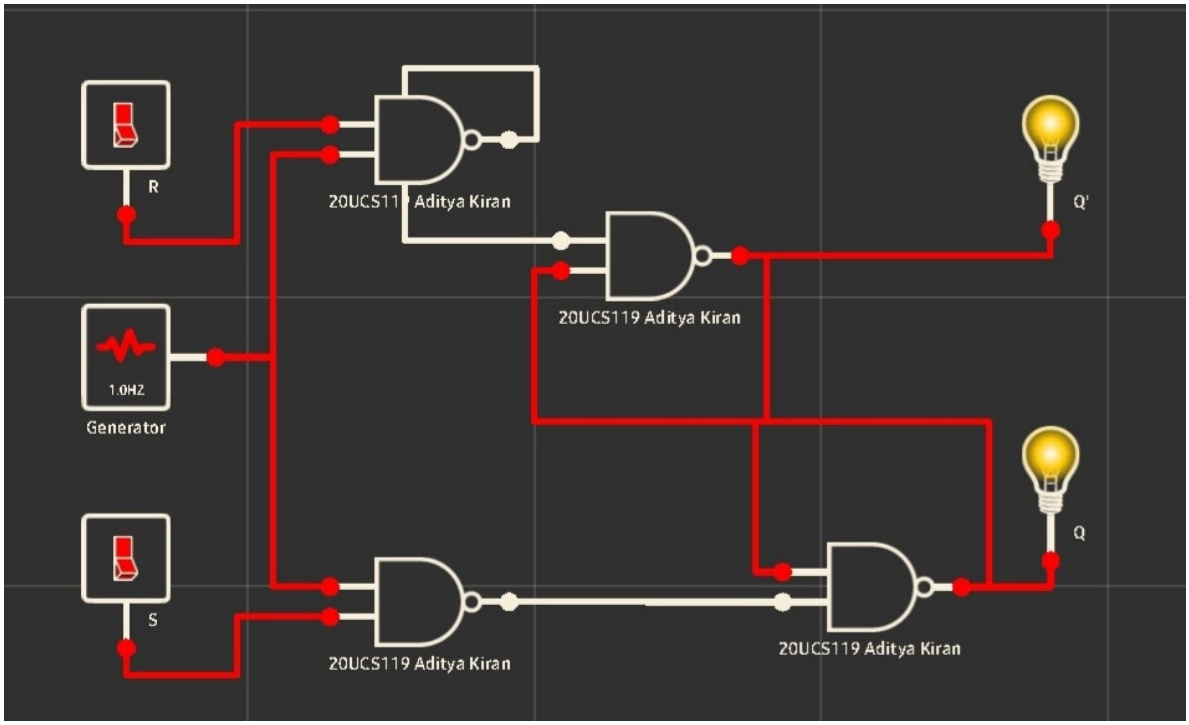
- Do the connection as per block diagram shown below and switch ON the power supply.
- Apply proper logic inputs to the S-R flip flop and observe the output on LEDs.
- Verify the function table of S-R Flip flop.

Logic Diagram :



Circuit Diagram :





TRUTH TABLE:

S	R	Q	Q’
0	0	0	1
0	1	0	1
1	0	1	0
1	1	∞	∞

Conclusion :

From the above experiment, we verified the characteristics of S-R flip flop.

EXPT NO. 10 :

STUDY OF J-K FLIP FLOP

Objective :

To study the J-K flip flop.

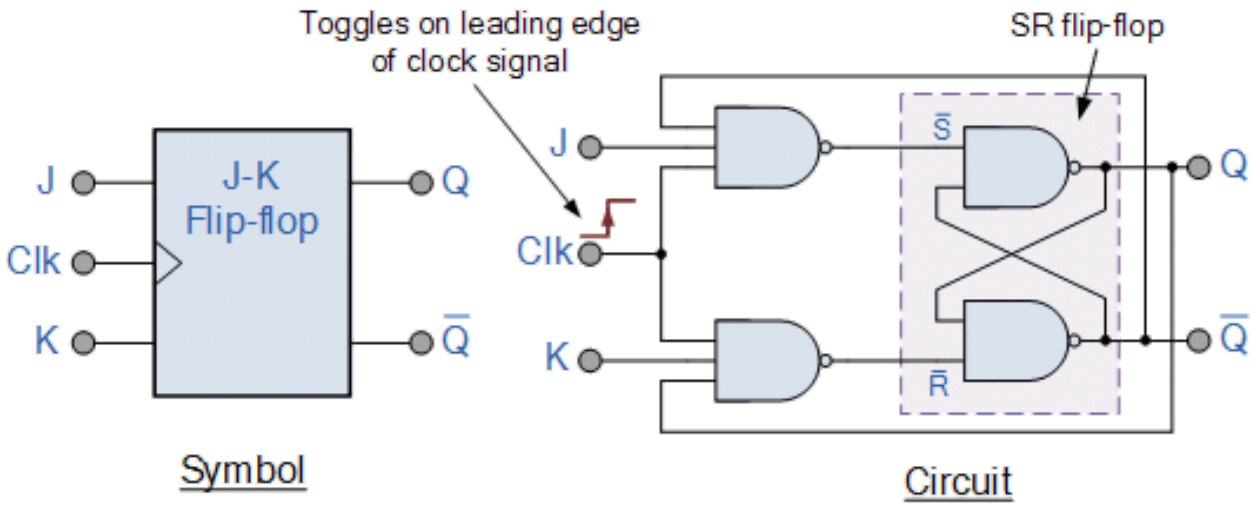
Equipments :

