NATIONAL UNIVERSITY OF MODERN LANGUAGES

“JANITA JIA ZAI”

BS COMPUTER SCIENCE III

Digital Logic Design

Practical 1

Universal Gates and Its Uses:

A gate is a fundamental component of a digital circuit that carries out a particular logical operation. A gate that may be used to make any other kind of gate is known as a universal gate. **NAND** gates and **NOR** gates are the two categories of universal gates.

NAND (Not-AND) Gates: Only when all of its inputs are low does a NAND gate produce a high output (1). (0). Any input that is high (1) will result in a low output (0). Because it may be used to create any digital circuit, the NAND gate is universal. Any other form of gate, such as AND, OR, NOT, and XOR gates, can be built using a combination of NAND gates.

NOR (Not-OR) Gates: Only when all of its inputs are low does a NOR gate produce a high output (1). (0). Any input that is high (1) will result in a low output (0). Because it may be used to create any digital circuit, the NOR gate is also universal. Any other gate type, such as AND, OR, NOT, and XOR gates, can be produced by combining NOR gates.

Any other form of gate can be built using either NOR or NAND gates. However, because they require fewer transistors than NOR gates and are more effective and less expensive to produce, NAND gates are more frequently utilised in contemporary digital circuits.

Truth Table AND Gate and OR Gate.

AND Gate

Input 1	Input 2	Input 3	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

OR Gate

Input 1	Input 2	Input 3	Output
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Truth Table NOR Gate and NAND Gate.

NOR Gate

A	B	C	Output
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

NAND Gate

A	B	C	Output
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Truth Table XOR Gate and XNOR Gate.

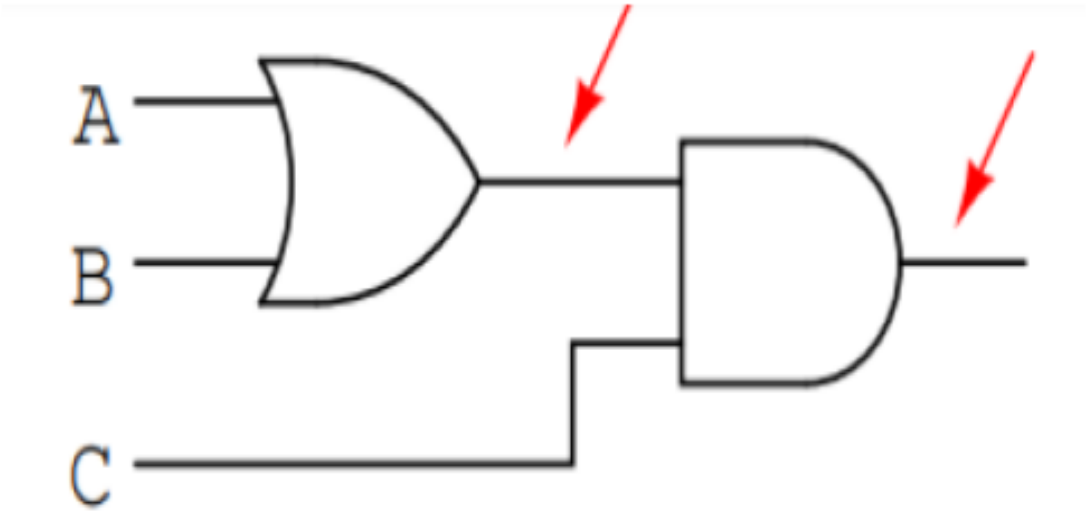
XOR Gates

A	B	C	Output
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

XNOR Gate

A	B	C	Output
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

Convert the following logic gate circuit into a Boolean expression, writing Boolean sub-expressions next to each gate output in the diagram:



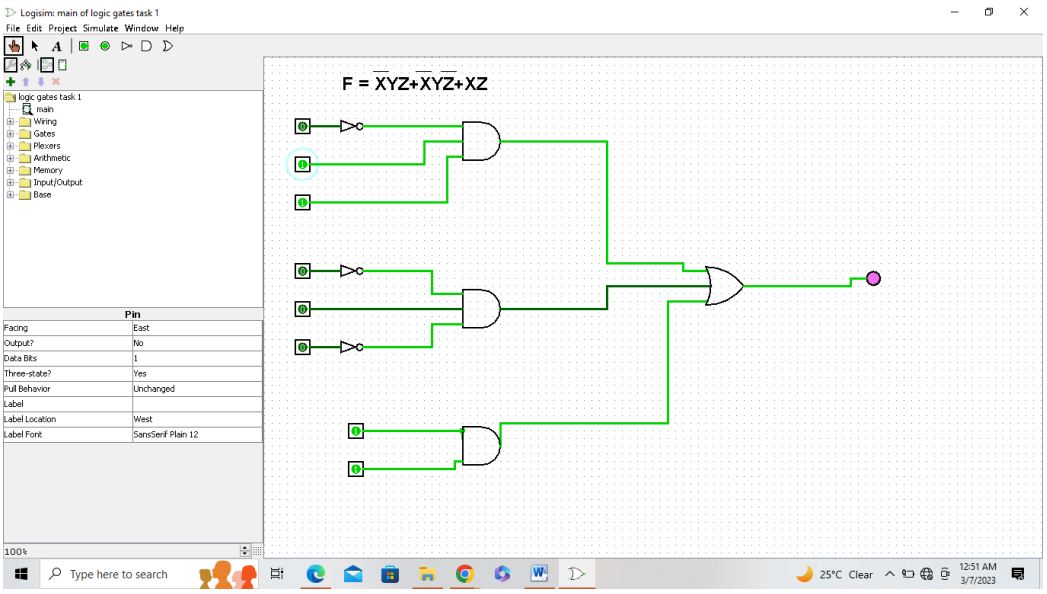
Answer

(A+B)+C

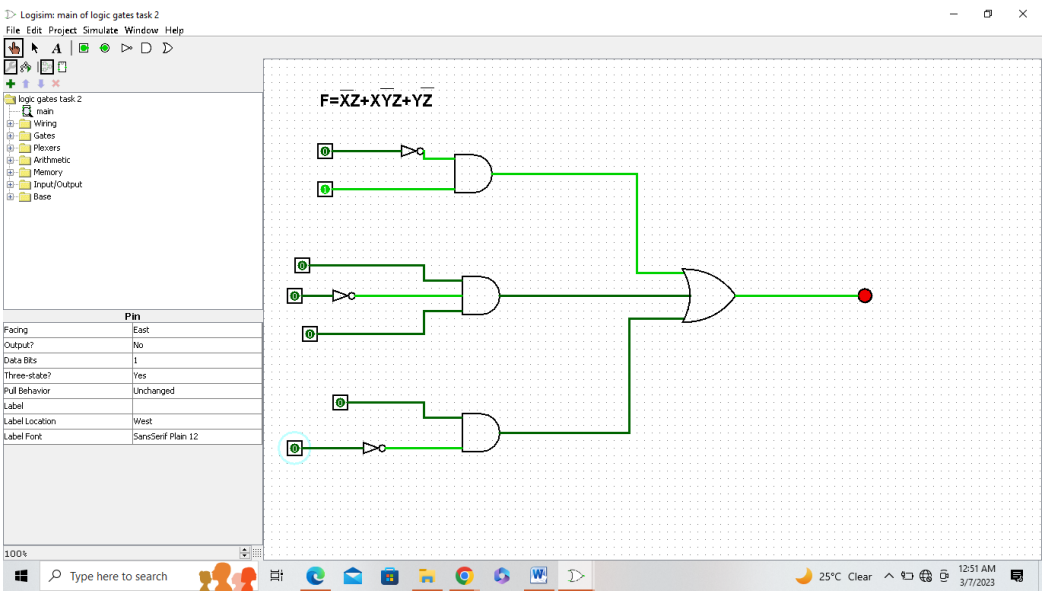
here is boolean expression of this Diagram

Q6. Draw the following function in Circuit maker.

