Shreya Sriram
195001106

Aim :

To design and implement 4 bit adder and subtractor using IC 7483.

Apparatus Required :

S.No	Component Name	IC Number	Quantity
1	IC	7483	1
2	EX-OR GATE	7486	1
3	NOT GATE	7404	1
4	IC TRAINER KIT	-	1
5	PATCH CORDS	-	40

Theory :

4 BIT BINARY ADDER:

A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders connected in cascade, with the output carry from each full adder connected to the input carry of next full adder in chain. The augends bits of 'A' and the addend bits of 'B' are designated by subscript numbers from right to left, with subscript 0 denoting the least significant bits. The carries are connected in chain through the full adder. The input carry to the adder is C0 and it ripples through the full adder to the output carry C4.

4 BIT BINARY SUBTRACTOR:

The circuit for subtracting A-B consists of an adder with inverters, placed between each data input 'B' and the corresponding input of full adder. The input carry C0 must be equal to 1 when performing subtraction.

4 BIT BINARY ADDER/SUBTRACTOR:

The addition and subtraction operation can be combined into one circuit with one common binary adder. The mode input M controls the operation. When M=0, the circuit is adder circuit. When M=1, it becomes subtractor.

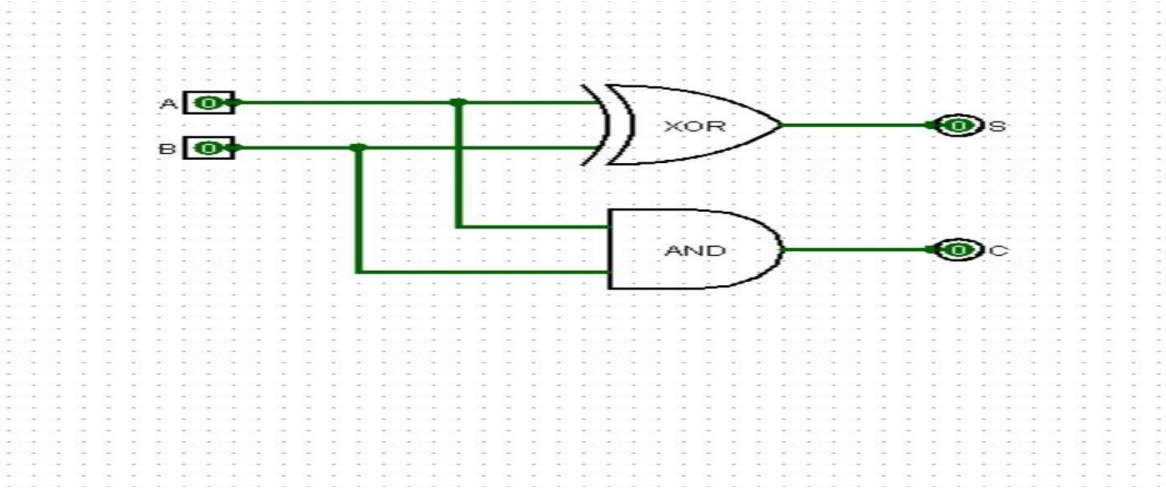
Values :

A = 1001 (9)

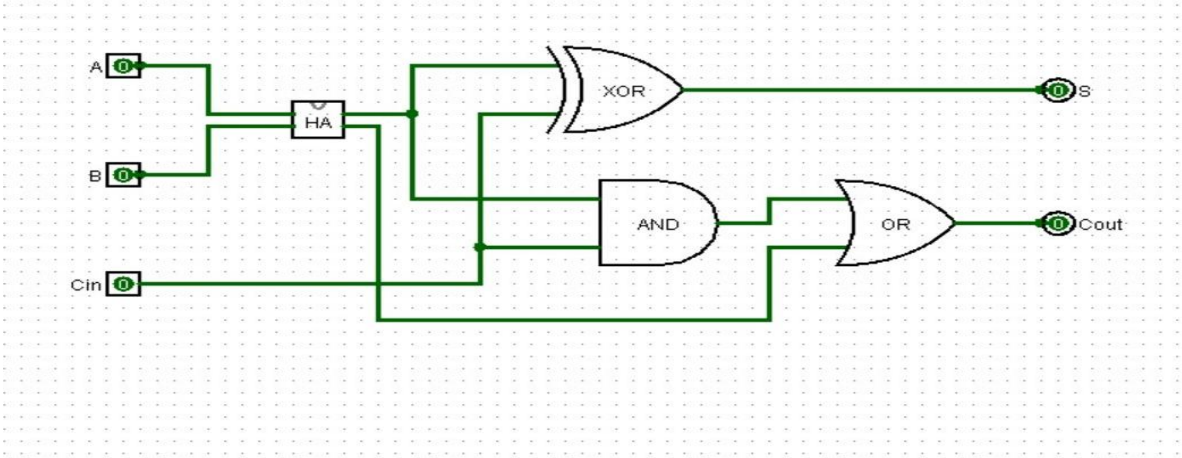
B = 0011 (3)

Shreya Sriram
195001106

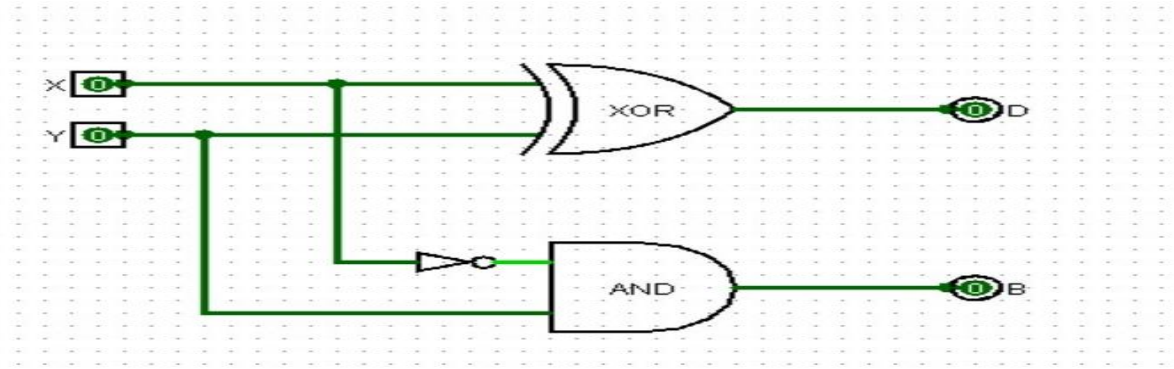
Half Adder :



Full Adder :

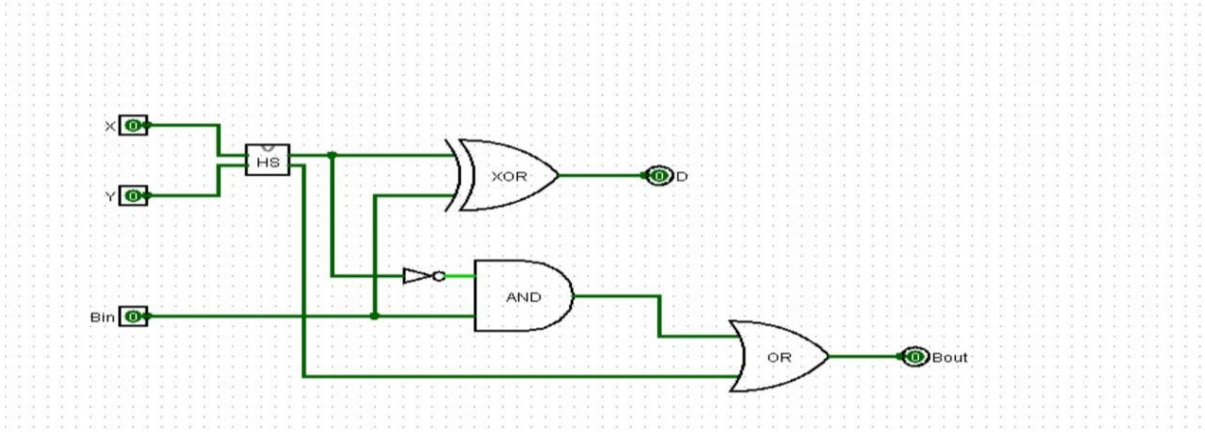


Half Subtractor :

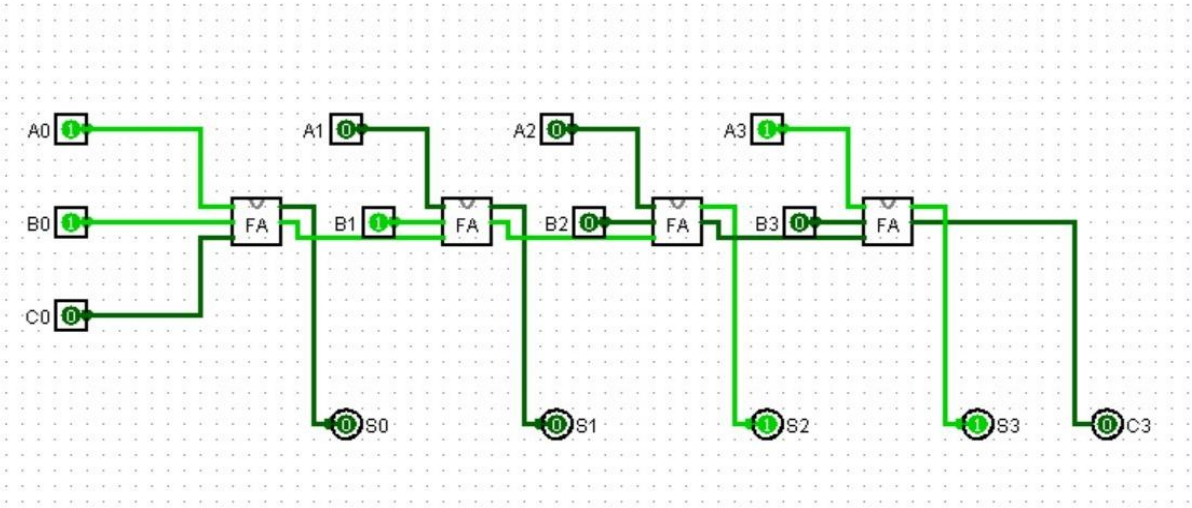


Shreya Sriram
195001106

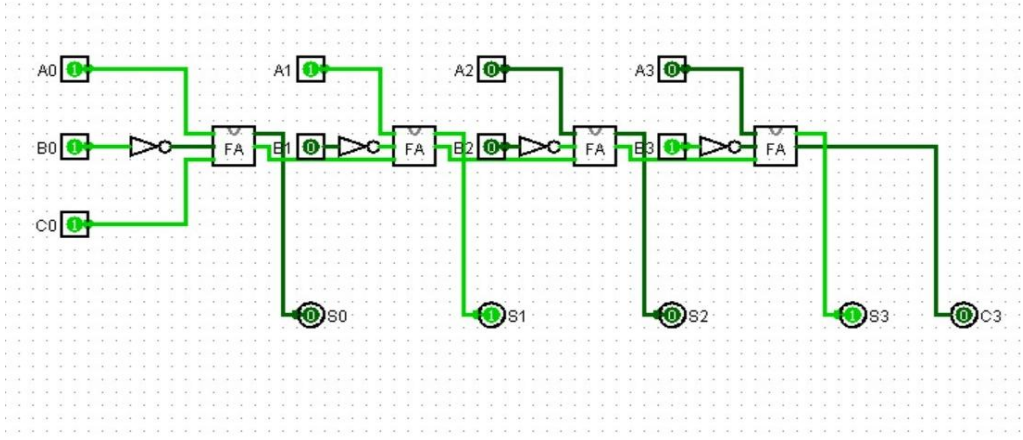
Full Subtractor :



4 Bit Binary Adder :

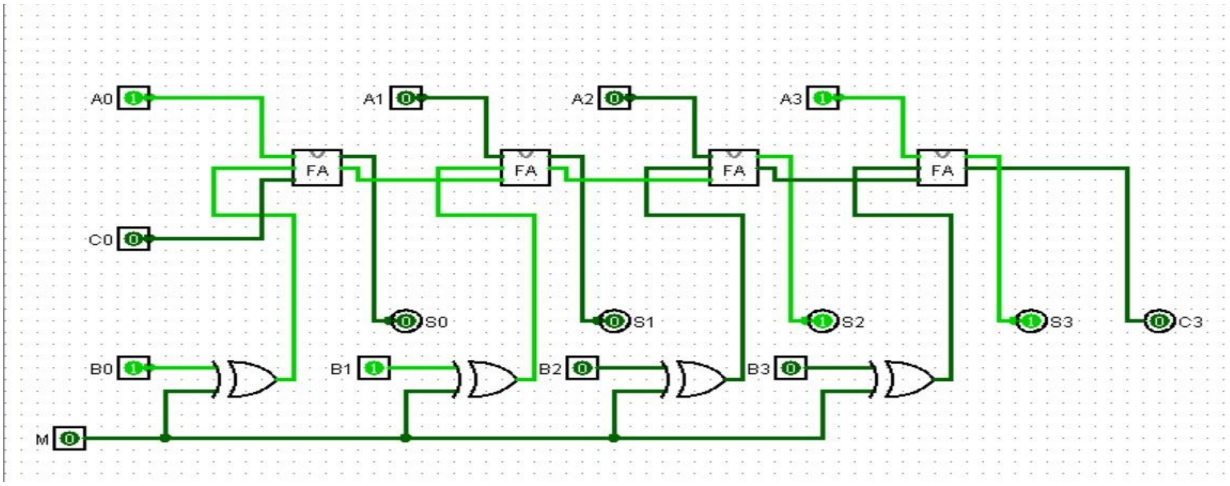


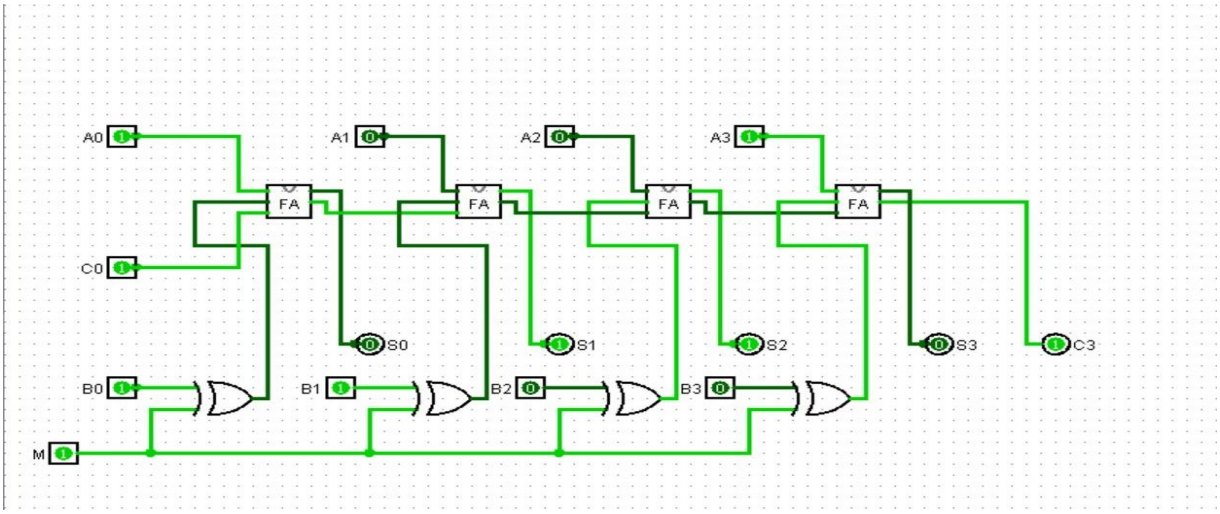
4 Bit Binary Subtractor :



Shreya Sriram
195001106

4 Bit Binary Adder/Subtractor :





Output :

	C_{in}	A_3	A_2	A_1	A_0	B_3	B_2	B_1	B_0	S_3	S_2	S_1	S_0	C_{out}
A + B	0	1	0	0	1	0	0	1	1	1	1	0	0	0
A - B	1	1	0	0	1	0	0	1	1	0	1	1	0	1
B - A	1	1	0	0	1	0	0	1	1	1	0	1	0	0

B – A gives no carry. Therefore, we have to take 2's complement of S and add a negative sign which results in – 0110 (-6).

Result :

The 4 bit adder and subtractor were designed and implemented successfully.

Shreya Sriram
195001106

Aim :

To design and implement the below given Boolean Function using a Multiplexer.

$F(A,B,C) = \sum (0,2,4,5,7)$

Apparatus Required :

S.No	Component Name	IC Number	Quantity
1	NOT Gate	7404	2
2.	Ex-OR Gate	7486	3
3	AND Gate	7408	1
4	OR Gate	7432	2
5	IC Trainer Kit	-	1
6	Connecting Wires	-	30

Truth Table for Boolean Function :

A	B	C	F	
0	0	0	1	F = C’
0	0	1	0	
0	1	0	1	F = C’
0	1	1	0	
1	0	0	1	F = 1
1	0	1	1	
1	1	0	0	F = C
1	1	1	1	

We need a 4:1 MUX and 2 selector lines to implement the Boolean Expression. A and B are taken as the selector lines S1 and S0 respectively.

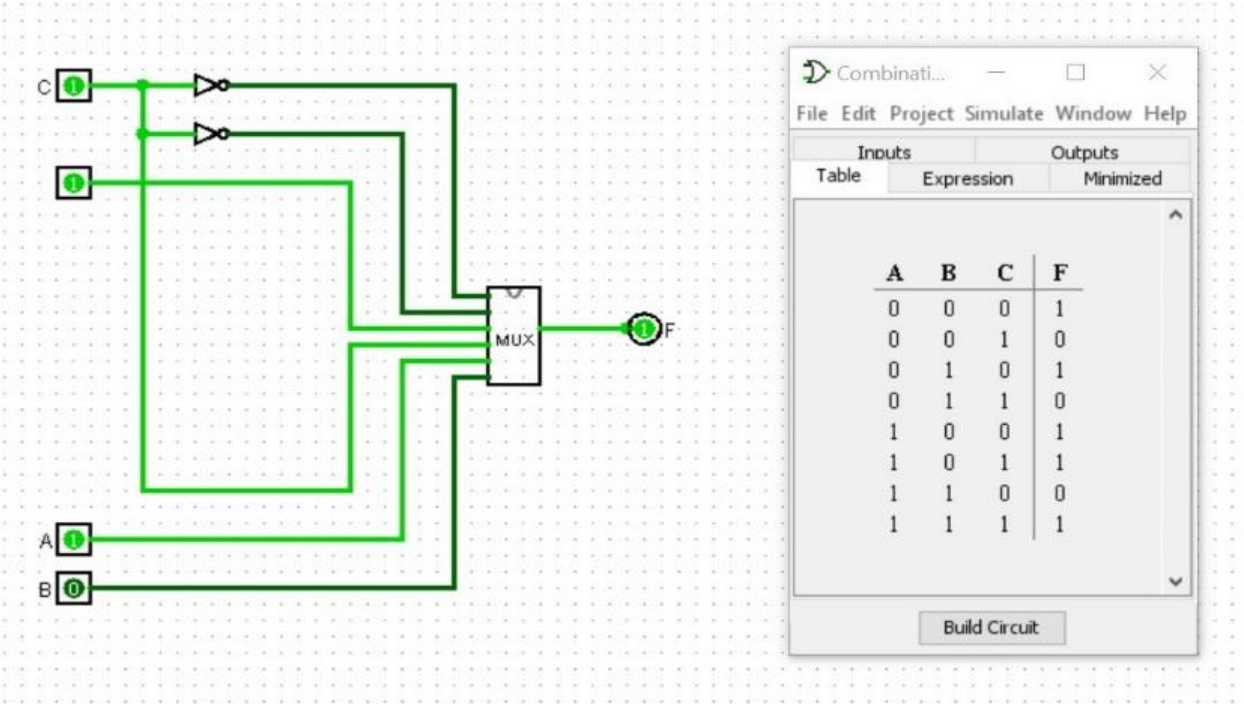
Inputs for MUX are : D0 = C’ D1 = C’ D2 = 1 AND D3 = C

Truth Table for MUX :

S1	S0	O/P
0	0	D0 = C’
0	1	D1 = C’
1	0	D2 = 1
1	1	D3 = C

Shreya Sriram
195001106

Logic Circuit :



Result :

The given Boolean Function was designed and implemented successfully using a Multiplexer.

Shreya Sriram
195001106

Objective

To design and implement a combinational circuit for the below given code converters.

Components Required

S.No	Component Name	IC Number	Quantity
1	AND	7408	1
2	OR	7432	1
3	Connecting Wires	-	-
4	LogiSIM Software	-	-

Specifications:

1. Draw the truth table for the corresponding code converter with required number of input bits (4 bits) and output bits (Not restricted)
2. Write the simplified Boolean Expression, for each output variable as the function of input variables using Karnaugh map.
3. Draw the combinational logic circuit using basic gates and universal gates
4. Verify the truth table

Truth Table

DECIMAL	8 4 -2 -1 CODE				8 4 2 1 BCD CODE			
	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	1
2	0	1	1	0	0	0	1	0
3	0	1	0	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	1	0	1	1	0	1	0	1
6	1	0	1	0	0	1	1	0
7	1	0	0	1	0	1	1	1
8	1	0	0	0	1	0	0	0
9	1	1	1	1	1	0	0	1

Shreya Sriram
195001106

KMap Simplification

AB vertically CD horizontally

	00	01	11	10
00	0	x	x	x
01	0	0	0	0
11	x	x	1	x
10	1	0	0	0

W = AB + AC'D'

	00	01	11	10
00	0	x	x	X
01	1	0	0	0
11	x	x	0	x
10	0	1	1	1

X = BC'D' + B'D + B'C

	00	01	11	10
00	0	x	x	x
01	0	1	0	1
11	x	x	0	x
10	0	1	0	1

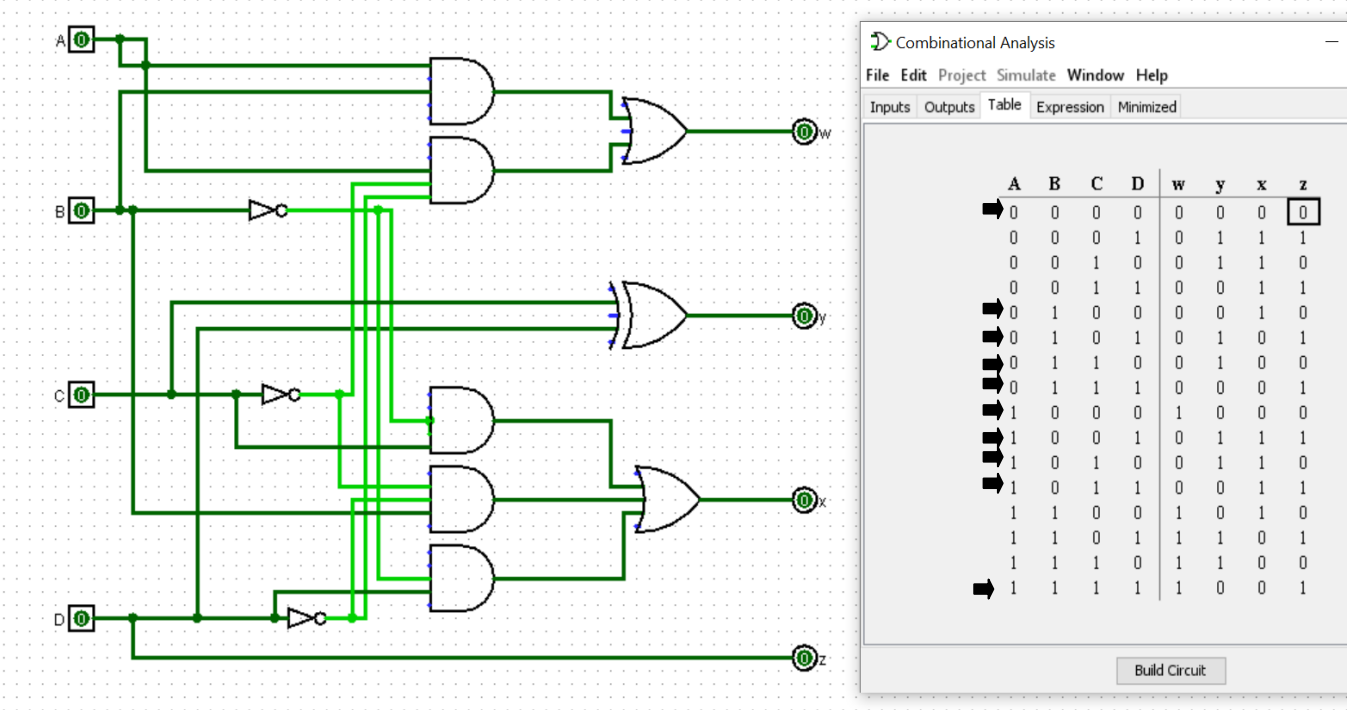
Y = C'D + CD'

	00	01	11	10
00	0	x	x	x
01	0	1	1	0
11	x	x	1	x
10	0	1	1	0

Z = D

Shreya Sriram
195001106

Output:



Results

The truth table has been verified with that of the simplified expression.

Shreya Sriram
195001106

Aim :

To design and implement absolute differentiator circuit using IC 7485 (4-bit Magnitude Comparator) and IC 7483 (4-bit Binary Adder).

Apparatus Required :

S.No	Component Name	IC Number	Quantity
1	4-bit Magnitude Comparator	7485	1
2	4-bit Binary Adder	7483	1
3	Ex-OR Gate	7486	2
4	IC Trainer Kit	-	1
5	Patch Cords	-	30

Theory :

Absolute Differentiator is a combinational circuit that takes two binary inputs and computes the difference between the two magnitudes and outputs the difference value irrespective of the sign. The serial circuit is constructed by connecting Magnitude Comparator and Binary Adder circuits in serial. The output of magnitude comparator is A=B, A<B and A>B. The outputs A<B and A>B are given as input to the Binary Adder circuit to obtain the difference of two magnitudes.

If A<B, the value of A is given as one of the inputs of Ex-OR gate and the other input is 1 which is taken from A<B output of Magnitude Comparator. This computes the one’s complement of A value and given as input to Binary Adder circuit.

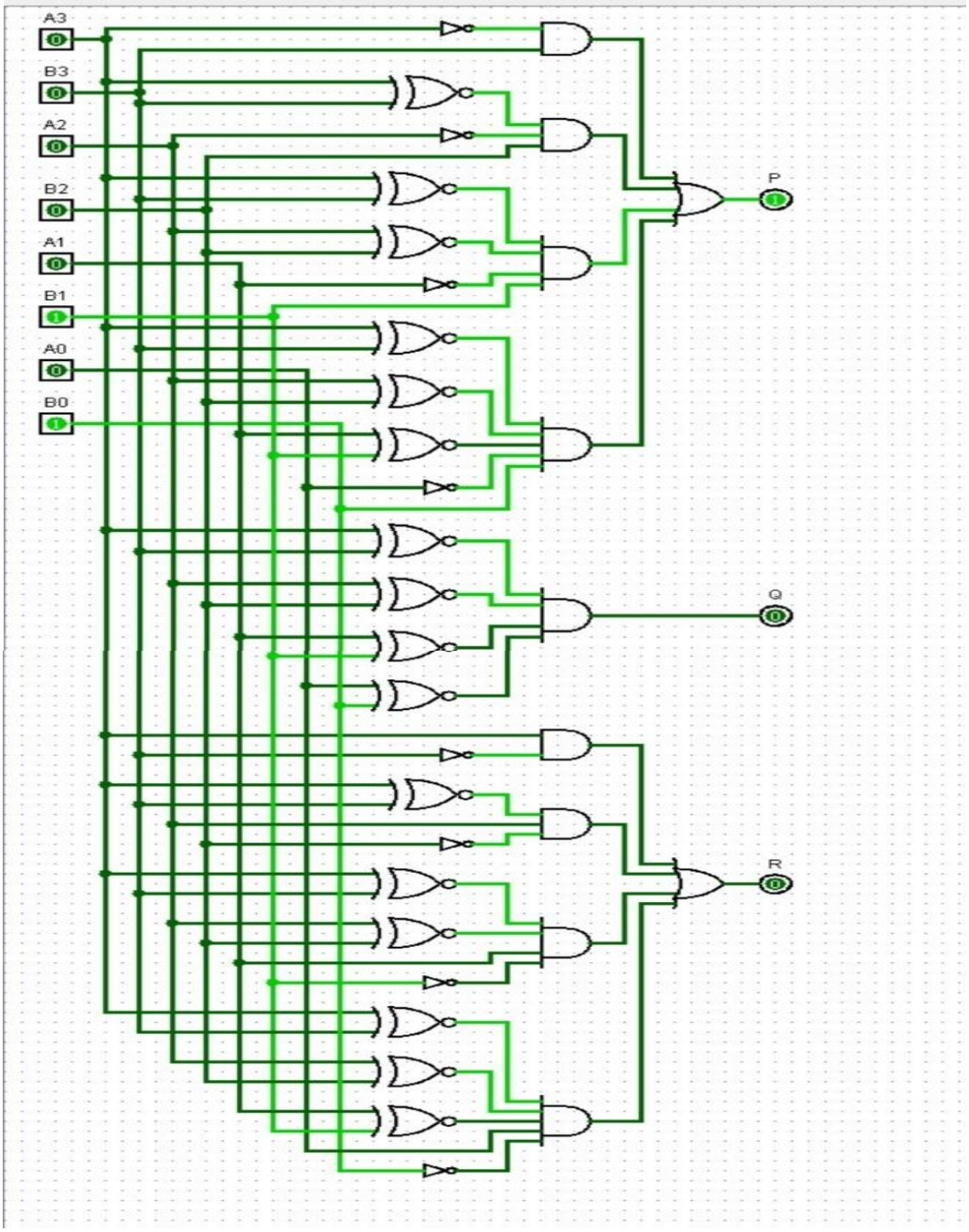
Similarly when A>B, the value of B is given as one of the inputs of Ex-OR gate and the other input is 1 which is taken from A>B output of Magnitude Comparator. This computes the one’s complement of B and is given as input to Binary Adder circuit.

