Unit -IV
CHAPTER- 13
BOOLEAN ALGEBRA

Boolean algebra

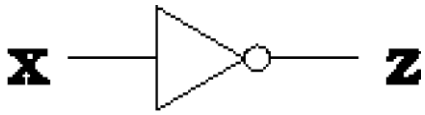
Boolean algebra:- a modern algebra which uses the set of numbers 0 and 1,the logic numbers used to solved logic problems.
Binary decision:-. The decision which results into yes or no. Also called logical statements or truth function.
Truth table:- a table representing all possible input-output combinations for a given logical problem/expression.
Tautology:-a Boolean expression that always results in true or 1.
Fallacy:-a Boolean expression that always results in false or 0.
Cononical expression :- a Boolean expression having all minterms or maxterms.
Minterm:-product of all the literals(with or without the bar) within the logic system.
Maxterm:-sum of all the literals(with or without the bar) within the logic system.
Karnaugh Map:- it is a graphical representation of the truth table of the given expression.

Logic Gates

Logic gates serve as the building blocks to digital logic circuits using combinational logic. We're going to consider the following gates: NOT gates (also called inverters), AND gates, OR gates, NAND gates, NOR gates, XOR gates, and XNOR gates.
We'll also discuss the concept of gate deltay.

NOT gates

NOT gates or *inverters* have a single bit input and a single bit of output.
This is a diagram of a NOT gate. It is a triangle with a circle on the right. The circle indicates "negation".



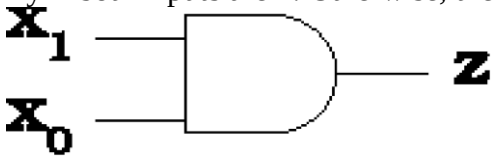
The truth table defines the behavior of this gate.

x	z
0	1
1	0

where **x** is the input and **z** is the output.

AND_2 gates

AND_2 gates have two bits of input and a single bit of output. The subscript, 2, indicates how many inputs this AND gate has. For example, AND_3 gates have 3 inputs.
The output of AND_2 gate is 1 only if both inputs are 1. Otherwise, the output is 0.



The truth table defines the behavior of this gate.

x_1	x_0	z
0	0	0
0	1	0
1	0	0
1	1	1

The function implmented by **AND_2** gates has interesting properties:

1. The function is symmetric. Thus, **x * y == y * x**. This can be verified by using truth tables. We use * to represent AND_2.
2. The function is associative. Thus, **(x * y) * z == x * (y * z)**. This can be verified by using truth tables.

Because of these properties, it's easy to define **AND_n**, which is an n-input **AND** gate.

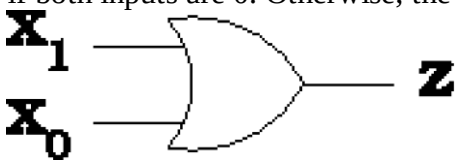
$$AND_n(x_1, x_2,...,x_n) = x_1 * x_2 * ... * x_n$$

That is, an AND gate with n-inputs is the AND of all the bits. This is not ambiguous because the AND function is associative (all parenthesization of this expression are equivalent).

OR_2 gates

OR_2 gates have two bits of input and a single bit of output. The subscript, 2, indicates how many inputs this OR gate has. For example, OR_3 gates have 3 inputs.

The output of OR_2 gate is 0 only if both inputs are 0. Otherwise, the output is 1.



The truth table defines the behavior of this gate.

x_1	x_0	z
0	0	0
0	1	1
1	0	1
1	1	1

The function implemented by **OR_2** gates has interesting properties:

1. The function is symmetric. Thus, **x + y == y + x**. This can be verified by using truth tables. We use "+" to represent **OR_2**
2. The function is associative. Thus, **(x + y) + z == x + (y + z)**. This can be verified by using truth tables.

Because of these properties, it's easy to define **OR_n**, which is an n-input **OR** gate.

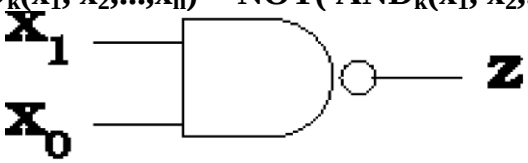
$$OR_n(x_1, x_2,...,x_n) = x_1 + x_2 + ... + x_n$$

That is, an AND gate with n-inputs is the AND of all the bits. This is not ambiguous because the AND function is associative (all parenthesization of this expression are equivalent).