1) F = $\bar{X}$YZ + $\bar{X}$Y$\bar{Z}$ + XZ



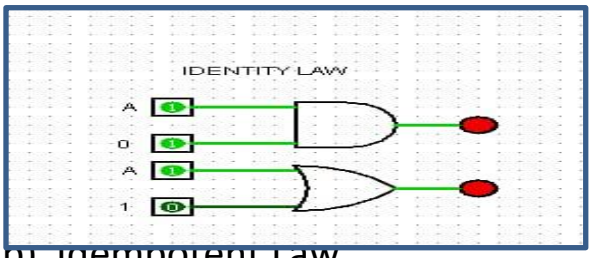
2) F=$\bar{X}$Z + X$\bar{Y}$Z +Y$\bar{Z}$



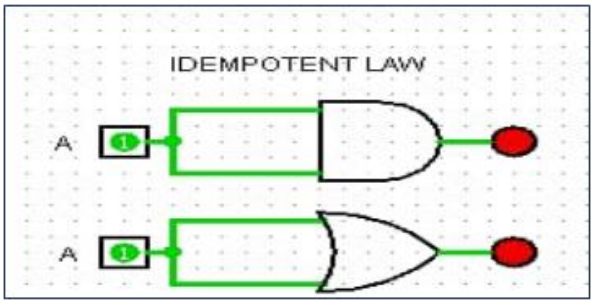
Practical 2

Q1. Verify following Rules of Boolean Algebra by designing them using Circuit Makers.