The C input to Binary Adder is 1 as it computes the 2’s complement value of smallest magnitude (either A or B) and adds it with the minuend.

The output of the Binary Adder circuit is the absolute difference between two magnitude values of A and B.

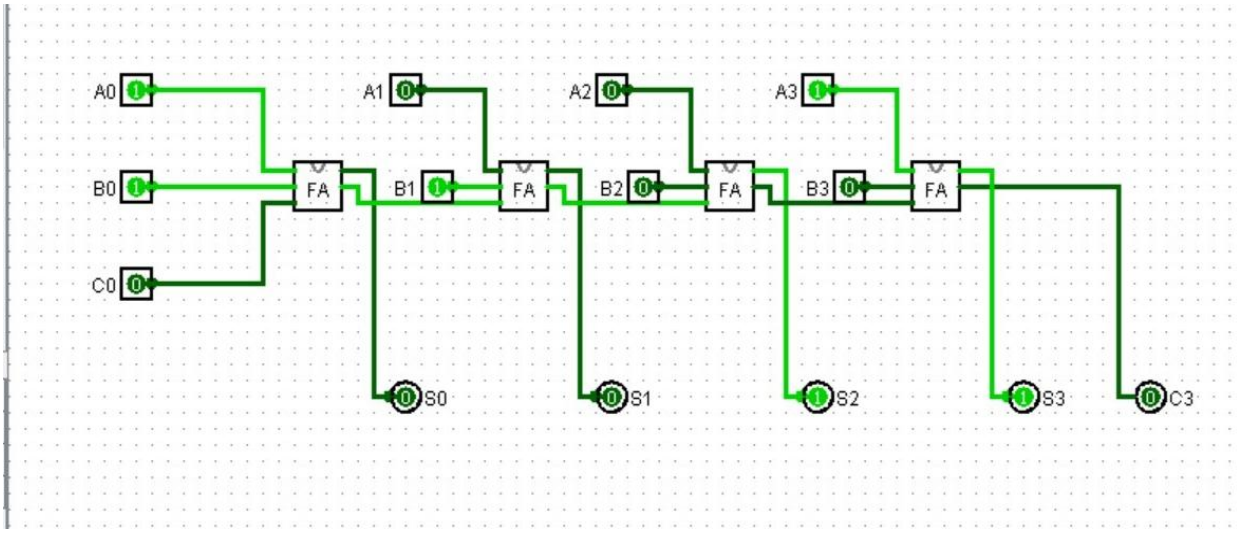
Shreya Sriram
195001106

Magnitude Comparator :

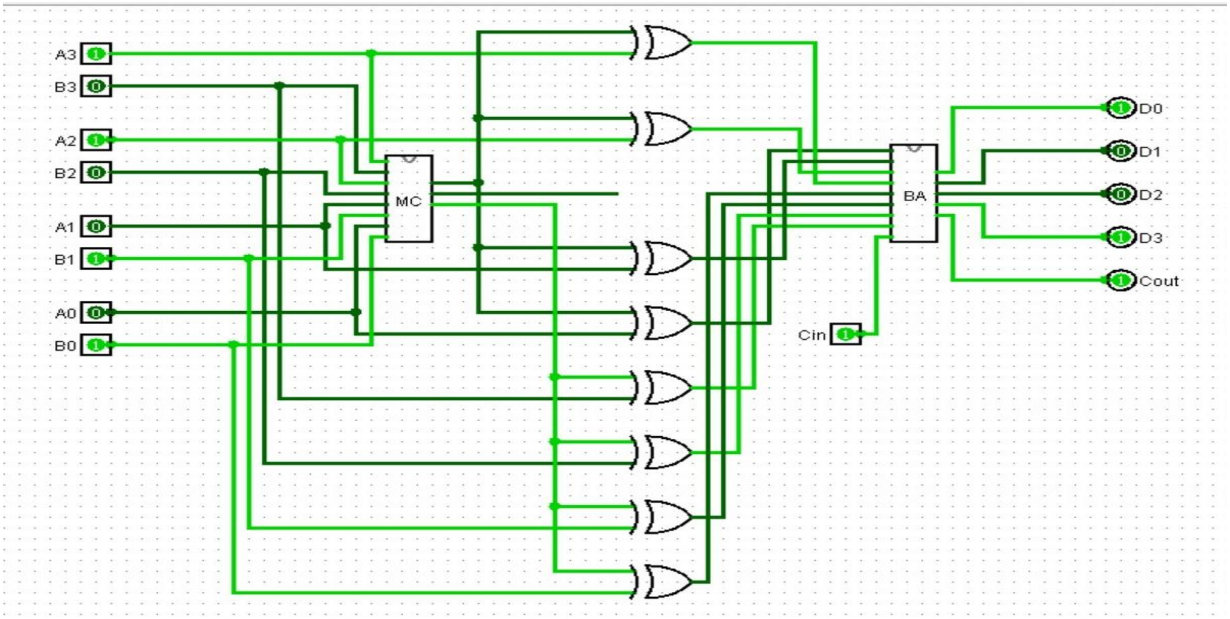


Shreya Sriram
195001106

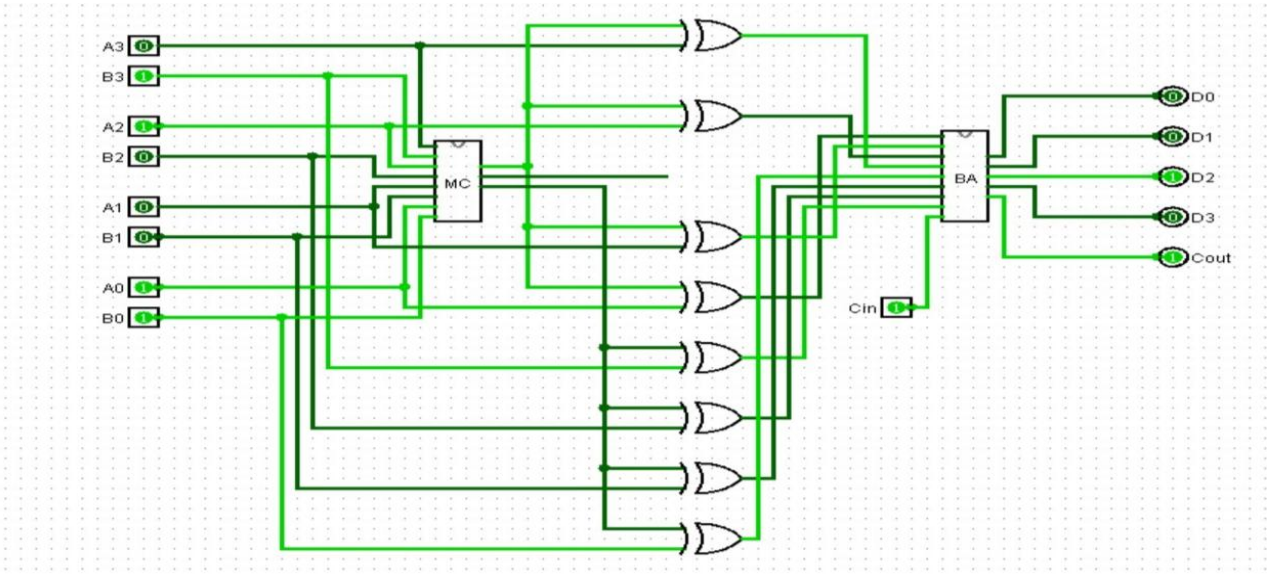
4 bit Binary Adder



Absolute Differentiator :



Shreya Sriram
195001106



Truth Table :

	C_{in}	A3	A2	A1	A0	B3	B2	B1	B0	D3	D2	D1	D0	C_{out}
12-3	1	1	1	0	0	0	0	1	1	1	0	0	1	1
5-9	1	0	1	0	1	1	0	0	1	0	1	0	0	1

Result :

The absolute differentiator circuit was designed and implemented successfully.

Shreya Sriram
195001106

Aim :

To design and implement a combinational circuit for 2:4 Decoder and 4:2 Encoder.

Apparatus Required :

S.No	Component Name	IC Number	Quantity
1	NOT Gate	7404	2
2	Ex-OR Gate	7486	3
3	AND Gate	7408	1
4	OR Gate	7432	2
5	IC Trainer Kit	-	1
6	Connecting Wires	-	30

Theory :

ENCODER

An encoder is a digital circuit that performs inverse operation of a decoder. An encoder has 2^n input lines and n output lines. In encoder the output lines generate the binary code corresponding to the input value. In octal to binary encoder it has eight inputs, one for each octal digit and three outputs that generate the corresponding binary code. In encoder it is assumed that only one input has a value of one at any given time otherwise the circuit is meaningless. It has an ambiguity that when all inputs are zero the outputs are zero. The zero outputs can also be generated when D0 = 1.

DECODER

A decoder is a multiple input multiple output logic circuit which converts coded input into coded output where input and output codes are different. The input code generally has fewer bits than the output code. Each input code word produces a different output code word i.e. there is one to one mapping can be expressed in truth table. In the block diagram of decoder circuit, the encoded information is present as n input producing 2^n possible outputs. 2^n output values are from 0 to 2^n – 1.

Truth Table :

4:2 ENCODER

X3	X2	X1	X0	Y1	Y0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

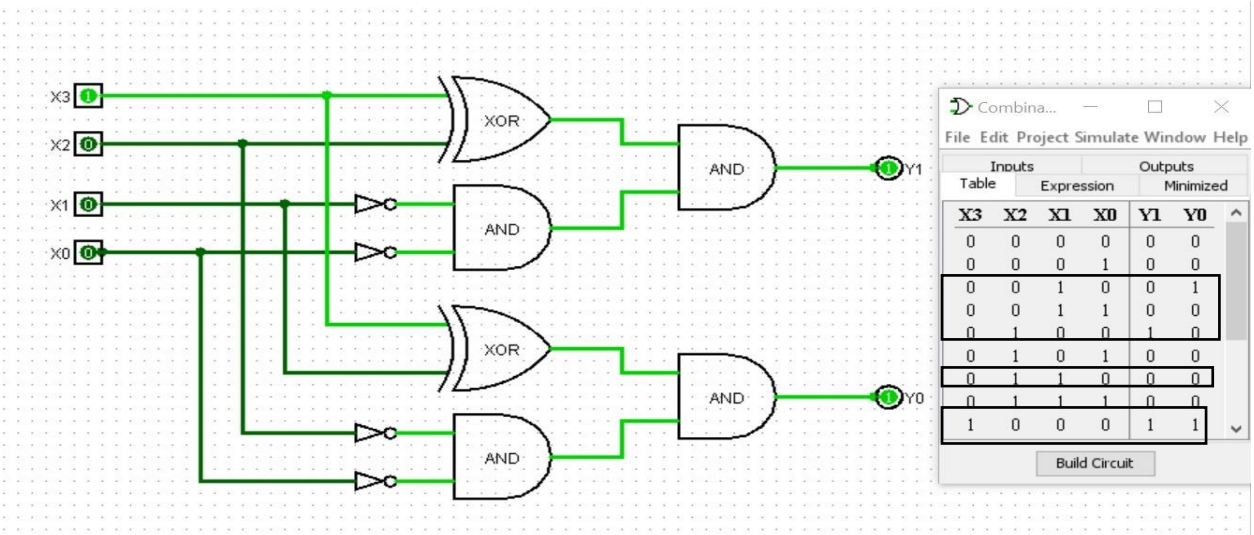
Shreya Sriram
195001106

2:4 DECODER

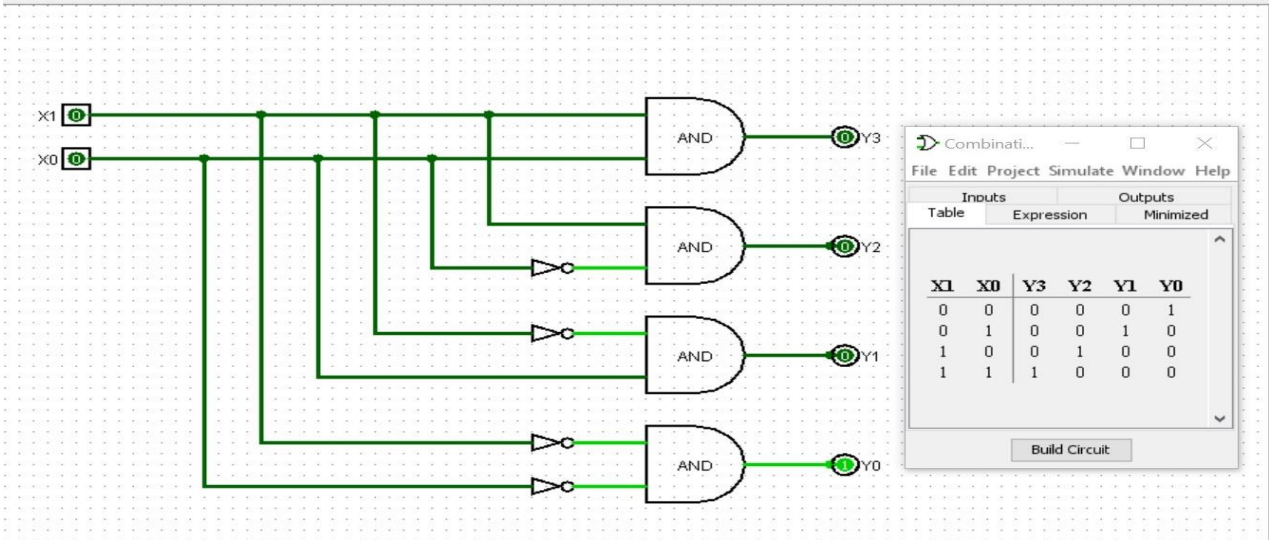
X1	X0	Y3	Y2	Y1	Y0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Logic Circuit :

4:2 ENCODER



2:4 DECODER

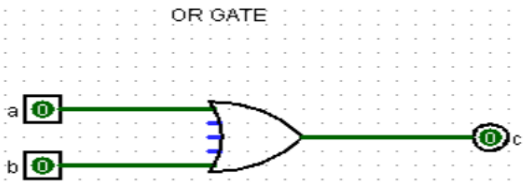


Result :

The combinational circuit for 2:4 Decoder and 4:2 Encoder were designed and implemented successfully.

EXPERIMENT -1
VERIFICATION OF LOGIC GATES

OR GATE



Combinational Analysis

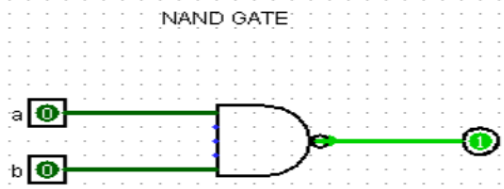
File Edit Project Simulate Window Help

Inputs Outputs Table Expression Minimized

a	b	c
0	0	0
0	1	1
1	0	1
1	1	1

Build Circuit

NAND GATE



Combinational Analysis

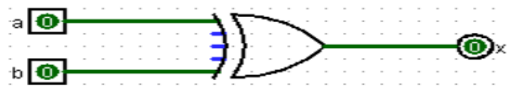
File Edit Project Simulate Window Help

Inputs Outputs Table Expression Minimized

a	b	x
0	0	1
0	1	1
1	0	1
1	1	0

Build Circuit

XOR GATE



Combinational Analysis

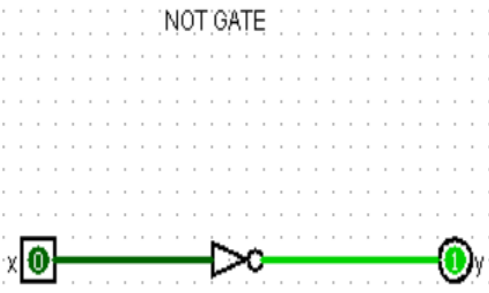
File Edit Project Simulate Window Help

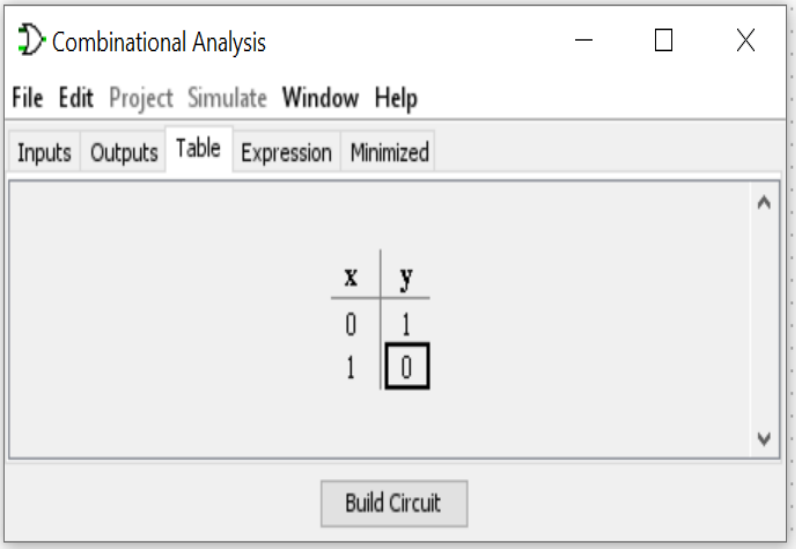
Inputs Outputs Table Expression Minimized

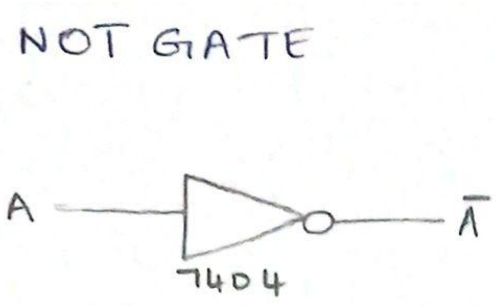
a	b	x
0	0	0
0	1	1
1	0	1
1	1	0

Build Circuit

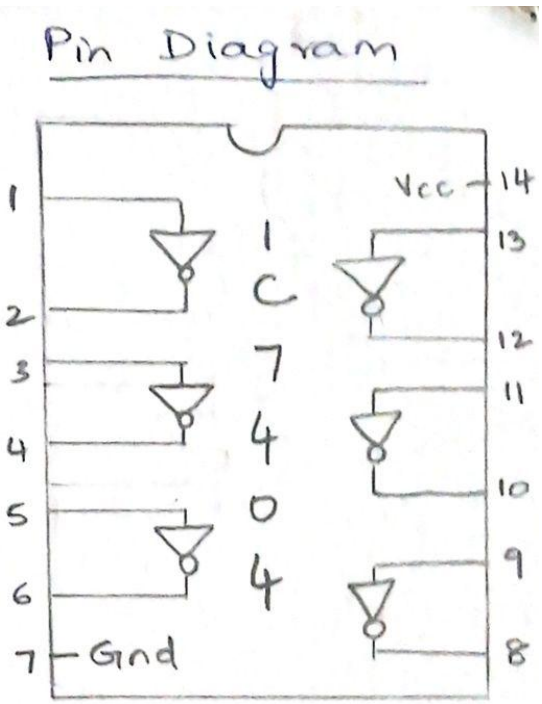
NOT GATE



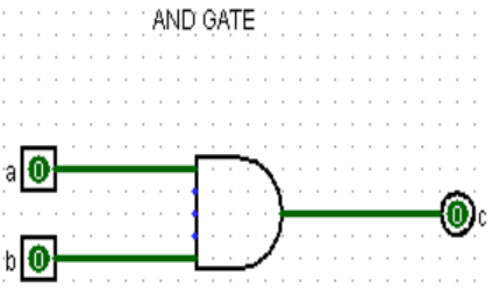


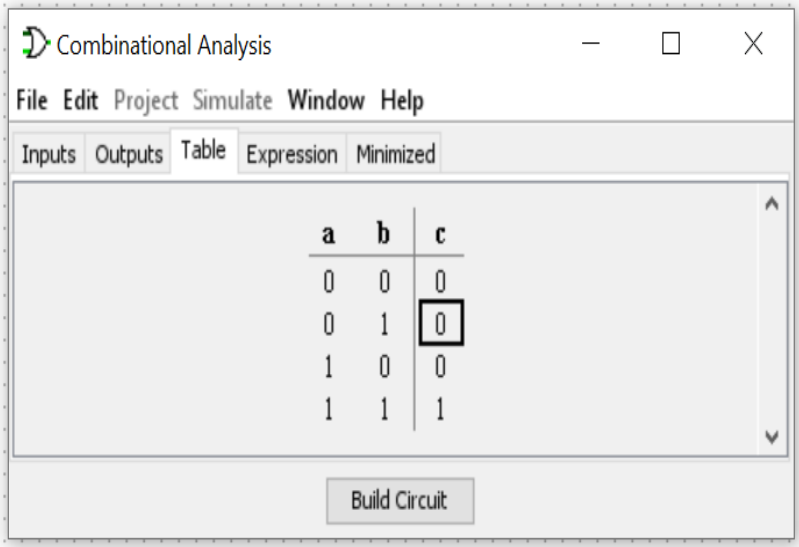


A	$\bar{A}$
0	1
1	0

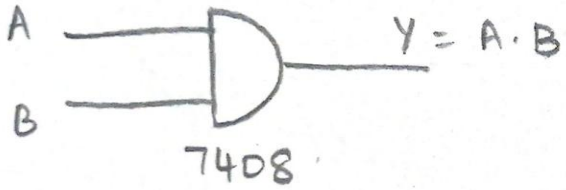


AND GATE

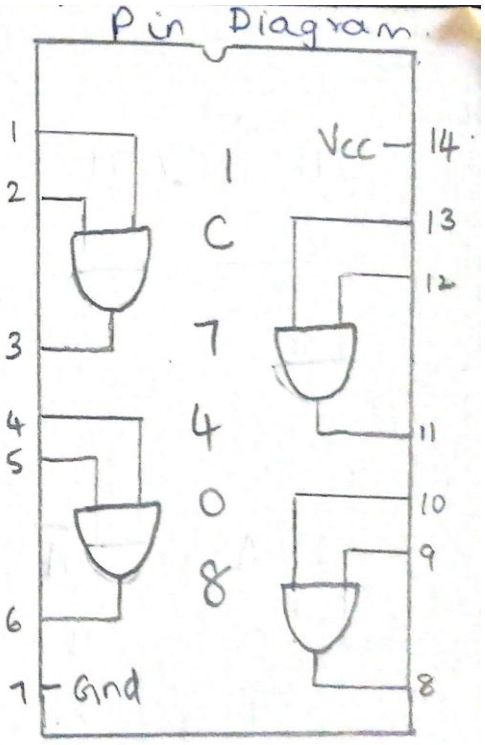




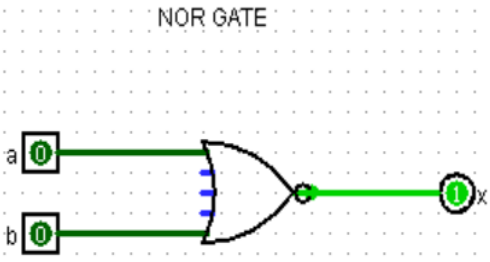
AND GATE

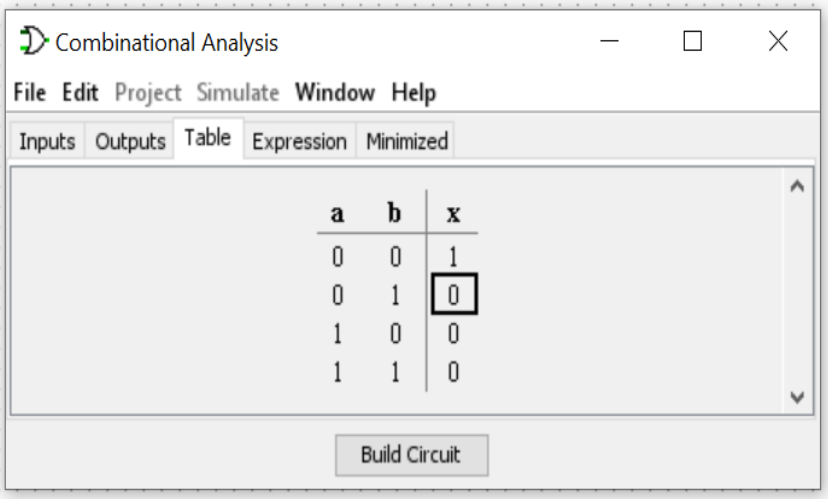


A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

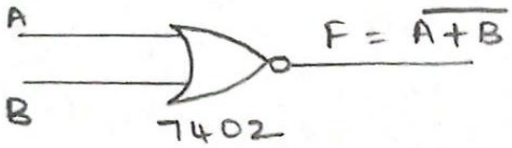


NOR GATE



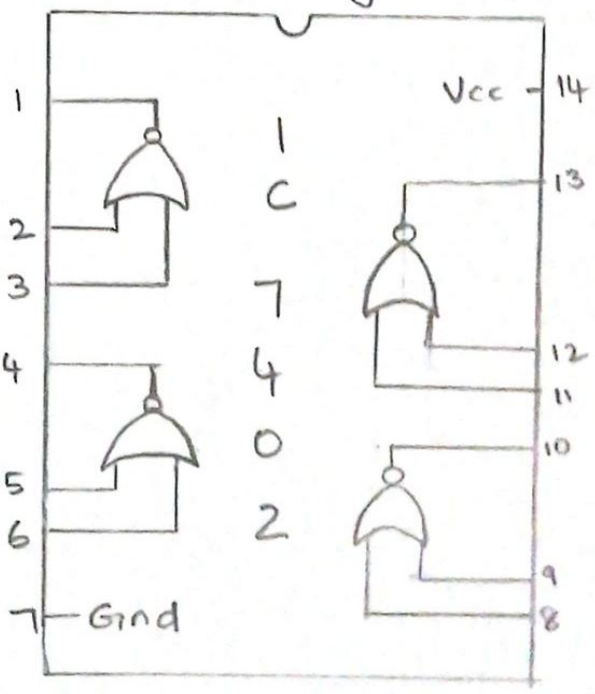


NOR GATE



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

Pin Diagram



<Student Name> Shreya Sriram
<Register Number> 195001106

Objective

To verify the Boolean theorems and postulates of Boolean algebra.

Specifications

1. Draw logic circuit for below theorems using basic logic gates
2. Verify the truth table for all the below mentioned theorems.

