UNIT- I

INTRODUCTION:

In 1854, George Boole, an English mathematician, proposed algebra for symbolically representing problems in logic so that they may be analyzed mathematically. The mathematical systems founded upon the work of Boole are called *Boolean algebra* in his honor.

The application of a Boolean algebra to certain engineering problems was introduced in 1938 by C.E. Shannon.

For the formal definition of Boolean algebra, we shall employ the postulates formulated by E.V. Huntington in 1904.

Fundamental postulates of Boolean algebra:

The postulates of a mathematical system forms the basic assumption from which it is possible to deduce the theorems, laws and properties of the system.

The most common postulates used to formulate various structures are—

i) **Closure**:

A set S is closed w.r.t. a binary operator, if for every pair of elements of S, the binary operator specifies a rule for obtaining a unique element of S.

The result of each operation with operator (+) or (.) is either 1 or 0 and 1, 0 ∈B.

ii) **Identity element**:

A set S is said to have an identity element w.r.t a binary operation * on S, if there exists an element e ∈ S with the property,

$$\boxed{e^* \, x = x \, * \, e = x}$$

Eg: 0+ 0 = 0   0+ 1 = 1+ 0 = 1   a) **x+ 0= x**

   1 . 1 = 1   1 . 0 = 0 . 1 = 1   b) **x. 1 = x**

iii) **Commutative law**:

A binary operator * on a set S is said to be commutative if,

$$\boxed{x \, * \, y = y \, * \, x}$$ for all x, y ∈ S

Eg: 0+ 1 = 1+ 0 = 1   a) **x+ y= y+ x**
            0 . 1 = 1 . 0 = 0   b) **x. y= y. x**

iv) **Distributive law**:

   If * and • are two binary operation on a set S, • is said to be distributive over **+** whenever,

> **x . (y+ z) = (x. y) + (x. z)**

Similarly, **+** is said to be distributive over • whenever,

> **x + (y. z) = (x+ y). (x+ z)**

v) **Inverse**:

   A set S having the identity element e, w.r.t. binary operator * is said to have an inverse, whenever for every x $\in$ S, there exists an element x' $\in$ S such that,

> **x. x' $\in$ e**

   a) x+ x' = 1, since 0 + 0' = 0+ 1 and 1+ 1' = 1+ 0 = 1
   b) x. x' = 1, since 0 . 0' = 0. 1 and 1. 1' = 1. 0 = 0

Summary:
Postulates of Boolean algebra:

POSTULATES	(a)	(b)
Postulate 2 (Identity)	**x + 0 = x**	**x . 1 = x**
Postulate 3 (Commutative)	**x+ y = y+ x**	**x . y = y. x**
Postulate 4 (Distributive)	**x (y+ z) = xy+ xz**	**x+ yz = (x+ y). (x+ z)**
Postulate 5 (Inverse)	**x+x' = 1**	**x. x' = 0**

Basic theorem and properties of Boolean algebra:

Basic Theorems:

The theorems, like the postulates are listed in pairs; each relation is the dual of the one paired with it. The postulates are basic axioms of the algebraic structure and need no proof. The theorems must be proven from the postulates. The proofs of the theorems with one variable are presented below. At the right is listed the number of the postulate that justifies each step of the proof.

1) a) **x+ x = x**

$x+ x = (x+ x) . 1$ ------------------ by postulate 2(b) [x. 1 = x]

$= (x+ x). (x+ x')$ ------------------ 5(a) [x+ x' = 1]

$= x+ xx'$ ------------------ 4(b) [x+yz = (x+y)(x+z)]

$= x+ 0$ ------------------ 5(b) [x. x' = 0]

$= x$ ------------------ 2(a) [x+0 = x]

b) x. x = x

$x. x = (x. x) + 0$ ------------------ by postulate 2(a) [x+ 0 = x]

$= (x. x) + (x. x')$ ------------------ 5(b) [x. x' = 0]

$= x (x+ x')$ ------------------ 4(a) [x (y+z) = (xy)+ (xz)]

$= x (1)$ ------------------ 5(a) [x+ x' = 1]

$= x$ ------------------ 2(b) [x.1 = x]

2) a) **x+ 1 = 1**

$x+ 1 = 1 . (x+ 1)$ ------------------ by postulate 2(b) [x. 1 = x]

$= (x+ x'). (x+ 1)$ ------------------ 5(a) [x+ x' = 1]

$= x+ x'.1$ ------------------ 4(b) [x+yz = (x+y)(x+z)]

$= x+ x'$ ------------------ 2(b) [x. 1 = x]

$= 1$ ------------------ 5(a) [x+ x'= 1]

b) **x .0 = 0**

3) **(x')' = x**

From postulate 5, we have x+ x' = 1 and x. x' = 0, which defines the complement of x. The complement of x' is x and is also (x')'.

Therefore, since the complement is unique,

$(x')' = x.$

4) Absorption Theorem:

a) **x+ xy = x**

x+ xy = x. 1 + xy   ------------------ by postulate 2(b)   [x. 1 = x]

= x (1+ y)   ------------------   4(a) [x (y+z) = (xy)+ (xz)]

= x (1)   ------------------ by theorem 2(a)   [x+ 1 = x]

= x.   ------------------ by postulate 2(a)   [x. 1 = x]

b)   x. (x+ y) = x

x. (x+ y) = x. x+ x. y   ------------------   4(a) [x (y+z) = (xy)+ (xz)]

= x + x.y   ------------------ by theorem 1(b)   [x. x = x]

= x.   ------------------ by theorem 4(a)   [x+ xy = x]

c) x+ x′y = x+ y

x+ x′y = x+ xy+ x′y   ------------------ by theorem 4(a)   [x+ xy = x]

= x+ y (x+ x′)   ------------------ by postulate 4(a) [x (y+z) = (xy)+ (xz)]

= x+ y (1)   ------------------   5(a)   [x+ x′ = 1]

= x+ y   ------------------   2(b)   [x. 1= x]

d) x. (x′+y) = xy

x. (x′+y) = x.x′+ xy   ------------------ by postulate 4(a) [x (y+z) = (xy)+ (xz)]

= 0+ xy   ------------------   5(b)   [x. x′ = 0]

= xy.   ------------------   2(a)   [x+ 0= x]

Properties of Boolean algebra:

1. Commutative property:

Boolean addition is commutative, given by

$$x+ y = y+ x$$

According to this property, the order of the OR operation conducted on the variables makes no difference.

Boolean algebra is also commutative over multiplication given by,

$$x. y = y. x$$

This means that the order of the AND operation conducted on the variables makes no difference.

2. Associative property:

The associative property of addition is given by,

$$A+ (B+ C) = (A+B) + C$$

The OR operation of several variables results in the same, regardless of the grouping of the variables.

The associative law of multiplication is given by,

$$A. (B. C) = (A.B) . C$$

It makes no difference in what order the variables are grouped during the AND operation of several variables.

3. <u>Distributive property</u>:

The Boolean addition is distributive over Boolean multiplication, given by

$$A+ BC = (A+B) (A+C)$$

The Boolean addition is distributive over Boolean addition, given by

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

4. <u>Duality</u>:

It states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged.

If the dual of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

$x+ x' = 1$ is $x. x' = 0$

Duality is a very important property of Boolean algebra.

Summary:

Theorems of Boolean algebra:

	THEOREMS	(a)	(b)
1	Idempotency	x + x = x	x . x = x
2		x + 1 = 1	x . 0 = 0
3	Involution	(x′)′ = x	
4	Absorption	x+ xy = x	x (x+ y) = x
		x+ x′y = x+ y	x. (x′+ y)= xy
5	Associative	x+(y+ z)= (x+ y)+ z	x (yz) = (xy) z
6	DeMorgan's Theorem	(x+ y)′= x′. y′	(x. y)′= x′+ y′

DeMorgan's Theorems:

Two theorems that are an important part of Boolean algebra were proposed by DeMorgan.

The first theorem states that the complement of a product is equal to the sum of the complements.

$$(AB)' = A' + B'$$

The second theorem states that the complement of a sum is equal to the product of the complements.

$$(A + B)' = A'. B'$$

Consensus Theorem:

In simplification of Boolean expression, an expression of the form **AB+ A′C+ BC**, the term BC is redundant and can be eliminated to form the equivalent expression AB+ A′C. The theorem used for this simplification is known as consensus theorem and is stated as,

$$AB+ A'C+ BC = AB+ A'C$$

The dual form of consensus theorem is stated as,

$$(A+B) (A'+C) (B+C) = (A+B) (A'+C)$$

BOOLEAN FUNCTIONS:

Minimization of Boolean Expressions:

The Boolean expressions can be simplified by applying properties, laws and theorems of Boolean algebra.

Simplify the following Boolean functions to a minimum number of literals:

1. x (x′+y)

= xx′+ xy	[x. x′= 0]
= 0 + xy	[x+ 0 = x]
= xy.	

2. x+ x′y

= x + xy + x′y	[x+ xy= x]
= x+ y (x+x′)	
= x+ y (1)	[x+ x′ = 1]
= x+ y.	

3. (x+ y) (x+ y′)

= x.x+ xy′+ xy+ yy′	
= x+ xy′+ xy+ 0	[x. x= 0]; [y. y′= 0]
= x (1+ y′+ y)	
= x (1)	[1+y= 1]
= x.	

4. xy + x′z + yz.

= xy + x′z + yz(x+ x′)	[x+ x′= 1]
= xy + x′z + xyz + x′yz	

Re-arranging,

= xy + xyz + x′z +x′yz	
= xy (1+ z) + x′z (1+y)	[1+y= 1]
= xy+ x′z.	

5. xy+ yz+ y′z

= xy+ z (y+ y′)	
= xy+ z (1)	[y+ y′ = 1]
= xy+ z.	

6. **(x+ y) (x′+ z) (y+ z)**
        = (x+ y) (x′+ z)   [dual form of consensus theorem,
                                             (A+ B) (A′+ C) (B+ C) = (A+ B) (A′+ C)]

7. **x′y+ xy+ x′y′**
        = y (x′+ x) + x′y′   [x (y+ z) = xy+ xz]
        = y (1) + x′y′   [x+ x′ = 1]
        = y+ x′y′   [x+ x′y′ = x+ y′]
        = y+ x′.

8. **x+ xy′+ x′y**
        = x (1+ y′)+ x′y
        = x (1) + x′y   [1+ x = 1]
        = x+ x′y   [x+ x′y = x+ y]
        = x+ y.

9. **AB + (AC)' + AB′C (AB + C)**
        = AB + (AC)' + AAB'BC + AB'CC
        = AB + (AC)' + 0+ AB'CC   [B.B' = 0]
        = AB + (AC)' + AB'C   [C.C = 1]
        = AB + A' + C' +AB'C   [(AC)' = A' + C']
        = AB + A′ + C' + AB'   [C′ + AB′C = C′ + AB′]
        = A' + B+ C′+ AB′   [A′ + AB = A′ + B]
   Re- arranging,
        = A' + AB′+ B+ C'   [A′ + AB = A′ + B]
        = A' + B′+ B+ C'   [B′+ B= 1]
        = A' +1+ C′   [A+ 1= 1]
        = 1

10. **(x′+ y) (x+ y)**
         = x′.x+ x′y+ yx+ y.y
         = 0+ x′y+ xy+ y   [x.x′= 0]; [x. x= x]
         = y (x′+ x+ 1)
         = y(1)   [1+ x = 1]
         = y.

11. **xy+ xyz+ xy (w+ z)**
         = xy (1+ z+ w+ z)

$$= xy\ (1\ ) \qquad\qquad\qquad\qquad\qquad [\ 1+ x = 1\ ]$$
$$= xy.$$

12. $xy+ xyz+ xyz'+ x'yz$
$$= xy\ (1+ z+ z')+ x'yz$$
$$= xy\ (1\ ) + x'yz \qquad\qquad\qquad [\ 1+ x = 1\ ]$$
$$= xy+ x'yz$$
$$= y\ (x+ x'z\ ) \qquad\qquad\qquad\qquad [\ x+ x'y = x+ y]$$
$$= y\ (x+ z\ ).$$

13. $xyz+ xy'z+ xyz'$
$$= xy\ (z+ z') + xy'z$$
$$= xy+ xy'z \qquad\qquad\qquad\qquad\quad [\ x+ x'= 1]$$
$$= x(y+ y'z) \qquad\qquad\qquad\qquad\quad [\ x+ x'y = x+ y]$$
$$= x(y+ z)$$

14. $x'y'z'+ x'yz'+ xy'z'+ xyz'$
$$= x'z'\ (y'+ y)+ xz'\ (y'+ y)$$
$$= x'z'+ xz' \qquad\qquad\qquad\qquad\quad [\ x+ x'= 1]$$
$$= z'\ (x'+ x)$$
$$= z' \qquad\qquad\qquad\qquad\qquad\qquad [\ x+ x'= 1]$$

15. $w'xyz'+ xyz'+ xy'z'+ xy'z$
$$= xyz'\ (w'+ 1) + xy'z'+ xy'z$$
$$= xyz'+ xy'z'+ xy'z \qquad\qquad [\ 1+ x = 1\ ]$$
$$= xz'\ (y+ y') + xy'z$$
$$= xz'+ xy'z \qquad\qquad\qquad\qquad [\ x+ x'= 1]$$
$$= x\ (z'+ y'z)$$
$$= x\ (z'+ y').\qquad\qquad\qquad\qquad [\ x'+ xy' = x'+ y']$$

16. $w'xy'z+ w'xyz+ wxz$
$$= w'xz\ (y'+ y)+ wxz$$
$$= w'xz\ (1)+ wxz \qquad\qquad\qquad [\ x+ x'= 1]$$
$$= w'xz+ wxz$$
$$= xz\ (w'+ w)$$
$$= xz. \qquad\qquad\qquad\qquad\qquad\qquad [\ x+ x'= 1]$$

17. $x'y'z'+ x'y'z+ x'yz'+ x'yz+ xy'z'$
$$= x'y'\ (z'+z) + x'y\ (z'+z)+ xy'z'$$
$$= x'\ y'\ (1) + x'y\ (1)+ xy'z' \qquad\quad [\ x+ x'= 1]$$
$$= x'y' + x'y + xy'z'$$
$$= x'(y'+y) + xy'z'$$

$= x' (1) + xy'z'$ [x+ x'= 1]
$= x' + xy'z'$
$= x'+ y'z'.$ [x'+ xy' = x'+ y']

18. $w'y (w'xz)' + w'xy'z' + wx'y$
$= w'y (w''+ x'+ z') + w'xy'z' + wx'y$
$= w'y (w+ x'+ z') + w'xy'z' + wx'y$ [x'' = x]
$= w'yw+ w'y x'+ w'y z' + w'xy'z' + wx'y$
$= 0 + w'x'y+ w'y z' + w'xy'z' + wx'y$ [x. x'= 0]
Re-arranging,
$= w'x'y+ wx'y + w'y z' + w'xy'z'$
$= x'y (w'+ w) + w'z' (y+ xy')$
$= x'y (1) + w'z' (y+ xy')$ [x+ x'= 1]
$= x'y+ w'z' (y+x)$ [x+ x'y = x+ y]

19. $xy+ x (y+ z) + y (y+ z)$
$= xy+ xy+ xz+ yy+ yz$
$= xy+ xz+ y+ yz$ [x+ x= x]; [x. x= x]
$= xy+ xz+ y$ [x+ xy= x]
$= y+ xz$ [x+ xy= x]

20. $[xy' (z+ wy) + x'y'] z$
$= [xy'z+ xy'wy+ x'y'] z$
$= [xy'z+ 0+ x'y'] z$ [x. x'= 0]
$= xy'z. z+ x'y'z$
$= xy'z+ x'y'z$ [x. x= x]
$= y'z (x+ x')$
$= y'z (1)$ [x+ x'= 1]
$= y'z.$

21. $x'yz+ xy'z'+ x'y'z'+ xy'z+ xyz$
$= yz (x'+x) + xy'z'+ x'y'z'+ xy'z$
$= yz (1) + y'z' (x+ x') + xy'z$ [x+ x'= 1]
$= yz+ y'z' (1) + xy'z$ [x+ x'= 1]
$= yz+ y'z'+ xy'z$
$= yz+ y' (z'+ xz)$
$= yz+ y' (z'+ x)$ [x'+ xy = x'+ y]
$= yz+ y'z'+ xy'$

22. $[(xy)'+ x'+ xy]'$
$= [x'+ y'+ x'+ xy]'$
$= [x'+ y'+ xy]'$ [x+ x= x]

$$= [x'+ y'+ x]' \qquad\qquad\qquad [\, x'+ xy = x'+ y]$$
$$= [y'+ 1]' \qquad\qquad\qquad\qquad [\, x+ x'= 1]$$
$$= [\, 1\,]' \qquad\qquad\qquad\qquad\quad [\, 1+ x = 1\,]$$
$$= 0.$$

23. $[\, xy+ xz]'+ x'y'z$

$$= (xy)'.\, (xz)'+ x'y'z$$
$$= (x'+ y').\, (x'+ z')+ x'y'z$$
$$= x'x'+ x'z'+ x'y'+ y'z'+ x'y'z$$
$$= x'+ x'z'+ x'y'+ y'z'+ x'y'z \qquad\qquad [x+ x= x]$$
$$= x'+ x'z'+ x'y'+ y'\, [z'+ x'z]$$
$$= x'+ x'z'+ x'y'+ y'\, [z'+ x'] \qquad\qquad [\, x'+ xy = x'+ y]$$
$$= x'+ x'y'+ y'\, [z'+ x'] \qquad\qquad\qquad [x+ xy = x]$$
$$= x'+ x'y'+ y'z'+ x'y'$$
$$= x'+ y'z'+ x'y' \qquad\qquad\qquad\qquad [x+ xy = x]$$
$$= x'+ y'z'. \qquad\qquad\qquad\qquad\quad [x+ xy = x]$$

24. $xy+ xy'(\, x'z')'$

$$= xy+ xy'\, (x''+ z'')$$
$$= xy+ xy'\, (x+ z) \qquad\qquad\qquad [x'' = x]$$
$$= xy+ xy'x+ xy'z$$
$$= xy+ xy'+ xy'z \qquad\qquad\qquad [x.\, x= x]$$
$$= xy+ xy'\, [1+ z]$$
$$= xy+ xy'\, [1] \qquad\qquad\qquad\qquad [\, 1+ x = 1\,]$$
$$= xy+ xy'$$
$$= x(\, y+ y')$$
$$= x\, [1] \qquad\qquad\qquad\qquad\qquad\quad [\, x+ x'= 1]$$
$$= x.$$

25. $[(\, xy'+ xyz)'+ x\, (y+ xy')]'$

$$= [\, x(\, y'+yz)'+ x\, (y+ xy')]'$$
$$= [\, x(\, y'+z)'+ x\, (y+ x)]' \qquad\qquad [\, x'+ xy = x'+ y];\ [\, x+ x'y = x+ y]$$
$$= [\, x(\, y'+z)'+ xy+ x.x)]'$$
$$= [\, (xy'+xz)'+ xy+ x)]' \qquad\qquad [x.\, x= x]$$
$$= [\, (\, xy'+xz)'+ x)]' \qquad\qquad\qquad [x+ xy = x]$$
$$= [\, (xy')'.\, (xz)'+ x]'$$
$$= [\, (x'+y'').\, (x'+z')+ x]'$$
$$= [\, (x'+y).\, (x'+z')+ x]' \qquad\qquad [x'' = x]$$
$$= [\, (x'+ yz')+ x]' \qquad\qquad\qquad [\, (x+ y)\, (x+ z)= x+ yz]$$
$$= [\, x'+ yz'+ x]'$$
$$= [\, 1+ yz']' \qquad\qquad\qquad\qquad [\, x+ x'= 1]$$
$$= [1]' \qquad\qquad\qquad\qquad\qquad [\, 1+ x = 1\,]$$

= 0.