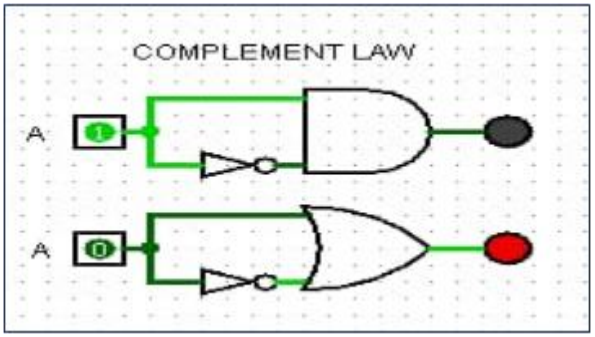
a) Identity Law



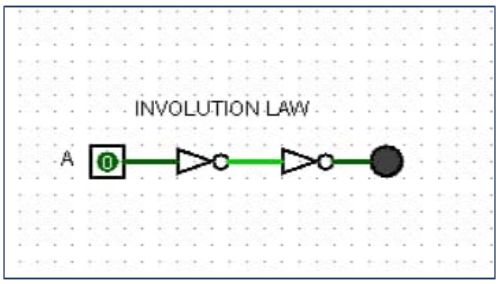
b) Idempotent Law



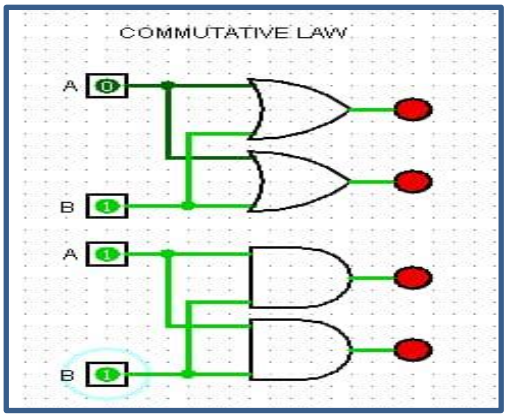
c) Complement



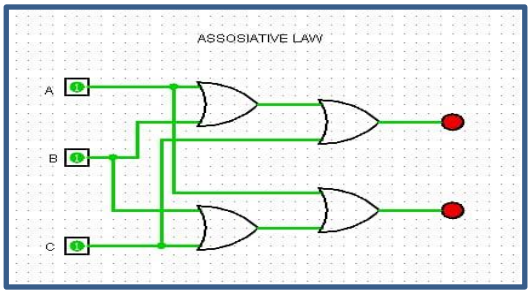
d) Involution Law



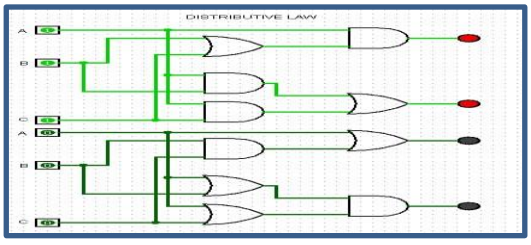
e) Commutative Law



f) Associative Law



g) Distributive Law



Q2. Simplify the expressions

- **F=(A+(BC)')**

Ans:

F = [A]' [(BC)']'

F = A'BC

- **F=(AB+CD)'**

Ans:

F = (AB)'(CD)'

Q3. Use DE Morgan‟s Theorem to prove that this NAND gate circuit performs the exact same function:

SOL:

F = [(AB)' (CD)']' BREAKING LONGEST COMPLIMENT

F = [(AB)']' + [(CD)']' COMPLIMENT LAW

F = AB+CD

Q4. Apply the principles of DE Morgan's theorems to the simplification of a gate circuit:

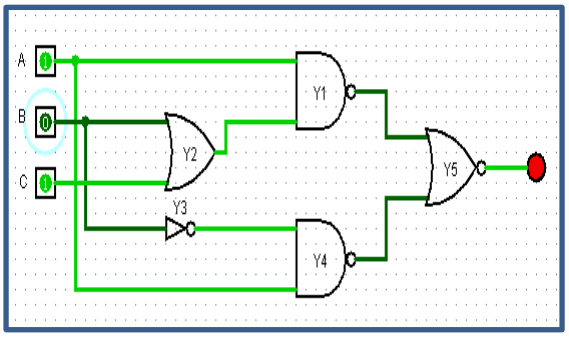
Y1 = [A(B+C)]'

Y2 = B+C

Y3 = B'

Y4 = (AB')'

Y5 = [{A(B+C)}' + (AB')']'



SOL:

= [{A (B+C)}' + (AB')']' //BREAKING LONGEST COMPLIMENT

= [{A (B+C)}']' . [(AB')']' //COMPLIMENT LAW

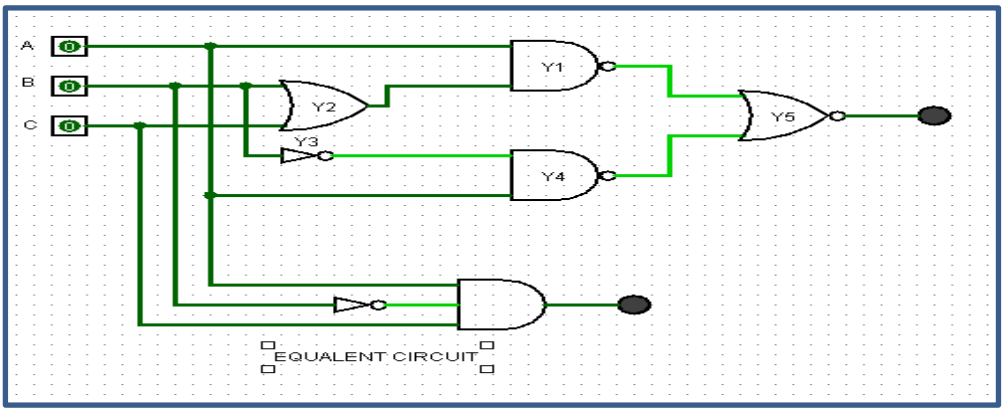
= [A(B+C)] . (AB') //DISTRIBUTIVE LAW

= (AB + AC) . AB' //MULTIPLYING

= (AB')(AB) + (AB')(AC) //A.A=A , B.B'=0

= A(0) +AB'C

= AB'C



Practical 3

Q.1 Verify Truth table for Three Basic Gates in Logisim.