Components Required

S.No	Component	Specification	Quantity
1.	NOT GATE	IC 7404	2
2.	EX-OR GATE	IC 7486	3
3.	AND GATE	IC 7408	1
4.	OR GATE	IC 7432	2
5.	IC TRAINER KIT	-	1
6.	Connecting wires	-	30

<Student Name> Shreya Sriram
<Register Number> 195001106

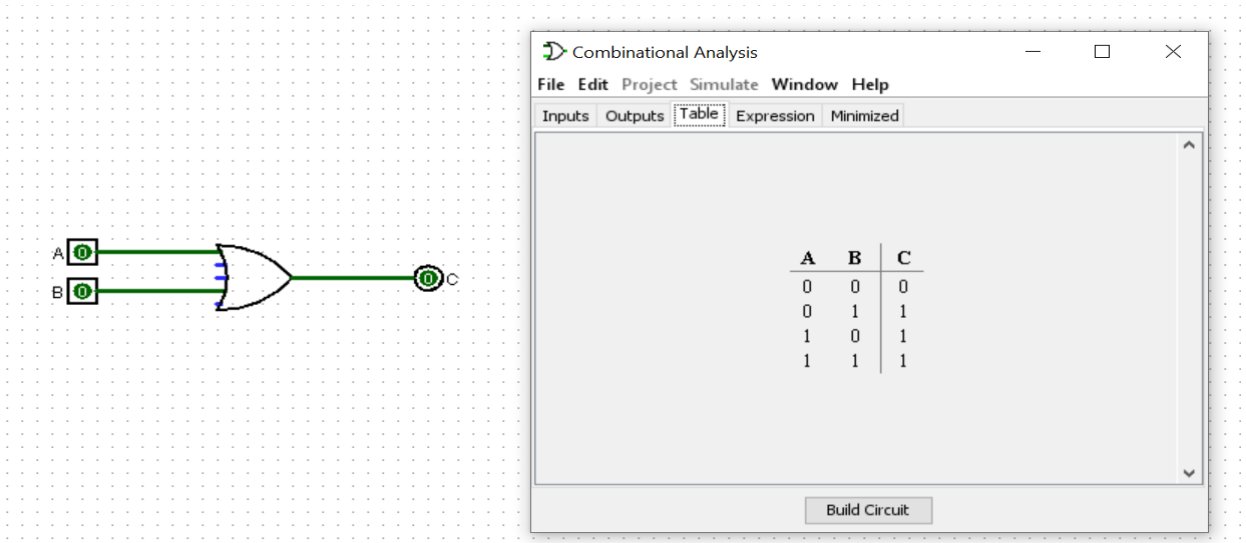
1. Commutative law

i. A+B = B+A

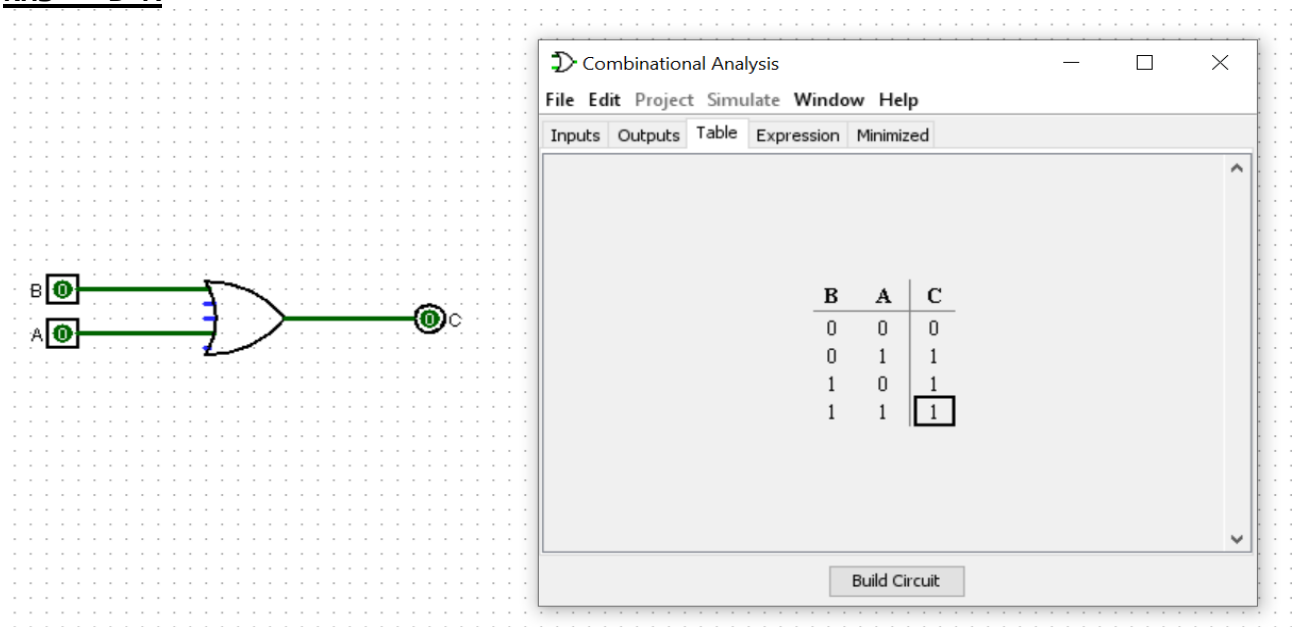
Output

Screenshot of LogiSIM - verifying Commutative Law

LHS – A+B



RHS – B+A



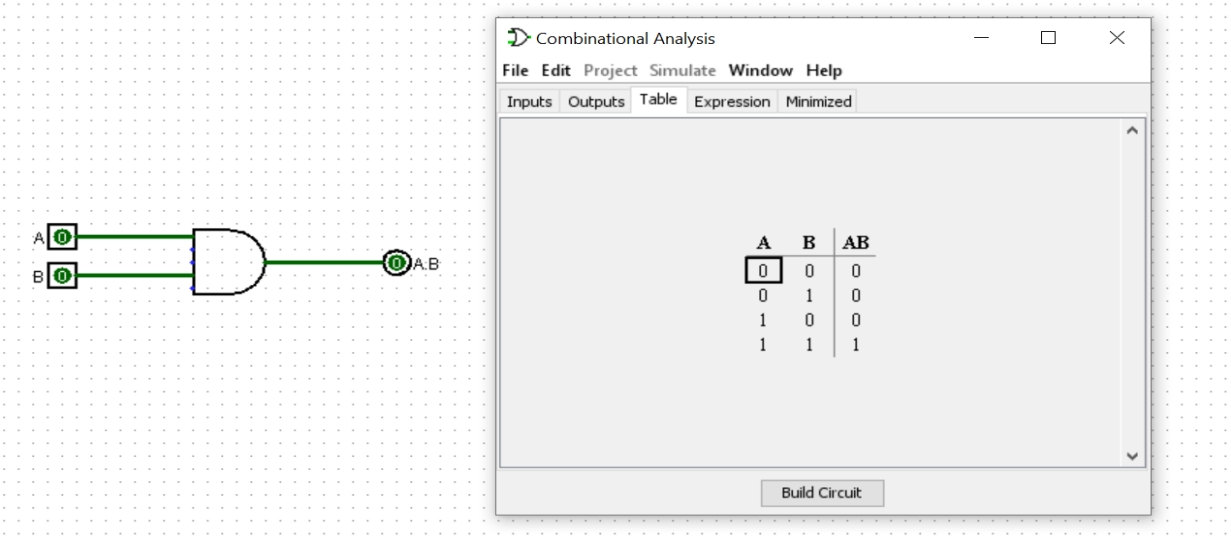
<Student Name> Shreya Sriram
<Register Number> 195001106

ii. A.B=B.A

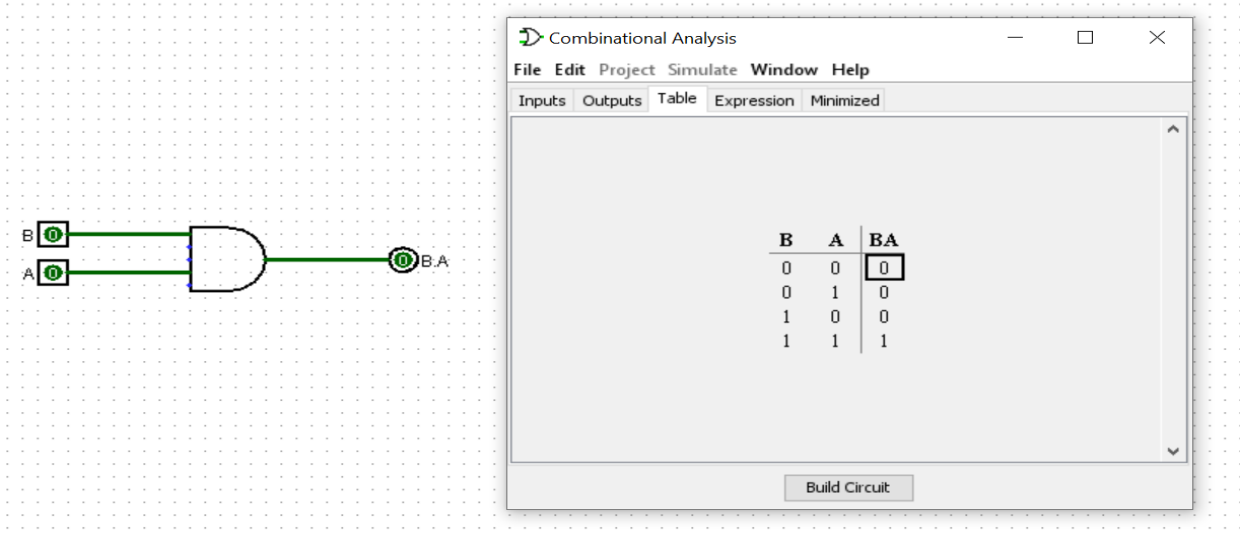
Output

Screenshot of LogiSIM - verifying Commutative Law

LHS-A.B



RHS-B.A



<Student Name> Shreya Sriram
<Register Number> 195001106

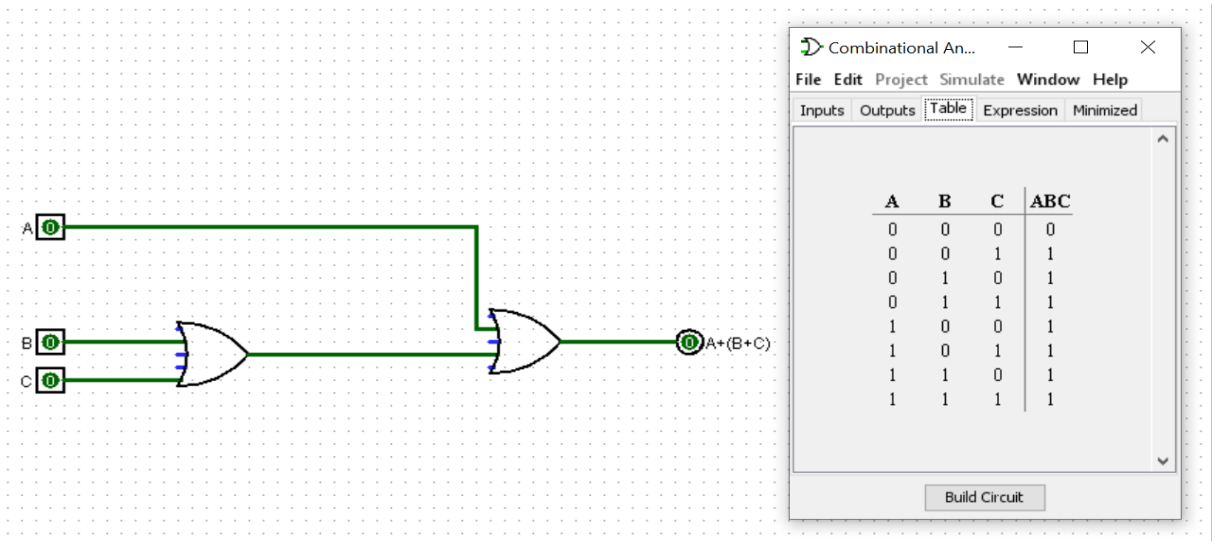
2. Associative law

i. A+(B+C) = (A+B)+C

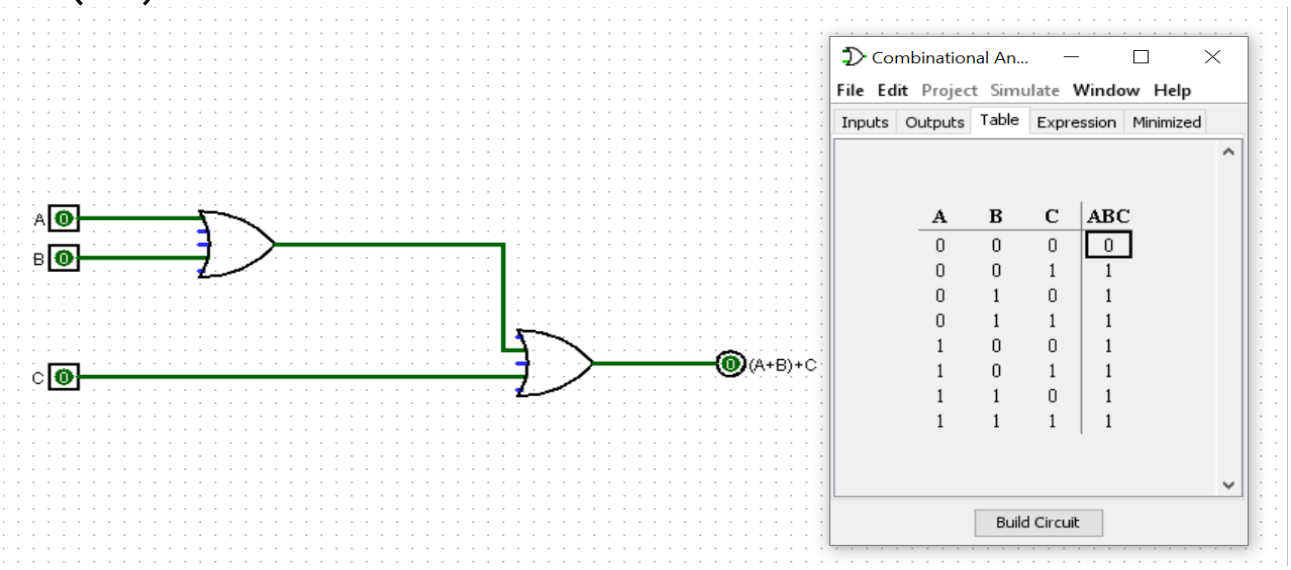
Output

Screenshot of LogiSIM - verifying Associative Law

LHS= A+(B+C)



RHS=(A+B)+C



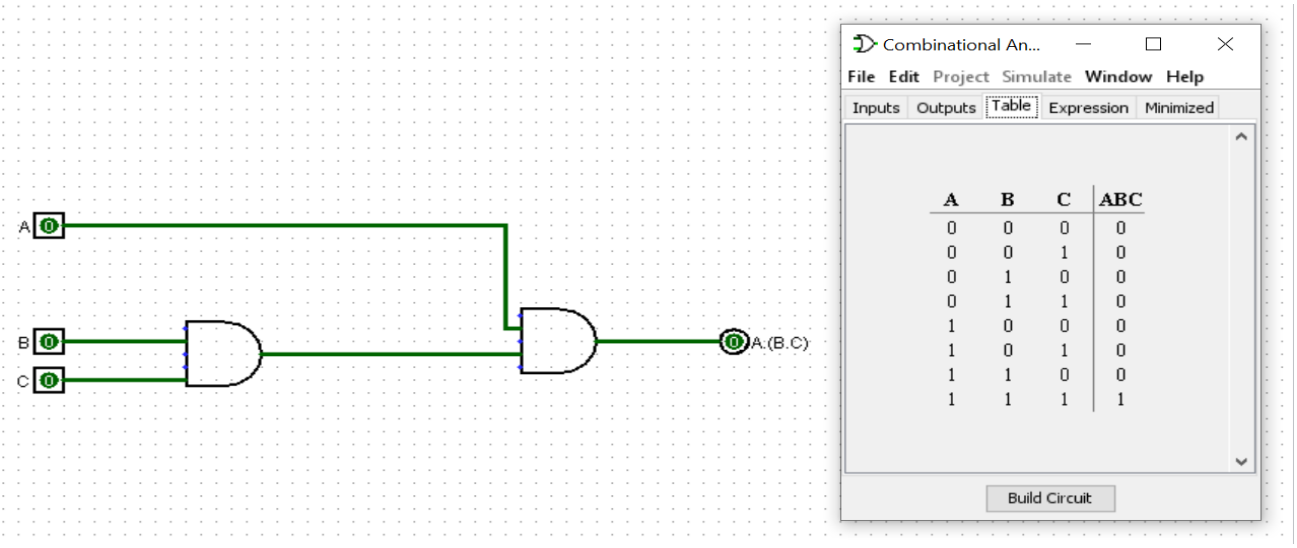
<Student Name> Shreya Sriram
<Register Number> 195001106

ii. A.(B.C) = (A.B).C

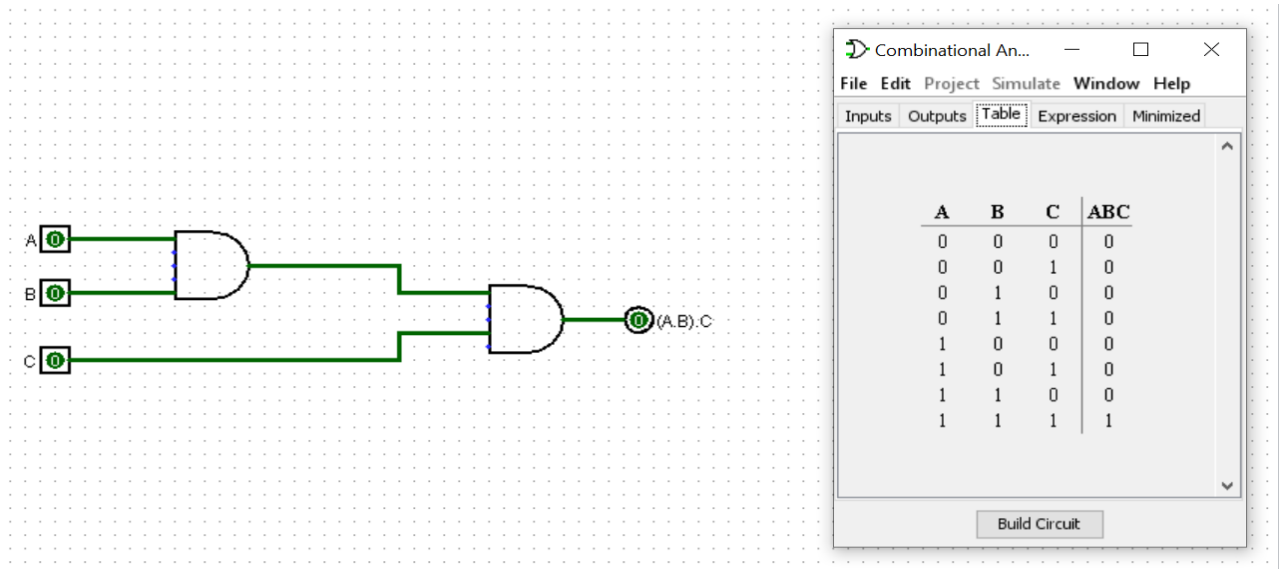
Output

Screenshot of LogiSIM - verifying Associative Law

LHS = A.(B.C)



RHS = (A.B).C



<Student Name> Shreya Sriram
<Register Number> 195001106

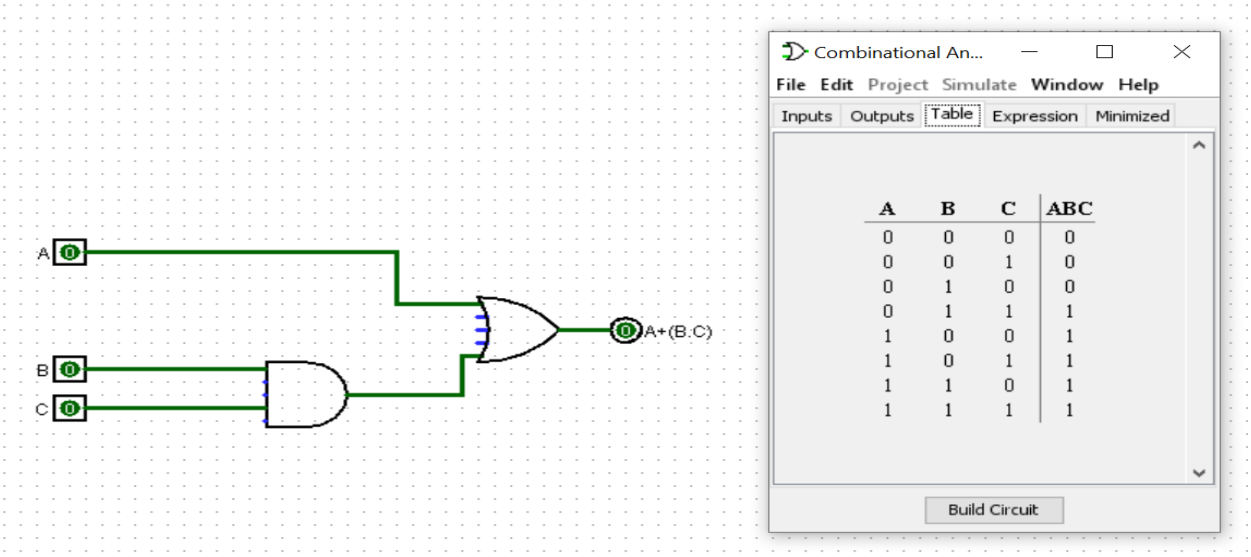
3. Distributive law

i. A+(B.C) = (A+B).(A+C)

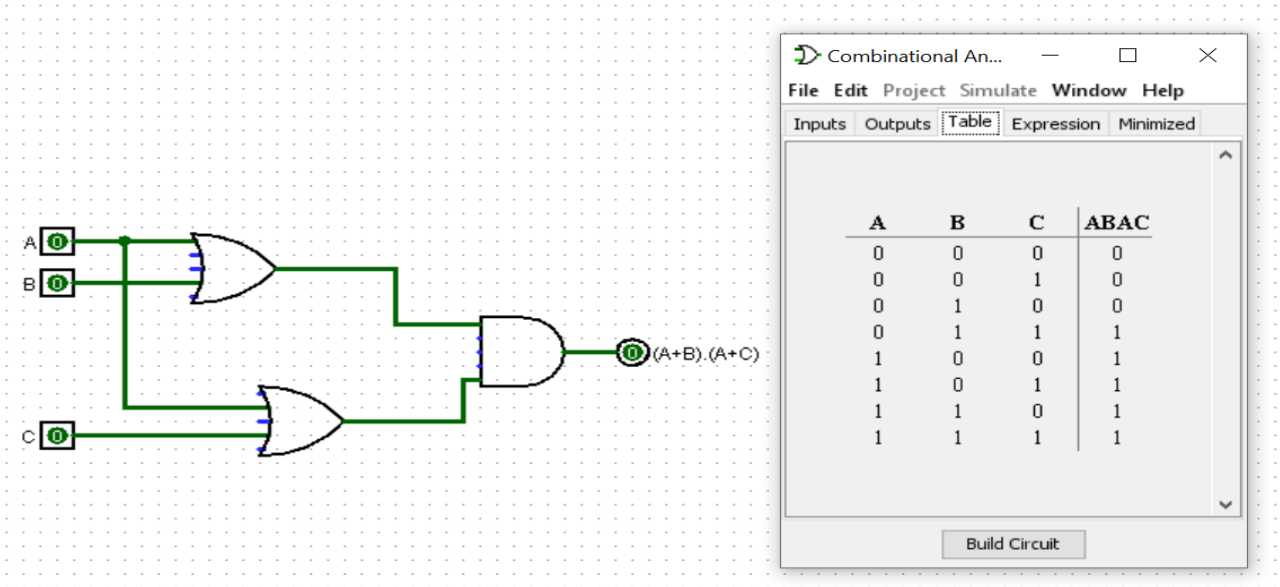
Output

Screenshot of LogiSIM - verifying Distributive Law

LHS = (A+(B.C))



RHS = (A+B).(A+C)



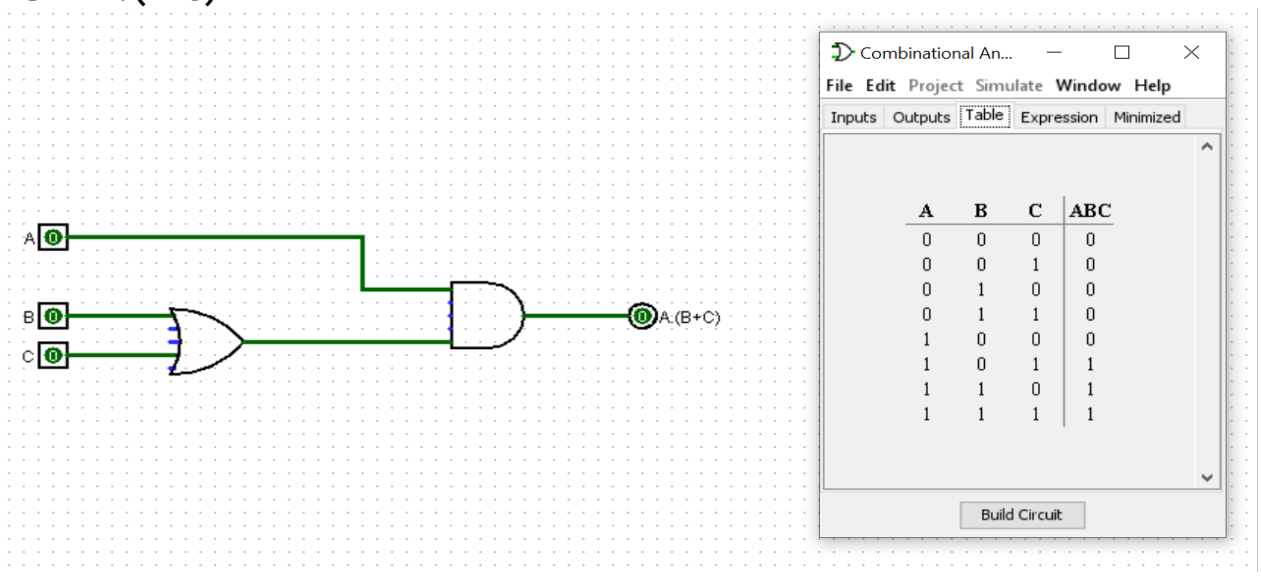
<Student Name> Shreya Sriram
<Register Number> 195001106

ii. A.(B+C) = (A.B)+(A.C)

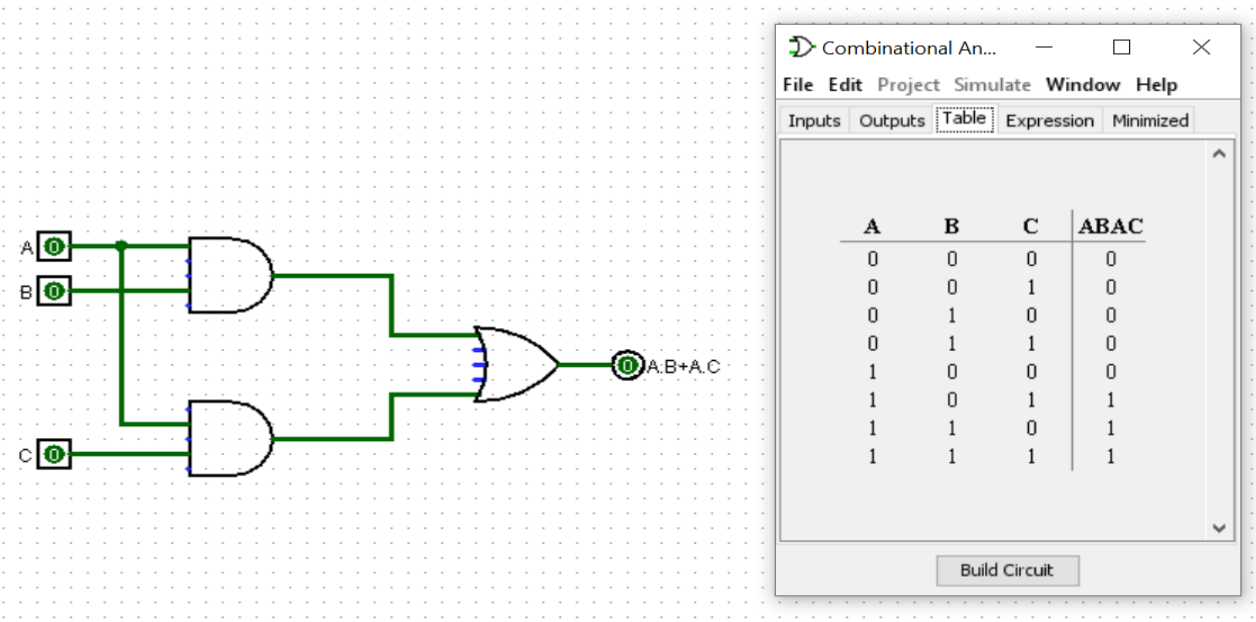
Output

Screenshot of LogiSIM - verifying Distributive Law

LHS = A.(B+C)