Logic Circuit Simulator Pro.

Theory :

J-K Flip Flop :

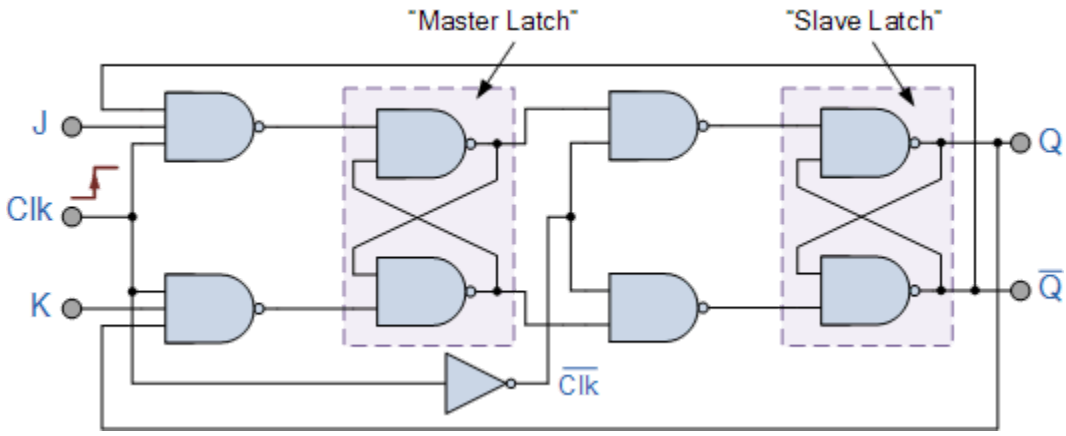
Fig 7(a) shows the clocked J-K flip-flop with clear (CR) and preset (PR) inputs. The small circle (inversion symbols) on these inputs indicates that logic '0' is required to clear or set the flip-flop. Thus the '0' applied to the clear input will reset the flip-flop to Q= 'O', and a 'o applied to the Preset input will set the flip-flop to Q J-K input. I.e. a 'o' applied to the clear input will reset the flip-flop regardless of the values of 1'. These inputs override the clock & J-K, and the clock. Under normal conditions, a 'O' should not be applied simultaneously to clear and preset. When the clear and preset inputs are both held at logic '1', the J, K and clock inputs operate in the normal manner.