$NAND_2$ gates

$NAND_2$ gates have two bits of input and a single bit of output. The subscript, 2, indicates how many inputs this NAND gate has. For example, $NAND_3$ gates have 3 inputs.
$NAND_k$ gates is define unusually. Since $NAND_2$ is *not* associative, the definition is based on AND_2. In particular

$$\mathbf{NAND_k(x_1, x_2,...,x_n) = NOT(AND_k(x_1, x_2,...,x_n))}$$



Thus, $NAND_k$ is the negation of AND_k.
The truth table defines the behavior of this gate. It's the negation of **AND_2**.

x_1	x_0	z
0	0	1
0	1	1
1	0	1
1	1	0

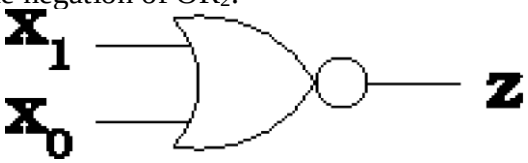
The function implemented by **$NAND_2$** gates has interesting properties:

1. The function is symmetric. Thus, **x NAND y == y NAND x**. This can be verified by using truth tables.
2. The function is *not* associative. This can be verified by using truth tables.

Because of these properties, **$NAND_k$** is defined from **AND_k**, and *not* built from **$NAND_2$** gates.

NOR_2 gates

OR_2 gates have two bits of input and a single bit of output. The subscript, 2, indicates how many inputs this OR gate has. For example, NOR_3 gates have 3 inputs.
The output of NOR_2 gate is the negation of OR_2.



The truth table defines the behavior of this gate.

x_1	x_0	z
0	0	1
0	1	0
1	0	0
1	1	0

The function implmented by **NOR_2** gates has interesting properties:

1. The function is symmetric. Thus, **x NOR y == y NOR x**. This can be verified by using truth

tables.

2. The function is *not* associative. This can be verified by using truth tables.

Because of these properties, **NOR_k** is defined from **OR_k**, and *not* built from **NOR_2** gates.

XOR_2 gates

XOR_2 gates have two bits of input and a single bit of output.

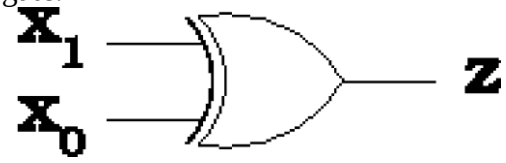
The output of XOR_2 gate is 1 only if the inputs have opposite values. That is, when one input has value 0, and the other has value 1.. Otherwise, the output is 0.

This is called *exclusive-or*. The definition of OR_2 is *inclusive-or*, where the output is 1 if either input is 1, or if both inputs are 1.

XOR_2 can be defined using AND_2, OR_2, and NOT.

x XOR y == (x AND (NOT y)) OR ((NOT x) AND y) == x\y + y\x

Here's a diagram of the XOR_2 gate.



If you look carefully at the drawing of the gate, there is a second arc behind the first one near the inputs. Since this second arc is hard to see, it's usually a good idea to write the word "XOR" inside the gate.

The truth table defines the behavior of this gate.

x_1	x_0	z
0	0	0
0	1	1
1	0	1
1	1	0

The function implmented by **XOR_2** gates has interesting properties:

1. The function is symmetric. Thus, **x (+) y == y (+) x**. This can be verified by using truth tables. (We use (+) to denote logical XOR--ideally, we'd draw it with a + sign inside a circle, but HTML doesn't seem to have a symbol for this).
2. The function is associative. Thus, **[x (+) y] (+) z == x (+) [y (+) z]**. This can be verified by using truth tables.

Because of these properties, it's easy to define **XOR_n**, which is an n-input **XOR** gate.

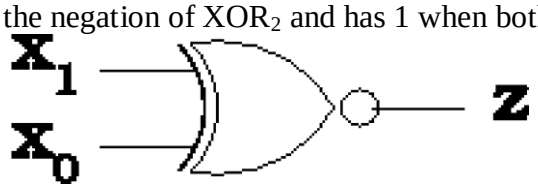
$$XOR_n(x_1, x_2,...,x_n) = x_1 (+) x_2 (+) ... (+) x_n$$

That is, an XOR gate with n-inputs is the XOR of all the bits. This is not ambiguous because the XOR function is associative (all parenthesization of this expression are equivalent).