RHS = (A.B)+(A.C)



<Student Name> Shreya Sriram
<Register Number> 195001106

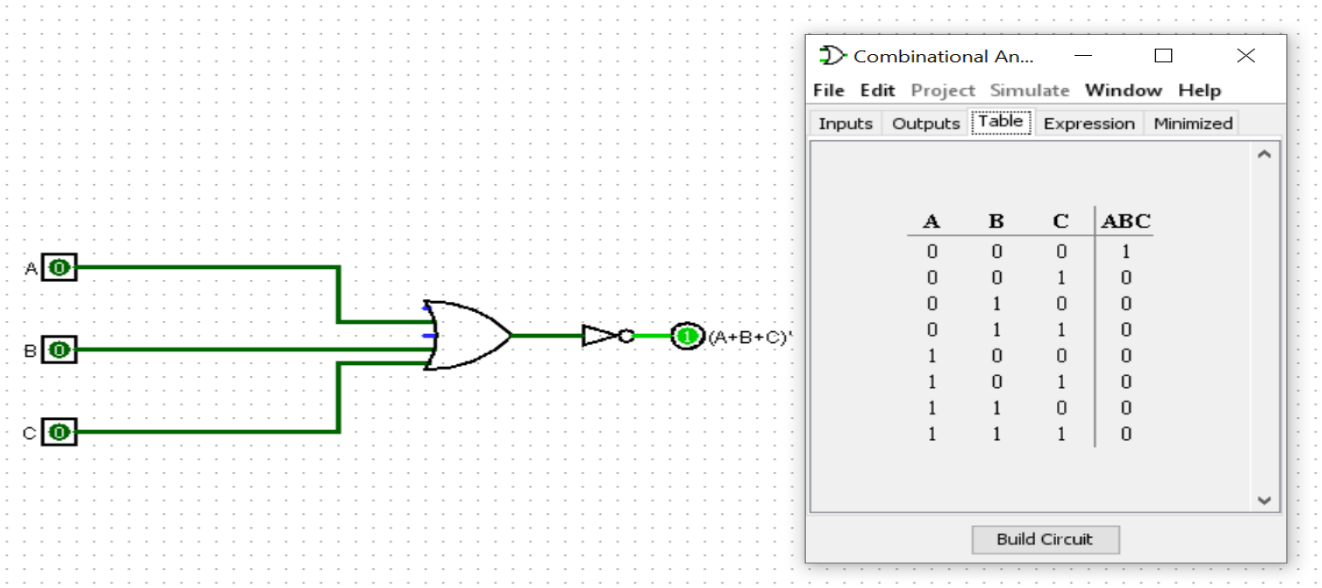
4. De - Morgan's law

i. (A + B + C)' = A' . B'. C'

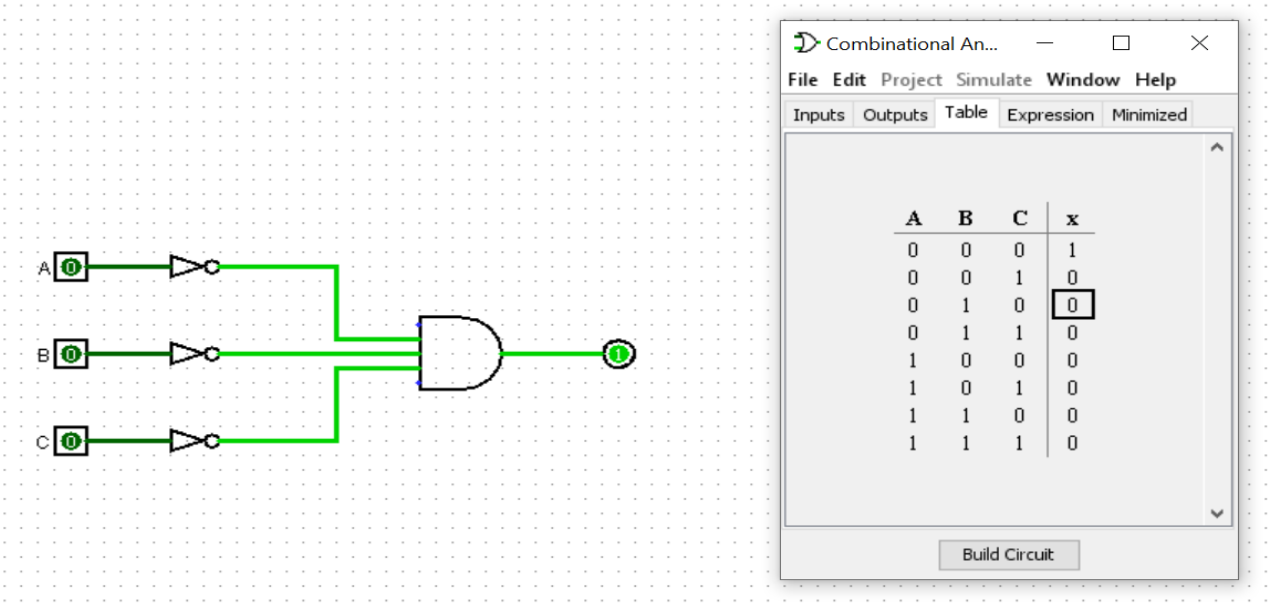
Output

Screenshot of LogiSIM - verifying De-Morgen's Law

LHS = (A+B+C)'



RHS = A'.B'.C'



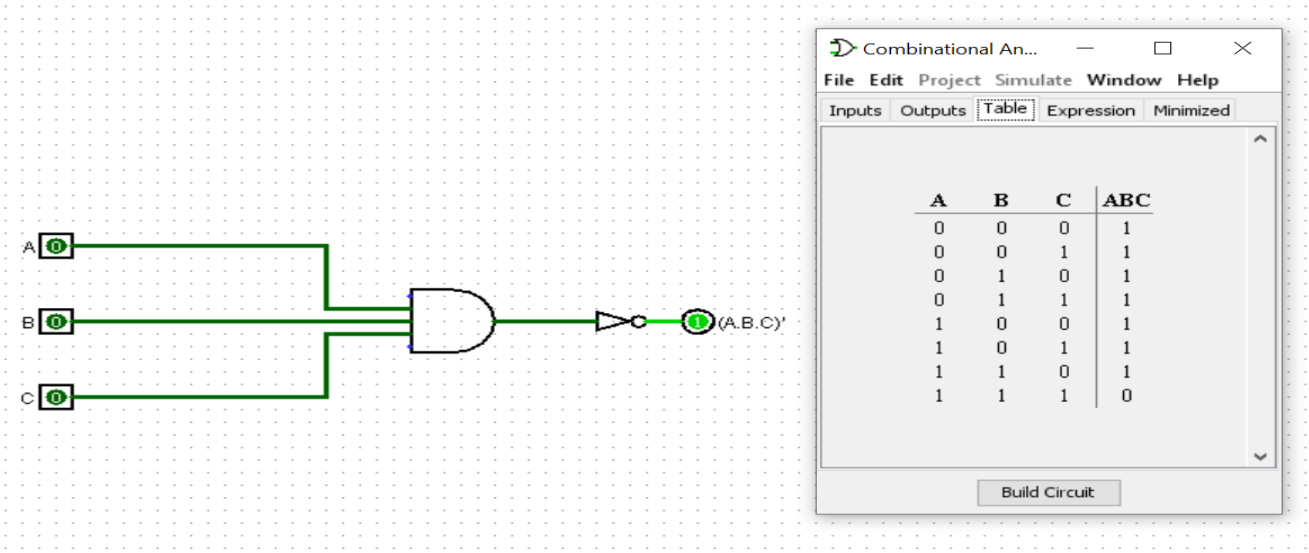
<Student Name> Shreya Sriram
<Register Number> 195001106

ii. (A.B.C)’ = A’ + B’ + C'

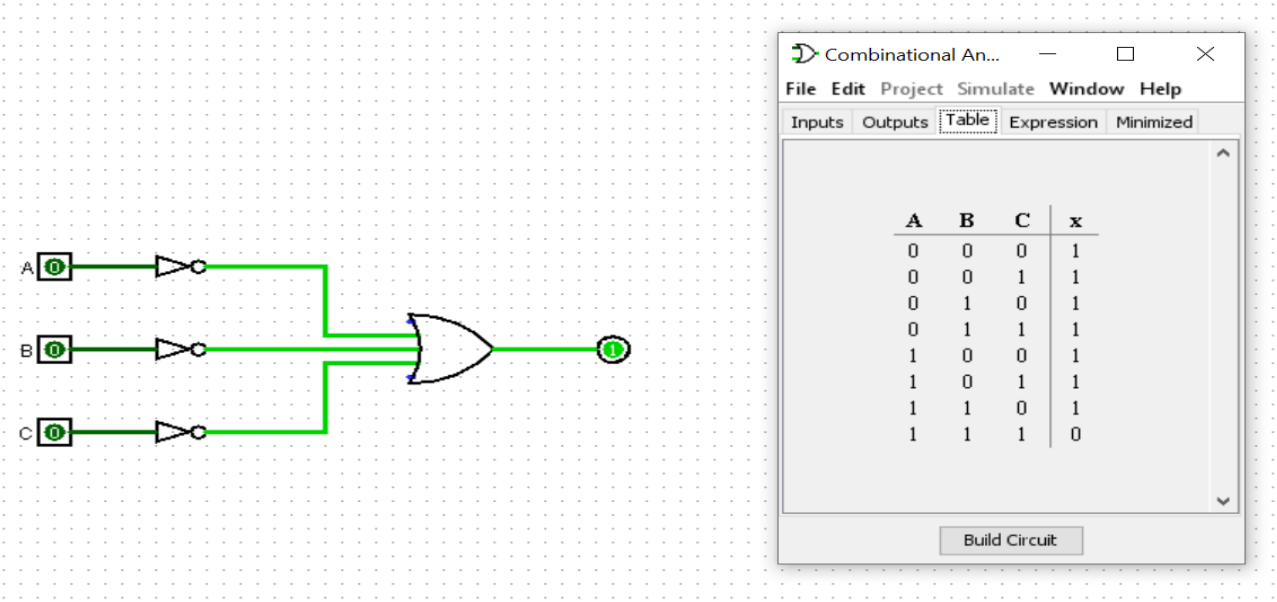
Output

Screenshot of LogiSIM - verifying De Morgens Law

LHS = (A.B.C)’



RHS = A’+B’+C’



<Student Name> Shreya Sriram
<Register Number> 195001106

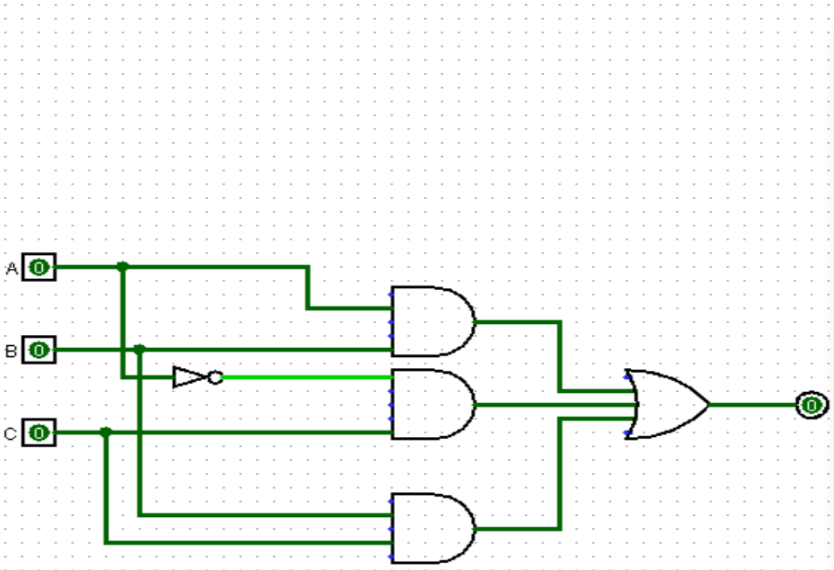
5. Consensus theorem

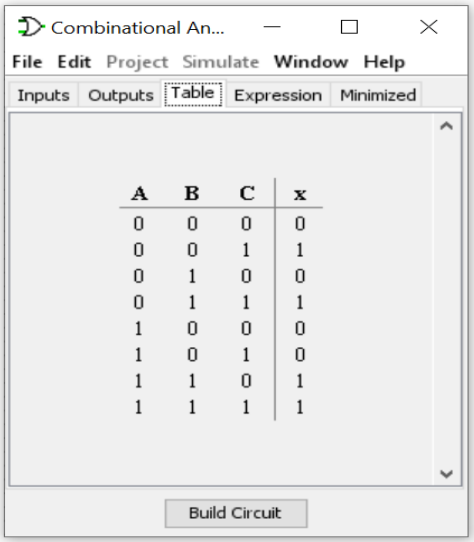
i. A.B+A’.C+B.C =A.B+A’.C

Output

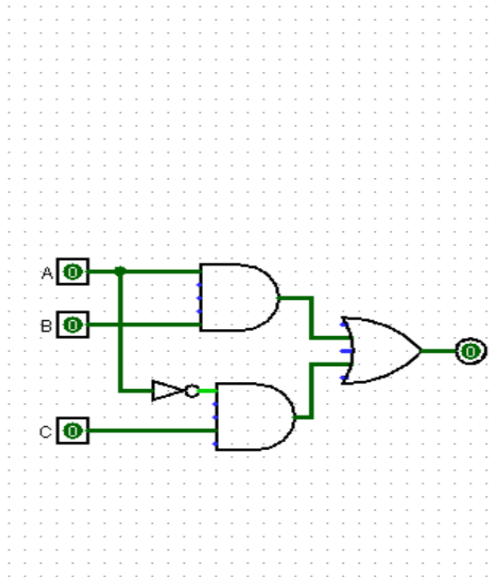
Screenshot of LogiSIM - verifying Consensus theorem

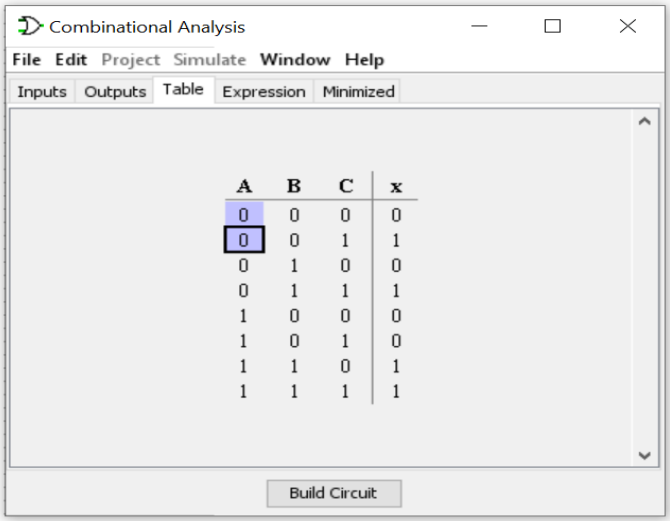
LHS = (A.B+A’.C+B.C)





RHS = (A.B+A’.C)





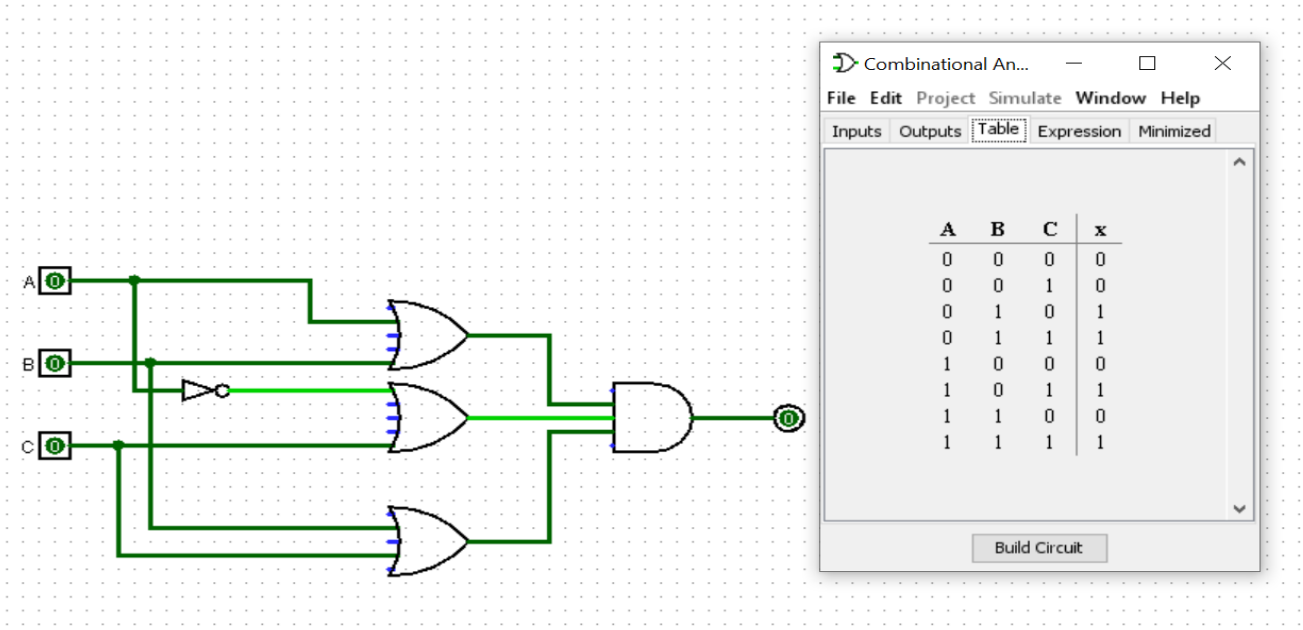
<Student Name> Shreya Sriram
<Register Number> 195001106

ii. (A+B).(A'+C).(B+C)=(A+B).(A'+C)

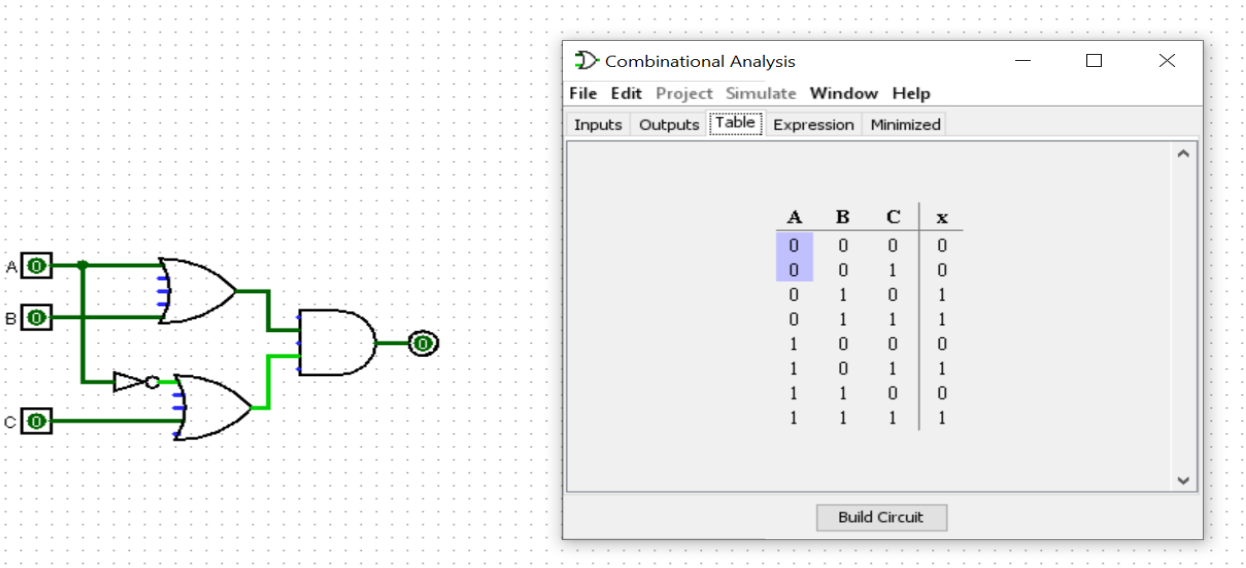
Output

Screenshot of LogiSIM - verifying Consensus Law

LHS = ((A+B).(A'+C).(B+C))



RHS = (A+B).(A'+C)



<Student Name> Shreya Sriram
<Register Number> 195001106

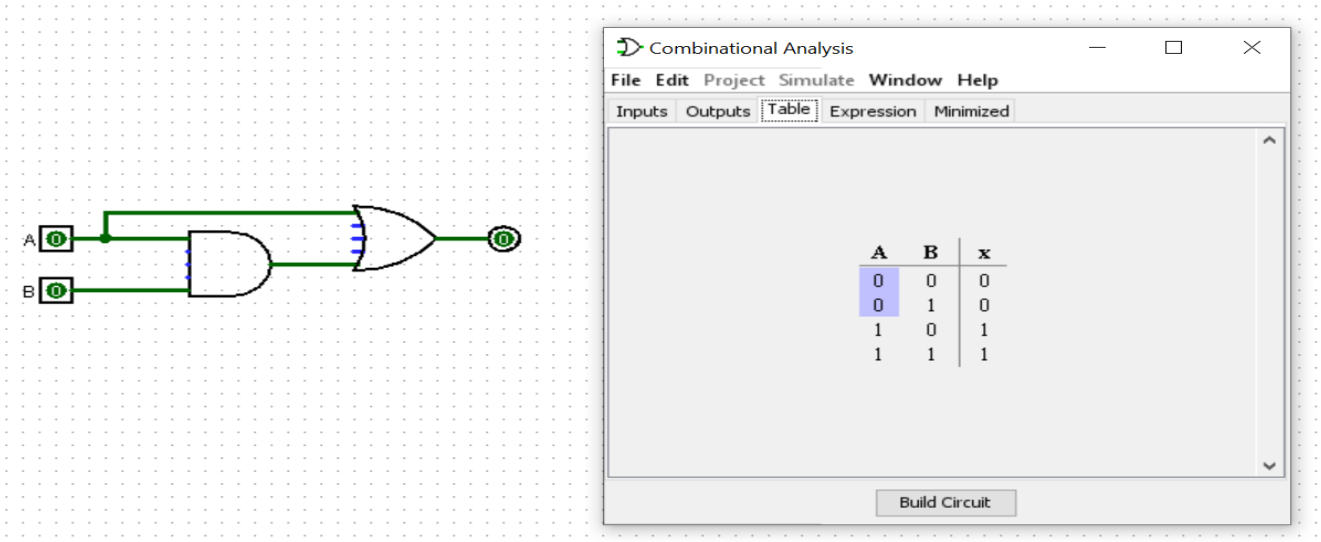
6. Absorption theorem

i. A+A.B = A

Output

Screenshot of LogiSIM - verifying Absorption theorem

LHS = (A+A.B) RHS = A

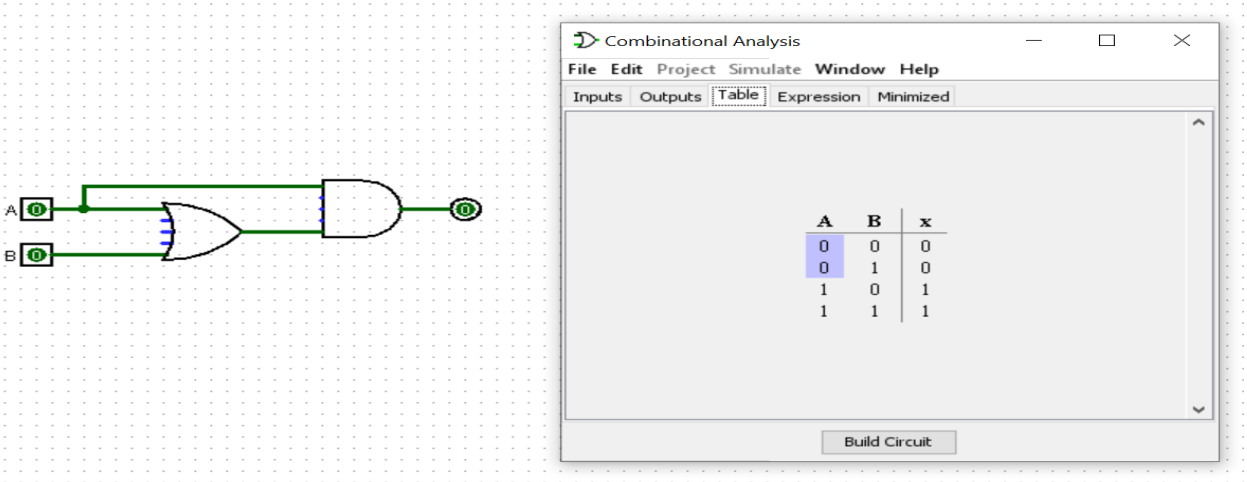


ii. A.(A+B) = A RHS = A

Output

Screenshot of LogiSIM - verifying Aborption theorem (A.(A+B)=A)

LHS = (A.(A+B))



<Student Name> Shreya Sriram
<Register Number> 195001106

7. Idempotency theorem

i. A + A =A
ii. A. A =A

8. Involution law or Double Complement Law

i. (A')' = A

9. Identity Law

i. A + 0 = A
ii. A .1 = A

10. Complementary or Inverse Law

i. A + A' = 1
ii. A . A' = 0

11. Null element for + and .(dot) operator

i. A + 1= 1
ii. A . 0 = 0

Results

The outputs of the given logic gates are verified, and the truth tables are drawn.

Name : Shreya Sriram
Register No: 195001106

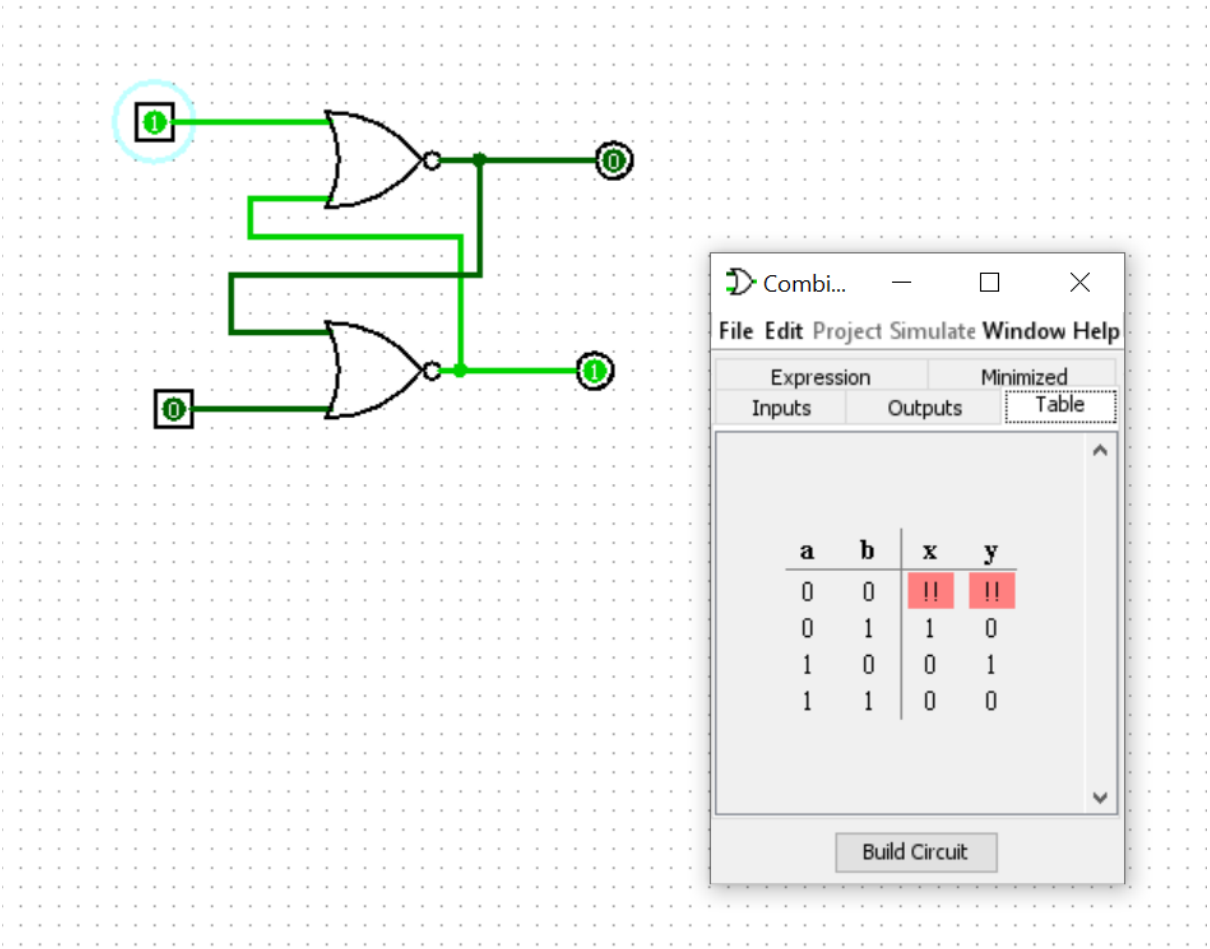
Aim :

To study different types of Flip Flops.