26. [(xy+ z′) ((x+ y)′+z)]′
    = [(xy+ z′) ((x′. y′)+z)]′
    = [xy. x′y′+ xy. z+ z′. x′y′+ z′. z]′
    = [0+ xyz+ x′y′z′+ 0]′   [x. x′= 0]
    = [xyz+ x′y′z′]′
    = (xyz)′. (x′y′z′)′
    = (x′+ y′+ z′). (x″+ y″+ z″)
    = (x′+ y′+ z′). (x+ y+ z).   [x″ = x]

27. (x+ y) (x′z′+ z) (y′+ xz)′
    = (x+ y) (x′z′+ z) (y″. (xz)′)
    = (x+ y) (x′+ z) (y. (xz)′)   [x+ x′y = x+ y]; [x″ = x]
    = (x+ y) (x′+ z) (y. (x′+z′))
    = (x.x′+ xz+ x′y+ yz) (x′y+ yz′)
    = (0+ xz+ x′y+ yz) (x′y+ yz′)
    = (xz+ x′y+ yz) (x′y+ yz′)
    = xz. x′y+ xz. yz′+ x′y. x′y+ x′y. yz′+ yz. x′y+ yz. yz′
    = 0+ 0+ x′y+ x′yz′+ x′yz+ 0   [x. x′= 0]; [x. x= x]
    = x′y+ x′yz′+ x′yz
    = x′y (1+ z′+ z)
    = x′y (1)   [1+ x = 1]
    = x′y.

28. Y= ∑m (1, 3, 5, 7)
    = x′y′z+ x′yz+ xy′z+ xyz
    = x′z(y′+y) + xz(y′+y)
    = x′z (1)+ xz (1)   [x+ x′= 1]
    = x′z+ xz
    = z(x′+ x)
    = z (1)   [x+ x′= 1]
    = z.

COMPLEMENT OF A FUNCTION:

The complement of a function F is F′ and is obtained from an interchange of 0′s for 1′s and 1′s for 0′s in the value of F. The complement of a function may be derived algebraically through DeMorgan's theorem.

DeMorgan's theorems for any number of variables resemble in form the two-variable case and can be derived by successive substitutions similar to the method used in the preceding derivation. These theorems can be generalized as –

$$(A+ B+ C+ D+ \ldots + F)' = A'\ B'\ C'\ D'\ \ldots\ F'$$

$$(A\ B\ C\ D\ \ldots\ F)' = A'+B'+ C'+ D'+ \ldots +F'.$$

Find the complement of the following functions,

1. **F= x′yz′+ x′y′z**

    F′= (x′yz′+ x′y′z)′
      = (x″+ y′+ z″) . (x″+ y″+z′)
      = (x+ y′+ z). (x+ y+ z′).

2. **F= (xy + y′z + xz) x.**

    F′ = [(xy + y′z + xz) x]′
       = (xy + y′z + xz)′ + x′
       = [(xy)′ . (y′z)′. (xz)′] + x′
       = [(x′+y′). (y+z′). (x′+z′)] + x′
       = [(x′y+ x′z′+ 0+ y′z′) (x′+z′)] + x′
       = x′x′y+ x′x′z′+ x′y′z′+ x′yz′+ x′z′z′+ y′z′z′+ x′
       = x′y+ x′z′+ x′y′z′+ x′yz′+ x′z′+ y′z′+ x′   [x+ x = x], [x. x = x]
       = x′y+ x′z′+ x′z′ (y′+ y) + y′z′+ x′   [x+ x′= 1]
       = x′y+ x′z′+ x′z′ (1) + y′z′+ x′
       = x′y+ x′z′+ y′z′+ x′
       = x′y+ x′+ x′z′+ y′z′

$$= x'(y+1) + x'z + y'z' \qquad\qquad [y+1= 1]$$
$$= x' (1+z) + y'z' \qquad\qquad\quad [y+1= 1]$$
$$= x'+ y'z'$$

3. **F= x (y′z′+ yz)**

$$F'= [x (y'z'+ yz)]'$$
$$= x'+ (y'z'+ yz)'$$
$$= x'+ (y'z')'. (yz)'$$
$$= x'+ (y''+ z'') . (y'+ z')$$
$$= x'+ (y+ z) . (y'+ z').$$

4. **F= xy′+ x′y**

$$F'= (xy'+ x'y)'$$
$$= (xy')'. (x'y)'$$
$$= (x'+y) (x+y')$$
$$= x'x+ x'y'+ yx+ yy'$$
$$= x'y'+ xy.$$

5. **f = wx′y + xy′+ wxz**

$$f' = (wx'y + xy'+ wxz)'$$
$$= (wx'y)' (xy')' (wxz)'$$
$$= (w'+x+ y') (x'+ y) (w'+ x'+ z')$$
$$= (w'x'+ w'y+ xx'+ xy+ x'y'+ yy') (w'+ x'+ z')$$
$$= (w'x'+ w'y+ xy+ x'y') (w'+ x'+ z')$$
$$= w'x'. w'+ w'y. w'+ xy. w'+ x'y'. w'+ w'x'. x'+w'y. x'+ xy. x'+ x'y'. x'+$$
$$\qquad w'x'. z'+ w'y. z'+ xy. z'+ x'y'.z'$$
$$= w'x'+ w'y+ w'xy+ w'x'y'+ w'x'+ w'x'y+ 0 + x'y'+ w'x'z'+ w'yz'+ xyz'+ x'y'z'$$
$$= w'x'+ w'y+ w'xy+ w'x'y'+ w'x'y+ x'y'+ w'x'z'+ w'yz'+ xyz'+ x'y'z'$$
$$= w'x'(1+ y'+ y+ z')+ w'y(1+ x+ z')+ x'y'(1+ z')+ xyz'$$
$$= w'x'(1)+ w'y(1)+ x'y'(1)+ xyz'$$
$$= w'x'+ w'y+ x'y'+ xyz'$$

CANONICAL AND STANDARD FORMS:

Minterms and Maxterms:

A binary variable may appear either in its normal form (x) or in its complement form (x'). Now either two binary variables x and y combined with an AND operation. Since each variable may appear in either form, there are four possible combinations:

x'y', x'y, xy' and xy

Each of these four AND terms is called a '*minterm*'.

In a similar fashion, when two binary variables x and y combined with an OR operation, there are four possible combinations:

x'+ y', x'+ y, x+ y' and x+ y

Each of these four OR terms is called a '*maxterm*'.

The minterms and maxterms of a 3- variable function can be represented as in table below.

Variables			Minterms	Maxterms
x	**y**	**z**	m_i	M_i
0	0	0	$x'y'z' = m_0$	$x+ y+ z= M_0$
0	0	1	$x'y'z = m_1$	$x+ y+ z'= M_1$
0	1	0	$x'yz' = m_2$	$x+ y'+ z= M_2$
0	1	1	$x'yz = m_3$	$x+ y'+ z'= M_3$
1	0	0	$xy'z' = m_4$	$x'+ y+ z= M_4$
1	0	1	$xy'z = m_5$	$x'+ y+ z'= M_5$
1	1	0	$xyz' = m_6$	$x'+ y'+ z= M_6$
1	1	1	$xyz = m_7$	$x'+ y'+ z'= M_7$

Sum of Minterm: (Sum of Products)

The logical sum of two or more logical product terms is called sum of products expression. It is logically an OR operation of AND operated variables such as:

1. Y= AB+ BC+ AC

Sum

2. Y= AB+ $\overline{B}$ C+ A$\overline{C}$

Product terms

Sum of Maxterm: (Product of Sums)

A product of sums expression is a logical product of two or more logical sum terms. It is basically an AND operation of OR operated variables such as,

1. Y= (A+B). (B+C). (A+C)

Product

2. Y= (A+B). ($\overline{B}$+C). (A+$\overline{C}$)

Sum terms

Canonical Sum of product expression:

If each term in SOP form contains all the literals then the SOP is known as standard (or) canonical SOP form. Each individual term in standard SOP form is called minterm canonical form.

F (A, B, C) = AB'C+ ABC+ ABC'

Steps to convert general SOP to standard SOP form:

1. Find the missing literals in each product term if any.
2. AND each product term having missing literals by ORing the literal and its complement.
3. Expand the term by applying distributive law and reorder the literals in the product term.
4. Reduce the expression by omitting repeated product terms if any.

Obtain the canonical SOP form of the function:

1. Y(A, B) = A+ B

$$= A. (B+ B')+ B (A+ A')$$
$$= \underline{AB}+ AB'+ \underline{AB}+ A'B$$
$$= AB+ AB'+ A'B.$$

2. $Y (A, B, C) = A+ ABC$

$$= A. (B+ B'). (C+ C')+ ABC$$
$$= (AB+ AB'). (C+ C')+ ABC$$
$$= \underline{ABC}+ ABC'+ AB'C+ AB'C'+ \underline{ABC}$$
$$= ABC+ ABC'+ AB'C+ AB'C'$$
$$= m_7+ m_6+ m_5+ m_4$$
$$= \sum m (4, 5, 6, 7).$$

3. $Y (A, B, C) = A+ BC$

$$= A. (B+ B'). (C+ C')+(A+ A'). BC$$
$$= (AB+ AB'). (C+ C')+ ABC+ A'BC$$
$$= \underline{ABC}+ ABC'+ AB'C+ AB'C'+ \underline{ABC}+ A'BC$$
$$= ABC+ ABC'+ AB'C+ AB'C'+ A'BC$$
$$= m_7+ m_6+ m_5+ m_4+ m_3$$
$$= \sum m (3, 4, 5, 6, 7).$$

4. $Y (A, B, C) = AC+ AB+ BC$

$$= AC (B+ B')+ AB (C+ C')+ BC (A+ A')$$
$$= \underline{ABC}+ AB'C+ \underline{ABC}+ ABC'+ \underline{ABC}+ A'BC$$
$$= ABC+ AB'C+ ABC'+ A'BC$$
$$= \sum m (3, 5, 6, 7).$$

5. $Y (A, B, C, D) = AB+ ACD$

$$= AB (C+ C') (D+ D') + ACD (B+ B')$$
$$= (ABC+ ABC') (D+ D') + ABCD+ AB'CD$$
$$= \underline{ABCD}+ ABCD'+ ABC'D+ ABC'D'+ \underline{ABCD}+ AB'CD$$
$$= ABCD+ ABCD'+ ABC'D+ ABC'D'+ AB'CD.$$

Canonical Product of sum expression:

If each term in POS form contains all literals then the POS is known as standard (or) Canonical POS form. Each individual term in standard POS form is called Maxterm canonical form.

- F (A, B, C) = (A+ B+ C). (A+ B′+ C). (A+ B+ C′)
- F (x, y, z) = (x+ y′+ z′). (x′+ y+ z). (x+ y+ z)

<u>Steps to convert general POS to standard POS form</u>:

1. Find the missing literals in each sum term if any.
2. OR each sum term having missing literals by ANDing the literal and its complement.
3. Expand the term by applying distributive law and reorder the literals in the sum term.
4. Reduce the expression by omitting repeated sum terms if any.

Obtain the canonical POS expression of the functions:

1. Y= A+ B′C
   = (A+ B′) (A+ C)   [**A+ BC = (A+B) (A+C)**]
   = (A+ B′+ C.C′) (A+ C+ B.B′)
   = <u>(A+ B′+C)</u> (A+ B′+C′) (A+ B+ C) <u>(A+ B′+ C)</u>
   = (A+ B′+C). (A+ B′+C′). (A+ B+ C)
   = M_2. M_3. M_0
   = $\prod M$ (0, 2, 3)

2. Y= (A+B) (B+C) (A+C)
   = (A+B+ C.C′) (B+ C+ A.A′) (A+C+B.B′)
   = <u>(A+B+C)</u> (A+B+C′) <u>(A+B+C)</u> (A′+B+C) <u>(A+B+C)</u> (A+B′+C)
   = (A+B+C) (A+B+C′) (A′+B+C) (A+B′+C)
   = M_0. M_1. M_4. M_2
   = $\prod M$ (0, 1, 2, 4)

3. Y= A. (B+ C+ A)
   = (A+ B.B′+ C.C′). (A+ B+ C)
   = <u>(A+B+C)</u> (A+B+C′) (A+B′+C) (A+ B′+C′) <u>(A+B+C)</u>
   = (A+B+C) (A+B+C′) (A+B′+C) (A+ B′+C′)
   = M_0. M_1. M_2. M_3
   = $\prod M$ (0, 1, 2, 3)

4. Y= (A+B′) (B+C) (A+C′)
    = (A+B′+C.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′)
    = (A+B′+C) (A+B′+C′) (A+B+C) (A′+B+C) (A+B+C′)
    = M_2. M_3. M_0. M_4. M_1
    = $\prod M$ (0, 1, 2, 3, 4)

5.   Y= xy+ x′z
     = (xy+ x′) (xy+ z)   Using distributive law, convert the function into OR terms.
     = (x+x′) (y+x′) (x+z) (y+z)   [x+ x′=1]
     = (x′+y) (x+z) (y+z)
     = (x′+y+ z.z′) (x+z+y.y′) (y+z+ x.x′)
     = (x′+ y+ z) (x′+ y+ z′) **(x+ y+ z)** (x+ y′+ z) **(x+ y+ z)** (x′+ y+ z)
     = (x′+ y+ z) (x′+ y+ z′) (x+ y+ z) (x+ y′+ z)
     = M_4. M_5. M_0. M_2
     = $\prod M$ (0, 2, 4, 5).

KARNAUGH MAP MINIMIZATION:

The simplification of the functions using Boolean laws and theorems becomes complex with the increase in the number of variables and terms. The map method, first proposed by Veitch and slightly improvised by Karnaugh, provides a simple, straightforward procedure for the simplification of Boolean functions. The method is called **Veitch diagram** or **Karnaugh map**, which may be regarded as a pictorial representation of a truth table.

The Karnaugh map technique provides a systematic method for simplifying and manipulation of Boolean expressions. **A** K-map is a diagram made up of squares, with each square representing one minterm of the function that is to be minimized. For n variables on a Karnaugh map there are 2^n numbers of squares. Each square or cell represents one of the minterms. It can be drawn directly from either minterm (sum-of-products) or maxterm (product-of-sums) Boolean expressions.
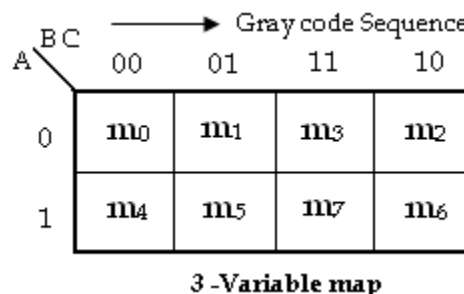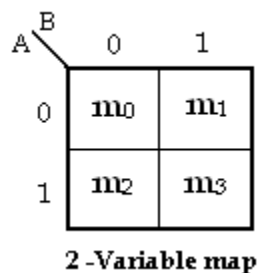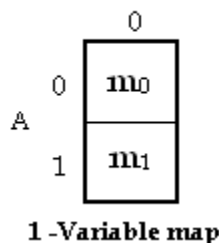
Two- Variable, Three Variable and Four Variable Maps

Karnaugh maps can be used for expressions with two, three, four and five variables. The number of cells in a Karnaugh map is equal to the total number of possible input variable combinations as is the number of rows in a truth table. For three variables, the number of cells is $2^3 = 8$. For four variables, the number of cells is $2^4 = 16$.

1 -Variable map

2 -Variable map

3 -Variable map

4 -Variable map
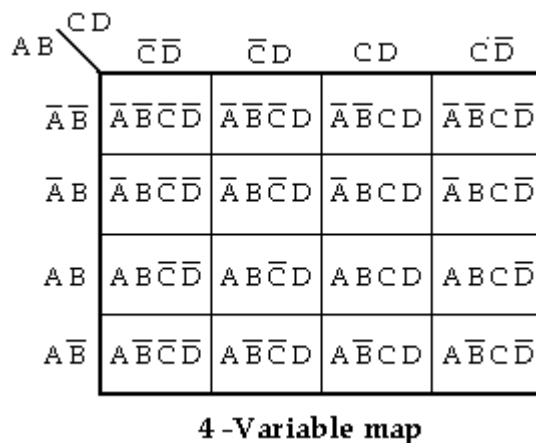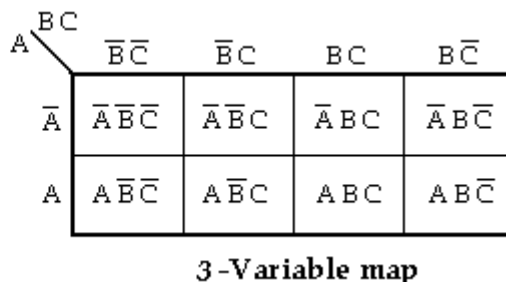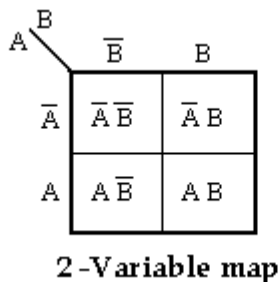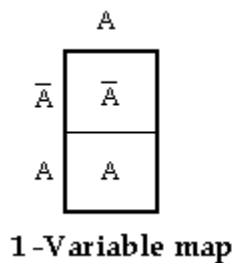
Product terms are assigned to the cells of a K-map by labeling each row and each column of a map with a variable, with its complement or with a combination of variables & complements. The below figure shows the way to label the rows & columns of a 1, 2, 3 and 4- variable maps and the product terms corresponding to each cell.

It is important to note that when we move from one cell to the next along any row or from one cell to the next along any column, one and only one variable in the product term changes (to a complement or to an uncomplemented form). Irrespective of number of variables the labels along each row and column must conform to a single change. Hence gray code is used to label the rows and columns of K-map as shown below.

1 -Variable map

2 -Variable map

3 -Variable map

4- Variable map

Grouping cells for Simplification:

The grouping is nothing but combining terms in adjacent cells. The simplification is achieved by grouping adjacent 1's or 0's in groups of 2i, where i = 1, 2, …, n and n is the number of variables. When adjacent 1's are grouped then we get result in the sum of product form; otherwise we get result in the product of sum form.