$XNOR_2$ gates

$XNOR_2$ gates have two bits of input and a single bit of output.

The output of $XNOR_2$ gate is the negation of XOR_2 and has 1 when both inputs are the same.



If you look carefully at the drawing of the gate, there is a second arc behind the first one near the inputs. Since this second arc is hard to see, it's usually a good idea to write the word "XNOR" inside the gate.

The truth table defines the behavior of this gate.

x_1	x_0	z
0	0	0
0	1	1
1	0	1
1	1	0

The function implmented by **$XNOR_2$** gates has interesting properties:

1. The function is symmetric. Thus, **x XNOR y == y XNOR x**. This can be verified by using truth tables.
2. The function is associative. Thus, **(x XNOR y) XNOR z == x XNOR (y XNOR z)**. This can be verified by using truth tables.

Because of these properties, it's easy to define **$XNOR_n$**, which is an n-input **XNOR** gate.

$$XNOR_n(x_1, x_2,...,x_n) = x_1 \text{ XNOR } x_2 \text{ XNOR } ... \text{ XNOR } x_n$$

That is, an XNOR gate with n-inputs is the XNOR of all the bits. This is not ambiguous because the XNOR function is associative (all parenthesization of this expression are equivalent).

(Error-checkers! You may wish to verify this, and email me if this is incorrect!).

Solved questions

Q1. State the principal of duality in Boolean algebra.

Ans Principal of duality states that from every Boolean relation, another Boolean relation can be derived by

i) Changing each OR sign(+) to an AND sign(.)
ii) Changing each an AND sign(.) to an OR sign(+)
iii) Replacing each 1 with 0 and each 0 with 1

Q2. Define the following terms:

(a) Logical constant
(b) Logical variable
(c) Binary valued quantity
(d) Boolean literal

Ans:- (a) the truth values true(1) or false(0)are known as logical constants.
(b) A variable that can store the truth-values (TRUE or FALSE) is called logical variable.
(c) quantity can be represented in terms of TRUE or FALSE is known as binary valued quantity.
(d) a single Boolean variable(logical variable) or its complement e.g X or Y or $\bar{Z}$ is known as literals.

Q3. State Demorgan's laws.

Ans Demorgan's first law: -it states that $\overline{X+Y} = \bar{X}.\bar{Y}$

Demorgan's second law: -it states that $\bar{X}.\bar{Y} = \overline{X+Y}$

Q4 . Why are AND and NOR gates called Universal Gates?

Ans. Nand and Nor gates are less expensive and easier to design.also,other switching function(And ,OR) can easily be implemented using NAND/NOR gates.thus,these (NAND,NOR) Gates are also referred to as universal gates

Q5. Given the following truth table, derive a sum of product (SOP) and product of sum (POS) form of Boolean expression from it:

X	Y	Z	G(X,Y,Z)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Ans. In SOP F= $\sum(1,2,3,5,7)$

$= \bar{X}\ \bar{Y}Z + \bar{X}\,Y\ \bar{Z} + X\ \bar{Y}Z + XYZ$

In POS F=$\pi(0,3,4,6)$

$=(X+Y+Z)(X+\bar{Y}+\bar{Z})\,(\bar{X}+Y+Z)(\bar{X}+\bar{Y}+Z).$

Q6. Prove that X.(X+Y)=X by truth table method.

Ans.

X	Y	X+Y	X.(X+Y)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

From the above table it is obvious that X.(X+Y) = X because both the columns are identical.

Q7. Prove that X.(X+Y)=X by algebric method

Ans. LHS = X.(X+Y)

= X.X+X.Y

=X+X.Y

=X.(1+Y)

=X.1 = X = RHS

Q8. State the distributive laws of Boolean algebra. How are they different from distributive laws of ordinary algebra.

Ans. Distributive laws of Boolean algebra state that

(i) X(Y+Z) = XY+XZ

(ii) X+YZ =(X+Y)(X+Z)

Ist law X(Y+Z) = XY+XZ holds good for all values of X, Y and Z in ordinary algebra whereas X+YZ =(X+Y)(X+Z) holds good only for two values (0,1) of X, Y and Z

Q9. In Boolean algebra, verify using truth table that (X + Y)' = X' Y' for each X, Y in (0, 1).

Ans. As it is a 2-variable expression, truth table will be as follows :

X	Y	X+Y	(X+Y)'	X'	Y'	X'Y'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

State Demorgan's laws. Verify one of the Demorgan's laws using truth tables.