Apparatus Required:

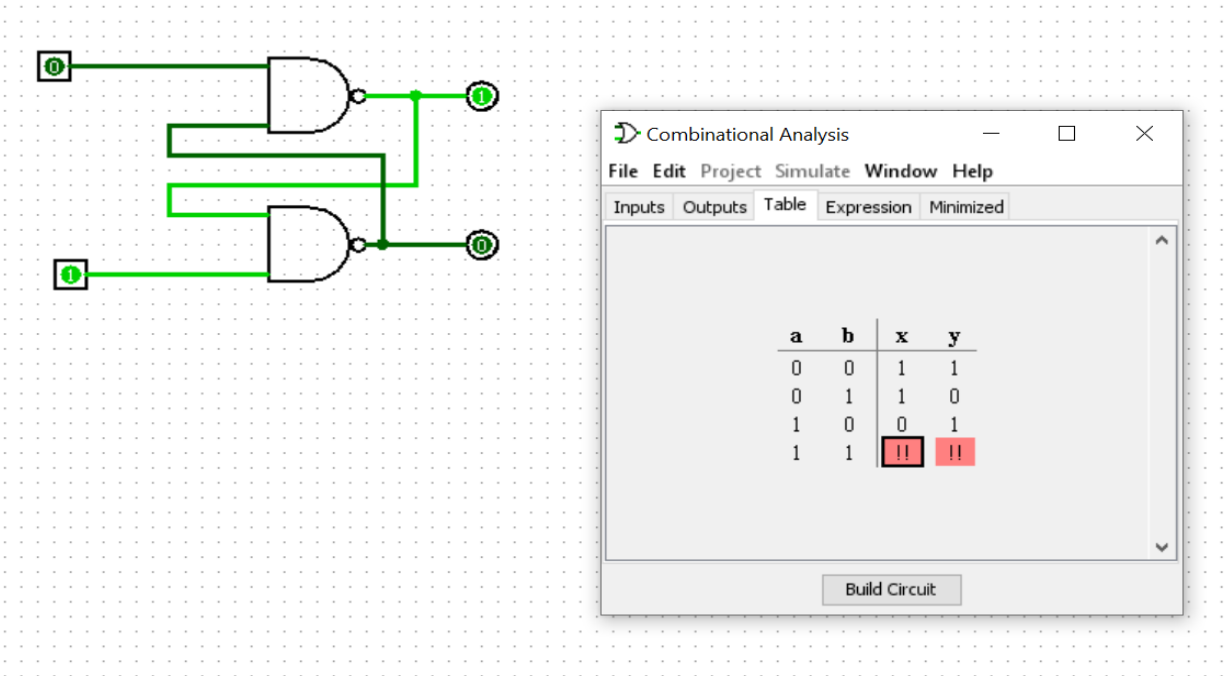
S.No	Component Name	IC Number	Quantity
1	NOT GATE	7404	2
2	EX- OR Gate	7486	3
3	AND Gate	7408	1
4	OR GATE	7432	2
5	NANAD	7400	2
6	NOR GATE	7402	2
7	IC TRAINER KIT	-	1
8	Connecting wires	-	30

1. SR Latch Using NOR

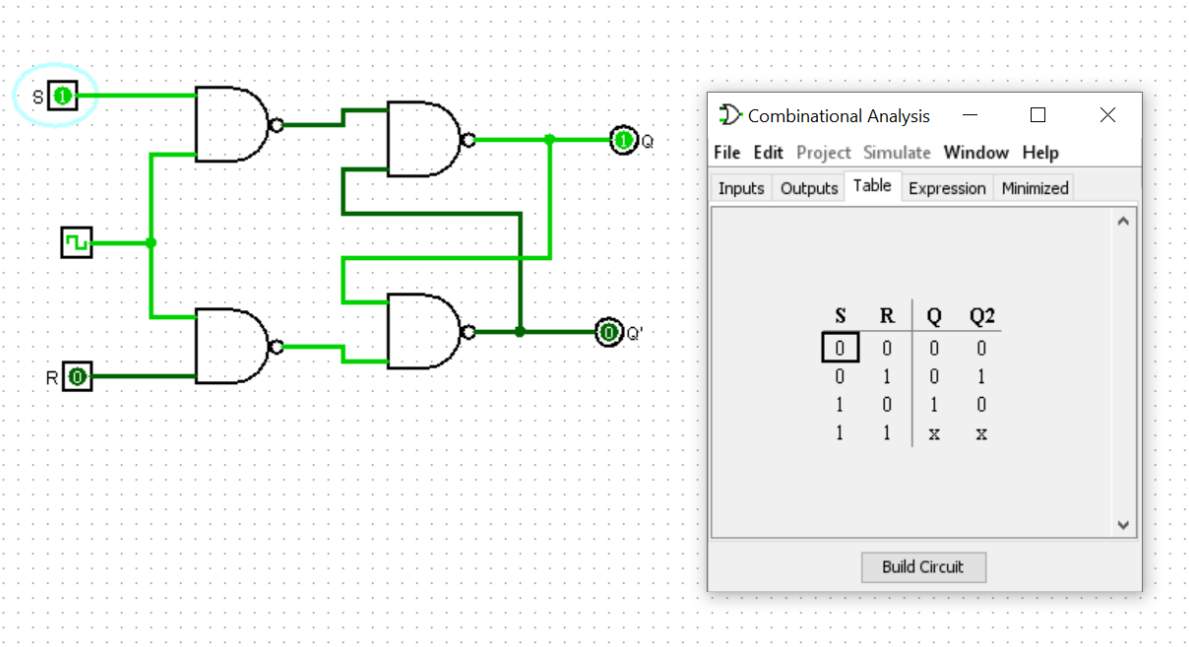


Name : Shreya Sriram
Register No: 195001106

2. SR Latch using NAND GATE



3. SR Flip Flop (Set Reset Flip Flop)

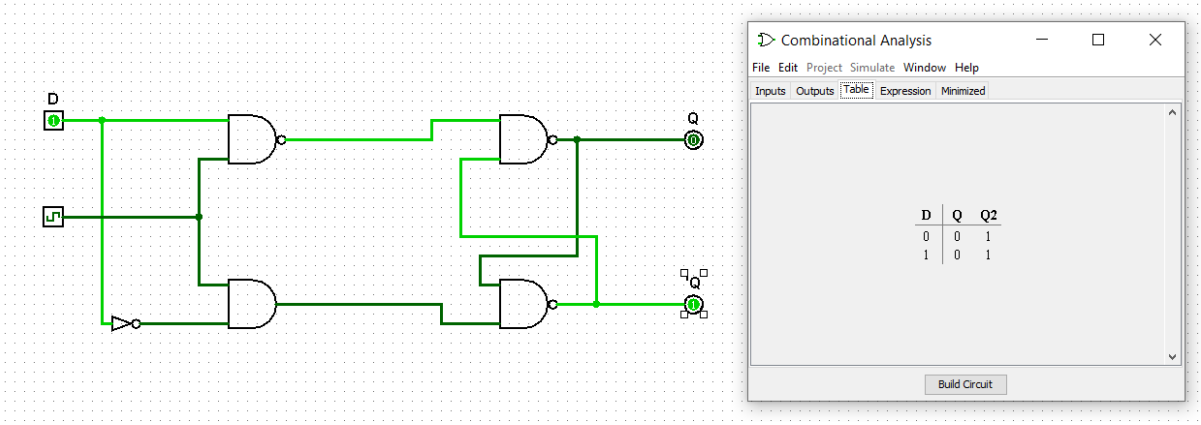


Characteristic Equation:

$$Q(t+1) = S + R' Q(t)$$

Name : Shreya Sriram
Register No: 195001106

4. D Flip Flop (Data / Delay Flip Flop)



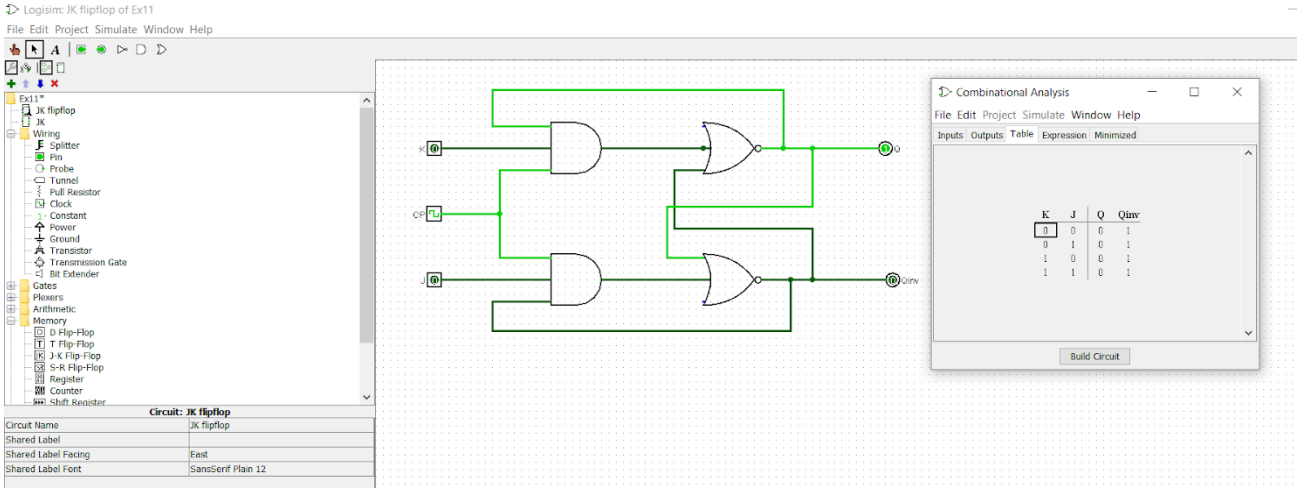
Characteristic Equation:

Q(t+1) = D

Characteristic Table of D Flip Flop

CLK	D	Q(t+1)
0	X	Q(t) No Change
1	0	0 Reset
1	1	1 Set

5. JK Flip Flop



Characteristic Equation:

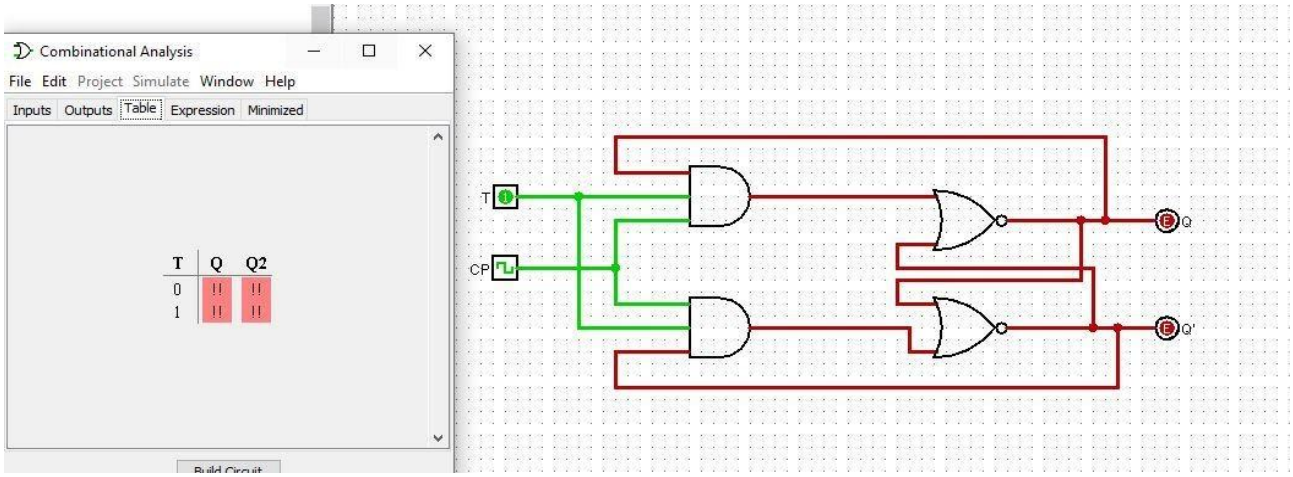
Q(t+1) = JQ(t)' + K' Q(t)

Name : Shreya Sriram
Register No: 195001106

Characteristic Table of JK Flip Flop

CLJ	J	K	Q(t+1)
0	X	X	Q(t) (No Change)
1	1	0	1 (Set)
1	0	1	0 (Reset)
1	1	1	Q(t)' (Complement)
1	0	0	Q(t) (No Change)

6. T Flip Flop (Toggle Flip Flop)



Characteristic Equation:

Q(t+1) = TQ(t)' + T'Q(t)

Q(t+1) = T⊕Q(t)

Characteristic Table of T Flip Flop

CLK	T	Q(t+1)
0	X	Q(t) (No Change)
1	1	Q(t)' (Complement)
1	0	Q(t) (No Change)

Result:

The different types of Flip Flops were designed and implemented successfully.

Shreya Sriram
195001106

Aim :

To design and implement a combinational circuit for 4:1 MUX and 1:4 DEMUX

Apparatus Required :

S.No	Component Name	IC Number	Quantity
1	NOT Gate	7404	2
2.	Ex-OR Gate	7486	3
3	AND Gate	7408	1
4	OR Gate	7432	2
5	IC Trainer Kit	-	1
6	Connecting Wires	-	30

Theory :

MULTIPLEXER

Multiplexer means transmitting a large number of information units over a smaller number of channels or lines. A digital multiplexer is a combinational 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. Normally there are 2^n input lines and n selection lines whose bit combination determines which input is selected.

DEMULTIPLEXER

The function of demultiplexer is in contrast to multiplexer function. It takes information from one line and distributes it to a given number of output lines. For this reason, the demultiplexer is also known as a data distributor. Decoder can also be used as a demultiplexer.

In the 1:4 demultiplexer circuit, the data input line goes to all of the AND gates. The data select lines enable only one gate at a time and the data input line will pass through the selected gate to the associated data output line.

Multiplexer :

FUNCTION TABLE

S1	S0	INPUTS Y
0	0	D0 -> D0S1'S0'
0	1	D1 -> D1S1'S0
1	0	D2 -> D2S1S0'
1	1	D3 -> D3S1S0

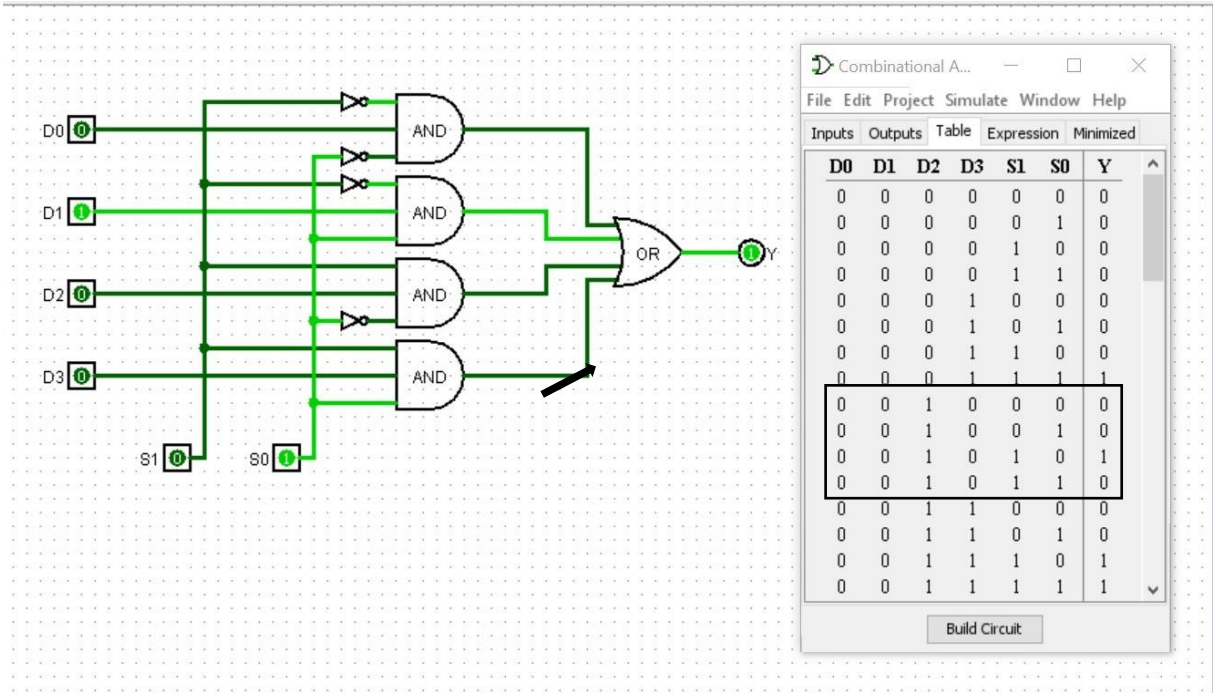
$$Y = D0S1'S0' + D1S1'S0 + D2S1S0' + D3S1S0$$

Shreya Sriram
195001106

TRUTH TABLE

S1	S0	Y = OUTPUT
0	0	D0
0	1	D1
1	0	D2
1	1	D3

LOGIC CIRCUIT



Demultiplexer :

FUNCTION TABLE

S1	S0	INPUTS Y
0	0	X -> D0 = XS1'S0'
0	1	X -> D1 = XS1'S0
1	0	X -> D2 = XS1S0'
1	1	X -> D3 = XS1S0

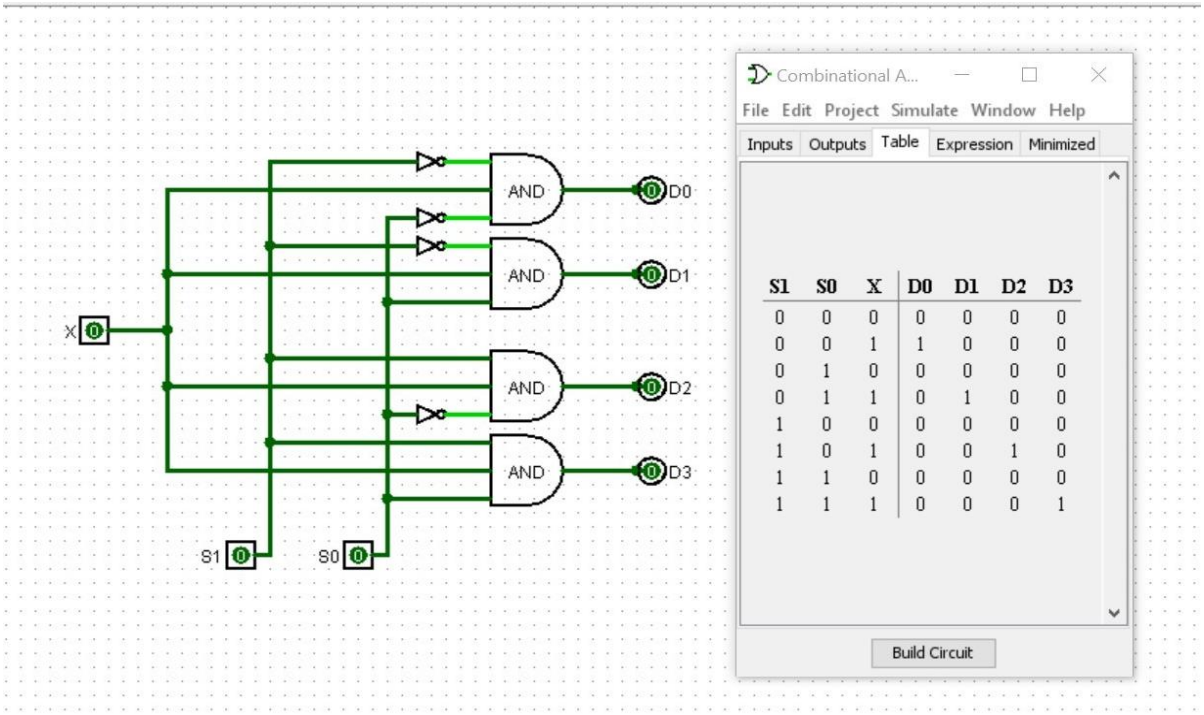
$$Y = XS1'S0' + XS1'S0 + XS1S0' + XS1S0$$

Shreya Sriram
195001106

TRUTH TABLE

INPUT			OUTPUT			
S1	S0	I/P	D0	D1	D2	D3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

LOGIC CIRCUIT



Result :

The combinational circuit for 4:1 MUX and 1:4 DEMUX were designed and implemented successfully.

<Student Name> Shreya Sriram
<Register Number>195001106

DESIGNING PARITY GENERATOR AND CHECKER

Objective

To design and implement a combinational circuit for odd and even parity generator and checker with 3-bit data.

APPARATUS REQUIRED:

S.No	Component Name	IC Number	Quantity
1	NOT GATE	IC 7404	2
2	EX-OR GATE	IC 7486	3
3	IC TRAINER KIT	-	1
4	PATCH CORDS	-	30

EVEN PARITY GENERATOR:

Truth table:

3-bit message			Even parity generator
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Sum of minterms (SOP):

F=A`B`C+A`BC`+AB`C`+ABC

<Student Name> Shreya Sriram
<Register Number>195001106

Truth Table:

A/BC	00	01	11	10
00	0	1	0	1
01	1	0	1	0

Simplified Boolean Expression

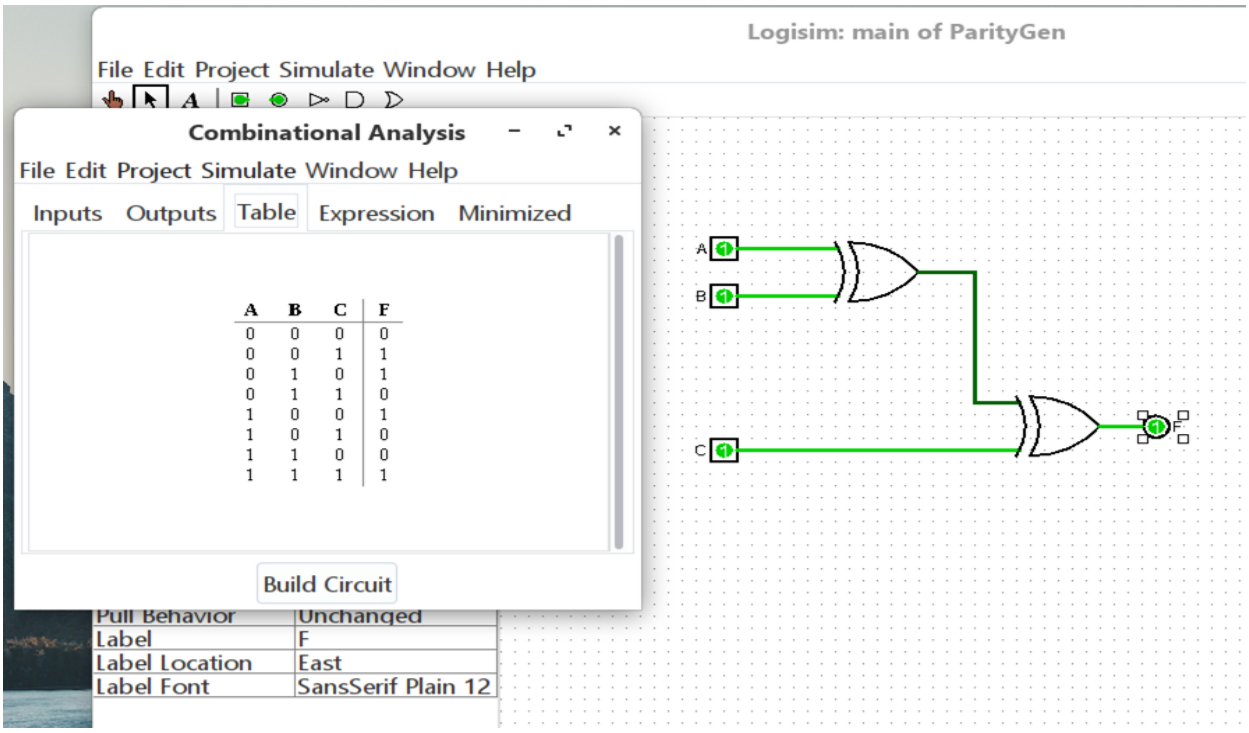
$$P = \bar{A}\,\bar{B}\,C + \bar{A}\,B\,\bar{C} + A\,\bar{B}\,\bar{C} + A\,B\,C$$

$$= \bar{A}\,(\bar{B}\,C + B\,\bar{C}) + A\,(\bar{B}\,\bar{C} + B\,C)$$

$$= \bar{A}\,(B \oplus C) + A\,(\overline{B \oplus C})$$

$$P = A \oplus B \oplus C$$

Logic Diagram:



<Student Name> Shreya Sriram
<Register Number>195001106

ODD PARITY GENERATOR:

Truth table:

3-bit message			Even parity generator
A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Sum of minterms (SOP):

F=ABC`+AB`C+A`BC+A`B`C`

Truth Table:

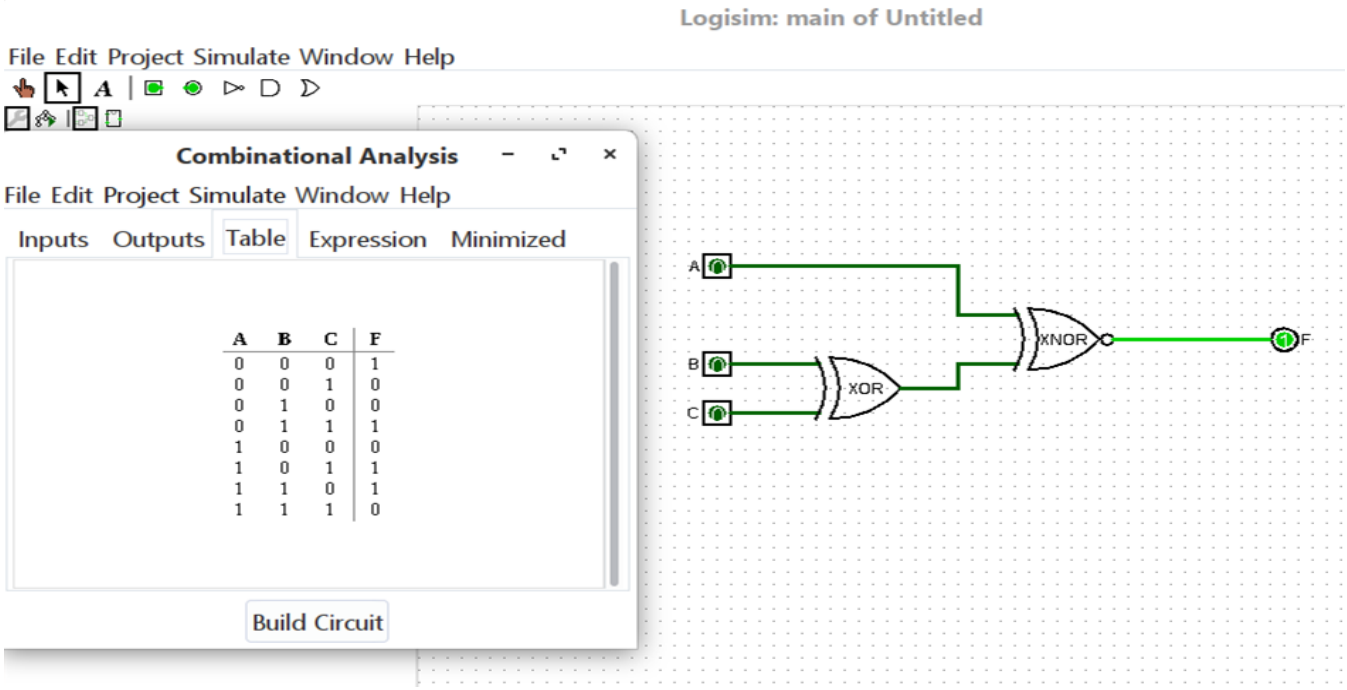
A/BC	00	01	11	10
00	1	0	1	0
01	0	1	0	1

Simplified Boolean Expression

F=ABC`+AB`C+A`BC+A`B`C`
=A(BC`+B`C) + A`(BC+B`C`)
=A (XOR (B, C)) + A` (XNOR (B, C))
=XNOR (A, XOR (B, C)).