Grouping Two Adjacent 1's: (Pair)

In a Karnaugh map we can group two adjacent 1's. The resultant group is called Pair.
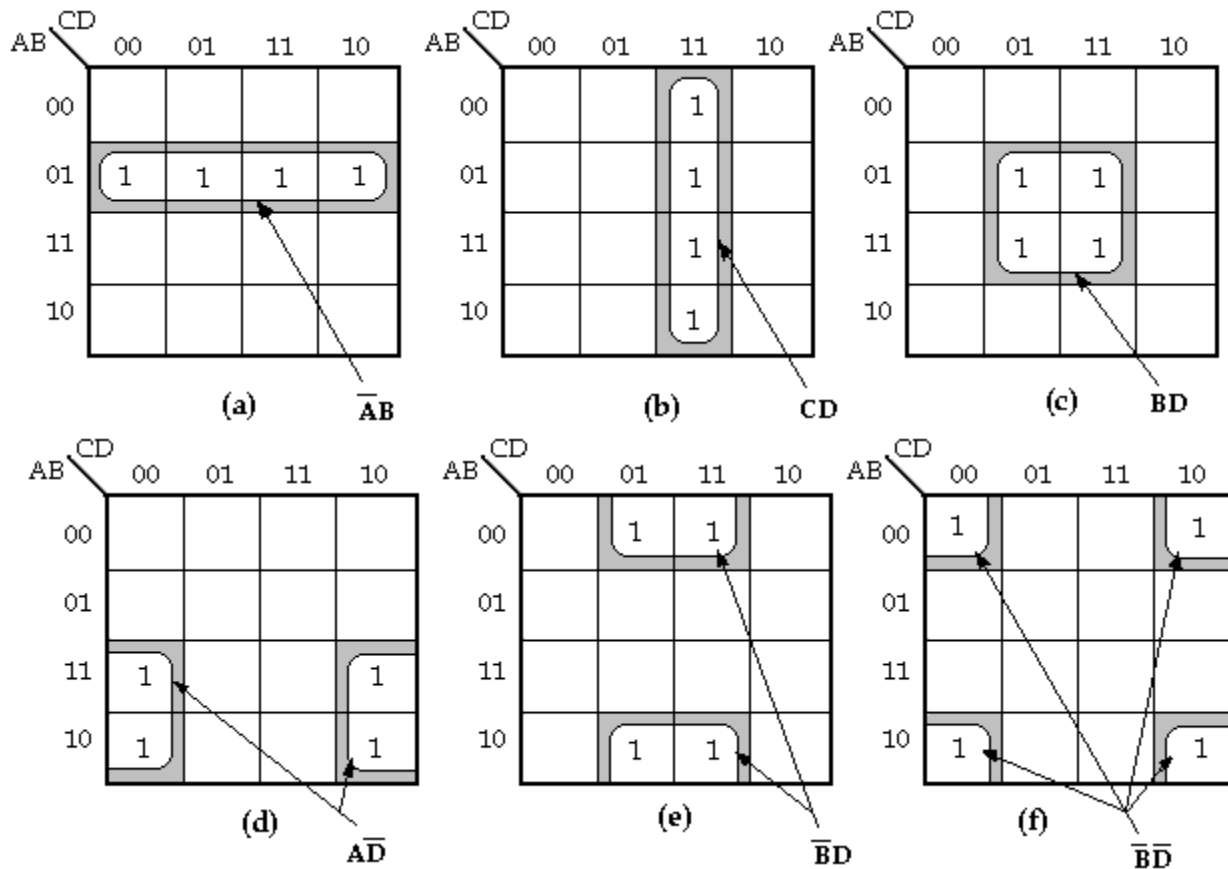

$\overline{A}BD$

BCD

$A\overline{B}\overline{D}$
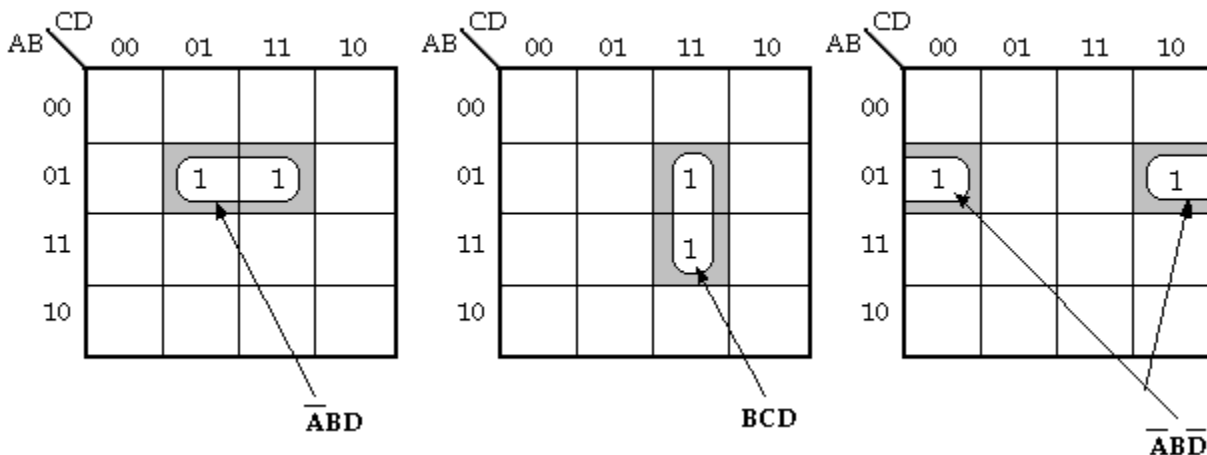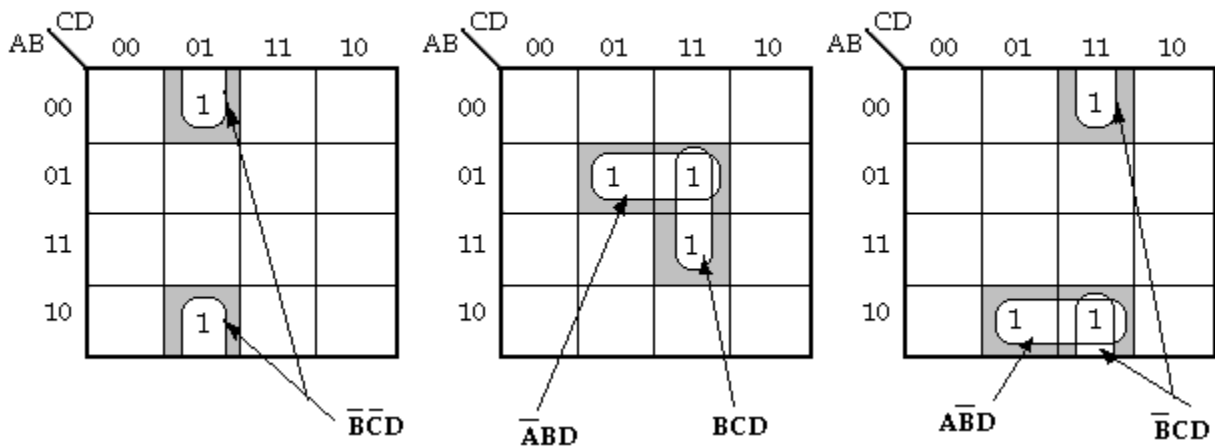
Examples of Pairs

Grouping Four Adjacent 1's: (Quad)

In a Karnaugh map we can group four adjacent 1's. The resultant group is called Quad. Fig (a) shows the four 1's are horizontally adjacent and Fig (b) shows they are vertically adjacent. Fig (c) contains four 1's in a square, and they are considered adjacent to each other.
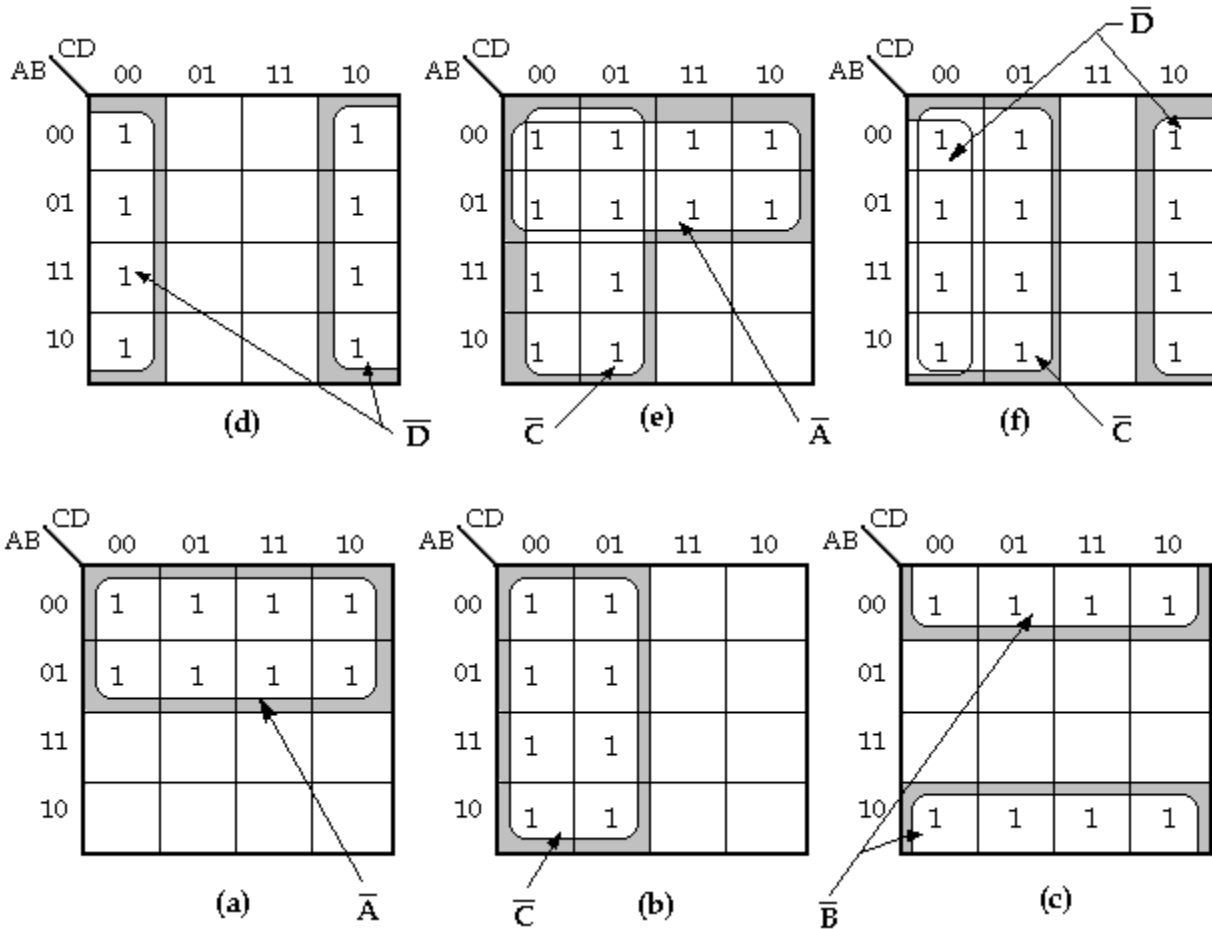
Examples of Quads

The four 1's in fig (d) and fig (e) are also adjacent, as are those in fig (f) because, the top and bottom rows are considered to be adjacent to each other and the leftmost and rightmost columns are also adjacent to each other.

Grouping Eight Adjacent 1's: (Octet)

In a Karnaugh map we can group eight adjacent 1's. The resultant group is called Octet.

Simplification of Sum of Products Expressions: (Minimal Sums)

The generalized procedure to simplify Boolean expressions as follows:

1. Plot the K-map and place 1's in those cells corresponding to the 1's in the sum of product expression. Place 0's in the other cells.
2. Check the K-map for adjacent 1's and encircle those 1's which are not adjacent to any other 1's. These are called **isolated 1's**.
3. Check for those 1's which are adjacent to only one other 1 and encircle such **pairs**.

4. Check for **quads** and **octets** of adjacent 1's even if it contains some 1's that have already been encircled. While doing this make sure that there are minimum number of groups.
5. Combine any pairs necessary to include any 1's that have not yet been grouped.
6. Form the simplified expression by summing product terms of all the groups.

Three- Variable Map:

1. Simplify the Boolean expression,
   $$F(x, y, z) = \sum m\ (3, 4, 6, 7).$$

Soln:
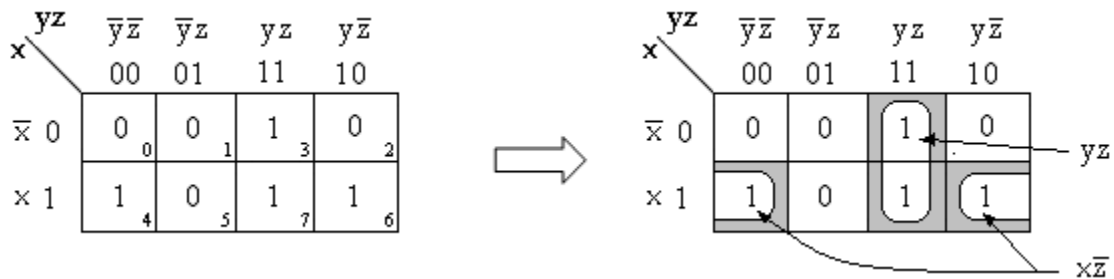
F = yz+ xz′

2. $F(x, y, z) = \sum m\ (0, 2, 4, 5, 6).$
Soln:
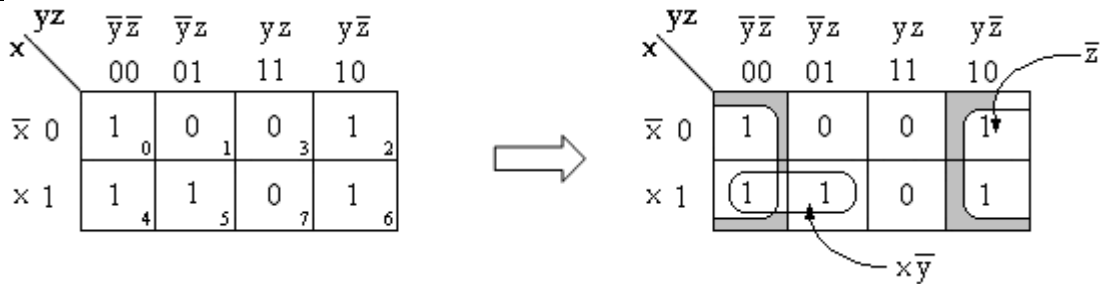
F = z′+ xy′

3. **F = A′C + A′B + AB′C + BC**
Soln:

$$= A′C\ (B+ B′) + A′B\ (C+ C′) + AB′C + BC\ (A+ A′)$$

$$= \underline{A′BC} + A′B′C + \underline{A′BC} + A′BC′ + AB′C + ABC + \underline{A′BC}$$

$$= A′BC + A′B′C + A′BC′ + AB′C + ABC$$

$$= m_3 + m_1 + m_2 + m_5 + m_7$$

$= \sum m\ (1, 2, 3, 5, 7)$

F = C + A′B

4. **AB′C + A′B′C + A′BC + AB′C′ + A′B′C′**

Soln:

$= m5 + m1 + m3 + m4 + m0$

$= \sum m\ (0, 1, 3, 4, 5)$

F = A′C + B′

Four - Variable Map:

1. Simplify the Boolean expression,

    Y = A′BC′D′ + A′BC′D + ABC′D′ + ABC′D + AB′C′D + A′B′CD′

Soln:
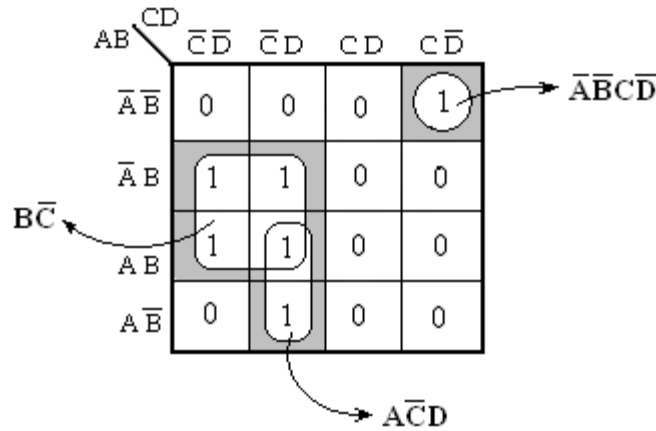
Therefore,

Y= A'B'CD'+ AC'D+ BC'

2. **F (w, x, y, z) = $\sum$ m(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)**
Soln:
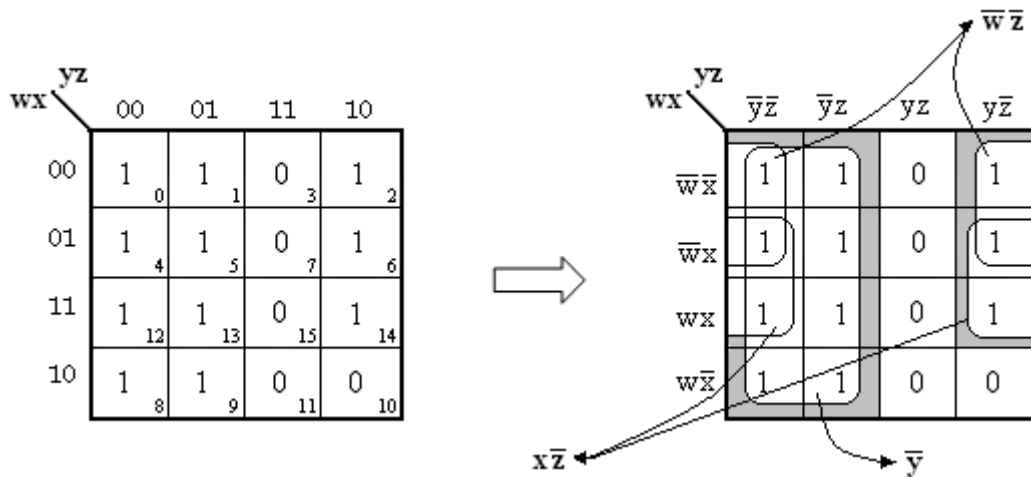
Therefore,

F= y'+ w'z'+ xz'

3. **F= A'B'C'+ B'CD'+ A'BCD'+ AB'C'**
      = A'B'C' (D+ D') + B'CD' (A+ A') + A'BCD'+ AB'C' (D+ D')
      = A'B'C'D+ A'B'C'D'+ AB'CD'+ A'B'CD'+ A'BCD'+ AB'C'D+ AB'C'D'
      = m_1+ m_0+ m_{10}+ m_2+ m_6+ m_9+ m_8

$$= \sum m\ (0, 1, 2, 6, 8, 9, 10)$$
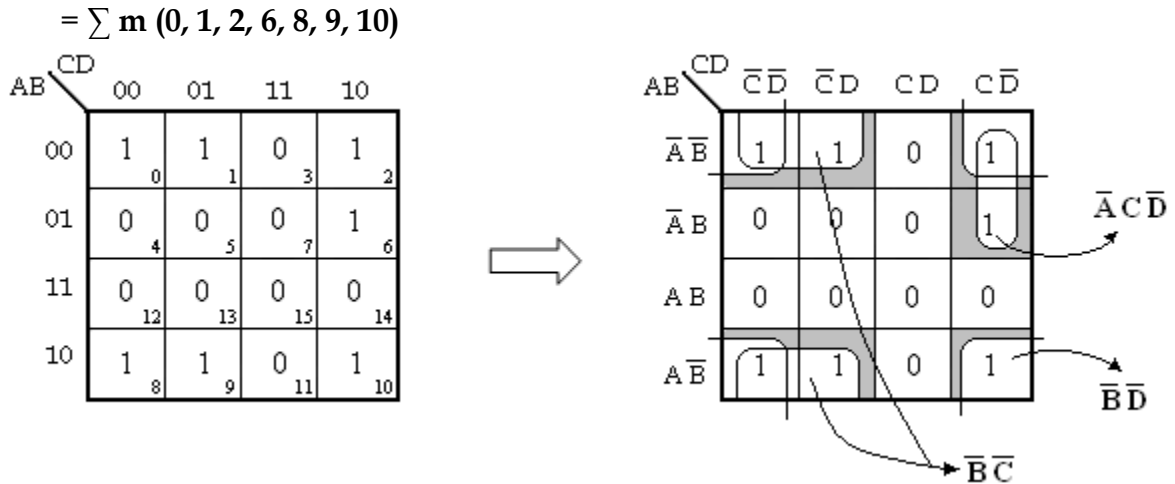
Therefore,

F= B′D′+ B′C′+ A′CD′.