Ans. De Morgan's first theorem. It states that $X + Y = X \cdot Y$

De Morgan's second theorem. It states that $X \cdot Y = X + Y$

Truth table for second theorem

X	Y	X.Y	X.Y	X	Y	X+Y
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

X.Y and X+Y are identical.

Q10. Why are AND and NOR gates called Universal gates?

Ans. NAND and NOR gates are less expensive and easier to design. Also, other switching functions (AND, OR) can easily be implemented using NAND/NOR gates. Thus, these (NAND, NOR) gates are also reffered to as Universal Gates.

Q11. By means of truth table, demonstrate the validity of the following Postulates / Laws of Boolean algebra:

(a) Commulative law
(b) Absorption law
(c) Idempotent law

Ans The commulative law states that

(i) X+Y = Y+X (ii) X.Y =Y.X

(i) Truth table for X+Y = Y+X is given below :

Input		**Output**	
X	**Y**	**X+Y**	**Y+X**
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Comparing the columns X+Y and Y+X, we see both of these are identical. Hence proved.

(ii) Truth table for X.Y = Y.X is given below :

Input		Output	
X	**Y**	**X.Y**	**Y.X**
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Comparing the columns X.Y and Y.X, we see both of these are identical. Hence proved.

The absorption law states that

(i) X+XY = X (ii) X(X+Y) =X

(i) Truth table for X+XY = X is given below :

Input		Output	
X	**Y**	**XY**	**X+XY**
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Both the columns X+XY and X are identical. Hence proved.

(ii) Truth table for X.(X+Y) = X is given below :

Input		Output	
X	**Y**	**X+Y**	**X(X+Y)**
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Column X and X(X+Y) are identical. Hence proved.

The Idempotent law states that

(i) X+X = X ii) X.X =X

(i) Truth table for X+X = X is given below :

Input		Output
X	**X**	**X+X**
0	0	0
1	1	1

(ii) Truth table for X.X = X is given below :

Input	Output

X	X	X.X
0	0	0
1	1	1

Q12. Obtain the simplified form of a boolean expression using Karnaugh map.

$F(u,v,w,x) = \sum (0, 3, 4, 5, 7, 11, 13, 15)$

[00]WZ	[01] WZ	[11]WZ	[10]WZ
1		1	
1	1	1	
	1	1	
		1	

2 quads, 1 pair.