<Student Name> Shreya Sriram
<Register Number>195001106

Logic Diagram:



<Student Name> Shreya Sriram
<Register Number>195001106

EVEN PARITY CHECKER

TRUTH TABLE:

A	B	C	P	EPC
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

SOP EXPRESSION

EPC = (A’B’C’P) + (A’B’CP’) + (A’BC’P’) + (A’BCP) + (AB’C’P’) + (AB’CP) + (ABC’P) + (ABCP’)

K MAP

AB/CD	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

<Student Name> Shreya Sriram
<Register Number>195001106

Simplified Expression

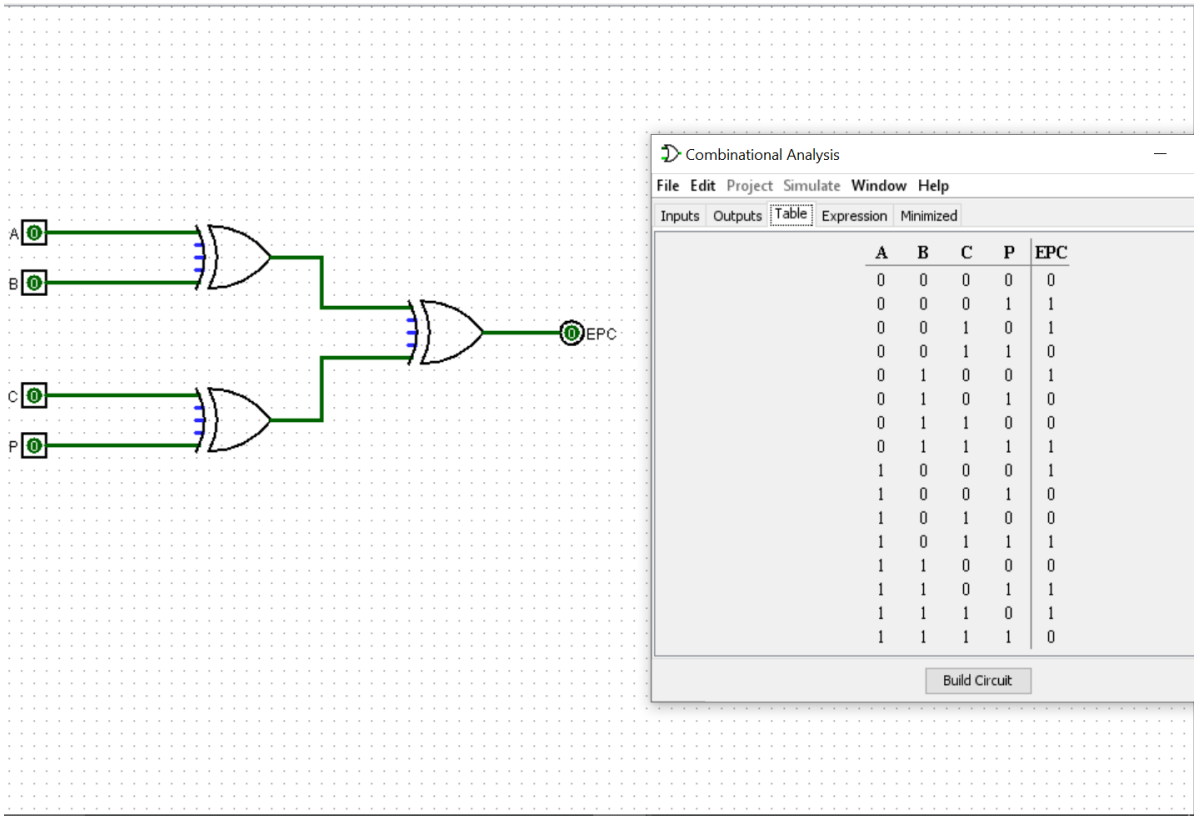
$$PEC = \bar{A}\,\bar{B}\,(\bar{C}\,D + C\,\bar{D}) + \bar{A}\,B\,(\bar{C}\,\bar{D} + C\,D) + A\,B\,(\bar{C}\,D + C\,\bar{D}) + A\,\bar{B}\,(\bar{C}\,\bar{D} + C\,D)$$

$$= \bar{A}\,\bar{B}\,(C \oplus D) + \bar{A}\,B\,(\overline{C \oplus D}) + A\,B\,(C \oplus D) + A\,\bar{B}\,(\overline{C \oplus D})$$

$$= (\bar{A}\,\bar{B} + A\,B)\,(C \oplus D) + (\bar{A}\,B + A\,\bar{B})\,(\overline{C \oplus D})$$

$$= (A \oplus B) \oplus (C \oplus D)$$

Logic Diagram



A	B	C	P	EPC
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

<Student Name> Shreya Sriram
<Register Number>195001106

ODD PARITY CHECKER

TRUTH TABLE

A	B	C	P	OPC
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

SOP EXPRESSION

OPC = (A'B'C'P') + (A'B'CP) + (A'BC'P) + (A'BCP') + (AB'C'P) + (AB'CP') + (ABC'P') + (ABCP)

K MAP

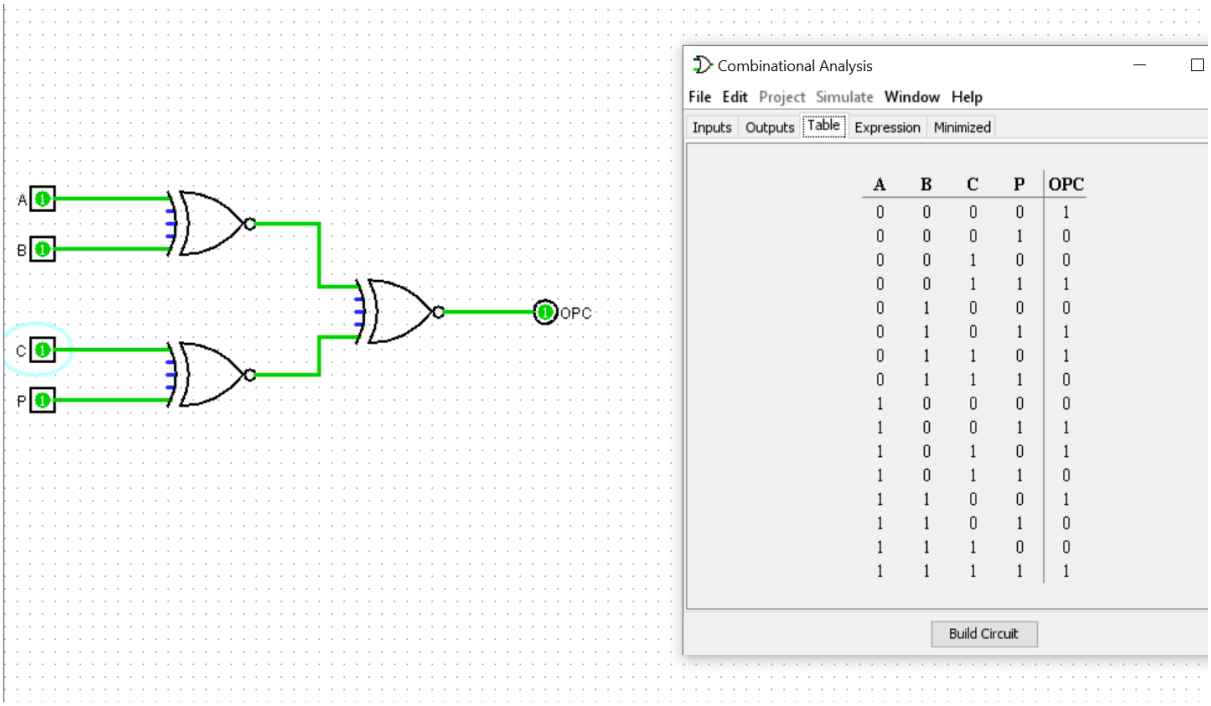
AB/CD	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	1	0	1	0
10	0	1	0	1

<Student Name> Shreya Sriram
<Register Number>195001106

Simplified Expression

OPC = (A Ex-NOR B) Ex-NOR (C Ex-NOR D)

Logic Diagram:



A	B	C	P	OPC
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

RESULT:

The combinational circuits for odd and even parity generator and checker with 3-bit data have been designed and implemented and truth tables have been verified.

Shreya Sriram
195001106

Aim :

To design and implement a 4:2 Priority Encoder.

Apparatus Required :

S.No	Component Name	IC Number	Quantity
1	NOT Gate	7404	2
2	AND Gate	7408	1
2	OR Gate	7432	2
4	IC Trainer Kit	-	1
6	Connecting Wires	-	30

Priority Encoder :

Encoder circuit is a combinational circuit which takes binary information in 2^n input lines and produces n outputs. Encoder produces the binary equivalent of input lines. At any given time, only one input line can be high in encoder circuit. If more than one input is high at the same time, encoder circuit must establish a priority on inputs. An encoder circuit, that gives priority to inputs, is called as Priority Encoder. The higher priority will be given to higher input suffix. For input lines 0000 and 0001, output will be 00. To differentiate between both or to resolve the ambiguity on both type of inputs, a Valid (V) bit is considered as one of the outputs of priority encoder circuit. V bit is set to high when there is at least one 1 in the input. V bit is low when all inputs are zero. The truth table of 4:2 priority encoder us given below.

Shreya Sriram
195001106

Truth Table of 4:2 Priority Encoder :

X3	X2	X1	X0	Y1	Y0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

Shreya Sriram
195001106

K-map :

X3X2 \ X1X0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

Y1 = X3 + X2

	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

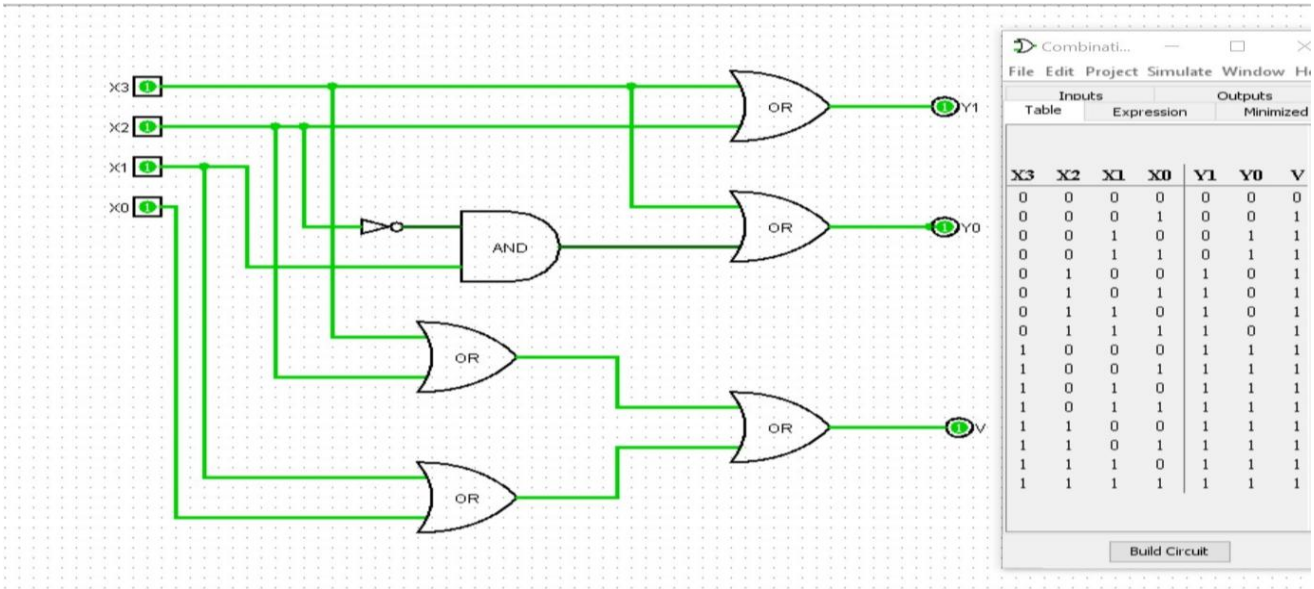
Y0 = X3 + X2'X1

Shreya Sriram
195001106

	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

V = X3 + X2 + X1 + X0

Logic Circuit :



Result :

4:2 Priority Encoder was designed and implemented successfully.

Name Shreya Sriram
Register No. 195001106

EX. NO.:17

DESIGN AND IMPLEMENTATION OF MOD 10 RIPPLE COUNTER

AIM:

To design and verify Mod 10 or BCD Ripple (Asynchronous) counter using

JK Flip Flop

APPARATUS REQUIRED:

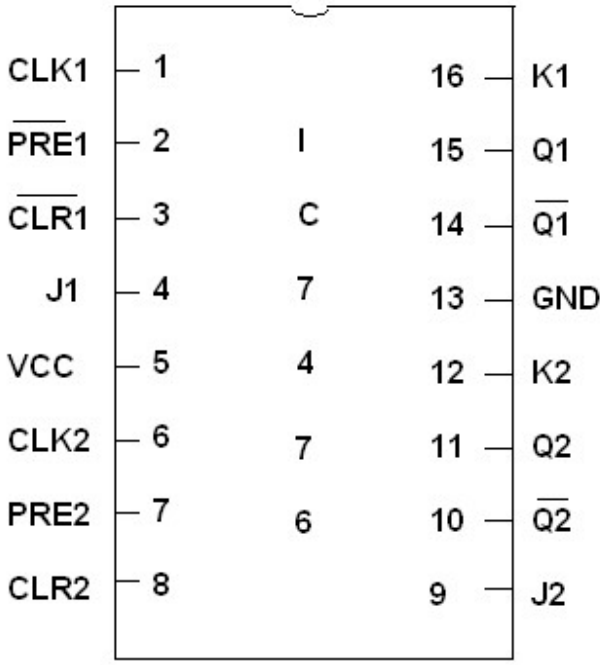
S.No.	COMPONENT	SPECIFICATION	QTY.
1.	JK FLIP FLOP	IC 7476	2
2.	NAND GATE	IC 7400	1
3.	NOT GATE	IC 7404	1
4.	IC TRAINER KIT	-	1
5.	PATCH CORDS	-	30

THEORY:

A counter is a register capable of counting number of clock pulse arriving at its clock input. Counter represents the number of clock pulses arrived. A specified sequence of states appears as counter output. This is the main difference between a register and a counter. There are two types of counter, synchronous and asynchronous. In synchronous common clock is given to all flip flop and in asynchronous first flip flop is clocked by external pulse and then each successive

flip flop is clocked by Q or Q output of previous stage. A soon the clock of second stage is triggered by output of first stage. Because of inherent propagation delay time all flip flops are not activated at same time which results in asynchronous operation.

PIN DIAGRAM FOR IC 7476:

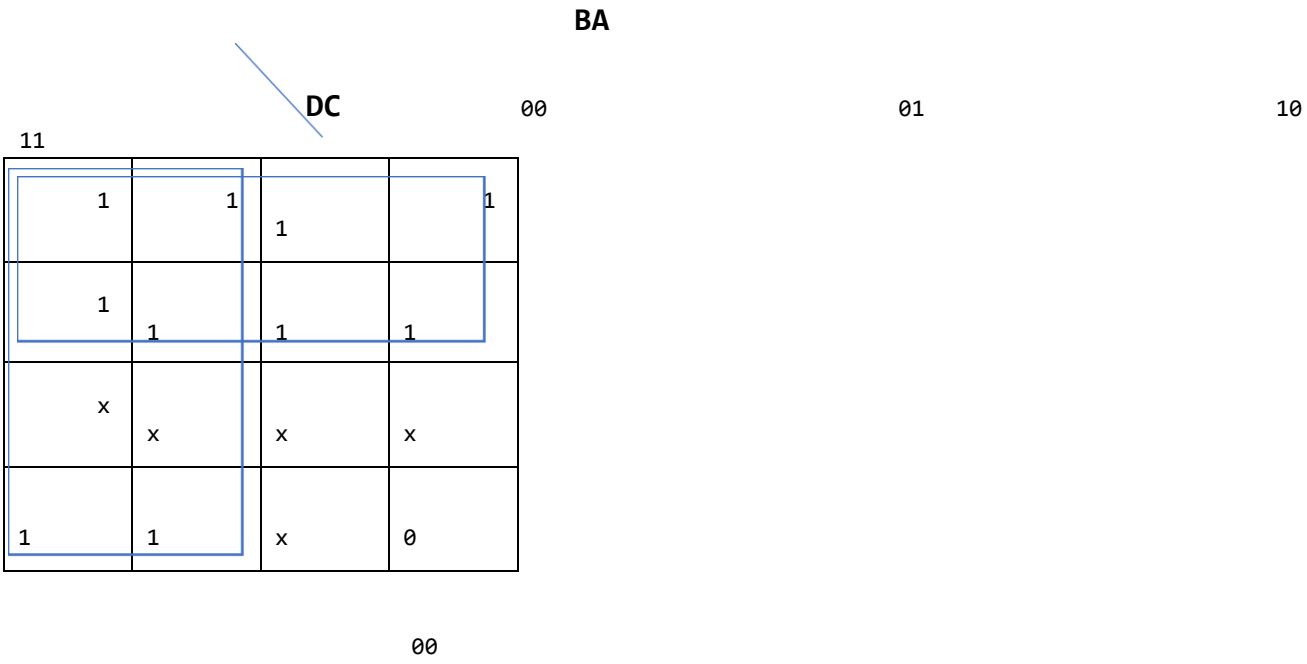


TRUTH TABLE FOR BCD RIPPLE COUNTER:

CLK Pulse	QD	QC	QB	QA	Y Output

Name Shreya Sriram
Register No. 195001106

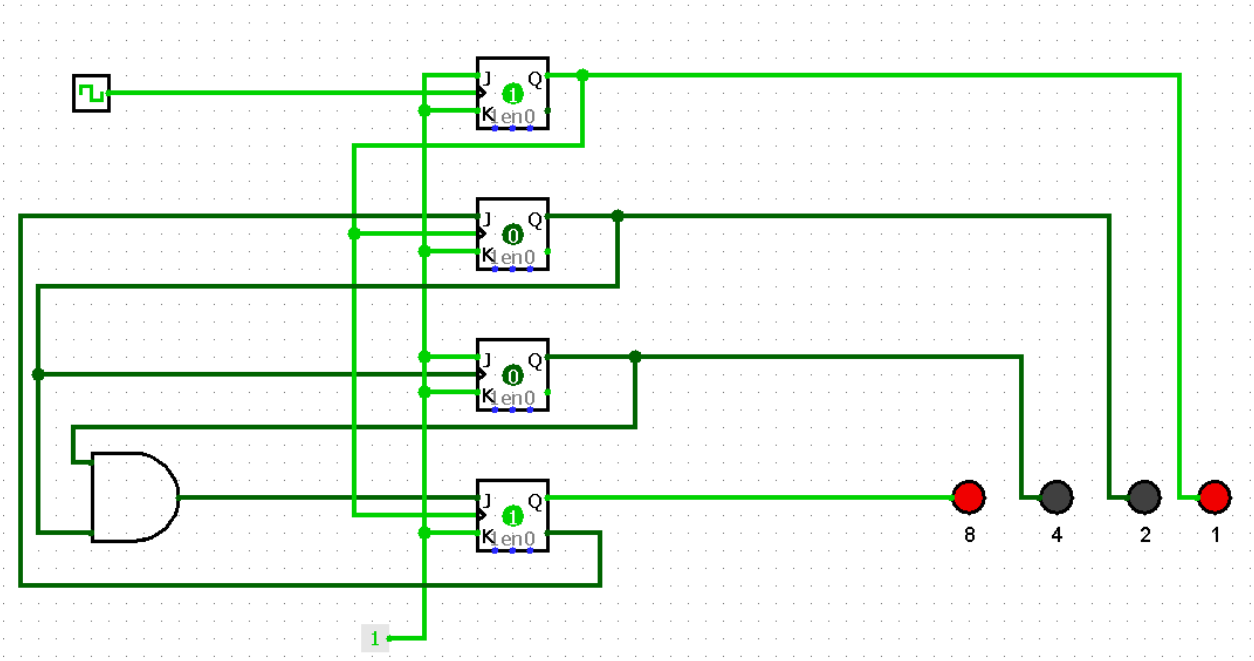
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	X
12	1	1	0	0	X
13	1	1	0	1	X
14	1	1	1	0	X
15	1	1	1	1	X



Name Shreya Sriram
Register No. 195001106

Equation of Y:

Y=D’+B’



Shreya S
195001106

EX. NO.:16

DESIGN AND IMPLEMENTATION OF 4 BIT UP/ DOWN RIPPLE COUNTER

AIM:

To design and verify 4 bit Up / Down Ripple (Asynchronous) counter using JK FlipFlop

Specifications:

1. Draw the logic diagram for 4 – Bit Ripple Up counter by giving output every JK flip flop as a clock input for next stage of JK flip flop.
2. Derive the truth table for 4 – Bit Ripple Up counter.
3. Draw the timing diagram for 4 - Bit Ripple Up counter.
4. Make connections using IC 7476 and verify the truth table for Ripple Up counter.

APPARATUS REQUIRED:

S.No.	COMPONENT	SPECIFICATION	QTY.
1.	JK FLIP FLOP	IC 7476	2
2.	NAND GATE	IC 7400	1
3.	NOT GATE	IC 7404	1
4.	IC TRAINER KIT	-	1
5.	PATCH CORDS	-	30

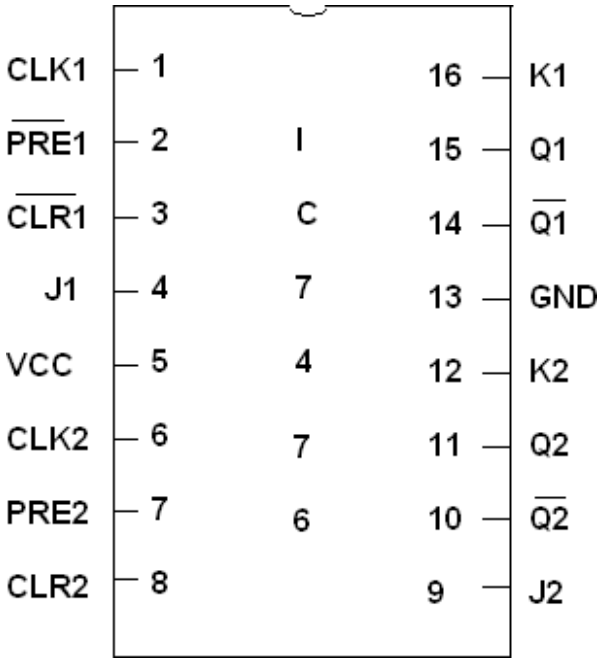
THEORY:

A counter is a register capable of counting number of clock pulse arriving at its clock input. Counter represents the number of clock pulses arrived. A specified sequence of states appears as counter output. This is the main difference between a register and a counter. There are two types of counter, synchronous and asynchronous. In synchronous common clock is given to all flip flop and in asynchronous first flip flop is clocked by external pulse and then each successive flip flop is clocked by Q or Q output of previous stage. A soon the clock of second stage is triggered by

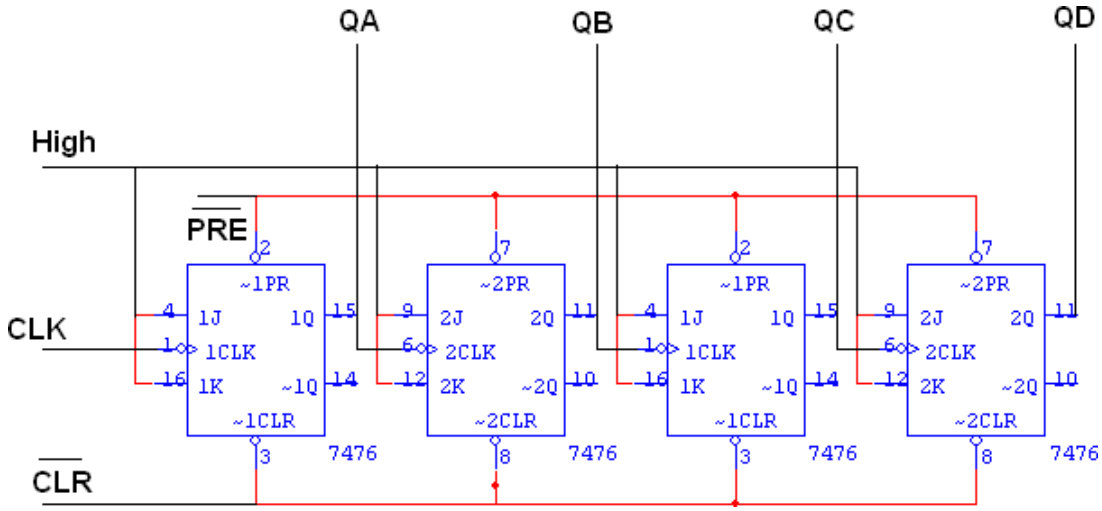
output of first stage. Because of inherent propagation delay time all flip flops are not activated at same time which results in asynchronous operation.

Shreya S
195001106

PIN DIAGRAM FOR IC 7476:



LOGIC DIAGRAM FOR 4 BIT RIPPLE UP COUNTER:



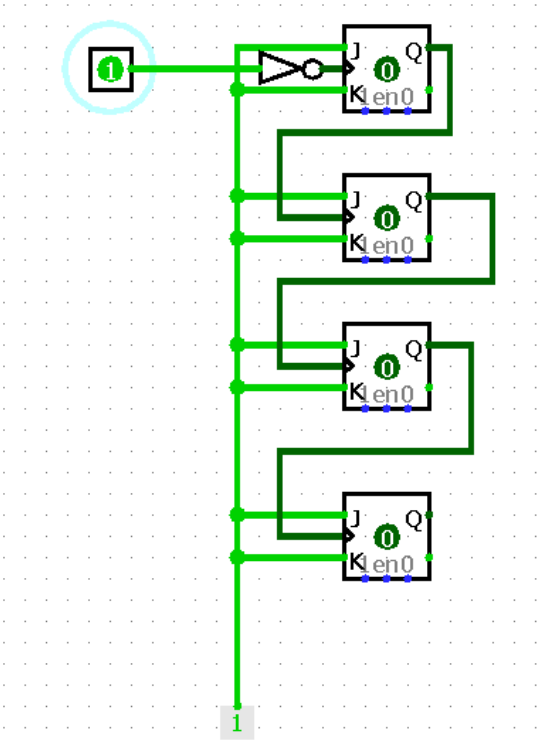
Shreya S
195001106

TRUTH TABLE FOR UP COUNTER:

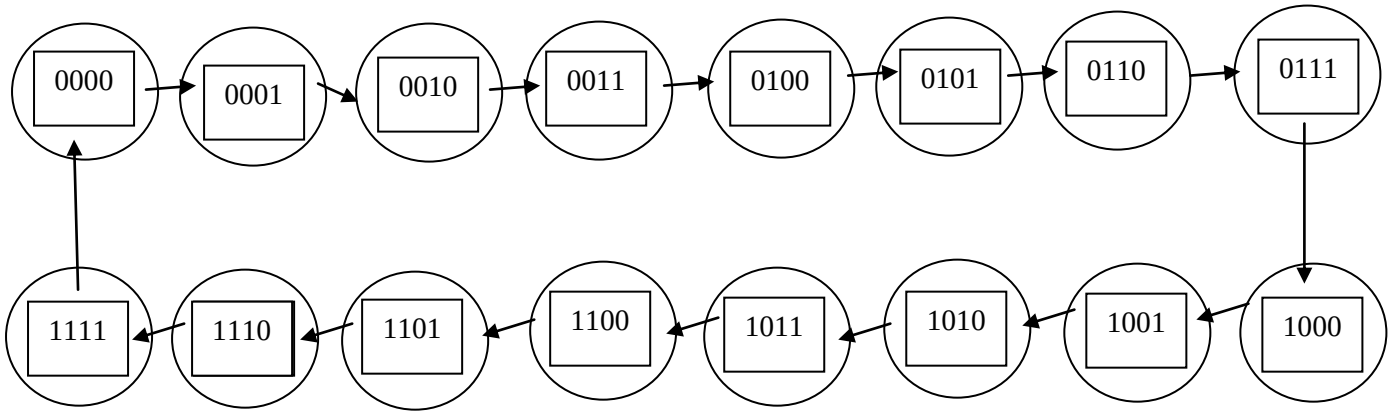
CLK Pulse	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Shreya S
195001106

Circuit Diagram:



State Diagram:



Result:
The circuit for ripple counter is designed and the state table and state diagram is recorded.