4. **Y= ABCD+ AB′C′D′+ AB′C+ AB**
   = ABCD+ AB′C′D′+ AB′C (D+D′)+ AB (C+C′) (D+D′)
   = ABCD+ AB′C′D′+ AB′CD+ AB′CD′+ (ABC+ ABC′) (D+ D′)
   = <u>ABCD</u>+ AB′C′D′+ AB′CD+ AB′CD′+ <u>ABCD</u>+ ABCD′+ ABC′D+ ABC′D′
   = ABCD+ AB′C′D′+ AB′CD+ AB′CD′+ ABCD′+ ABC′D+ ABC′D′
   = m_{15}+ m_8+ m_{11}+ m_{10}+ m_{14}+ m_{13}+ m_{12}
   = $\sum m\ (8, 10, 11, 12, 13, 14, 15)$
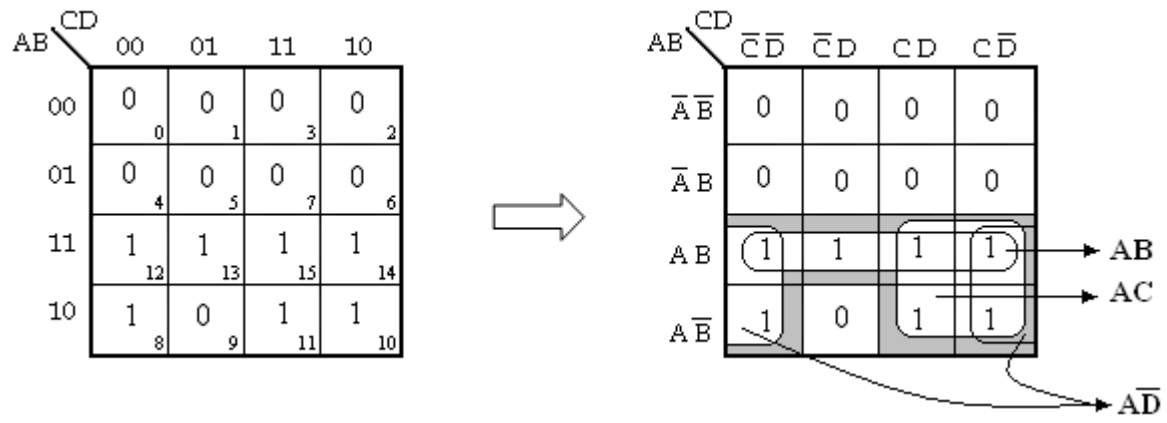
Therefore,

Y= AB+ AC+ AD′.

5. **Y (A, B, C, D)= $\sum m$ (7, 9, 10, 11, 12, 13, 14, 15)**
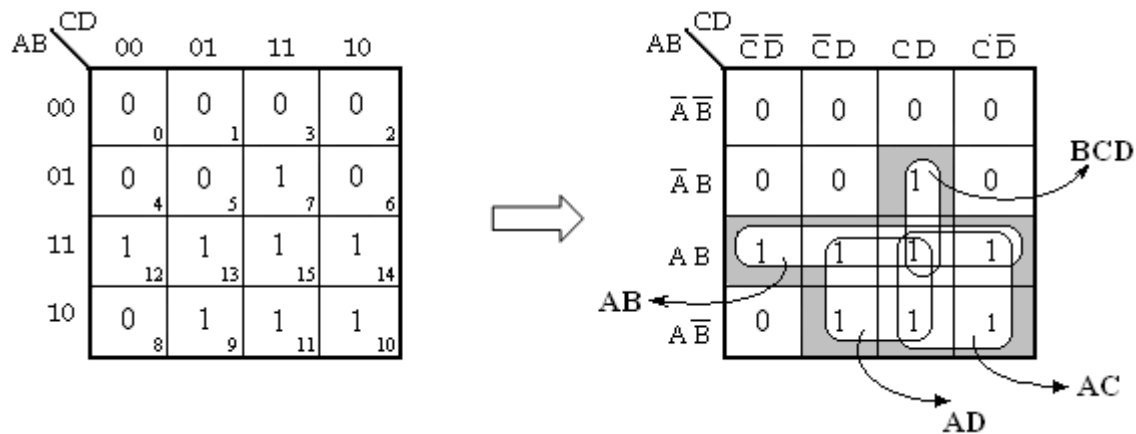
Therefore,

Y= AB+ AC+ AD+BCD.

6. Y= A'B'C'D+ A'BC'D+ A'BCD+ A'BCD'+ ABC'D+ ABCD+ AB'CD

$= m_1 + m_5 + m_7 + m_6 + m_{13} + m_{15} + m_{11}$

$= \sum m$ **(1, 5, 6, 7, 11, 13, 15)**
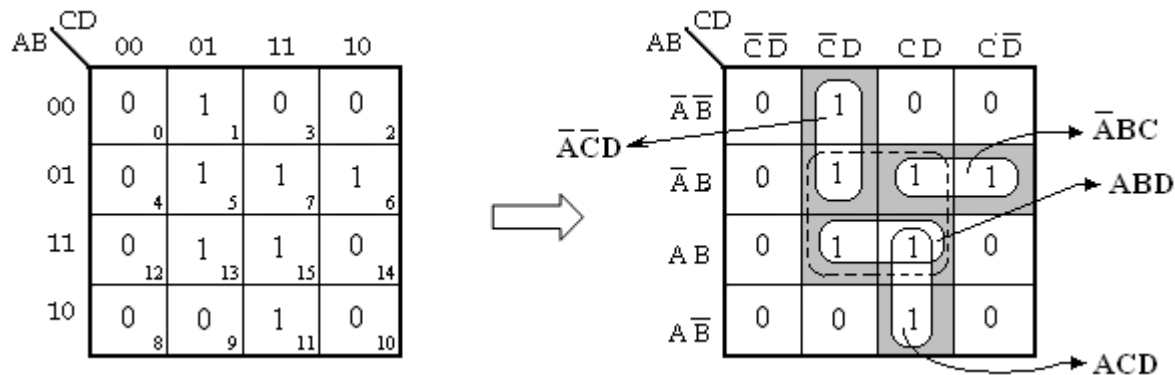
In the above K-map, the cells 5, 7, 13 and 15 can be grouped to form a quad as indicated by the dotted lines. In order to group the remaining 1's, four pairs have to be formed. However, all the four 1's covered by the quad are also covered by the pairs. So, the quad in the above k-map is redundant.

Therefore, the simplified expression will be,

Y = A'C'D+ A'BC+ ABD+ ACD.

7. **Y= $\sum m$ (1, 5, 10, 11, 12, 13, 15)**
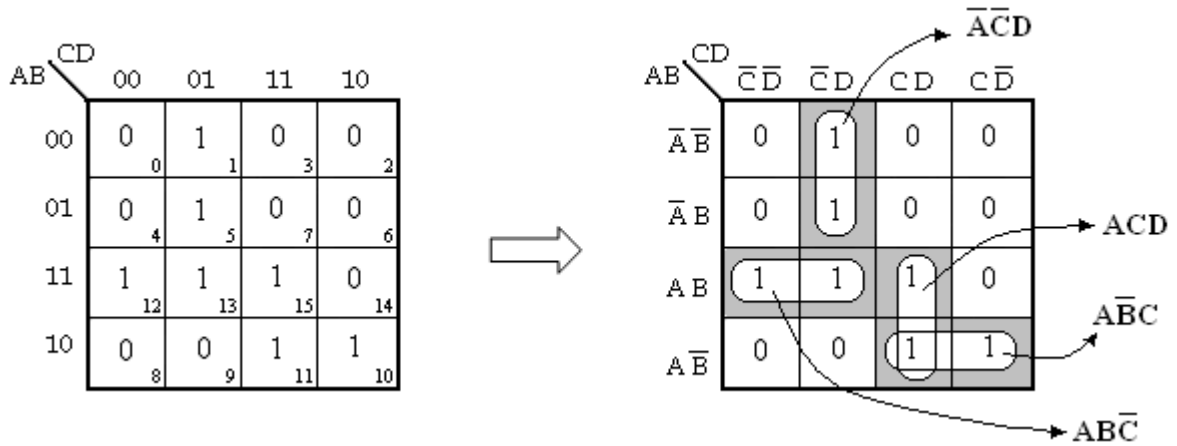
Therefore, **Y= A'C'D+ ABC'+ ACD+ AB'C.**

8. **Y= A'B'CD'+ ABCD'+ AB'CD'+ AB'CD+ AB'C'D'+ ABC'D'+ A'B'CD+ A'B'C'D'**
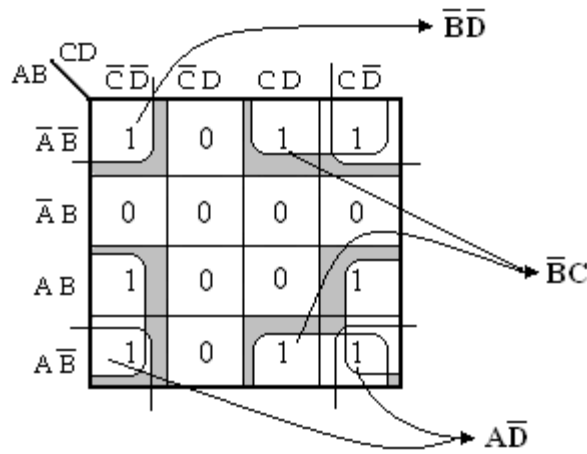
Therefore, **Y= AD'+ B'C+ B'D'**

9. **F (A, B, C, D) = $\sum$ m (0, 1, 4, 8, 9, 10)**
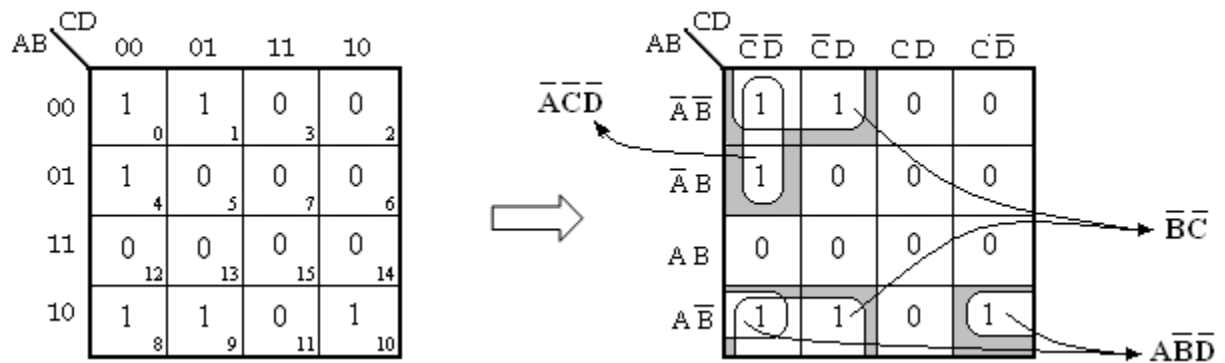
Therefore, **F= A'C'D'+ AB'D'+ B'C'.**

Simplification of Sum of Products Expressions: (Minimal Sums)

1. **Y= (A+ B+ C′) (A+ B′+ C′) (A′+ B′+ C′) (A′+ B+ C) (A+ B+ C)**

   = $M_1 . M_3 . M_7 . M_4 . M_0$

   =$\prod M (0, 1, 3, 4, 7)$

   = $\sum m (2, 5, 6)$
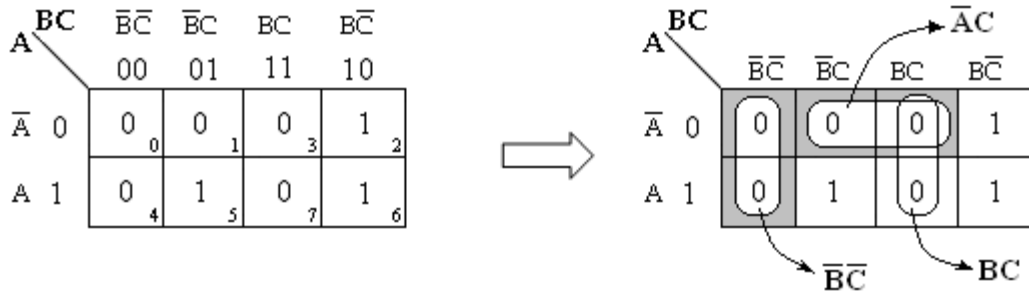
Y′ = B′C′+ A′C+ BC.

Y= Y″ = (B′C′+ A′C+ BC)′

        = (B′C′)′. (A′C)′. (BC)′

        = (B″+ C″). (A″+C′). (B′+ C′)

   Y = (B+ C). (A+C′). (B′+ C′)

2. **Y= (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′)

   = $M_{12} . M_{14} . M_{15} . M_8 . M_6 . M_7 . M_0 . M_{13}$

   = $\prod M (0, 6, 7, 8, 12, 13, 14, 15)$
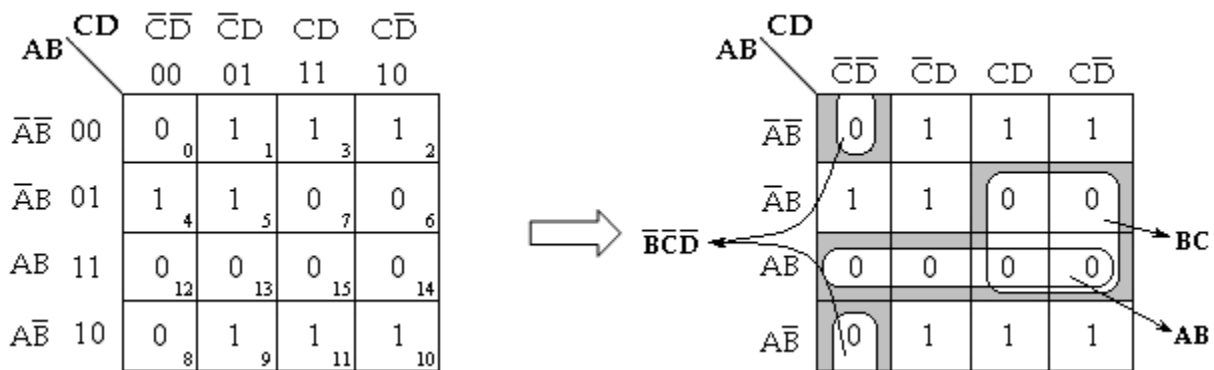
Y′ = B′C′D′+ AB+ BC

Y= Y″ = (B′C′D′+ AB+ BC)′

        = (B′C′D′)′. (AB)′. (BC)′

        = (B″+ C″+D″). (A′+B′). (B′+ C′)

        = (B+ C+ D). (A′+ B′). (B′+ C′)

Therefore, **Y= (B+ C+ D). (A′+ B′). (B′+ C′)**

3. **F(A, B, C, D)= $\prod$M (0, 2, 3, 8, 9, 12, 13, 14, 15)**
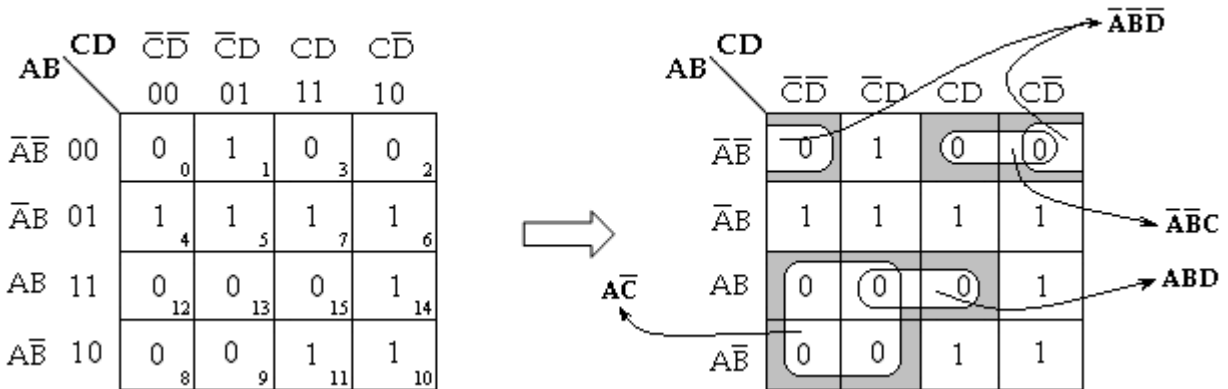
Y′ = A′B′D′+ A′B′C+ ABD+ AC′

Y= Y″ = (A′B′D′+ A′B′C+ ABD+ AC′)′
        = (A′B′D′)′. (A′B′C)′. (ABD)′. (AC′)′
        = (A″+ B″+ D″). (A″+ B″+C′). (A′+ B′+ D′). (A′+ C″)
        = (A+ B+ D). (A+ B+ C′). (A′+ B′+ D′). (A′+ C)

Therefore, **Y= (A+ B+ D). (A+ B+ C′). (A′+ B′+ D′). (A′+ C)**

4. **F(A, B, C, D)= $\sum$m (0, 1, 2, 5, 8, 9, 10)**
            = $\prod$M (3, 4, 6, 7, 11, 12, 13, 14, 15)
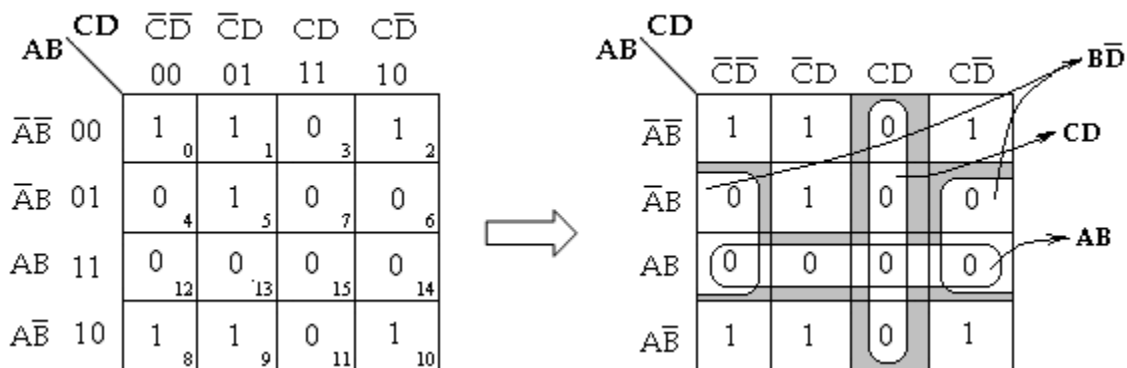
Y′ = BD′+ CD+ AB

Y= Y″ = (BD′+ CD+ AB)′
        = (BD′)′. (CD)′. (AB)′
        = (B′+ D″). (C′+ D′). (A′+ B′)
        = (B′+ D). (C′+ D′). (A′+ B′)

Therefore, **Y= (B′+ D). (C′+ D′). (A′+ B′)**

Don't care Conditions:

A don't care minterm is a combination of variables whose logical value is not specified. When choosing adjacent squares to simplify the function in a map, the don't care minterms may be assumed to be either 0 or 1. When simplifying the function, we can choose to include each don't care minterm with either the 1's or the 0's, depending on which combination gives the simplest expression.