NOT gate (also called an inverter)

Input	Output
0	1
1	0

AND gate

Input 1	Input 2	Output
0	0	0
0	1	0
1	0	0
1	1	1

OR gate

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	1

Q.2 Show how we can change/edit the properties of any device.

A device's settings, which can differ based on the model and operating system, can be changed or edited to change or alter the properties of a device. Following are some general guidelines:

Locate the device whose attributes you want to change.

Open the settings or device properties window.

Find the setting you wish to modify; it can be on a particular tab or page.

The setting can be changed as needed.

Close the window after saving the changes.

To modify some characteristics, you may not be able to or you may need sophisticated technical knowledge, therefore keep in mind that the precise steps may vary based on the device and operating system you're using. For detailed instructions or help, always check the device manufacturer's or manufacturer's documentation.

Q.3 Write down the functionality of Explorer Pane.

Many software programmes have the Explorer Pane, which facilitates file and folder management for users. Users can browse through the list of files and folders and take operations such as copying, moving, deleting, and renaming on them. Additionally, users can filter and sort the list, see the contents of files, search for files, and check properties like file size and creation date. Users may more easily organise and operate with their files and folders overall thanks to the Explorer Pane.

Q.4 What do you think Logisim is good software for manipulating Digital circuits or not?

Yes, Logisim is a good software for manipulating digital circuits, especially for beginners who want to learn about digital circuits and experiment with different designs. It has a user-friendly interface and allows users to create logic gates, combinational circuits, and sequential circuits using drag-and-drop tools. However, for more complex and professional projects, other software options may be more appropriate.

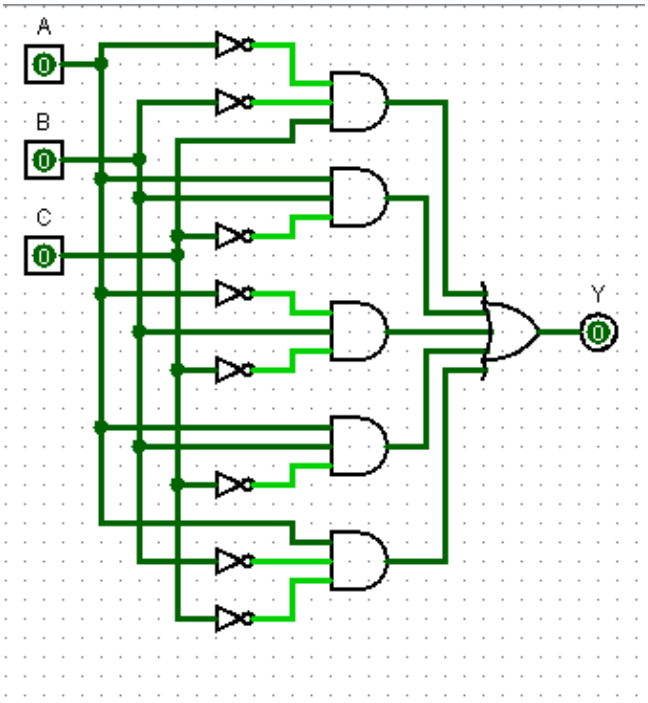
Practical 4

Q1. Simplify given expression using Standard Sum of Product, also show step by step process of building a circuit and designing a truth table.

i. F1(A,B,C) = A'B'C + BC' + AC'

SOL: ~A~B C + B ~C (A + ~A) + A ~C (B + ~B')