Quad 1(m3+m7+m11+m15) reduces to WZ

Quad 2(m5+m7+m13+m15) reduces to VZ

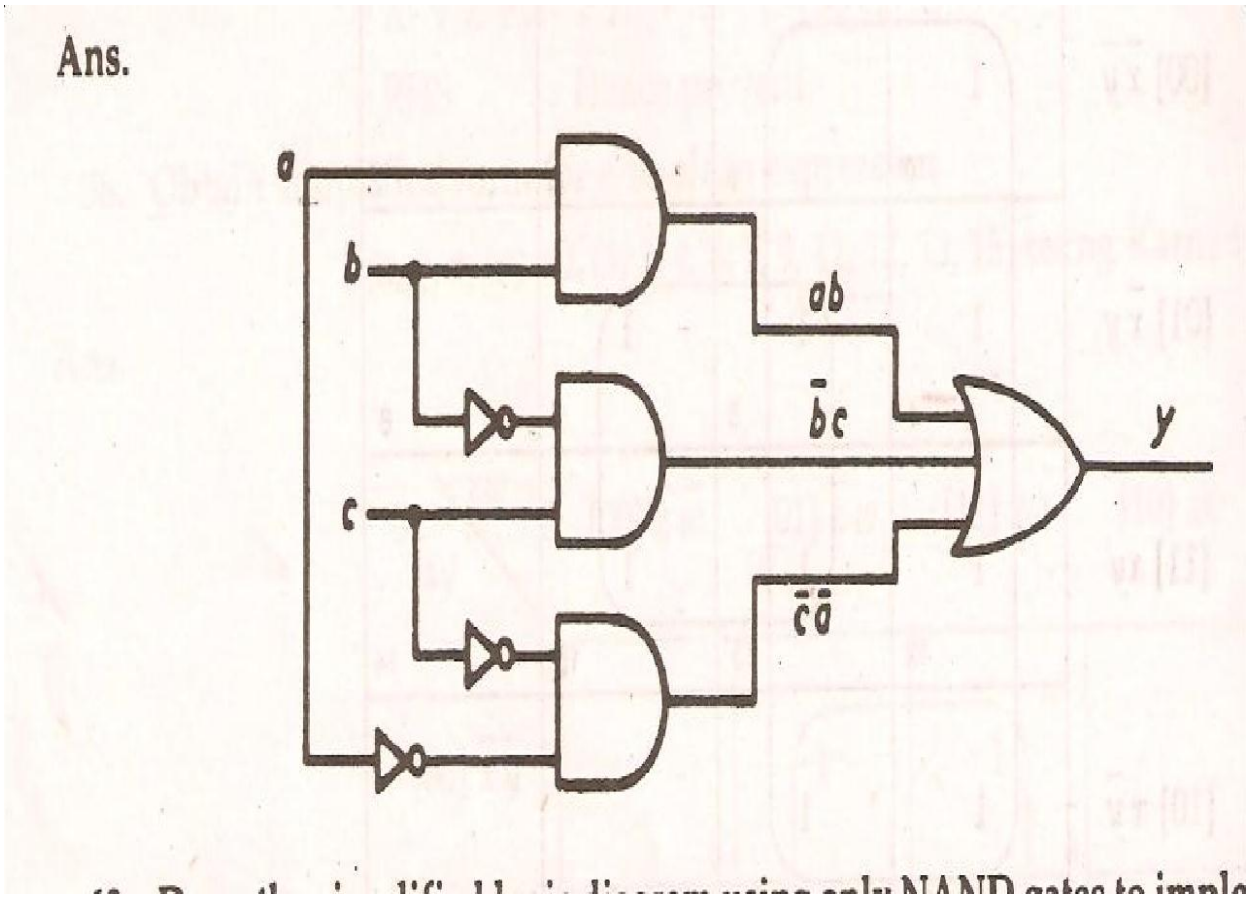
Pair 1(m0,m4) reduces to UWZ

Therefore F=WZ + VZ + UWZ

Q13 . Draw the logic circuit diagram for the following expression :

Y = a b + b c + c a

Ans.

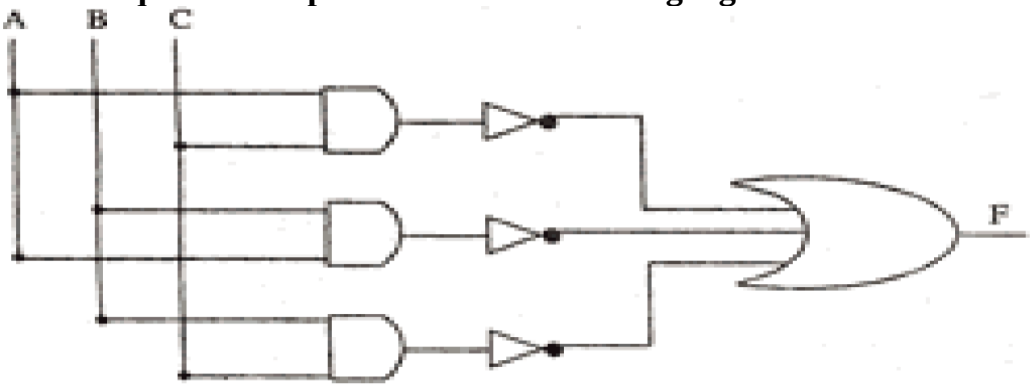


Q14. Prepare a truth table for X Y Z + X Y

Ans. Truth table for X Y Z + X Y is given below :

Input			Output					
X	**Y**	**Z**	**X'**	**Y'**	**Z'**	**X'YZ'**	**XY'**	**X'YZ' + XY'**
0	0	0	1	1	1	1	0	0
0	0	1	1	1	0	0	0	0
0	1	0	1	0	1	1	0	1
0	1	1	1	0	0	0	0	0
1	0	0	0	1	1	1	1	1
1	0	1	0	1	0	0	1	1
1	1	0	0	0	1	1	0	0
1	1	1	0	0	0	0	0	0

Q15. Write the equivalent expression for the following logic circuit :



Ans. F=(AC)' +(BA)'+(BC)'

Q16. Draw the circuit diagram for F = AB'C + C'B using NAND to NAND logic only.