<Student Name> Shreya Sriram
<Register Number> 195001106

AIM:

To implement a 4 – bit Shift Register using D FlipFlop (IC7474).

APPARATUS REQUIRED:

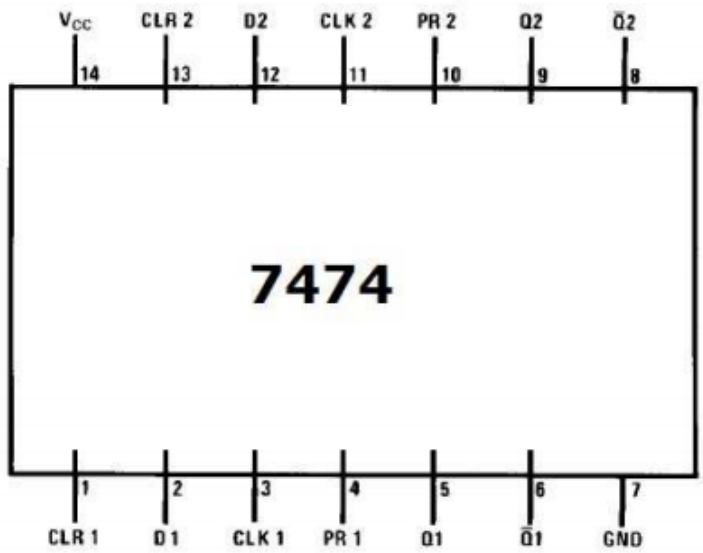
S.No.	COMPONENT	SPECIFICATION	QTY.
1	D FLIP FLOP	IC 7474	2
2	OR GATE	IC 7432	1
3	IC TRAINER KIT	-	1
4	PATCH CORDS	-	35

THEORY:

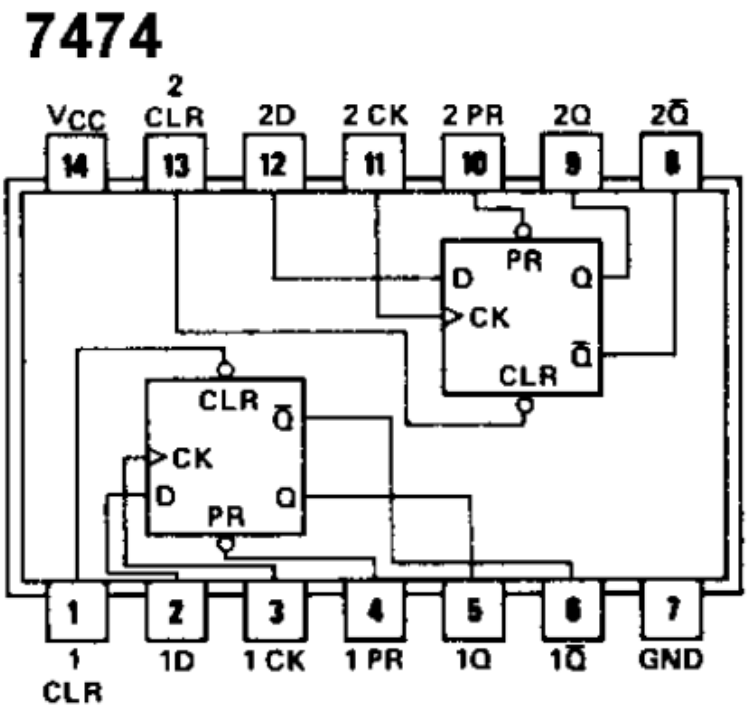
A register is capable of shifting its binary information in one or both directions is known as shift register. The logical configuration of shift register consist of a D-Flip flop cascaded with output of one flip flop connected to input of next flip flop. All flip flops receive common clock pulses which causes the shift in the output of the flip flop. The simplest possible shift register is one that uses only flip flop. The output of a given flip flop is connected to the input of next flip flop of the register. Each clock pulse shifts the content of register one bit position to right.

<Student Name> Shreya Sriram
<Register Number> 195001106

PIN DIAGRAM IC 7474



IC 7474has internally 2 - JK Flip Flops as shown below



Specifications:

<Student Name> Shreya Sriram
<Register Number> 195001106

1. Construct a 4 Bit Shift Register, by connecting (two IC 7474) serially, which involves 4 - D flipflops

2. Implement the following types of Shift Register using IC 7474.
1. **Serial In Serial Out**
2. **Serial In Parallel Out**
3. **Parallel In Serial Out**
4. **Parallel In Parallel Out**

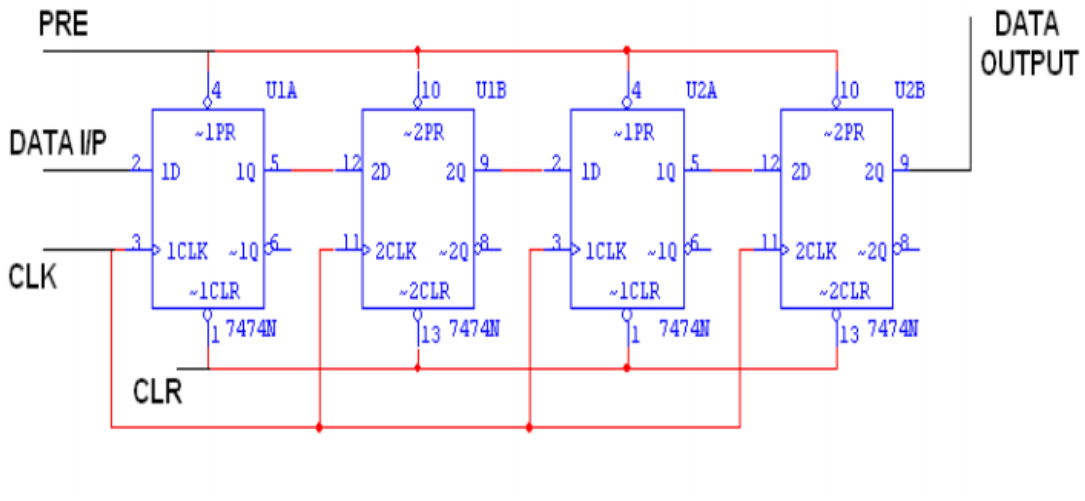
Preset and clear are active low signals

Preset	Clear	Operation
0	1	1 (Set to 1)
1	0	(Clear FF)
1	1	Normal Operation of FF
0	0	Undefined one (should not occur)

<Student Name> Shreya Sriram
<Register Number> 195001106

LOGIC DIAGRAM:

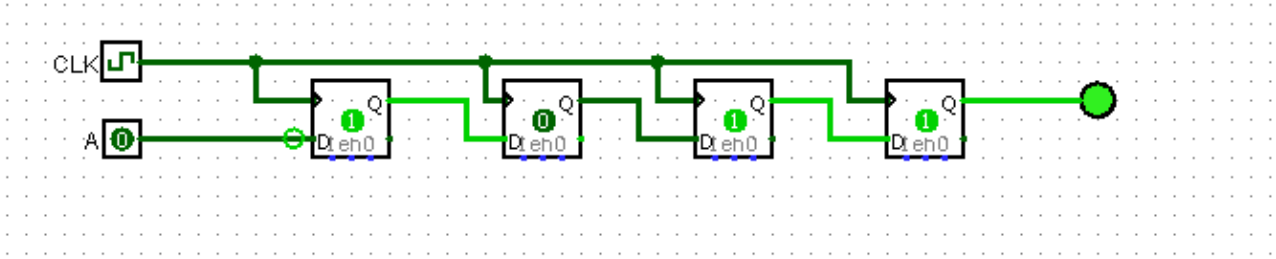
SERIAL IN SERIAL OUT:



TRUTH TABLE:

CLK	Serial in	Serial out
1	1	0
2	0	0
3	0	0
4	1	1
5	X	0
6	X	0
7	X	1

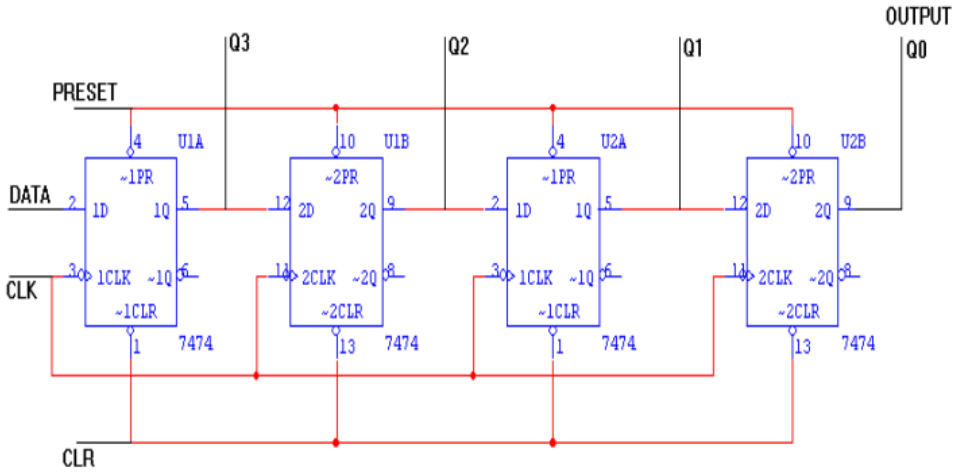
IMPLEMENTATION:



<Student Name> Shreya Sriram
<Register Number> 195001106

LOGIC DIAGRAM:

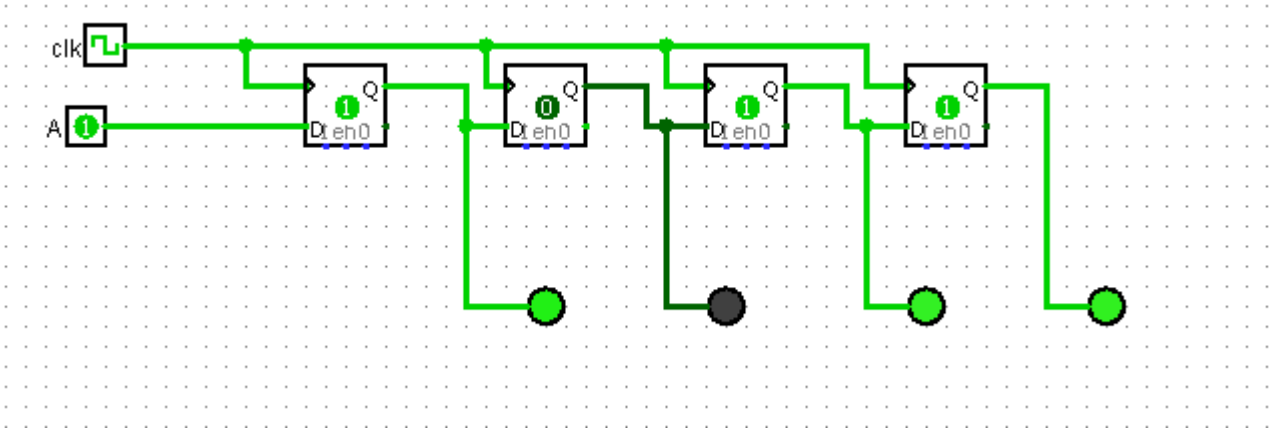
SERIAL IN PARALLEL OUT:



TRUTH TABLE:

CLK	DATA	OUTPUT			
		Q_A	Q_B	Q_C	Q_D
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	1	1	0	0	1

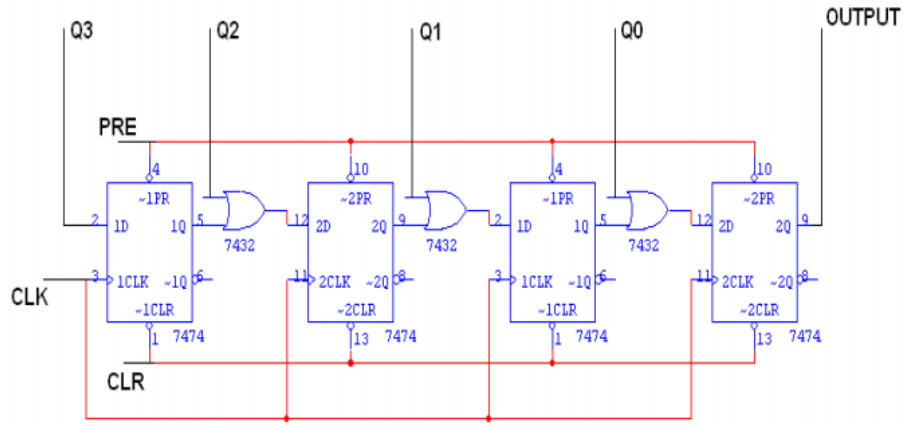
IMPLEMENTATION:



<Student Name> Shreya Sriram
<Register Number> 195001106

LOGIC DIAGRAM:

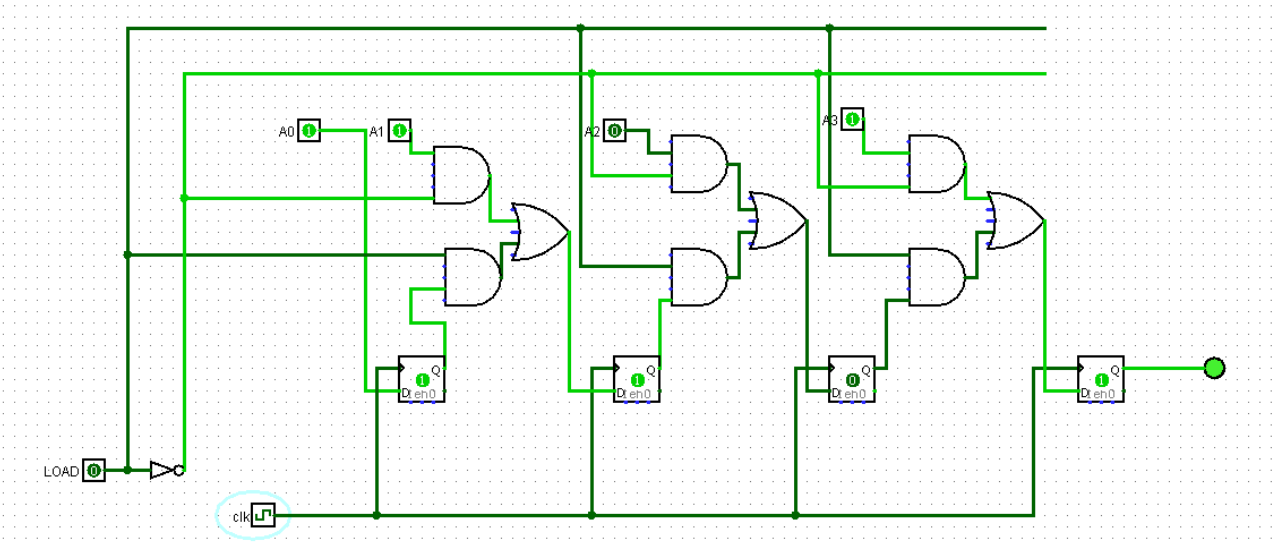
PARALLEL IN SERIAL OUT:



TRUTH TABLE:

CLK	Q3	Q2	Q1	Q0	O/P
0	1	0	0	1	1
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	1

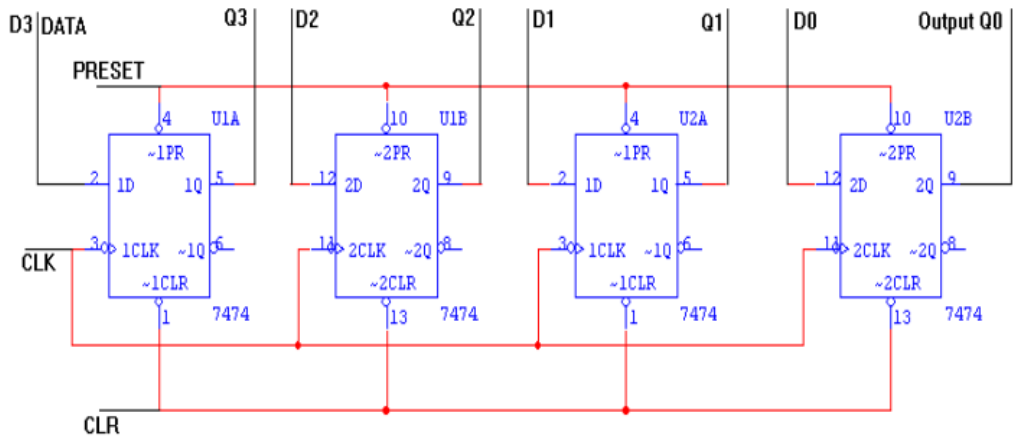
IMPLEMENTATION:



<Student Name> Shreya Sriram
<Register Number> 195001106

LOGIC DIAGRAM:

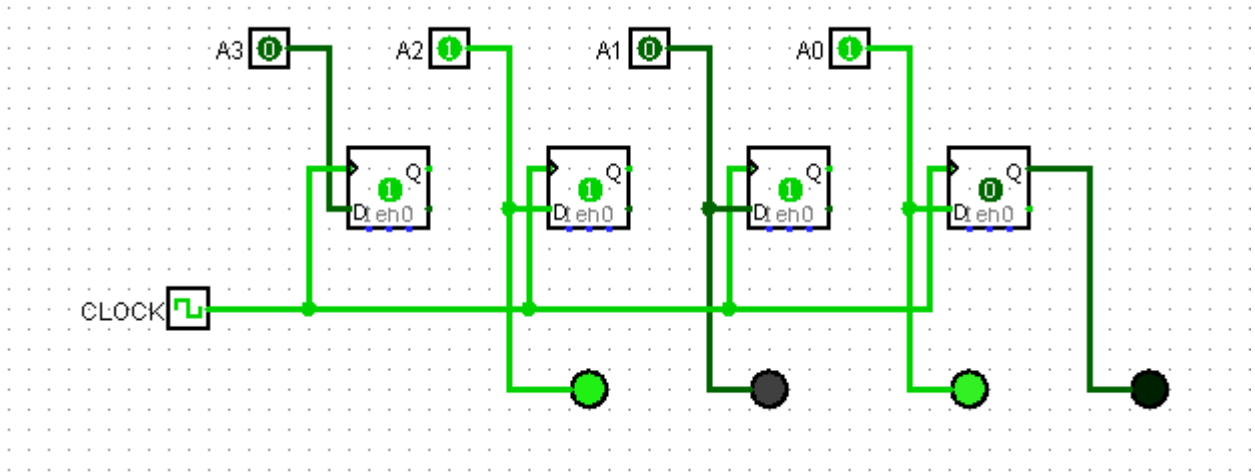
PARALLEL IN PARALLEL OUT:



TRUTH TABLE:

CLK	DATA INPUT				OUTPUT			
	D_A	D_B	D_C	D_D	Q_A	Q_B	Q_C	Q_D
1	1	0	0	1	1	0	0	1
2	1	0	1	0	1	0	1	0

IMPLEMENTATION:



<Student Name> Shreya Sriram
<Register Number> 195001106

Simplification of Boolean Expression using Karnaugh Map

Objective

To simplify the Boolean expression using Karnaugh map and to verify the truth table.

F(A,B,C,D)=π(0,2,3,8,9,12,13,15)

TRUTH TABLE:

	A	B	C	D	F
m0	0	0	0	0	0
m1	0	0	0	1	1
m2	0	0	1	0	0
m3	0	0	1	1	0
m4	0	1	0	0	1
m5	0	1	0	1	1
m6	0	1	1	0	1
m7	0	1	1	1	1
m8	1	0	0	0	0
m9	1	0	0	1	0
m10	1	0	1	0	1
m11	1	0	1	1	1
m12	1	1	0	0	0
m13	1	1	0	1	0
m14	1	1	1	0	1
m15	1	1	1	1	0

<Student Name> Shreya Sriram
<Register Number> 195001106

Minimize the given Boolean expression using Karnaugh map.

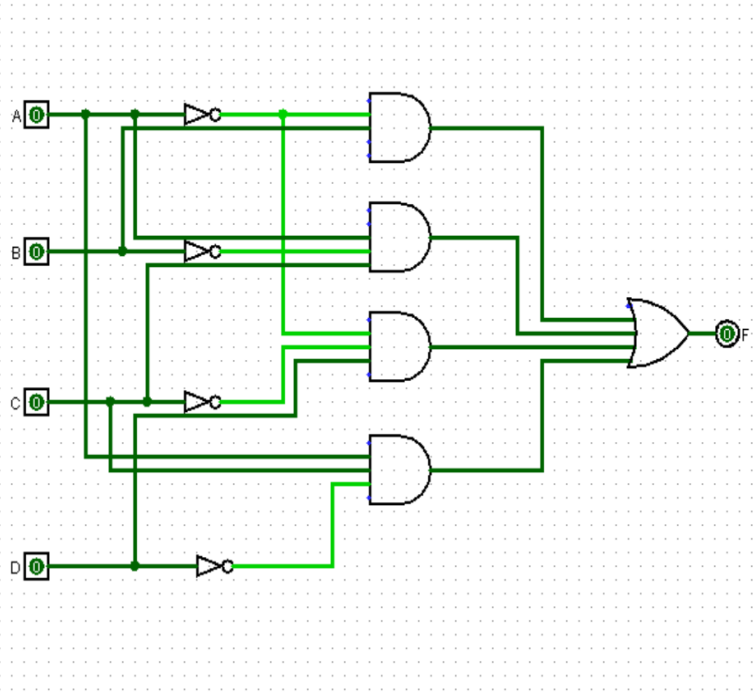
SOP FORM

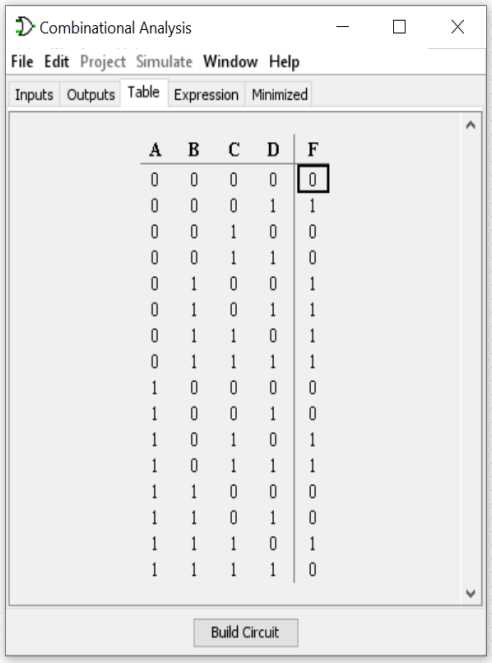
	C'D'	C'D	CD	CD'
A'B'	0	1	0	0
A'B	1	1	1	1
AB	0	0	0	1
AB'	0	0	1	1

ON GROUPING 1's IN THE KMAP

F = A'B + ACD' + AB'C +A'C'D

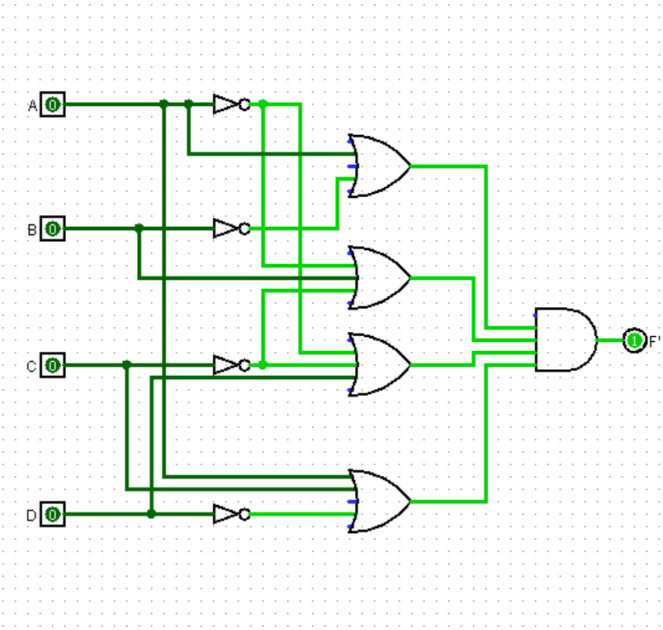
THIS IS THE REQUIRED SOP FORM

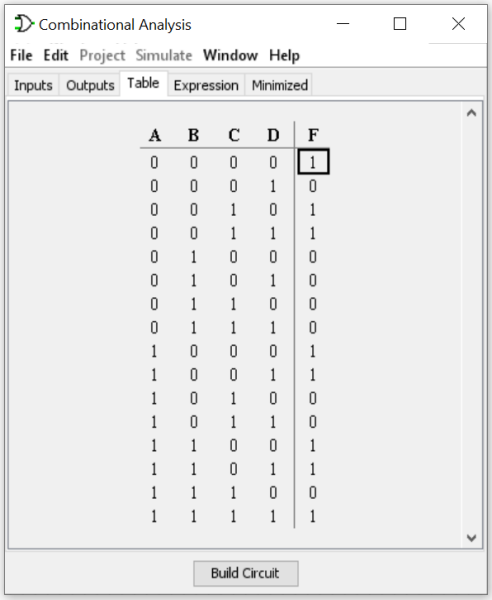




<Student Name> Shreya Sriram
<Register Number> 195001106

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





POS FORM

	C'D'	C'D	CD	CD'
A'B'	0	1	0	0
A'B	1	1	1	1
AB	0	0	0	1
AB'	0	0	1	1

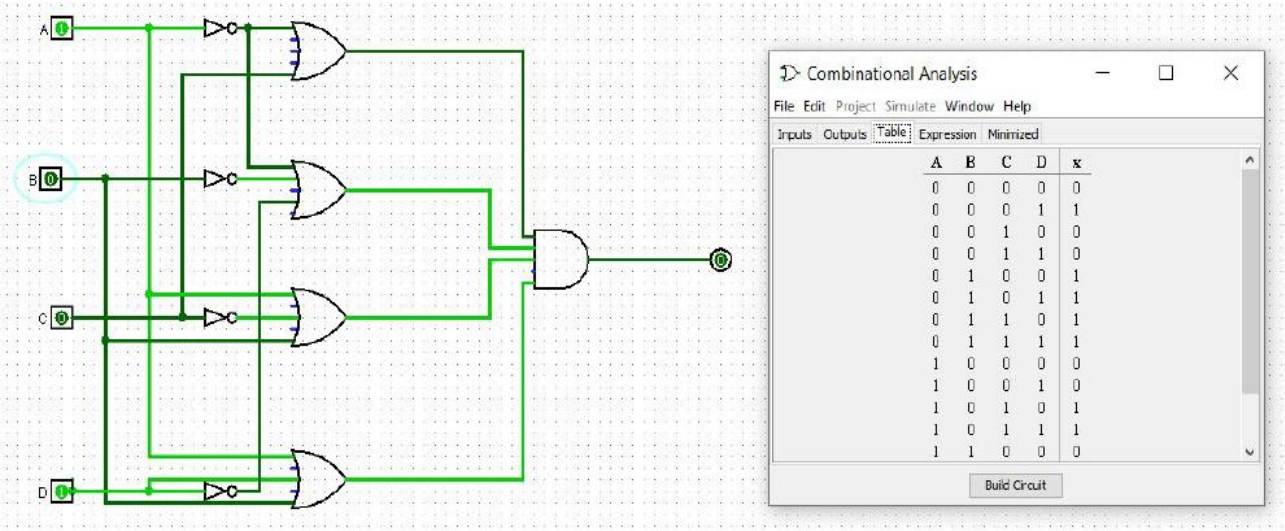
ON GROUPING 0's IN THE KMAP

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

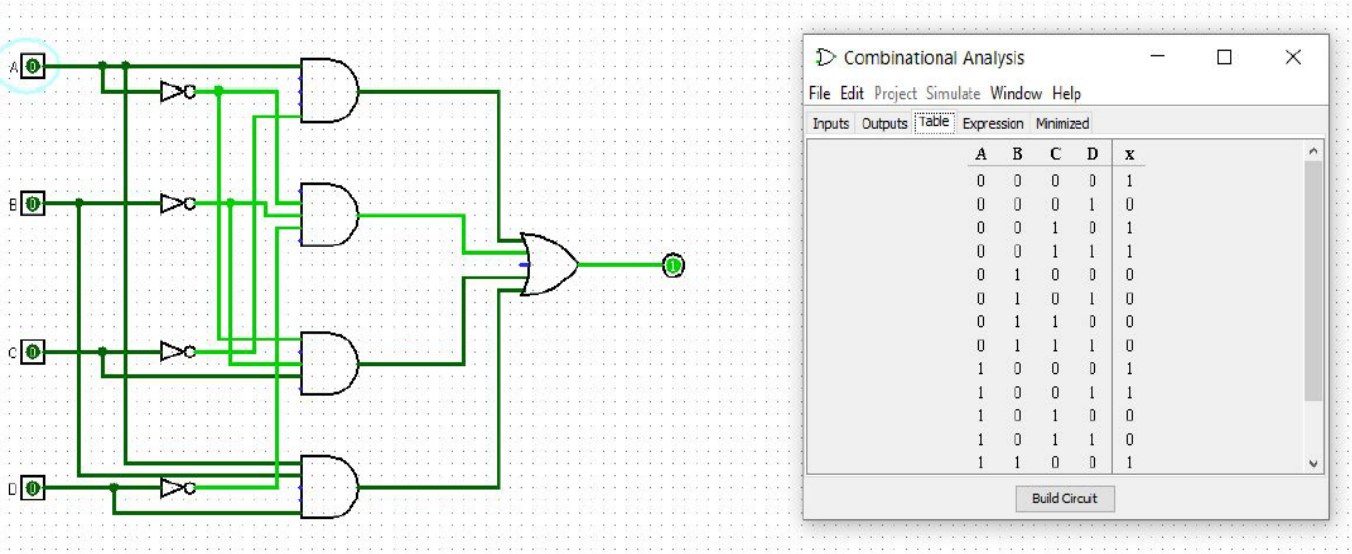
THIS IS THE REQUIRED POS FORM

<Student Name> Shreya Sriram
<Register Number> 195001106

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



(F)' = AC' + ABD + A'B'C + A'B'D'



RESULT:

The given Boolean expression is simplified using KMap and the truth tables are verified

<Student Name> Shreya S
<Register Number> 195001106

Design And Implementation Of 3 Bit Synchronous Counter Using Jk Flip Flop

Objective

To design and implement a 3 bit synchronous counter using JK Flip Flop.