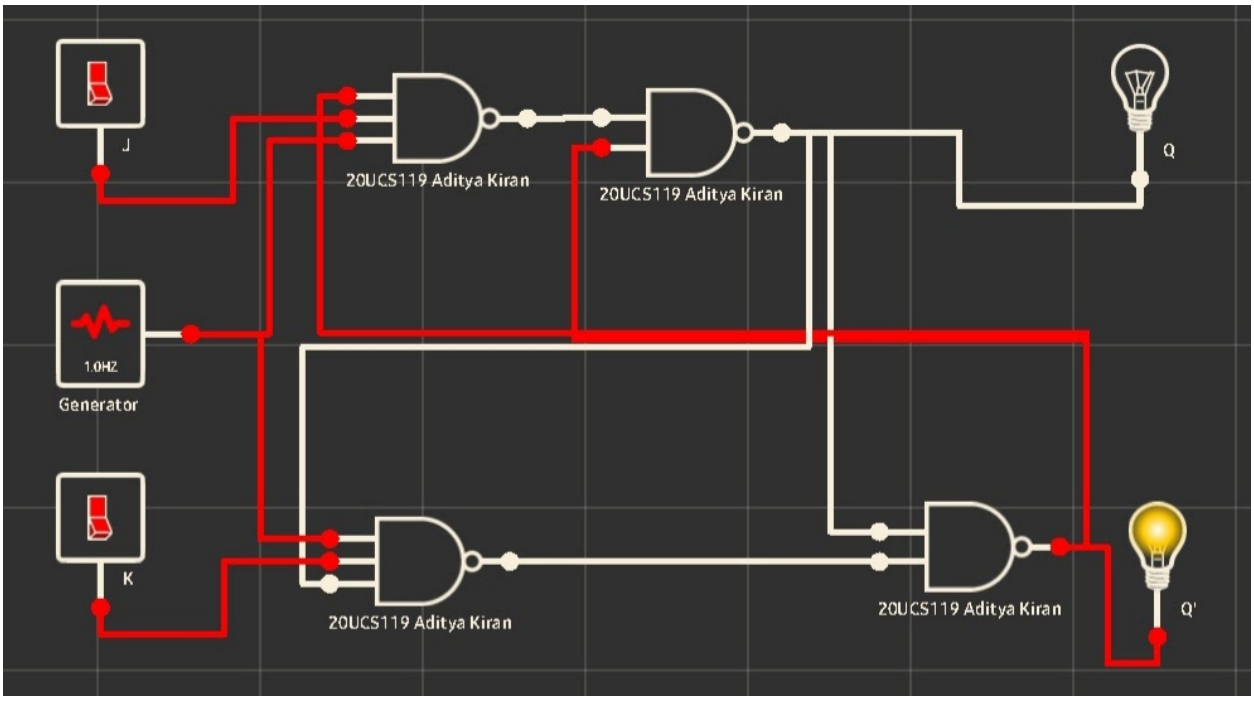
Procedure :

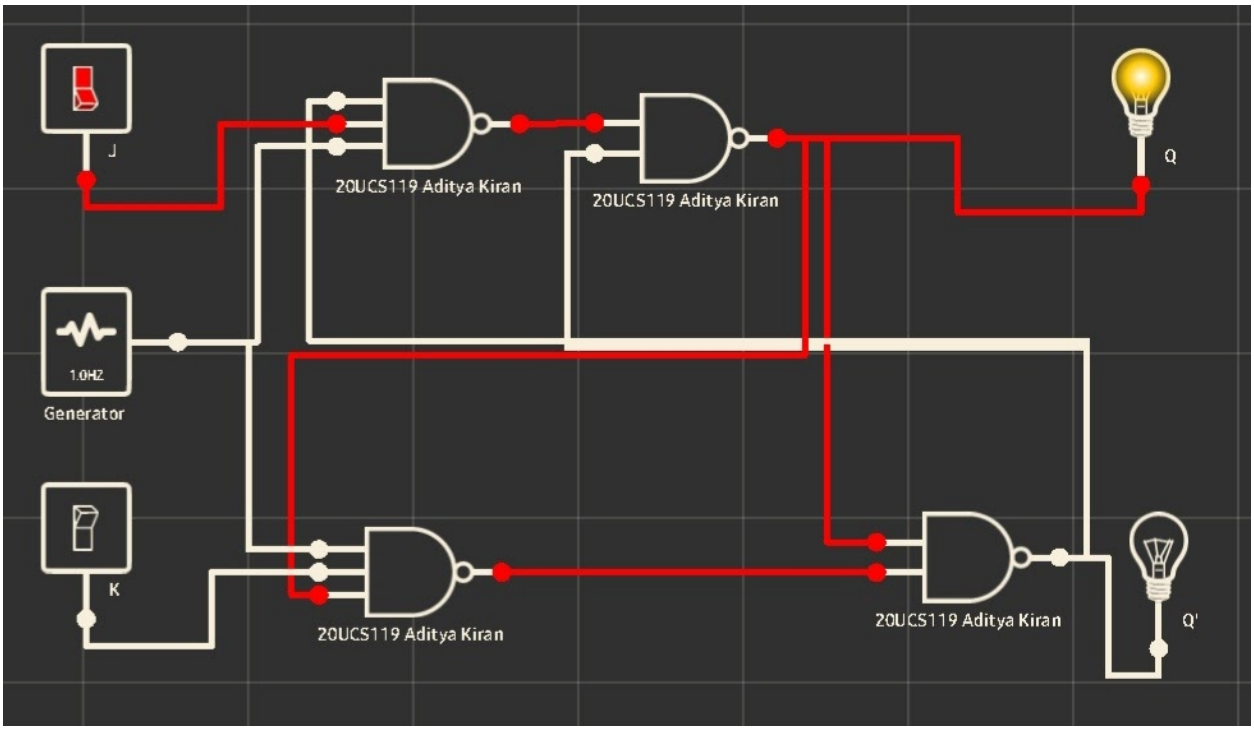
- Do the connection as per block diagram shown below and switch ON the power supply.
- Apply proper logic inputs to the J-K flip flop and observe the output on LEDs.
- Verify the function table of J-K Flip flop.