~A ~B C + B ~C A + B ~C ~A + A ~C B + A ~C ~B

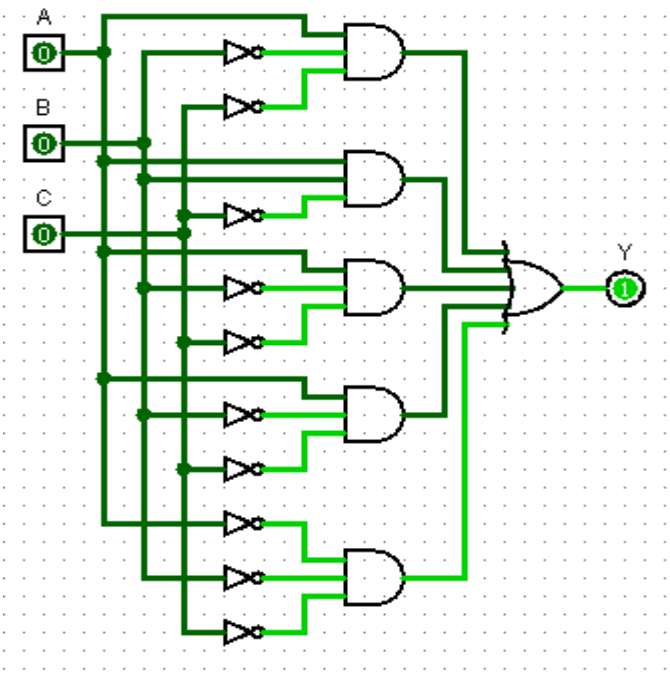


A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

ii. F2(A,B,C) = AB'C' + AC' + B'C'

SOL: A ~B ~C + A ~C (B + ~B) + ~B ~C (A + ~A)

A ~B ~C + A B ~C + A ~B ~C + A ~B ~C + ~A ~B ~C



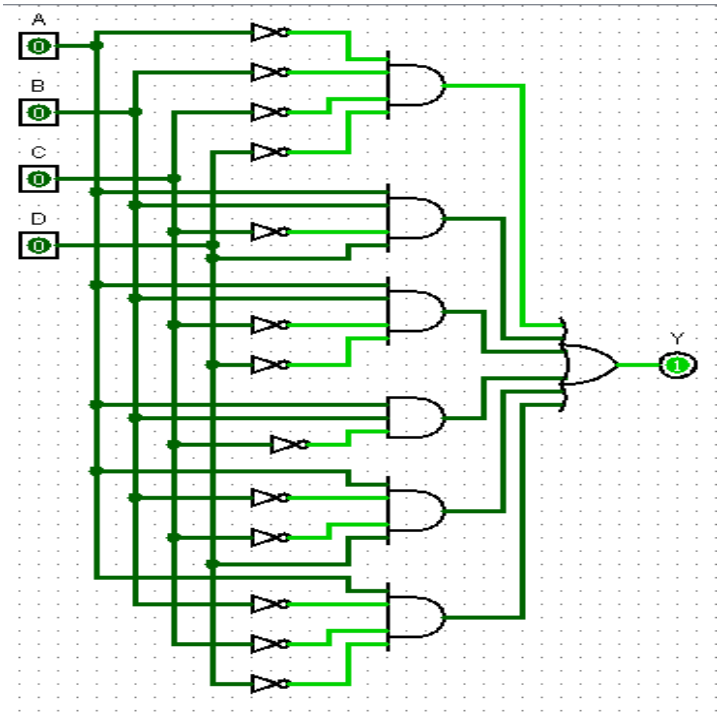
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

iii. F1(A,B,C,D) = A'B'C'D' + ABC' + AC'

SOL: = ~A ~B ~C ~D + A B ~C (D + ~D) + A ~C (B + ~B)

~A ~B ~C ~D + A B ~C D + A B ~C ~D + A B ~C + A ~B ~C (D + ~D)