Specifications

1. Draw the state diagram for 3 bit - Synchronous UP/ Down counter
2. Derive the state table for 3 bit synchronous UP/ Down counter.
3. Obtain simplified Boolean function for input equations of JK flip flop
4. Draw the synchronous sequential circuit with JK flip flop for a 3 bit - Synchronous UP/ Down counter
5. Verify using LogiSIM software

Components Required

S.No	Component Name	IC Number	Quantity
1	1. JK FLIP FLOP IC 7476 2	7408	1
2	OR GATE IC 7432 1	7432	1
3	Connecting Wires	-	-
4	LogiSIM Software	-	-
5	3 I/P AND GATE IC 7411 1		
6	XOR GATE IC 7486 1		
7	NOT GATE IC 7404 1		
8	IC TRAINER KIT - 1		
9	PATCH CORDS - 35		

Theory:

A counter is a register capable of counting number of clock pulse arriving at its clock input.
Counter represents the number of clock pulses arrived. A specified sequence of states appears as counter output. This is the main difference between a register and a counter. There are two types of counter, synchronous and asynchronous. In synchronous common clock is given to all flip flop and in asynchronous first flip flop is clocked by external pulse and then each successive flip flop is clocked by Q or Q output of previous stage. A soon the clock of second stage is triggered by output of first stage. Because of inherent propagation delay time all flip flops are not activated at same time which results in asynchronous operation.

<Student Name> Shreya S
<Register Number> 195001106

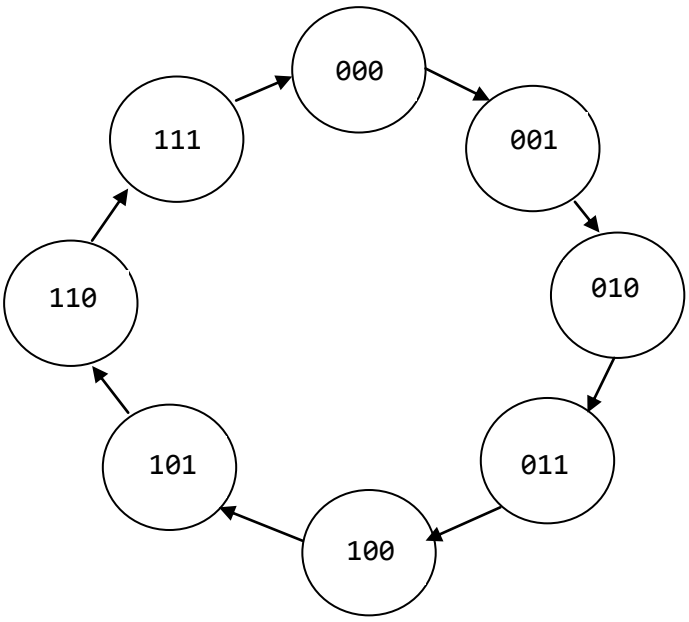
The counter can be either UP / DOWN counter. The 4 bit UP counter starts from 0000 and increments by 1 for every clock input reaches 1111 and repeats to 0000. The DOWN counter starts from 1111 and decrements by 1 for every clock input, reaches 0000 and repeats to 111

Excitation table for JK flip flop

Q	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

UP COUNTER

State Diagram



<Student Name> Shreya S
<Register Number> 195001106

State table

PS			NS			Jc	Kc	Jb	Kb	Ja	Ka
Qc	Qb	Qa	Qc+	Qb+	Qa+						
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

Simplification using KMAP

QbQa

Qc	00	01	11	10
0	0	0	1	0
1	X	X	X	X

Jc = QbQa

QbQa

Qc	00	01	11	10
0	X	X	X	X
1	0	0	1	0

Kc = QbQa

QbQa

Qc	00	01	11	10
0	0	1	X	X
1	0	1	X	X

Jb = Qa

<Student Name> Shreya S
<Register Number> 195001106

QbQa

Qc	00	01	11	10
0	X	X	1	0
1	X	X	1	0

Kb = Qa

QbQa

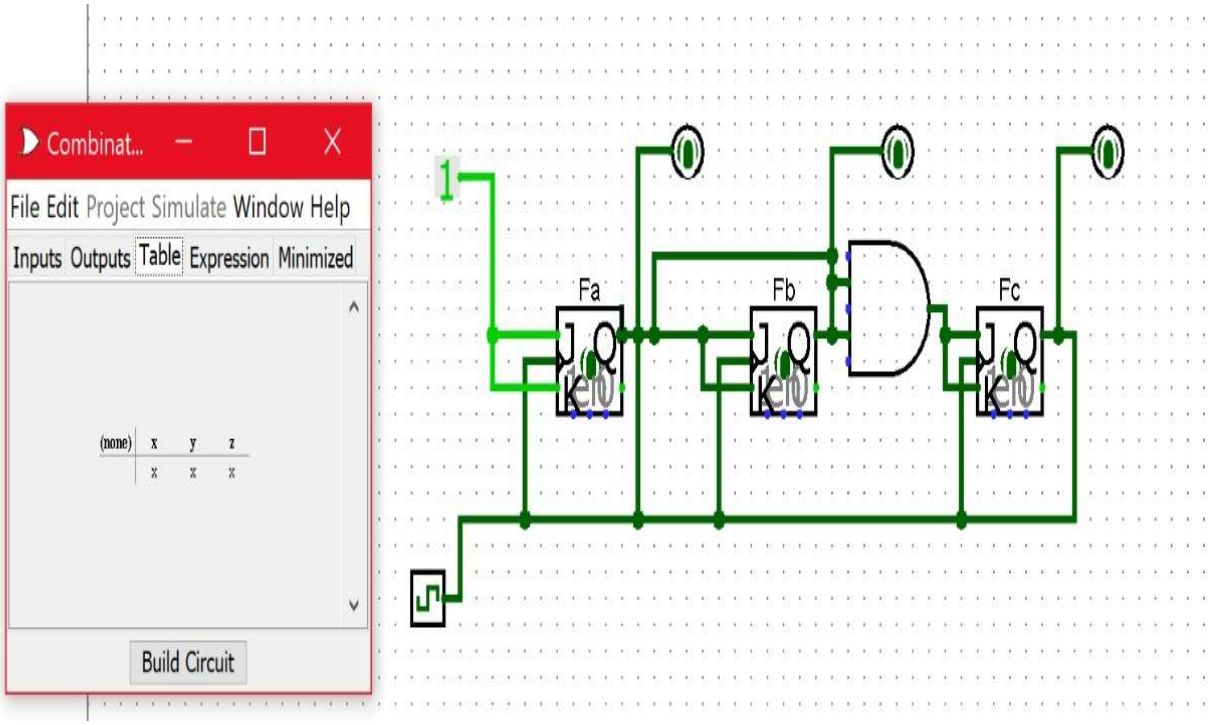
Qc	00	01	11	10
0	1	X	X	1
1	1	X	X	1

Ja = 1

QbQa

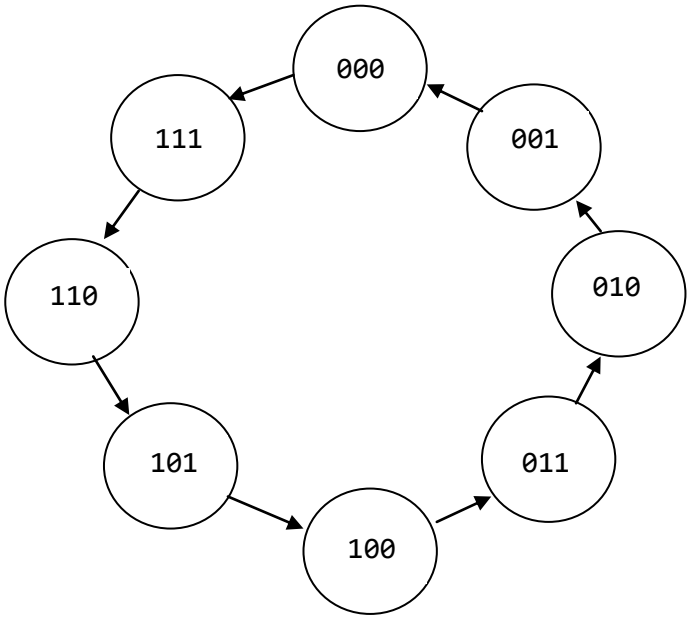
Qc	00	01	11	10
0	X	1	1	X
1	X	1	1	X

Ka = 1



<Student Name> Shreya S
<Register Number> 195001106

DOWN COUNTER



State table

PS			NS			Jc	Kc	Jb	Kb	Ja	Ka
Qc	Qb	Qa	Qc+	Qb+	Qa						
0	0	0	1	1	1	1	X	1	X	1	X
0	0	1	0	0	0	0	X	0	X	X	1
0	1	0	0	0	1	0	X	X	1	1	X
0	1	1	0	1	0	0	X	X	0	X	1
1	0	0	0	1	1	X	1	1	X	1	X
1	0	1	1	0	0	X	0	0	X	X	1
1	1	0	1	0	1	X	0	X	1	1	X
1	1	1	1	1	0	X	0	X	0	X	1

Kmap Simplification

QbQa

Qc	00	01	11	10
0	1	0	0	0
1	X	X	X	X

Jc = Qb'Qa'

<Student Name> Shreya S
<Register Number> 195001106

QbQa

Qc	00	01	11	10
0	X	X	X	X
1	1	0	0	0

Kc = Qb'Qa'

QbQa

Qc	00	01	11	10
0	1	0	X	X
1	1	0	X	X

Jb = Qa'

QbQa

Qc	00	01	11	10
0	X	X	0	1
1	X	X	0	1

Kb = Qa'

QbQa

Qc	00	01	11	10
0	1	X	X	1
1	1	X	X	1

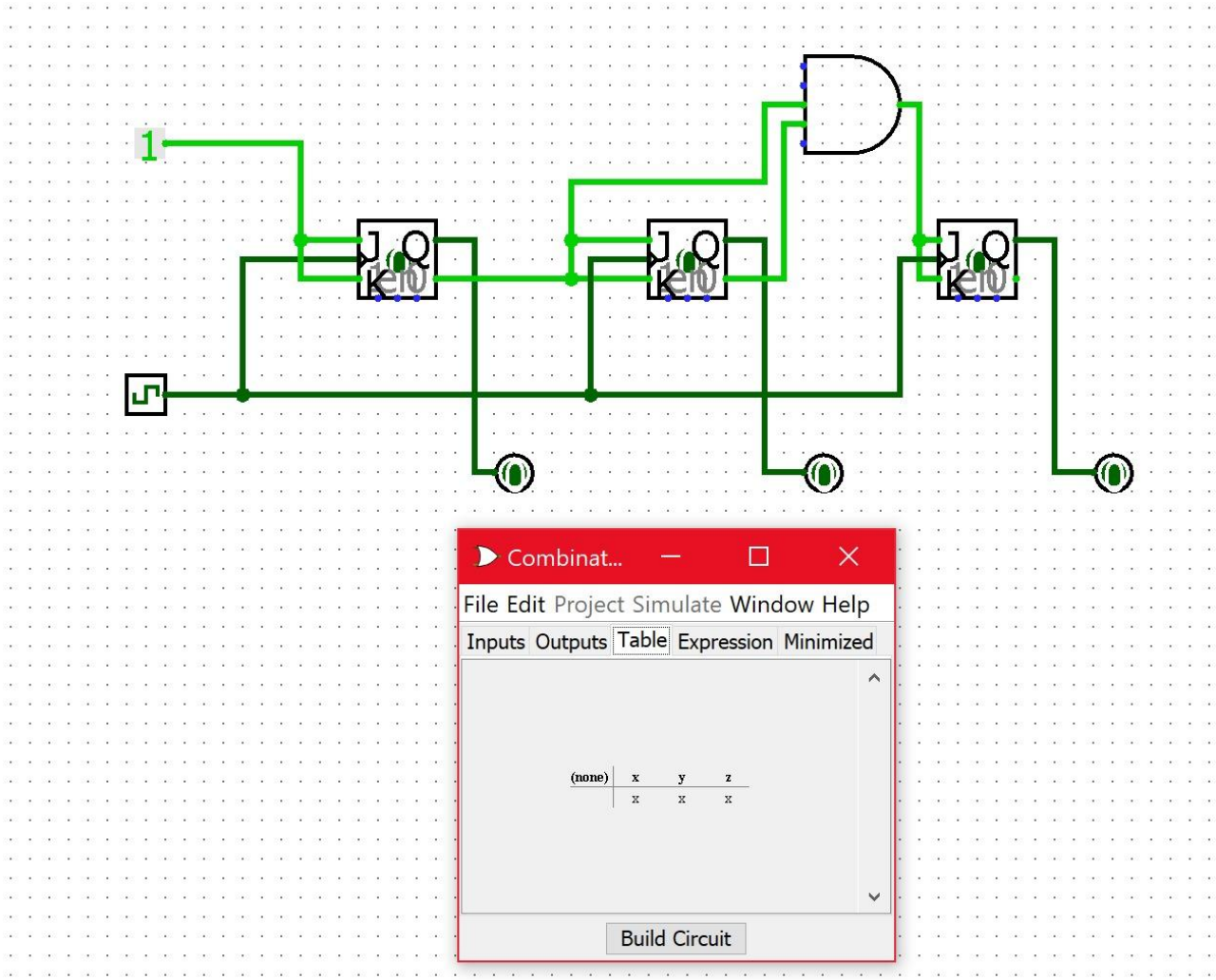
Ja = 1

QbQa

Qc	00	01	11	10
0	X	1	1	X
1	X	1	1	X

Ka = 1

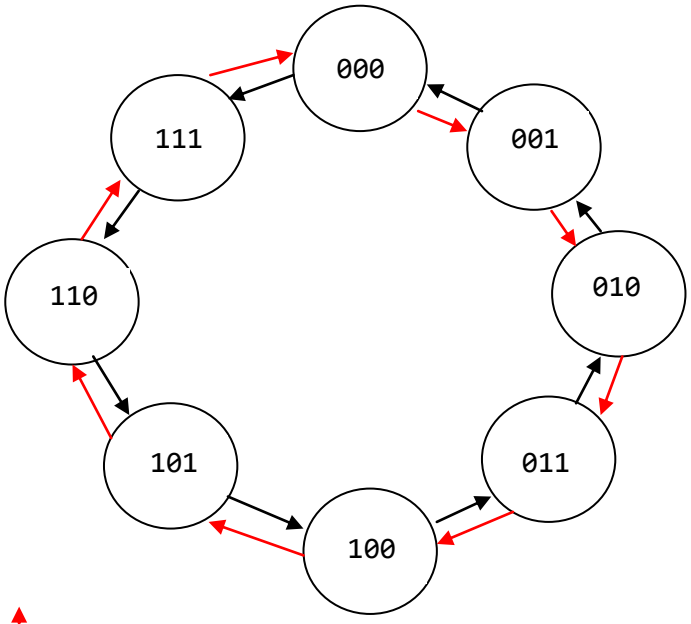
<Student Name> Shreya S
<Register Number> 195001106



<Student Name> Shreya S
<Register Number> 195001106

UP DOWN COUNTER

State Diagram



Down Up

<Student Name> Shreya S
<Register Number> 195001106

State table

M	PS			NS			Jc	Kc	Jb	Kb	Ja	Ka
	Qc	Qb	Qa	Qc+	Qb+	Qa						
0	0	0	0	0	0	1	0	X	0	X	1	X
0	0	0	1	0	1	0	0	X	1	X	X	1
0	0	1	0	0	1	1	0	X	X	0	1	X
0	0	1	1	1	0	0	1	X	X	1	X	1
0	1	0	0	1	0	1	X	0	0	X	1	X
0	1	0	1	1	1	0	X	0	1	X	X	1
0	1	1	0	1	1	1	X	0	X	0	1	X
0	1	1	1	0	0	0	X	1	X	1	X	1
1	0	0	0	1	1	1	1	X	1	X	1	X
1	0	0	1	0	0	0	0	X	0	X	X	1
1	0	1	0	0	0	1	0	X	X	1	1	X
1	0	1	1	0	1	0	0	X	X	0	X	1
1	1	0	0	0	1	1	X	1	1	X	1	X
1	1	0	1	1	0	0	X	0	0	X	X	1
1	1	1	0	1	0	1	X	0	X	1	1	X
1	1	1	1	1	1	0	X	0	X	0	X	1

Kmap Simplification

For all kmaps QbQa columns and mQc are rows

	00	01	11	10
00	0	0	1	0
01	x	x	x	x
11	x	x	x	x
10	1	0	0	0

Jc = Qb’Qa’m + m’QbQa

<Student Name> Shreya S
<Register Number> 195001106

	00	01	11	10
00	x	x	x	X
01	0	0	1	0
11	1	0	0	0
10	x	x	x	x

Kc = Qb'Qa'm + m'QbQa

	00	01	11	10
00	0	1	x	x
01	0	1	x	x
11	1	0	x	x
10	1	0	x	x

Jb = mQa' + m'Qa

	00	01	11	10
00	x	x	1	0
01	x	x	1	0
11	x	x	0	1
10	x	x	0	1

Kb = mQa' + m'Qa

<Student Name> Shreya S
<Register Number> 195001106

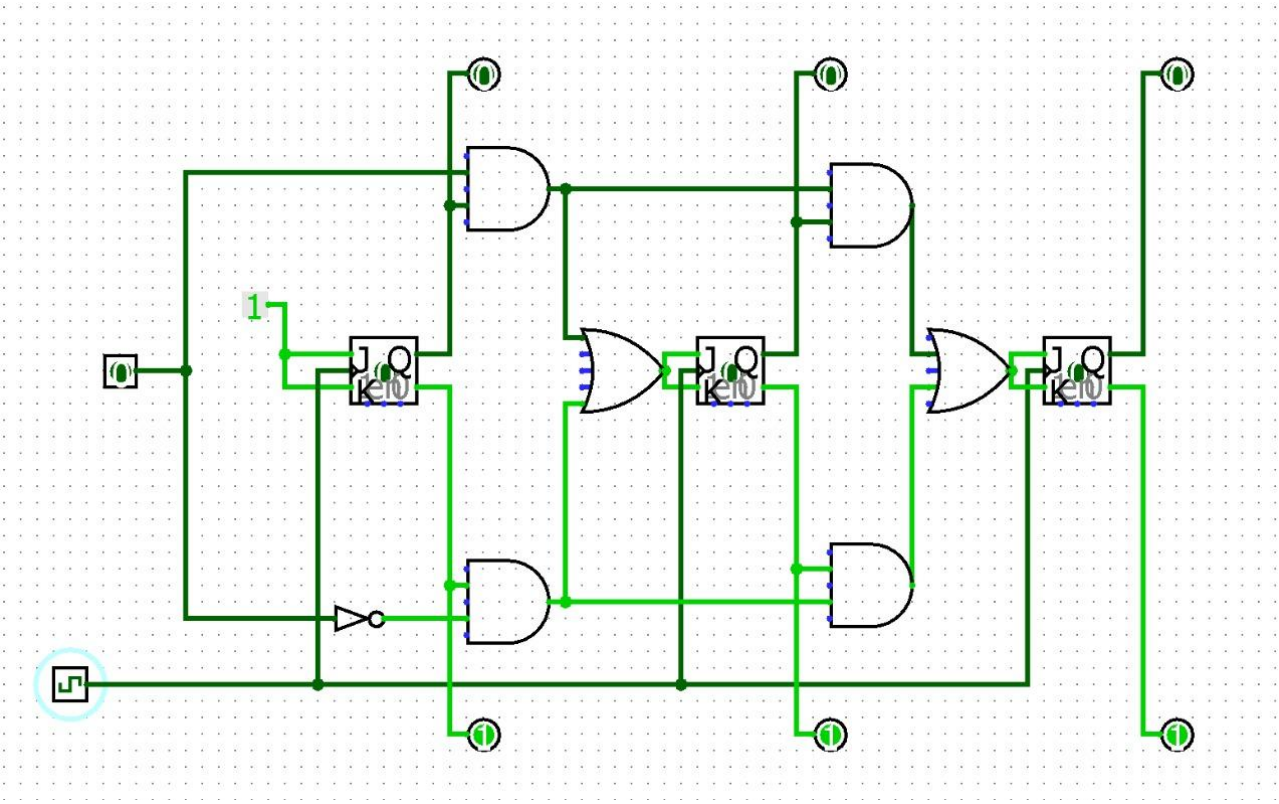
	00	01	11	10
00	1	x	x	1
01	1	x	x	1
11	1	x	x	1
10	1	x	x	1

Ja = 1

	00	01	11	10
00	x	1	1	x
01	x	1	1	x
11	x	1	1	x
10	x	1	1	x

Ka = 1

<Student Name> Shreya S
<Register Number> 195001106



Result:

The implementation has been done for 3bit synchronous counters

<Student Name> Shreya Sriram
<Register Number>195001106

Realization of Boolean Expression using Basic and Universal gates

Aim: To realize the simplified Boolean expression using the basic and Universal gates (NAND and NOR) for the below given Truth Table.

F(A,B,C,D)=π(0,2,3,8,9,12,13,15)

TRUTH TABLE:

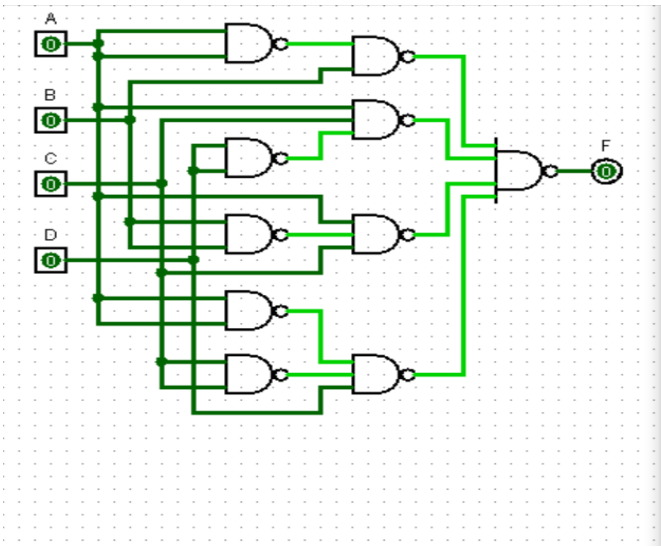
A	B	C	D	F	POS	SOP
0	0	0	0	0		A+B+C+D
0	0	0	1	1	A'B'C'D	
0	0	1	0	0		A+B+C'+D
0	0	1	1	0		A+B+C'+D'
0	1	0	0	1	A'BC'D'	
0	1	0	1	1	A'BC'D	
0	1	1	0	1	A'BCD'	
0	1	1	1	1	A'BCD	
1	0	0	0	0		A'+B+C+D
1	0	0	1	0		A'+B+C+D'
1	0	1	0	1	AB'CD'	
1	0	1	1	1	AB'CD	
1	1	0	0	0		A'+B'+C+D
1	1	0	1	0		A'+B'+C+D'
1	1	1	0	1	ABCD'	
1	1	1	1	0		A'+B'+C'+D'

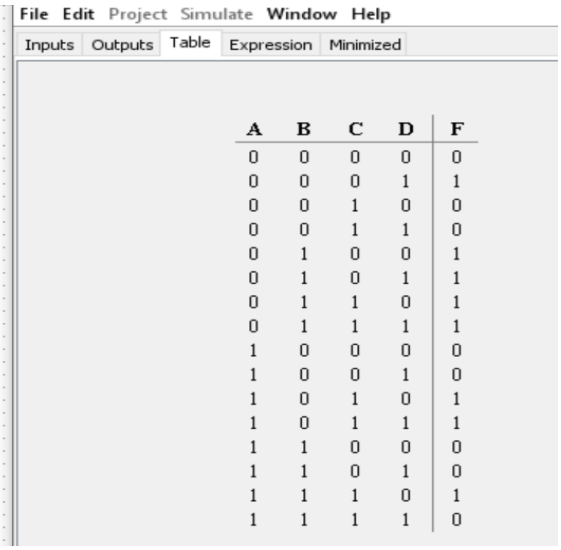
USING SOP FORM **F = A'B + ACD' + AB'C +A'C'D**

USING POS FORM **F = (A'+C).(A'+B'+D').(A+B+C').(A+B+D)**

<Student Name> Shreya Sriram
<Register Number>195001106

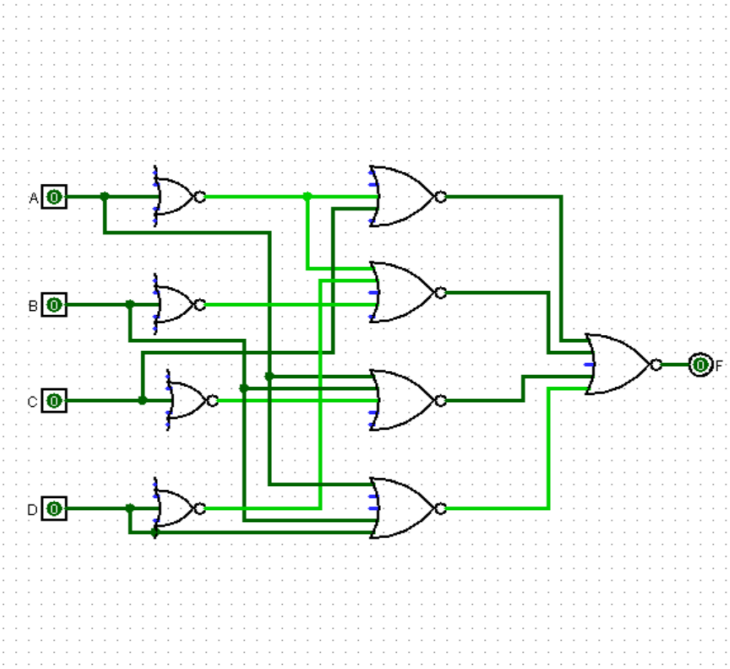
SOP USING NAND GATES:

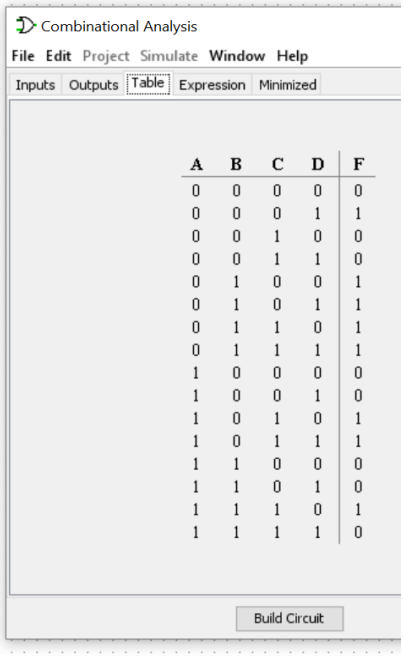




A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

SOP USING NOR GATE





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

Results

The Boolean expressions are simplified and realised using NAND and NOR gates.