1. $F(x, y, z) = \sum m(0, 1, 2, 4, 5) + \sum d(3, 6, 7)$
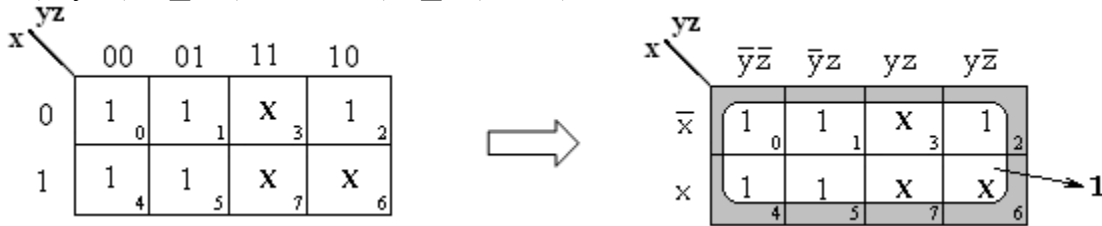
F (x, y, z) = 1

2. $F(w, x, y, z) = \sum m(1, 3, 7, 11, 15) + \sum d(0, 2, 5)$
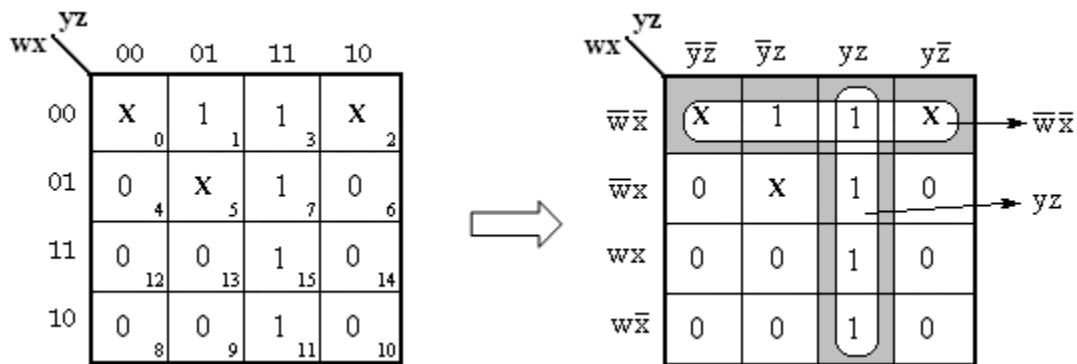
F (w, x, y, z) = w'x'+ yz

3. $F(w, x, y, z) = \sum m(0, 7, 8, 9, 10, 12) + \sum d(2, 5, 13)$
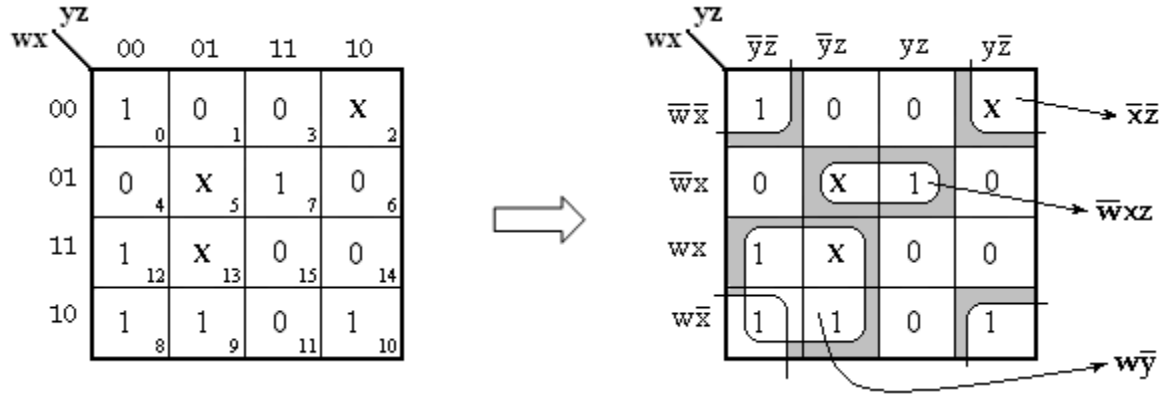
F (w, x, y, z) = w′xz+ wy′+ x′z′.
4. F (w, x, y, z) = ∑m (0, 1, 4, 8, 9, 10)+ ∑d (2, 11)
Soln:
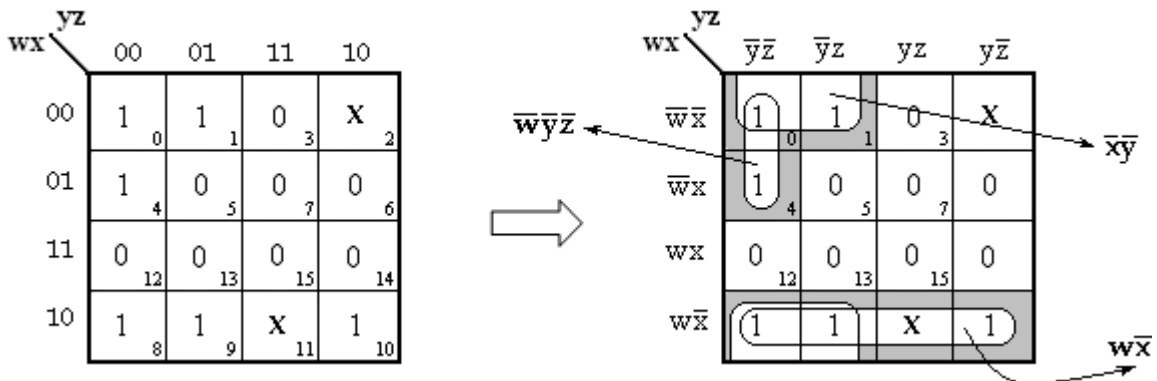
F (w, x, y, z) = wx′+ x′y′+ w′y′z′.

5. **F(A, B, C, D) = ∑m (0, 6, 8, 13, 14)+ ∑d (2, 4, 10)**
Soln:
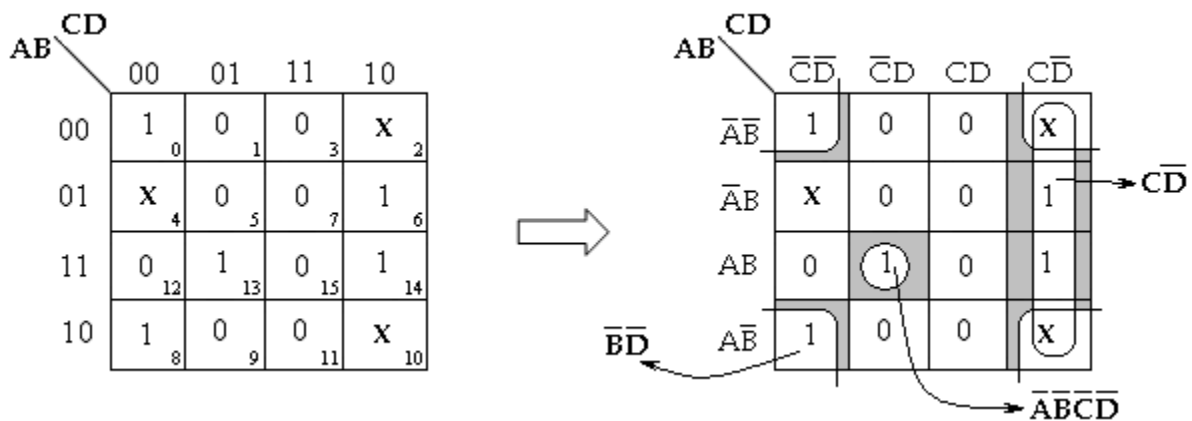
F(A, B, C, D) = CD′+ B′D′+ A′B′C′D′.

Five- Variable Maps:

A 5- variable K- map requires 25= 32 cells, but adjacent cells are difficult to identify on a single 32-cell map. Therefore, two 16 cell K-maps are used.

If the variables are A, B, C, D and E, two identical 16- cell maps containing B, C, D and E can be constructed. One map is used for A and other for A′.

In order to identify the adjacent grouping in the 5- variable map, we must imagine the two maps superimposed on one another ie., every cell in one map is adjacent to the corresponding cell in the other map, because only one variable changes between such corresponding cells.
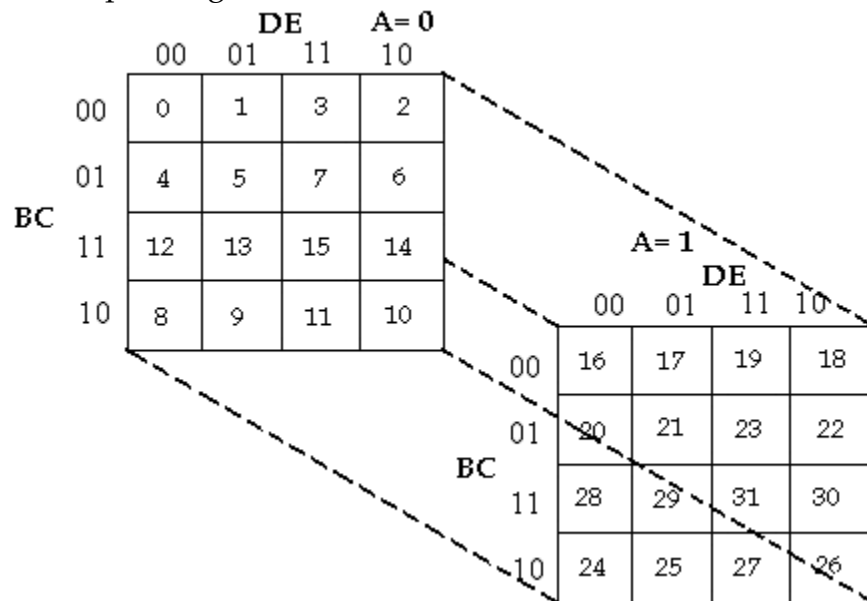
Five- Variable Karnaugh map (Layer Structure)

Thus, every row on one map is adjacent to the corresponding row (the one occupying the same position) on the other map, as are corresponding columns. Also,

the rightmost and leftmost columns within each 16- cell map are adjacent, just as they are in any 16- cell map, as are the top and bottom rows.
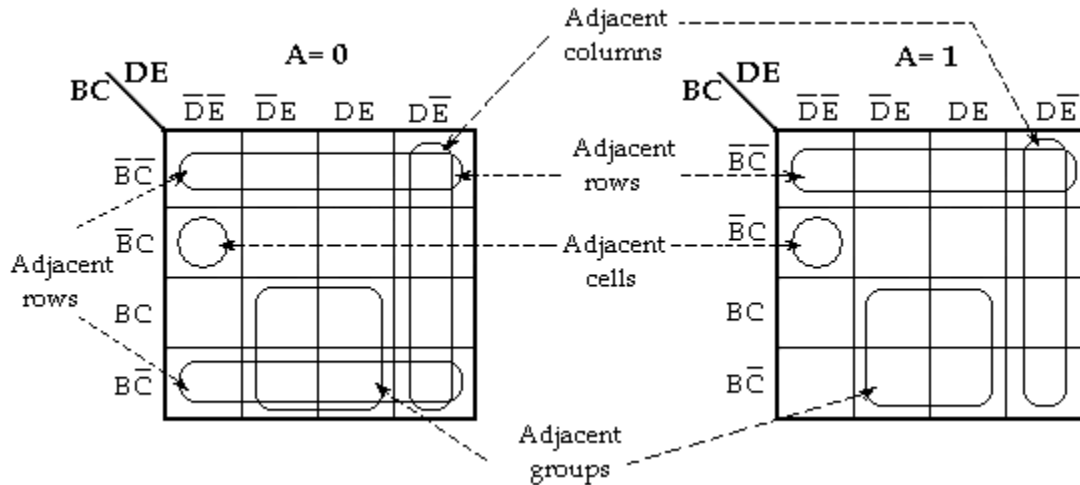
Typical subcubes on a five-variable map

However, the rightmost column of the map is not adjacent to the leftmost column of the other map.

1. Simplify the Boolean function
   F (A, B, C, D, E) = ∑m (0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31)
Soln:
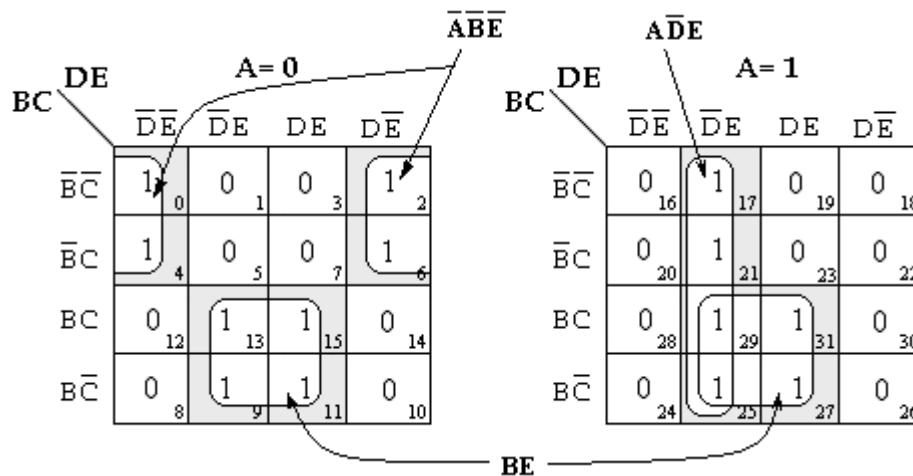
F (A, B, C, D, E) = A′B′E′+ BE+ AD′E

2. F (A, B, C, D, E) = ∑m (0, 5, 6, 8, 9, 10, 11, 16, 20, 24, 25, 26, 27, 29, 31)

Soln:
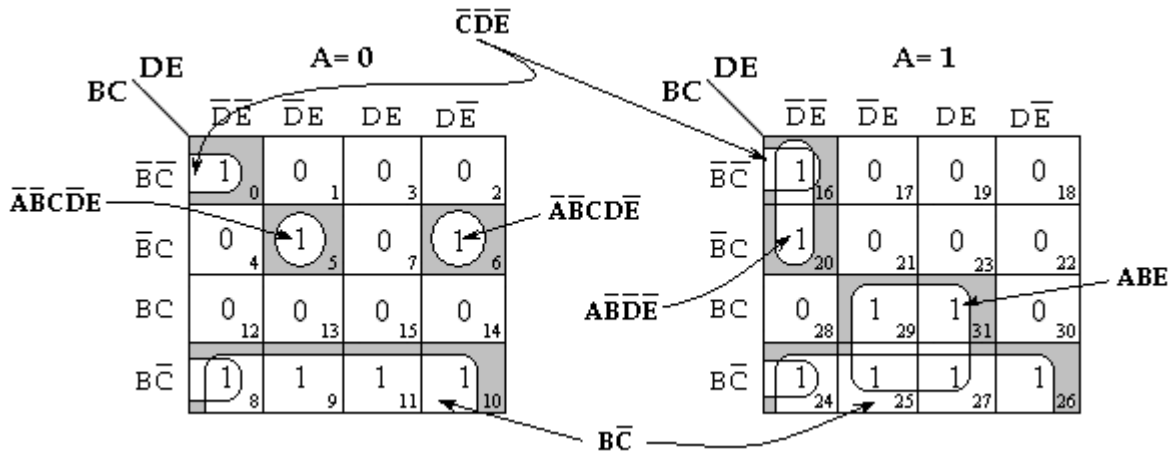
F (A, B, C, D, E) = C′D′E′+ A′B′CD′E+ A′B′CDE′+ AB′D′E′+ ABE+ BC′

3. F (A, B, C, D, E) = $\sum$m (1, 4, 8, 10, 11, 20, 22, 24, 25, 26)+$\sum$d (0, 12, 16, 17)
Soln:
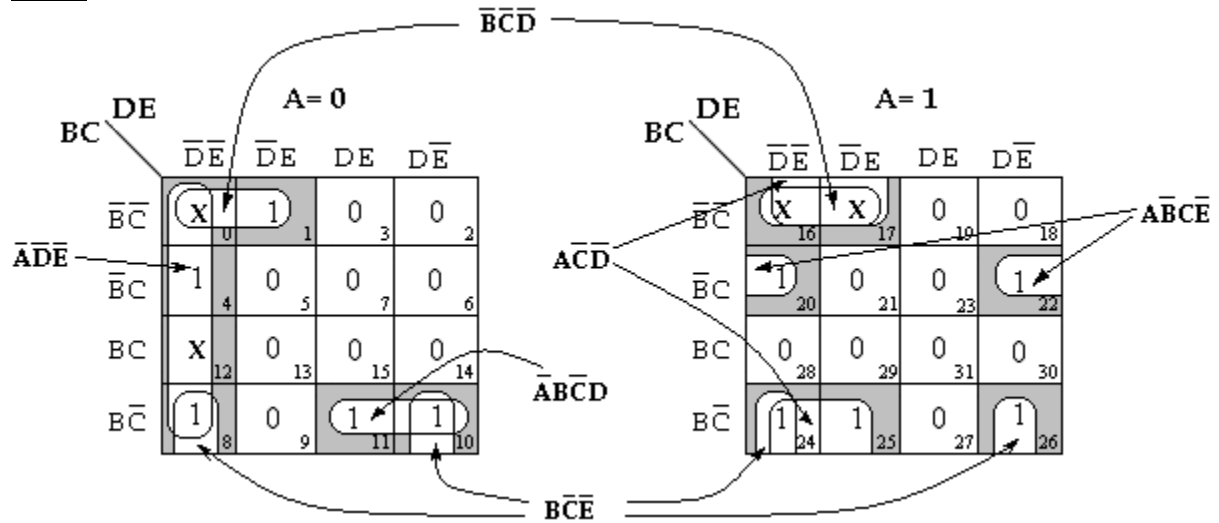
F (A, B, C, D, E) = B′C′D′+ A′D′E′+ BC′E′+ A′BC′D+ AC′D′+ AB′CE′