Ans. F=AB'C + C'B =((A)NAND(B')NAND(C))NAND((C')NAND B)

Q17. Write the Sum of Products form of the function G(U,V,W). Truth table representation of G is as follows :

U	V	W	G
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Ans. To get the product of sums form, we need to add maxterms for all those input combinations that produce output as 0. Thus ,

G(U,V,W) = (U + V + W) (U + V + W') (U + V' + W') (U' + V + W')

Ans. (X + Y')(X' + Y)(X' + Y')

Unsolved Questions:

1. Define Binary logic.
2. What is a Boolean Operation ?
3. Define a boolean function.
4. Define a Boolean Expression.
5. Name the three primary and secondary operators of Boolean Algebra.
6. State any four postulates of boolean algebra.
7. Define Idempotent Law & Absorptive Law.
8. Define Involution Law.
9. What is De Morgan's Theorem ?
10. State the principle of duality.
11. Define the Sum Of Products format of a boolean expression.
12. Define the Product of Sums format of a boolean expression.
13. What is a Full Adder ?
14. Differentiate between an Encoder and a Decoder ?
15. Define the working of a XOR gate ?
16. What is a Canonical Sum of Products ?
17. What is a Canonical Product of Sums ?
18. State and verify duality principle.
19. What do you understand by a minterm and a maxterm?
20. F(x,y,z,w)=∑(1,3,4,5,7,9,11,12,13,15) using k-map.

High Order Thinking Skills(HOTS)

Q1. **State the principle of duality in Boolean Algebra and give the dual of the Boolean ex-pression (X + Y) . (Xf + Zf) . (Y + Z)**

Ans: Principle of Duality states that from every Boolean relation, another boolean relation can be derived by Changing each OR sign to AND and vice versa Replacing each 1 by 0 and vice versa. The new derived relation is known as the dual of the original relation.
Dual of (X + Y) . (X□f + Z□f) . (Y + Z) is X.Y + X□f.Z□f + Y.Z

Q2. **Sev**en inverters are cascaded one after another. What is the output if the input is 1?

Ans: 0.

Q3. **Why are NAND and NOR gates called Universal Gates?**

Ans: NAND and NOR gates are less expensive and easier to design. Also other functions (NOT, AND, OR) can easily be implemented using NAND/NOR gates.

Q4. **Obtain a simplified form for the following Boolean ex-pression using Karnaugh□fs Map:**
F(a, b, c, d) = ‡"(0, 1, 2, 4, 5, 7, 8, 9, 10, 11, 14).

Ans:

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

Quad 1 = afcf , Quad 2 = abf , Pair 1 is afbd, Pair 2 is bfcdf , Pair 3 is acdf
The simplified form is: afcf + abf + afbd + bfcdf + acdf

Q5. Prove XY+YZ+YZ□f=Y algebraically.

Ans:- XY+YZ+YZ□f
=XY+Y(Z+Z□f)
=XY+Y=Y(1+X)=Y hence proved

Q6. Simplified ABfCDf+ABfCD+ABCDf+ABCD.

Ans. ABfCDf+ABfCD+ABCDf+ABCD
= ABfC(Df+D)+ABC(Df+D)
= ABfC.1+ABC.1
= AC(Bf+B)
= AC.1=AC.

Q7. Draw the diagram of digital circuit for F(a,b,c)= AB+BC+CD using NAND-to- NAND logic.

Ans. F(a,b,c)= AB+BC+CD
=(A NAND B) NAND (B NAND C) NAND (C NAND D)
Thus the logic circuit is

Q8. Prepare a truth table for XfYf+XfY

Ans XfYf+XfY is a 2- Variable ex-pression ,its truth table is as follows

X	Y	Xf	Yf	XfYf	XfY	XfYf+XfY
0	0	1	1	1	0	1
0	1	1	0	0	1	1
1	0	0	1	0	0	0
1	1	0	0	0	0	0

Q9. Convert X+Y into minterms.

Ans X+Y=X.1+Y.1
=X(Y+Y□f)+Y(X+X□f)
=XY+XY□f+XY+X□fY
=XY+XY+XY□f+X□fY
=XY+XY□f+X□fY.

Q10. Convert the following function into canonical product of sums form F(X,Y,Z)=Π(0,2,4,5).

Ans $F(X,Y,Z)=\Pi(0,2,4,5)= M_O.M_2.M_4.M_5$
M0=000=X+Y+Z
M2=010=X+Y□f+Z
M4=100=X□f+Y+Z
M5=101=X□f+Y+Z□f
F= (X+Y+Z)(X+Y□f+Z)(X□f+Y+Z)(X□f+Y+Z□f)