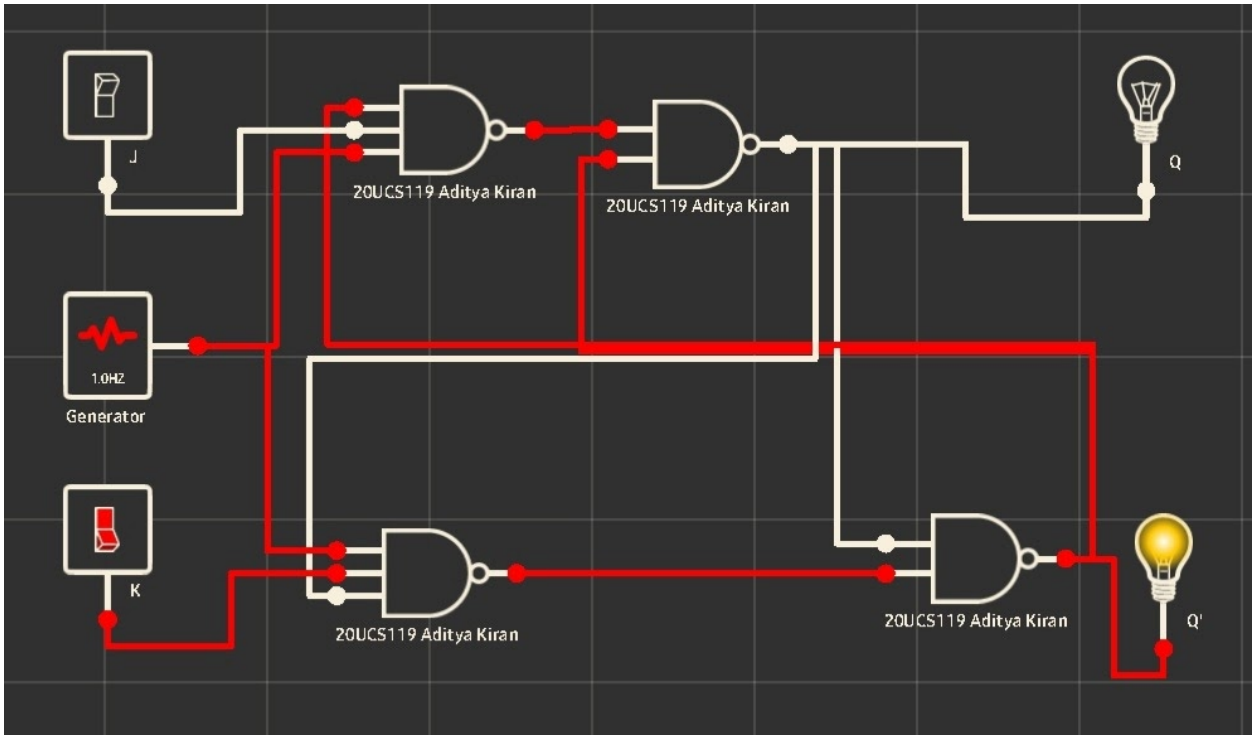
Logic Diagram :



Circuit Diagram :




J
20UCS119 Aditya Kiran
20UCS119 Aditya Kiran
Q
1.0HZ
Generator
K
20UCS119 Aditya Kiran
20UCS119 Aditya Kiran
Q'


J
20UCS119 Aditya Kiran
20UCS119 Aditya Kiran
Q
1.0HZ
Generator
K
20UCS119 Aditya Kiran
20UCS119 Aditya Kiran
Q'

Truth Table :

PR	CL	CLK	J	K	Q	Q'
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	-	-
1	1	1	0	0	Q_0	Q_0'
1	1	1	1	0	1	0
1	1	1	0	1	0	1
1	1	1	1	1	Toggle	
1	1	0	X	X	Q_0	Q_0'

Conclusion :

From the above experiment, we verified the characteristics of J-K flip flop.