4. F (A, B, C, D, E) = $\sum$m (0, 1, 2, 6, 7, 9, 12, 28, 29, 31)
Soln:
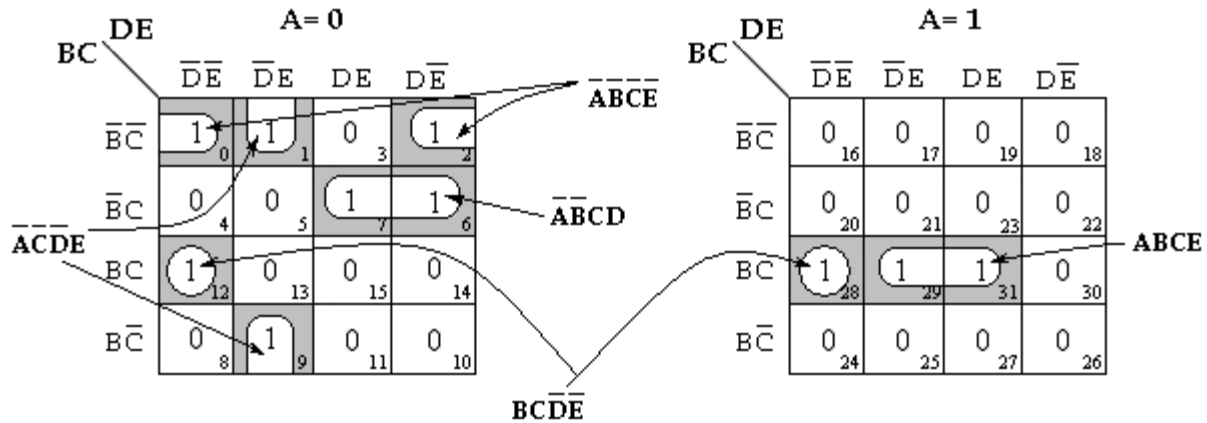
F (A, B, C, D, E) = BCD′E′+ ABCE+ A′B′C′E′+ A′C′D′E+ A′B′CD

5. $F(x_1, x_2, x_3, x_4, x_5) = \sum m\ (2, 3, 6, 7, 11, 12, 13, 14, 15, 23, 28, 29, 30, 31\ )$
Soln:
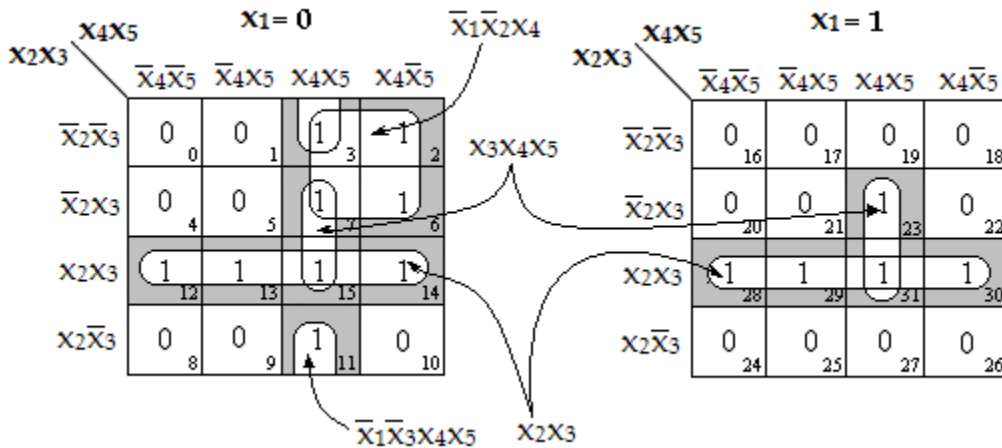
F $(x_1, x_2, x_3, x_4, x_5)$ = x2x3+ x3x4x5+ x1′x2′x4+ x1′x3′x4x5

6. $F(x_1, x_2, x_3, x_4, x_5) = \sum m\ (1, 2, 3, 6, 8, 9, 14, 17, 24, 25, 26, 27, 30, 31\ )+ \sum d\ (4, 5)$
Soln:
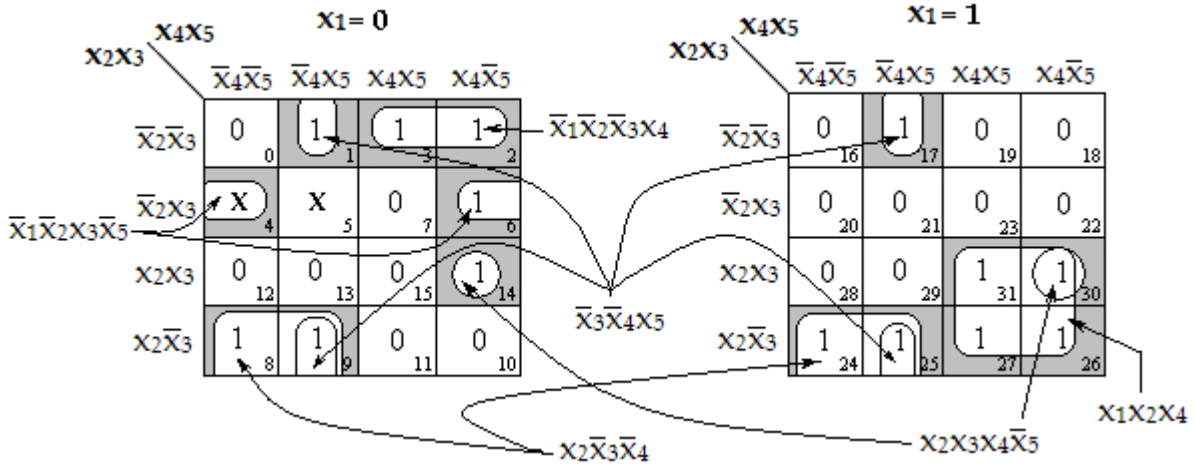
$$F\ (x_1,\ x_2,\ x_3,\ x_4,\ x_5) = x_2x_3'x_4' + x_2x_3x_4x_5' + x_3'x_4'x_5 + x_1x_2x_4 + x_1'x_2'x_3x_5' + x_1'x_2'x_3'x_4$$

QUINE-MCCLUSKEY (OR) TABULAR METHOD:

The K-map is a very effective tool for minimization of logic functions with 4 or less variables. For logic expressions with more than 4 variables, the visualization of adjacent cells and drawing of the K-map becomes more difficult.

The Quine-McCluskey method, also known as the tabular method, can be employed in such cases to minimize switching functions. This method employs a systematic, step-by-step procedure to produce a simplified standard form of expression for a function with any number of variables.

The steps to be followed in Quine-McCluskey method are –
1. A set of all prime implicants of the function must be obtained.
2. From the set of all prime implicants, a set of essential implicants must be determined by preparing a prime implicants chart.
3. The minterms which are not covered by the essential implicants are taken into consideration and a minimum cover is obtained from the remaining prime implicants.

Selecting prime implicants:

1. List all minterms in the binary form.

2. Arrange the minterms according to the number of 1's.
3. Compare each binary number with every term in the adjacent next higher category and if they differ only by one position, put a check mark (✓) and copy the term in the next column with '—' in the position that they differed.
4. Apply the same process described in step 3 for the resultant column and continue these cycles until a single pass through cycle yields no further elimination of literals.
5. List all prime implicants.
6. Select the minimum number of prime implicants which must cover all the minterms.

1. Find the minimal SOP expression for the following function
   F (a, b, c, d) = $\sum$ m(0, 3, 4, 6, 11) + $\sum$ d(8, 10, 12, 13) using Quine-McCluskey method.
Soln:
<u>Step 1:</u>
   These minterms are represented in the binary form.

Minterm	a	b	c	d
0	0	0	0	0
3	0	0	1	1
4	0	1	0	0
6	0	1	1	0
11	1	0	1	1
8	1	0	0	0
10	1	0	1	0
12	1	1	0	0
13	1	1	0	1

Binary representation of minterms

Step 2:

The above binary representations are grouped into a number of sections in terms of the number of 1's.

No of 1's	Minterm	a	b	c	d
0	0 ✔	0	0	0	0
1	4 ✔	0	1	0	0
	8 ✔	1	0	0	0
2	3 ✔	0	0	1	1
	6 ✔	0	1	1	0
	10 ✔	1	0	1	0
	12 ✔	1	1	0	0
3	11 ✔	1	0	1	1
	13 ✔	1	1	0	1

Group of minterms for different number of 1's

Step 3:

Minterm Combination	a	b	c	d
(0, 4) ✔	0	-	0	0
(0, 8) ✔	-	0	0	0
(4, 6)	0	1	-	0
(4, 12) ✔	-	1	0	0
(8, 10)	1	0	-	0
(8, 12) ✔	1	-	0	0
(3, 11)	-	0	1	1
(10, 11)	1	0	1	-
(12, 13)	1	1	0	-

2-cell combinations

Step 4:

From the two-cell combinations, one variable and a dash (-) in the same position can be combined to form 4-cell combination.

Minterm Combination	a	b	c	d
(0, 4, 8, 12)	-	-	0	0

4-cell combination

Step 5:

The column having only one cross mark corresponds to the essential prime implicants. A tick mark is put against every column which has only one cross mark. A star mark is put against every essential prime implicants.

Prime Implicants		0	3	4	6	11	8	10	12	13
a'bd' *	(4, 6)			X	⊗					
ab'd'	(8, 10)						X	X		
b'cd *	(3, 11)		⊗			X				
ab'c	(10, 11)					X		X		
abc'	(12, 13)								X	X
c'd' *	(0, 4, 8, 12)	⊗		X			X		X	
		✓	✓		✓					

F (a, b, c, d) = a'bd'+ b'cd+ c'd'

2. Find the minimal SOP expression for the following function
   F (w, x, y, z) = $\sum$ m (0, 1, 2, 3, 10, 11, 12, 13, 14, 15) using Quine-McCluskey method.

Soln:

Step 1:

These minterms are represented in the binary form.

Minterm	w	x	y	z
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	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

Binary representation of minterms

Step 2:

The above binary representations are grouped into a number of sections in terms of the number of 1's.

No of 1's	Minterm	w	x	y	z
0	0 ✔	0	0	0	0
1	1 ✔	0	0	0	1
	2 ✔	0	0	1	0
2	3 ✔	0	0	1	1
	10 ✔	1	0	1	0
	12 ✔	1	1	0	0
3	11 ✔	1	0	1	1
	13 ✔	1	1	0	1
	14 ✔	1	1	1	0
4	15 ✔	1	1	1	1

Group of minterms for different number of 1's

Step 3:

Minterm Combination	w	x	y	z
(0, 1) ✔	0	0	0	-
(0, 2) ✔	0	0	-	0
(1, 3) ✔	0	0	-	1
(2, 3) ✔	0	0	1	-
(2, 10) ✔	-	0	1	0
(3, 11) ✔	-	0	1	1
(10, 11) ✔	1	0	1	-
(10, 14) ✔	1	-	1	0
(12, 13) ✔	1	1	0	-
(12, 14) ✔	1	1	-	0

	w	x	y	z
(11, 15) ✓	1	-	1	1
(13, 15) ✓	1	1	-	1
(14, 15) ✓	1	1	1	-

2-cell combinations

Step 4:

From the two-cell combinations, one variable and a dash (-) in the same position can be combined to form 4-cell combination.

Minterm Combination	w	x	y	z
(0, 1, 2, 3)	0	0	-	-
(2, 3, 10, 11)	-	0	1	-
(10, 11, 14, 15)	1	-	1	-
(12, 13, 14, 15)	1	1	-	-

4-cell combinations

Step 5:

The column having only one cross mark corresponds to the essential prime implicants. A tick mark is put against every column which has only one cross mark. A star mark is put against every essential prime implicants.

Prime Implicants		0	1	2	3	10	11	12	13	14	15
w′x′ *	(0, 1, 2, 3)	⊗	x	x	x						
x′y *	(2, 3, 10, 11)			x	x	x	x				
wy	(10, 11, 14, 15)					x	x			x	x
wx *	(12, 13, 14, 15)							⊗	x	x	x
		✓	✓					✓	✓		

F (w, x, y, z) = w′x′+ wx+ (x′y or wy).

3. Find the minimal SOP expression for the following function

F (A, B, C, D) = $\sum$ m (1, 3, 4, 7, 8, 10, 11, 13, 15) using Quine-McCluskey method.

Soln:

<u>Step 1:</u>

These minterms are represented in the binary form.

Minterm	A	B	C	D
1	0	0	0	1
3	0	0	1	1
4	0	1	0	0
7	0	1	1	1
8	1	0	0	0
10	1	0	1	0
11	1	0	1	1
13	1	1	0	1
15	1	1	1	1

Binary representation of minterms

<u>Step 2:</u>

The above binary representations are grouped into a number of sections in terms of the number of 1's.

No of 1's	Minterm	A	B	C	D
1	1 ✔	0	0	0	1
	4	0	1	0	0
	8 ✔	1	0	0	0
2	3 ✔	0	0	1	1
	10 ✔	1	0	1	0
3	7 ✔	0	1	1	1
	11 ✔	1	0	1	1
	13 ✔	1	1	0	1
4	15 ✔	1	1	1	1

Group of minterms for different number of 1's

<u>Step 3:</u>

Minterm Combination	A	B	C	D
(1, 3)	0	0	-	1

(8, 10)	1	0	-	0
(3, 7) ✔	0	-	1	1
(3, 11) ✔	-	0	1	1
(10, 11)	1	0	1	-
(7, 15) ✔	-	1	1	1
(11, 15) ✔	1	-	1	1
(13, 15)	1	1	-	1

2-cell combinations

Step 4:

From the two-cell combinations, one variable and a dash (-) in the same position can be combined to form 4-cell combination.

Minterm Combination	A	B	C	D
(3, 7, 11, 15)	-	-	1	1

4-cell combination

Step 5:

The column having only one cross mark corresponds to the essential prime implicants. A tick mark is put against every column which has only one cross mark. A star mark is put against every essential prime implicants.

Prime Implicants		1	3	4	7	8	10	11	13	15
A'BC'D' *	4			⊗						
A'B'D *	(1, 3)	⊗	x							
AB'D' *	(8, 10)					⊗	x			
AB'C	(10, 11)						x	x		
ABD *	(13, 15)								⊗	x
CD *	(3, 7, 11, 15)		x		⊗			x		x
		✔		✔	✔	✔			✔	

F (A, B, C, D) = A'BC'D'+ A'B'D+ AB'D'+ ABD+ CD.

4. Find the minimal SOP expression for the following function
    F (A, B, C, D) = $\sum$ m (0, 1, 2, 8, 10, 11, 14, 15) using Quine-McCluskey method.

Soln:

Step 1:

These minterms are represented in the binary form.

Minterm	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
8	1	0	0	0
10	1	0	1	0
11	1	0	1	1
14	1	1	1	0
15	1	1	1	1

Binary representation of minterms

Step 2:

The above binary representations are grouped into a number of sections in terms of the number of 1's.

No of 1's	Minterm	A	B	C	D
0	0 ✔	0	0	0	0
1	1 ✔	0	0	0	1
	2 ✔	0	0	1	0
	8 ✔	1	0	0	0
2	10 ✔	1	0	1	0
3	11 ✔	1	0	1	1
	14 ✔	1	1	1	0
4	15 ✔	1	1	1	1

Group of minterms for different number of 1's

Step 3:

Minterm Combination		A	B	C	D
(0, 1)		0	0	0	-
(0, 2)	✓	0	0	-	0
(0, 8)	✓	-	0	0	0
(2, 10)	✓	-	0	1	0
(8, 10)	✓	1	0	-	0
(10, 11)	✓	1	0	1	-
(10, 14)	✓	1	-	1	0
(11, 15)	✓	1	-	1	1
(14, 15)	✓	1	1	1	-

2-cell combinations

Step 4:

From the two-cell combinations, one variable and a dash (-) in the same position can be combined to form 4-cell combination.

Minterm Combination	A	B	C	D
(0, 2, 8, 10)	-	0	-	0
(10, 11, 14, 15)	1	-	1	-

4-cell combinations

Step 5:

The column having only one cross mark corresponds to the essential prime implicants. A tick mark is put against every column which has only one cross mark. A star mark is put against every essential prime implicants.

Prime Implicants			0	1	2	8	10	11	14	15
A'B'C'	*	(0, 1)	x	⊗						
B'D'	*	(0, 2, 8, 10)	x		⊗	x	x			
AC	*	(10, 11, 14, 15)					x	⊗	x	x
				✓	✓	✓		✓	✓	✓

F (A, B, C, D) = A′B′C′+ B′D′+ AC.

5. Find the minimal SOP expression for the following function
   F (v, w, x, y, z) = $\sum$ m (0, 1, 3, 5, 6, 9, 11, 14, 21, 23, 24, 31) + $\sum$ d (25, 30) using Quine-McCluskey method.

Soln:

Step 1:

These minterms are represented in the binary form.

Minterm	v	w	x	y	z
0	0	0	0	0	0
1	0	0	0	0	1
3	0	0	0	1	1
5	0	0	1	0	1
6	0	0	1	1	0
9	0	1	0	0	1
11	0	1	0	0	0
14	0	1	1	1	1
21	1	0	1	1	0
23	1	0	1	0	1
24	1	1	0	0	1
31	1	1	1	1	1
25	1	1	0	1	0
30	1	1	1	1	1

Binary representation of minterms

Step 2:

The above binary representations are grouped into a number of sections in terms of the number of 1's.

No of 1's	Minterm	v	w	x	y	z
0	0 ✓	0	0	0	0	0
1	1 ✓	0	0	0	0	1

2	3 ✓	0	0	0	1	1
	5 ✓	0	0	1	0	1
	6 ✓	0	0	1	1	0
	9 ✓	0	1	0	0	1
	24 ✓	1	1	0	0	0
3	11 ✓	0	1	0	1	1
	14 ✓	0	1	1	1	0
	21 ✓	1	0	1	0	1
	25 ✓	1	1	0	0	1
4	23 ✓	1	0	1	1	1
	30 ✓	1	1	1	1	0
5	31 ✓	1	1	1	1	1

Group of minterms for different number of 1's

Step 3:

Minterm Combination	v	w	x	y	z
(0, 1)	0	0	0	0	-
(1, 3) ✓	0	0	0	-	1
(1, 5)	0	0	-	0	1
(1, 9) ✓	0	-	0	0	1
(3, 11) ✓	0	-	0	1	1
(5, 21)	-	0	1	0	1
(6, 14)	0	-	1	1	0
(9, 11) ✓	0	1	0	-	1
(9, 25)	-	1	0	0	1
(24, 25)	1	1	0	0	-
(14, 30)	-	1	1	1	0
(21, 23)	1	0	1	-	1
(23, 31)	1	-	1	1	1
(30, 31)	1	1	1	1	-

2-cell combinations

Step 4:

From the two-cell combinations, one variable and a dash (-) in the same position can be combined to form 4-cell combination.