~A ~B ~C ~D + A B ~C D + A B ~C ~D + A B ~C + A ~B ~C D + A ~B ~C ~D



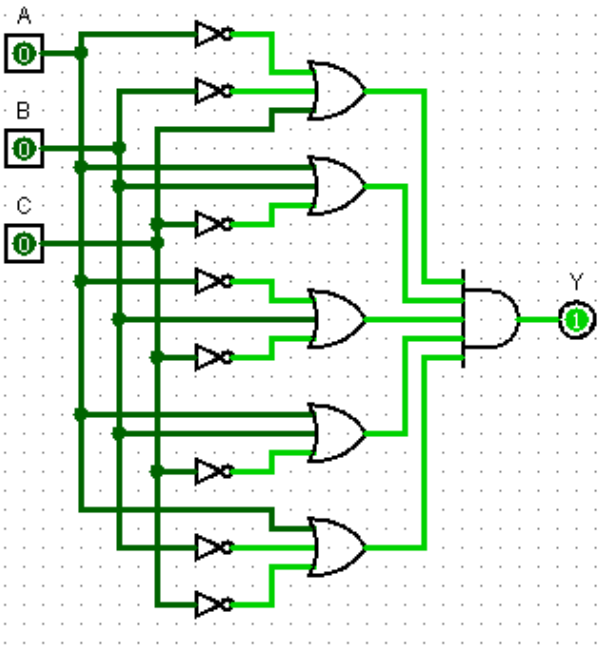
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Q 2. Simplify given expression using Standard Product of Sum, also show step by step process of building a circuit and designing a truth table

i. F1(A,B,C) = (A'+B'+C)(B+C') (A+C')

SOL: (~A + ~B + C) (B + ~C) (A + ~A) (A + ~C) (B + ~B)

(~A + ~B + C) (A + B + ~C) (~A + B + ~C) (A + B + ~C) (A + ~B + ~C)

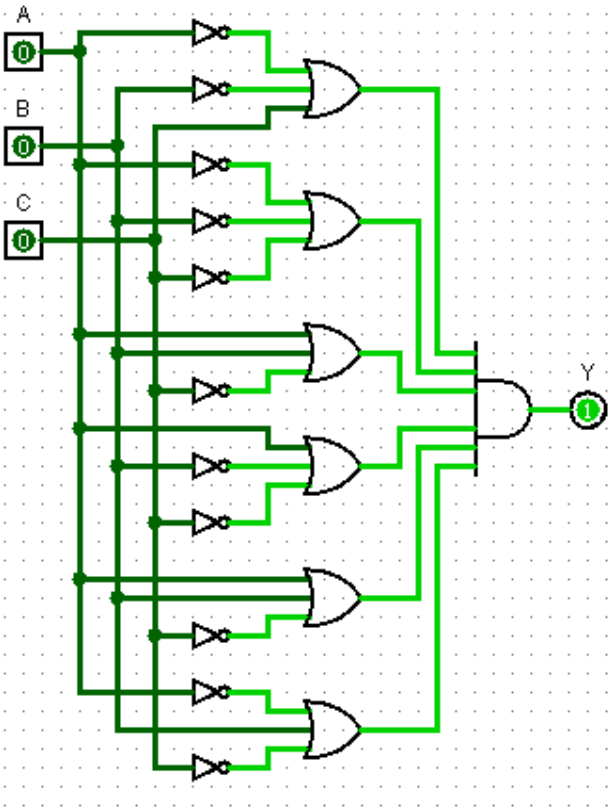


A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

ii. F2(A,B,C) = (A'+B')(A+C') (B+C')

SOL: (~A + ~B) (C + ~C) (A + ~C) (B + ~B) (B+ ~C) (A + ~A)

(~A + ~B + C) (~A + ~B + ~C) (A + B + ~C) (A + ~B + ~C) (A + B + ~C) (~A + B + ~C)