Minterm Combination	v	w	x	y	z
(1, 3, 9, 11)	0	-	0	-	1

4-cell combination

Step 5:

The column having only one cross mark corresponds to the essential prime implicants. A tick mark is put against every column which has only one cross mark. A star mark is put against every essential prime implicants.

Prime Implicants		0	1	3	5	6	9	11	14	21	23	24	31	25	30
v′w′x′y′ *	(0, 1)	⊗	x												
v′w′y′z	(1, 5)		x		x										
w′xy′z *	(5, 21)				x					x					
v′xyz′ *	(6, 14)					⊗			x						
wx′y′z	(9, 25)						x							x	
vwx′y′ *	(24, 25)											⊗		x	
wxyz′	(14, 30)								x						x
vw′xz	(21, 23)									x	x				
vxyz *	(23, 31)										x		x		
vwxy	(30, 31)												x		x
v′x′z *	(1, 3, 9, 11)		x	⊗			x	⊗							
		✓		✓		✓		✓				✓			

F (v, w, x, y, z) = v′w′x′y′+ w′xy′z+ v′xyz′+ vwx′y′+ vxyz+ v′x′z

6. Find the minimal SOP expression for the following function
    F (A, B, C, D) = ∑ m (0, 2, 3, 6, 7, 8, 10, 12, 13) using Tabulation method.
Soln:
Step 1:
    List all minterms in the binary form.

Minterm	A	B	C	D
0	0	0	0	0
2	0	0	1	0
3	0	0	1	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
10	1	0	1	0
12	1	1	0	0
13	1	1	0	1

Binary representation of minterms

Step 2:

The above binary representations are grouped into a number of sections in terms of the number of 1's.

No of 1's	Minterm	A	B	C	D
0	0 ✔	0	0	0	0
1	2 ✔	0	0	1	0
	8 ✔	1	0	0	0
2	3 ✔	0	0	1	1
	6 ✔	0	1	1	0
	10 ✔	1	0	1	0
	12 ✔	1	1	0	0
3	7 ✔	0	1	1	1
	13 ✔	1	1	0	1

Group of minterms for different number of 1's

Step 3:

Minterm Combination	A	B	C	D
(0, 2) ✔	0	0	-	0
	-	0	0	0

(0, 8) ✔				
(2, 3) ✔	0	0	1	-
(2, 6) ✔	0	-	1	0
(2, 10) ✔	-	0	1	0
(8, 10) ✔	1	0	-	0
(8, 12)	1	-	0	0
(3, 7) ✔	0	-	1	1
(6, 7) ✔	0	1	1	-
(12, 13)	1	1	0	-

2-cell combinations

Step 4:

From the two-cell combinations, one variable and a dash (-) in the same position can be combined to form 4-cell combination.

Minterm Combination	A	B	C	D
(0, 2, 8, 10)	-	0	-	0
(2, 3, 6, 7)	0	-	1	-

4-cell combinations

Step 5:

The column having only one cross mark corresponds to the essential prime implicants. A tick mark is put against every column which has only one cross mark. A star mark is put against every essential prime implicants.

Prime Implicants		0	2	3	6	7	8	10	12	13
AC′D′	(8, 12)						x		x	
ABC′ *	(12,13)								x	⊗
B′D′ *	(0, 2, 8, 10)	⊗	x				x	x		
A′C *	(2, 3, 6, 7)		x	⊗	⊗	⊗				
		✔		✔	✔	✔				✔

F (A, B, C, D) = ABC′+ B′D′+ A′C

7. Find the minimal SOP expression for the following function
   F (A, B, C, D) = $\sum$ m (1, 4, 6, 7, 8, 9, 10, 11, 15) using Quine-McCluskey method.
Soln:

Step 1:

List all minterms in the binary form.

Minterm	A	B	C	D
1	0	0	0	1
4	0	1	0	0
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
15	1	1	1	1

Binary representation of minterms

Step 2:

The above binary representations are grouped into a number of sections in terms of the number of 1's.

No of 1's	Minterm	A	B	C	D
1	1 ✔	0	0	0	1
	4 ✔	0	1	0	0
	8 ✔	1	0	0	0
2	6 ✔	0	1	1	0
	9 ✔	1	0	0	1
	10 ✔	1	0	1	0
3	7 ✔	0	1	1	1
	11 ✔	1	0	1	1
4	15 ✔	1	1	1	1

Group of minterms for different number of 1's

Step 3:

Minterm Combination	A	B	C	D

(1, 9)		-	0	0	1
(4, 6)		0	1	-	0
(8, 9)	✓	1	0	0	-
(8, 10)	✓	1	0	-	0
(6, 7)		0	1	1	-
(9, 11)	✓	1	0	-	1
(10, 11)	✓	1	0	1	-
(7, 15)		-	1	1	1
(11, 15)		1	-	1	1

2-cell combinations

Step 4:

From the two-cell combinations, one variable and a dash (-) in the same position can be combined to form 4-cell combination.

Minterm Combination	A	B	C	D
(8, 9, 10,11)	1	0	-	-

4-cell combination

Step 5:

The column having only one cross mark corresponds to the essential prime implicants. A tick mark is put against every column which has only one cross mark. A star mark is put against every essential prime implicants.

Prime Implicants			1	4	6	7	8	9	10	11	15
B′C′D	*	(1, 9)	⊗					x			
A′BD′	*	(4, 6)		⊗	x						
A′BC		(6, 7)			x	x					
BCD	*	(7, 15)				x					x
ACD		(11, 15)								x	x
AB′	*	(8, 9, 10, 11)					⊗	x	⊗	x	
			✓	✓			✓		✓		

F (A, B, C, D) = B′C′D+ A′BD′+ BCD+ AB′

8.   Using Tabulation method, simplify the Boolean function,

$$F(w, x, y, z) = \sum m(1, 2, 3, 5, 9, 12, 14, 15) + \sum d(4, 8, 11).$$

Soln:

Step 1:

List all minterms in the binary form.

Minterm	w	x	y	z
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
5	0	1	0	1
9	1	0	0	1
12	1	1	0	0
14	1	1	1	0
15	1	1	1	1
4	0	1	0	0
8	1	0	0	0
11	1	0	1	1

Binary representation of minterms

Step 2:

The above binary representations are grouped into a number of sections in terms of the number of 1's.

No of 1's	Minterm	w	x	y	z
1	1 ✔	0	0	0	1
	2 ✔	0	0	1	0
	4 ✔	0	1	0	0
	8 ✔	1	0	0	0
2	3 ✔	0	0	1	1
	5 ✔	0	1	0	1
	9 ✔	1	0	0	1
	12 ✔	1	1	0	0
3	14 ✔	1	1	1	0
	11 ✔	1	0	1	1
4	15 ✔	1	1	1	1

Group of minterms for different number of 1's

Step 3:

Minterm Combination	w	x	y	z
(1, 3) ✓	0	0	-	1
(1, 5)	0	-	0	1
(1, 9) ✓	-	0	0	1
(2, 3)	0	0	1	-
(4, 5)	0	1	0	-
(4, 12)	-	1	0	0
(8, 9)	1	0	0	-
(8, 12)	1	-	0	0
(3, 11) ✓	-	0	1	1
(9, 11) ✓	1	0	-	1
(12, 14)	1	1	-	0
(11, 15)	1	-	1	1
(14, 15)	1	1	1	-

2-cell combinations

Step 4:

From the two-cell combinations, one variable and a dash (-) in the same position can be combined to form 4-cell combination.

Minterm Combination	w	x	y	z
(1, 3, 9, 11)	-	0	-	1

4-cell combination

Step 5:

The column having only one cross mark corresponds to the essential prime implicants. A tick mark is put against every column which has only one cross mark. A star mark is put against every essential prime implicants.

Using 2-cell and 4-cell combination, prime implicants table can be drawn as,

Prime Implicants			1	2	3	5	9	12	14	15	4	8	11
w′y′z	*	(1, 5)	x			x							
w′x′y	*	(2, 3)		⊗	x								
w′xy′		(4, 5)				x					x		
xy′z′		(4, 12)						x			x		
wx′y′		(8, 9)					x					x	
wy′z′		(8, 12)						x				x	
wxz′	*	(12, 14)						x	x				
wyz		(11, 15)								x			x
wxy	*	(14, 15)							x	x			
x′z	*	(1,3,9,11)	x		x		x						x
				✓									

F (w, x, y, z) = w′y′z+ w′x′y+ wxz′+ wxy+ x′z

9.   Using Tabulation method, simplify the Boolean function,
        F (w, x, y, z) = $\sum$ m (2, 3, 4, 6, 7, 11, 12, 13, 15) + $\sum$ d (1, 5, 15).

Soln:

Step 1:

List all minterms in the binary form.

Minterm	w	x	y	z
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
6	0	1	1	0
7	0	1	1	1
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
1	0	0	0	1
5	0	1	0	1
15	1	1	1	1

Binary representation of minterms

Step 2:

No of 1's	Minterm	w	x	y	z
1	2 ✔	0	0	1	0
	4 ✔	0	1	0	0
	1 ✔	0	0	0	1
2	3 ✔	0	0	1	1
	6 ✔	0	1	1	0
	12 ✔	1	1	0	0
	5 ✔	0	1	0	1
3	7 ✔	0	1	1	1
	11 ✔	1	0	1	1
	13 ✔	1	1	0	1
	14 ✔	1	1	1	0
4	15 ✔	1	1	1	1

Group of minterms for different number of 1's

Step 3:

Minterm Combination	w	x	y	z
(2, 3) ✔	0	0	1	-
(2, 6) ✔	0	-	1	0
(4, 6) ✔	0	1	-	0
(4, 12) ✔	-	1	0	0
(4, 5) ✔	0	1	0	-
(1, 3) ✔	0	0	-	1
(1, 5) ✔	0	-	0	1
(3, 7) ✔	0	-	1	1
(3, 11) ✔	-	0	1	1
(6, 7) ✔	0	1	1	-
(6, 14) ✔	-	1	1	0
(12, 13) ✔	1	1	0	-
	1	1	-	0

(12, 14) ✔		0	1	-	1
(5, 7) ✔		-	1	0	1
(5, 13) ✔					
(7, 15) ✔		-	1	1	1
(11, 15) ✔		1	-	1	1
(13, 15) ✔		1	1	-	1
(14, 15) ✔		1	1	1	-

2-cell combinations

Step 4:

From the two-cell combinations, one variable and a dash (-) in the same position can be combined to form 4-cell combination.

Minterm Combination		w	x	y	z
(2, 3, 6, 7)		0	-	1	-
(4, 6, 12, 14)	✔	-	1	-	0
(4, 6, 5, 7)	✔	0	1	-	-
(4, 12, 5, 13)	✔	-	1	0	-
(1, 3, 5, 7)		0	-	-	1
(3, 7, 11, 15)		-	-	1	1
(6, 7, 14, 15)	✔	-	1	1	-
(12, 13, 14, 15)	✔	1	1	-	-
(5, 7, 13, 15)	✔	-	1	-	1

4-cell combinations

From the two-cell combinations, one variable and a dash (-) in the same position can be combined to form 8-cell combination.

Minterm Combination	w	x	y	z
(4, 5, 6, 7, 12, 13, 14, 15)	-	1	-	-

8-cell combination

Step 5:

The column having only one cross mark corresponds to the essential prime implicants. A tick mark is put against every column which has only one cross mark. A star mark is put against every essential prime implicants.

Using 4-cell and 8-cell combination, prime implicants table can be drawn as,

Prime Implicants		2	3	4	6	7	11	12	13	14	1	5	15
w'y *	(1, 5)	⊗	x		x	x							
w'z	(2, 3)		x			x					x	x	
yz *	(4, 5)		x			x	⊗						x
x *	(4, 5, 6, 7, 12, 13, 14, 15)			⊗	x	x		⊗	⊗	⊗		x	x
		✓		✓			✓	✓	✓	✓	✓		

F (w, x, y, z) = w'y+ yz+ x.

10. Find the minimal SOP expression for the following function
F (v, w, x, y, z) = $\sum$ m (0, 1, 8, 11, 12, 15, 20, 21, 22, 24, 29, 31) + $\sum$ d (9, 18, 30) using Quine-McCluskey method.
Soln:
Step 1:

These minterms are represented in the binary form.

Minterm	v	w	x	y	z
0	0	0	0	0	0
1	0	0	0	0	1
8	0	1	0	0	0
11	0	1	0	1	1
12	0	1	1	0	0
15	0	1	1	1	1
20	1	0	1	0	0
21	1	0	1	0	1
22	1	0	1	1	0
24	1	1	0	0	0
29	1	1	1	0	1
31	1	1	1	1	1

9	0	1	0	0	1
18	1	0	0	1	0
30	1	1	1	1	0

Binary representation of minterms

Step 2:

The above binary representations are grouped into a number of sections in terms of the number of 1's.

No of 1's	Minterm	v	w	x	y	z
0	0 ✓	0	0	0	0	0
1	1 ✓	0	0	0	0	1
	8 ✓	0	1	0	0	0
2	12 ✓	0	1	1	0	0
	20 ✓	1	0	1	0	0
	24 ✓	1	1	0	0	0
	9 ✓	0	1	0	0	1
	18 ✓	1	0	0	1	0
3	11 ✓	0	1	0	1	1
	21 ✓	1	0	1	0	1
	22 ✓	1	0	1	1	0
4	15 ✓	0	1	1	1	1
	29 ✓	1	1	1	0	1
	30 ✓	1	1	1	1	0
5	31 ✓	1	1	1	1	1

Group of minterms for different number of 1's

Step 3:

Minterm Combination	v	w	x	y	z
(0, 1) ✓	0	0	0	0	-
(0, 8) ✓	0	-	0	0	0
(1, 9) ✓	0	-	0	0	1
(8, 12)	0	1	-	0	0
(8, 24)	-	1	0	0	0

(8, 9) ✓	0	1	0	0	-
(20, 21)	1	0	1	0	-
(20, 22)	1	0	1	-	0
(9, 11)	0	1	0	-	1
(18, 22)	1	0	-	1	0
(11, 15)	0	1	-	1	1
(21, 29)	1	-	1	0	1
(22, 30)	1	-	1	1	0
(15, 31)	-	1	1	1	1
(29, 31)	1	1	1	-	1
(30, 31)	1	1	1	1	-

2-cell combinations

Step 4:

From the two-cell combinations, one variable and a dash (-) in the same position can be combined to form 4-cell combination.

Minterm Combination	v	w	x	y	z
(0, 1, 8, 9)	0	-	0	0	-

4-cell combination

Step 5:

Using 2-cell and 4-cell combination, prime implicants table can be drawn as,

Prime Implicants		0	1	8	11	12	15	20	21	22	24	29	31	9	18	30
v'wy'z' *	(8, 12)			x		⊗										
wx'y'z' *	(8, 24)			x							⊗					
v'wx'z'	(9, 11)				x									x		
vw'yz'	(18, 22)									x					x	
vw'xy' *	(20, 21)							x	x							
vw'xz' *	(20, 22)							x		x						
v'wyz *	(11, 15)				x		x									
vxy'z	(21, 29)								x			x				
vxyz'	(22, 30)									x						x
wxyz	(15, 31)						x						x			

vwxz	*	(29, 31)											x	x			
vwxy		(30, 31)												x			x
v'x'y'	*	(0,1,8,9)	⊗	⊗	x										x		
			✓	✓		✓				✓							
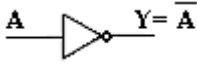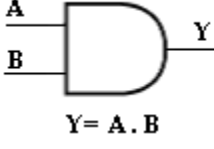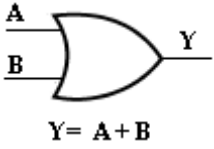
F (v, w, x, y, z) = v'wy'z'+ wx'y'z'+ vw'xy'+ vw'xz'+ v'wyz+ vwxz+ v'x'y'.
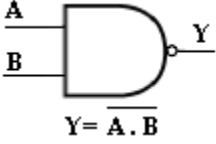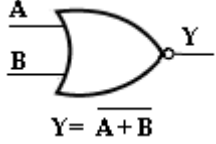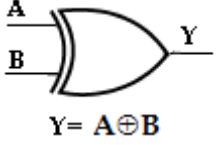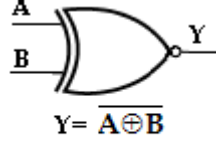
LOGIC GATES

BASIC LOGIC GATES:

Logic gates are electronic circuits that can be used to implement the most elementary logic expressions, also known as Boolean expressions. The logic gate is the most basic building block of combinational logic.

There are three basic logic gates, namely the OR gate, the AND gate and the NOT gate. Other logic gates that are derived from these basic gates are the NAND gate, the NOR gate, the EXCLUSIVE- OR gate and the EXCLUSIVE-NOR gate.

GATE	SYMBOL	OPERATION	TRUTH TABLE
NOT (7404)	A ▷o— $Y=\overline{A}$	NOT gate (Invertion), produces an inverted output pulse for a given input pulse.	A | $Y=\overline{A}$ 0 | 1 1 | 0
AND (7408)	A, B →[ ]→ Y, $Y= A.B$	AND gate performs logical **multiplication**. The output is HIGH only when all the inputs are HIGH. When any of the inputs are low, the output is LOW.	A | B | $Y= A.B$ 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1
OR (7432)	A, B →⟩→ Y, $Y= A+B$	OR gate performs logical **addition**. It produces a HIGH on the output when any of the inputs are HIGH. The output is LOW only when all inputs are LOW.	A | B | $Y= A+B$ 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1

		It is a universal gate. When any of the inputs are LOW, the output will be HIGH. LOW output occurs only when all inputs are HIGH.	A	B	$Y= \overline{A.B}$
NAND (7400)	$Y= \overline{A.B}$		0	0	1
			0	1	1
			1	0	1
			1	1	0
		It is a universal gate. LOW output occurs when any of its input is HIGH. When all its inputs are LOW, the output is HIGH.	A	B	$Y= \overline{A+B}$
NOR (7402)	$Y= \overline{A+B}$		0	0	1
			0	1	0
			1	0	0
			1	1	0
		The output is HIGH only when odd number of inputs is HIGH.	A	B	$Y= A \oplus B$
EX- OR (7486)	$Y= A \oplus B$		0	0	0
			0	1	1
			1	0	1
			1	1	0
		The output is HIGH only when even number of inputs is HIGH. Or when all inputs are zeros.	A	B	$Y= A \odot B$
EX- NOR	$Y= \overline{A \oplus B}$ (or) $Y= A \odot B$		0	0	1
			0	1	0
			1	0	0
			1	1	1

UNIVERSAL GATES:

The NAND and NOR gates are known as universal gates, since any logic function can be implemented using NAND or NOR gates. This is illustrated in the following sections.

a)   **NAND Gate**:

The NAND gate can be used to generate the NOT function, the AND function, the OR function and the NOR function.

i) NOT function:

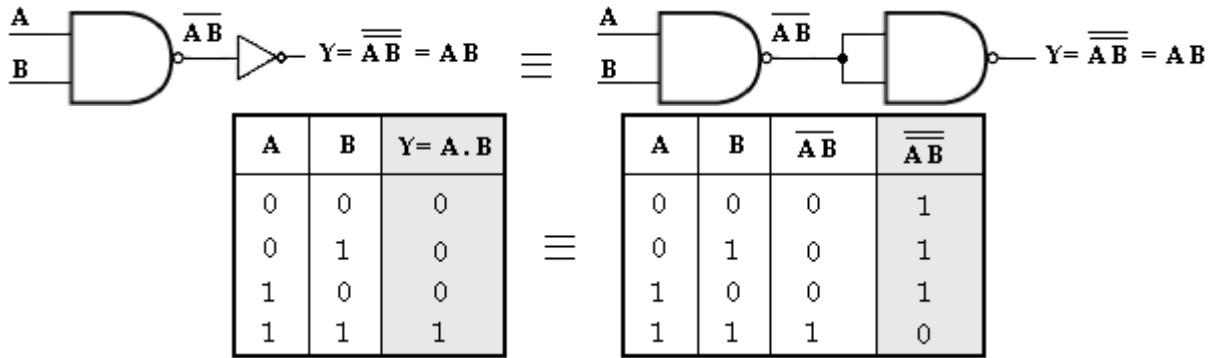By connecting all the inputs together and creating a single common input.