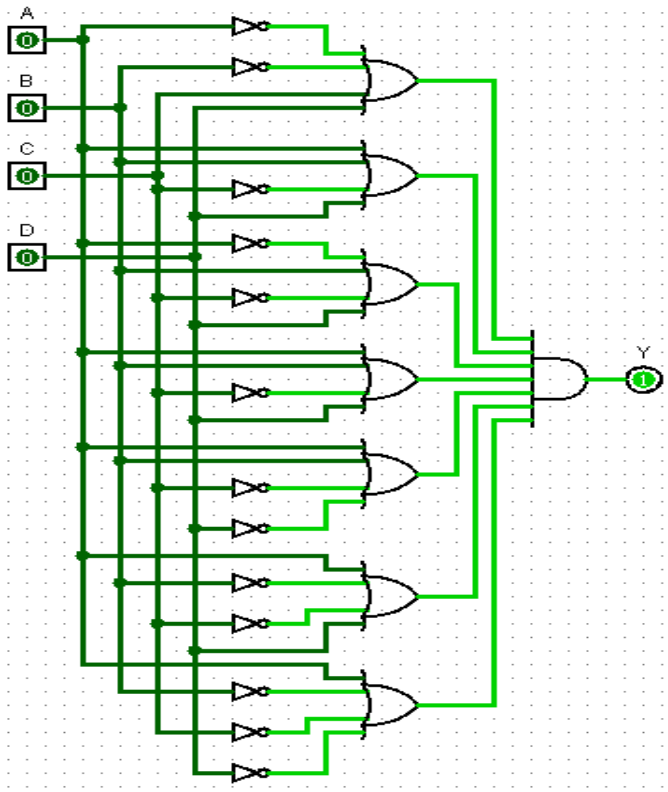
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

iii. F1(A,B,C,D) = (A'+B'+C+D)(B+C'+D) (A+C')

SOL: (~A + ~B + C + D) (B + ~C + D) (A + ~A) (A + ~C) (B + ~B)

(~A + ~B + C + D) (A + B + ~C + D) (~A + B + ~C +D) (A + B + ~C) (D + ~D) (A + ~B + ~C) (D + ~D)

(~A + ~B + C + D) (A + B + ~C + D) (~A + B + ~C +D) (A + B + ~C + D) (A + B + ~C + ~D) (A + ~B + ~C + D) (A + ~B + ~C + ~D)



A	B	C	D	Y
0	0	0	0	1
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	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
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	1

Q3. Why do we convert SOF & POS into their Canonical form?

Sum of Products (SOP) and Product of Sums (POS) expressions are transformed into their Canonical forms in order to make them simpler, implement them using logic gates, and assess whether they are equivalent to other expressions. A Boolean statement can be represented in what is known as canonical form,

which enables us to use Boolean algebraic methods for simplification. We can compare two expressions to see if they are the same and build smaller, more effective logic circuits by translating expressions into their canonical form.

Q4. What is Combinational Analysis?

Combinatorial analysis is the study of combinational circuits, which are circuits composed of logic gates that carry out a particular function based on their input signals, in digital logical architecture. Combinatorial analysis is the study and creation of circuits that carry out particular logical operations, such as AND, OR, NOT, and XOR. It also entails figuring out how many input and output signals are needed for a specific circuit and figuring out what input signals might be combined to produce what output signal. A crucial component of digital logical design is combinatorial analysis, which makes sure that the circuits are built to carry out the specified logical operations precisely and effectively.

Q5. What are minterms and Maxterms?

In digital logic design, concepts called minterms and maxterms are employed to analyse and simplify logic operations.

A minterm is a logical expression that symbolises the ANDing of every input variable into a logic function, where each variable can either be in its complemented or uncomplemented form. A minterm, then, is a product term in which each input variable only appears once, either in complemented form or uncomplemented form.

For Example, the minterms in a two-input logic function with inputs A and B would be A'B', A'B, AB', and AB, reflecting all potential input combinations.

The ORing of all input variables in a logic function is represented by a maxterm, on the other hand, where each input variable is represented as either its

complemented or uncomplemented form. A maxterm, then, is a sum term in which each input variable only appears once, either in complemented form or uncomplemented form.

For instance, the maxterms in a two-input logic function with inputs A and B would be (A+B), (A'+B), (A+B'), and (A'+B'), which reflect all potential input combinations.

In Boolean algebra, minterms and maxterms are used to condense and transform between several representations of the same logic function, such as from a truth table to a Boolean expression.