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

	A	B	Y = $\overline{A.B}$	
X=0	0	0	1	Y=1
	0	1	1	
	1	0	1	
X=1	1	1	0	Y=0
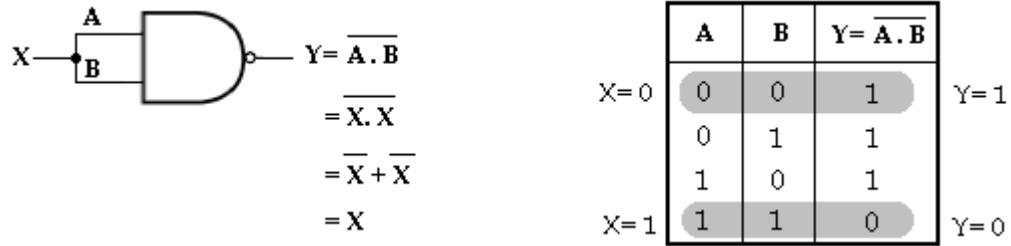
NOT function using NAND gate

ii) <u>AND function</u>:

By simply inverting output of the NAND gate. i.e.,

$$\overline{\overline{AB}} = AB$$

A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

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

AND function using NAND gates

iii) <u>OR function</u>:

By simply inverting inputs of the NAND gate. i.e.,
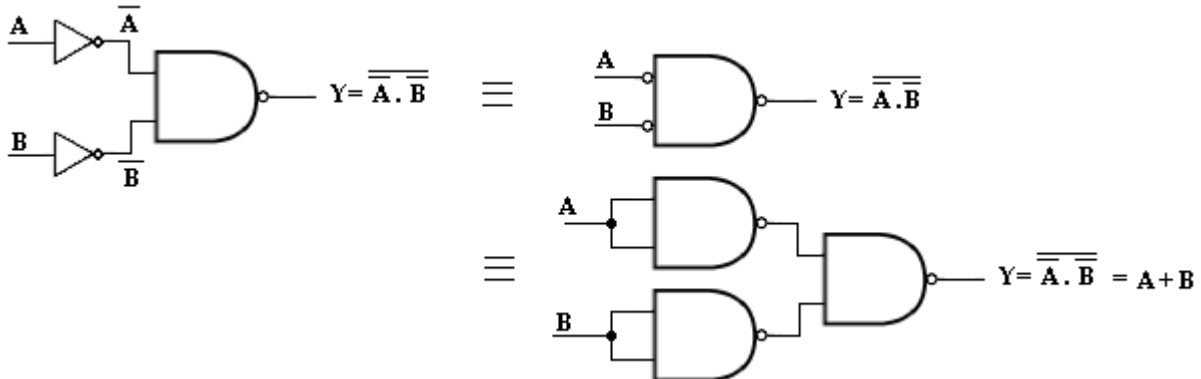
OR function using NAND gates

Bubble at the input of NAND gate indicates inverted input.

A	B	Y= A+B
0	0	0
0	1	1
1	0	1
1	1	1

$\equiv$

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

iv) <u>NOR function</u>**:**

By inverting inputs and outputs of the NAND gate.

$\equiv$
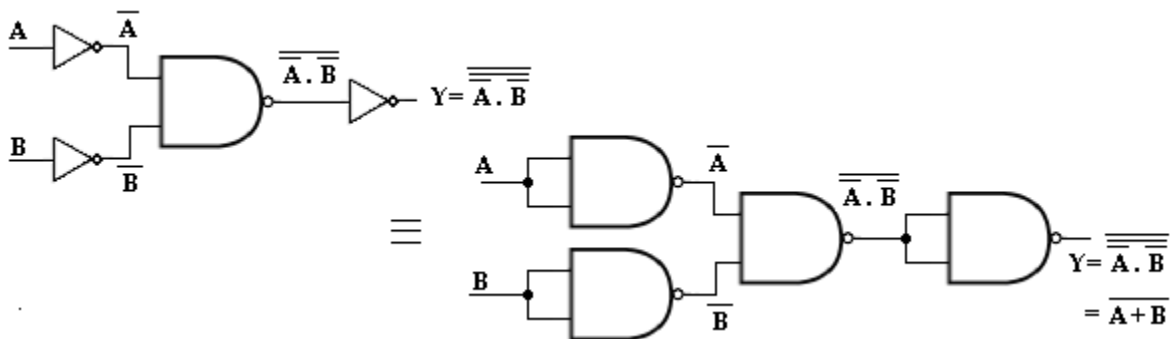
A	B	Y= $\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

$\equiv$

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

NOR function using NAND gates

b)   **<u>NOR Gate</u>**:

Similar to NAND gate, the NOR gate is also a universal gate, since it can be used to generate the NOT, AND, OR and NAND functions.

i) <u>NOT function</u>:

By connecting all the inputs together and creating a single common input.

A	B	Y= $\overline{A+B}$	
0	0	1	Y= 1
0	1	0	
1	0	0	
1	1	0	Y= 0
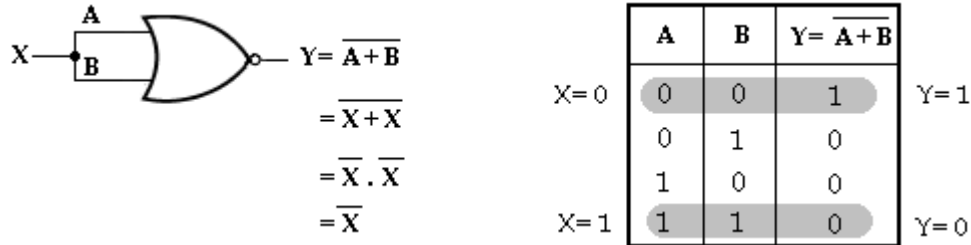
NOT function using NOR gates

ii) <u>OR function:</u>

By simply inverting output of the NOR gate. i.e.,

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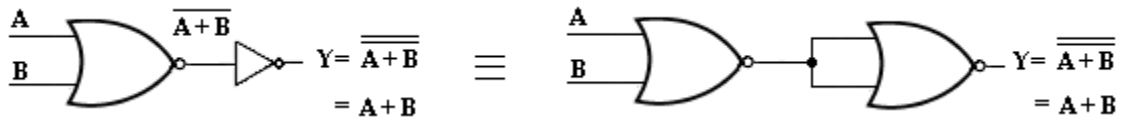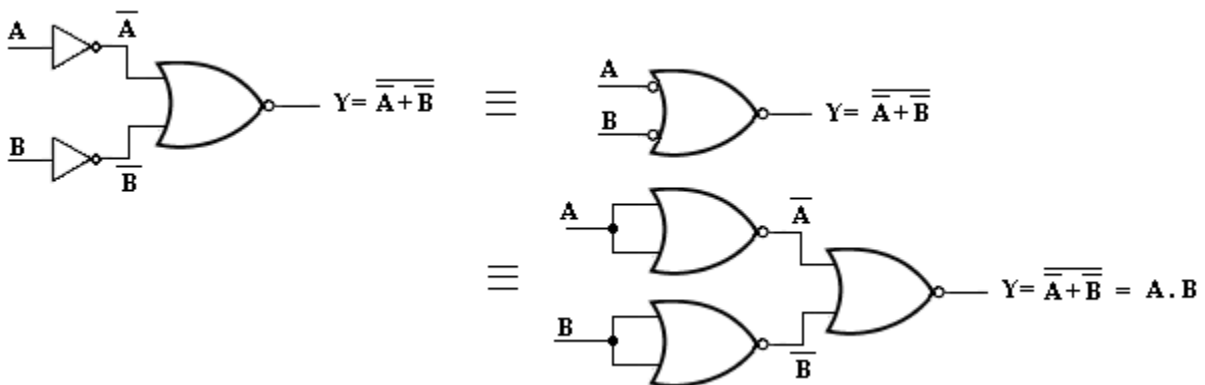

OR function using NOR gates

A	B	Y= A+B
0	0	0
0	1	1
1	0	1
1	1	1

≡

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

iii) <u>AND function:</u>

By simply inverting inputs of the NOR gate. i.e.,

AND function using NOR gates

Bubble at the input of NOR gate indicates inverted input.

A	B	Y= A . B		A	B	$\overline{A+B}$	$\overline{\overline{A+B}}$
0	0	0		0	0	1	0
0	1	0	≡	0	1	1	0
1	0	0		1	0	1	0
1	1	1		1	1	0	1

Truth table

iv) <u>NAND Function</u>:

By inverting inputs and outputs of the NOR gate.

NAND function using NOR gates

A	B	$Y= \overline{A . B}$		A	B	$\overline{A}+\overline{B}$	$\overline{\overline{A}+\overline{B}}$	$\overline{A+B}$
0	0	1		0	0	1	0	1
0	1	1	≡	0	1	1	0	1
1	0	1		1	0	1	0	1
1	1	0		1	1	0	1	0

<u>Conversion of AND/OR/NOT to NAND/NOR</u>:

1. Draw AND/OR logic.
2. If NAND hardware has been chosen, add bubbles on the output of each AND gate and bubbles on input side to all OR gates.
   If NOR hardware has been chosen, add bubbles on the output of each OR gate and bubbles on input side to all AND gates.
3. Add or subtract an inverter on each line that received a bubble in step 2.
4. Replace bubbled OR by NAND and bubbled AND by NOR.
5. Eliminate double inversions.
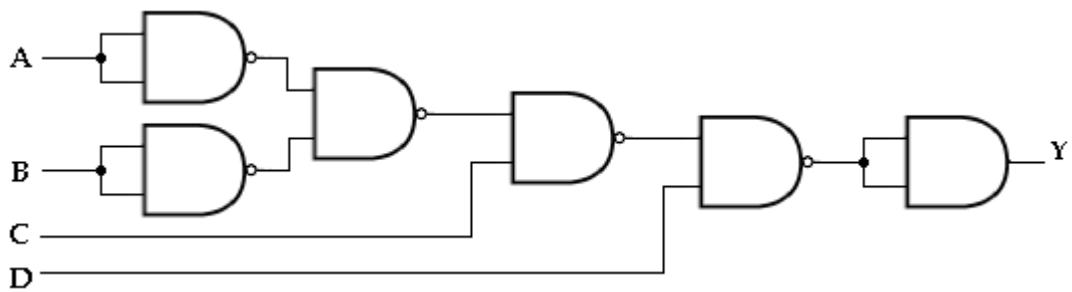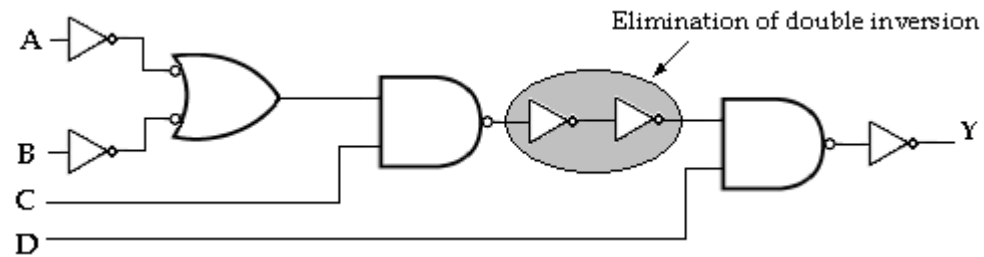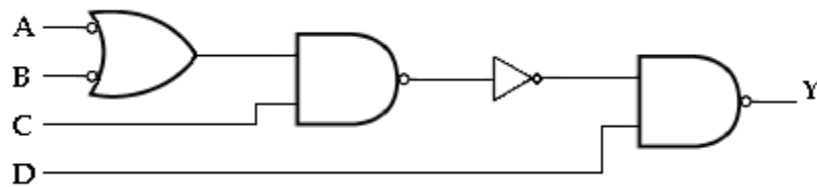

1.   Implement Boolean expression using NAND gates:
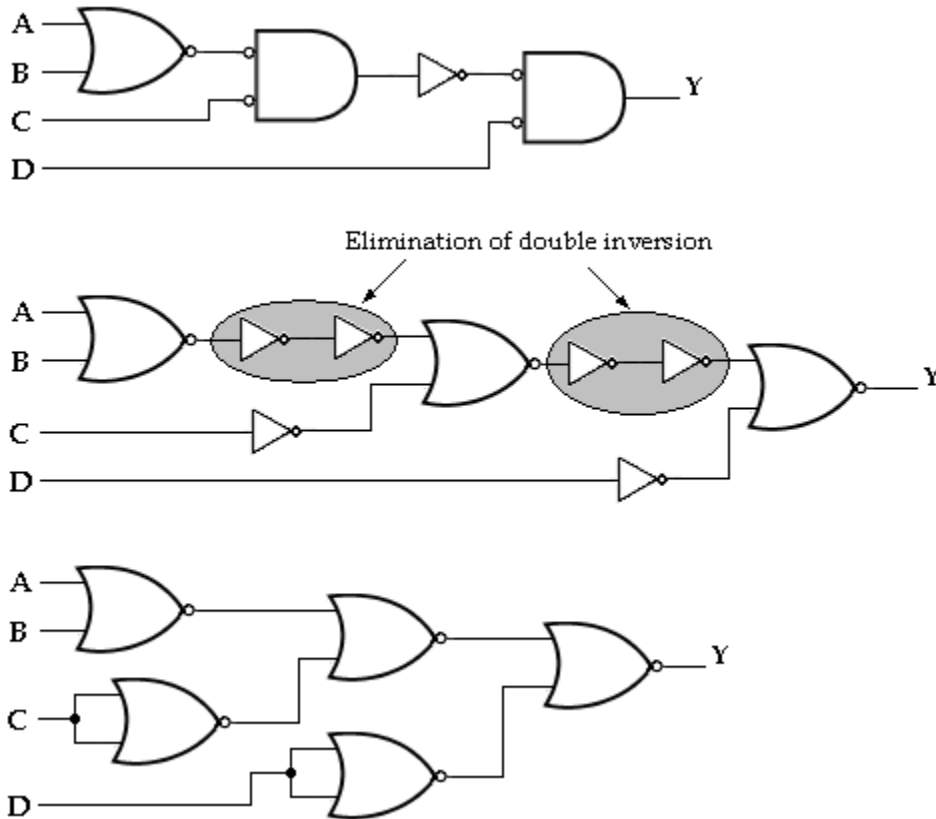
$\overline{((A+ B)\ C)}\ D$

<u>Original Circuit:</u>
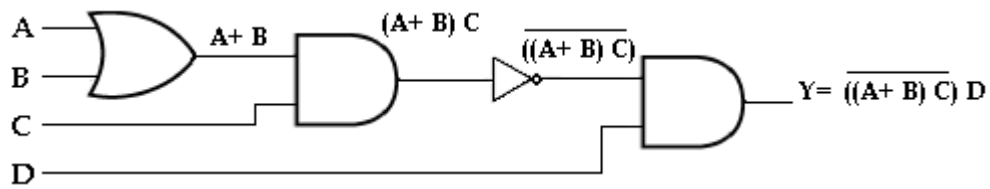
<u>**Soln**</u>:

<u>NAND Circuit:</u>

<u>NOR Circuit:</u>

Elimination of double inversion

2.   Implement Boolean expression for EX-OR gate using NAND gates.

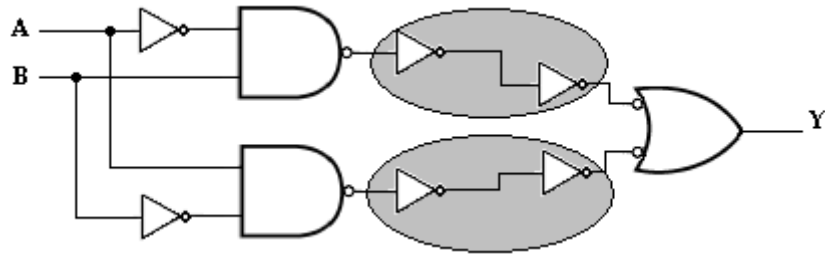
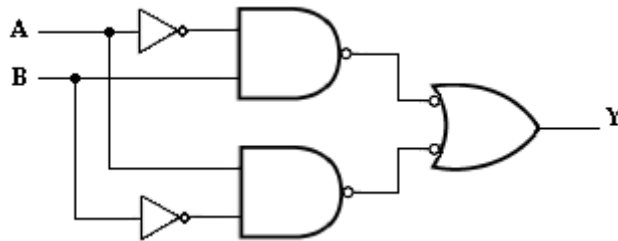$Y = \overline{A}B + A\overline{B}$
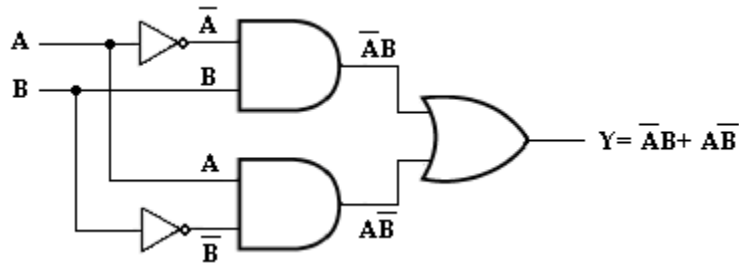
Soln:

Adding bubbles on the output of each AND gates and on the inputs of each OR gate.
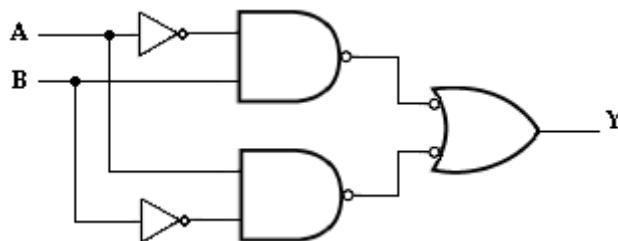
Adding an inverter on each line that received bubble,

Eliminating double inversion,

Replacing inverter and bubbled OR with NAND, we have

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

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