Digital Design and Computer Organization

Module - 1

Introduction to Digital Design :

Binary Logic :-

Binary Logic deals with variables that take on two discrete values and with operations that assume logical meaning. The two values the variables assume may be called by different names (true and false, yes and no, etc), but for our purpose, it is convenient to think in terms of bits and assign the values 1 and 0. The binary logic introduced in this section is equivalent to an algebra called Boolean algebra.

Definition of Binary Logic :-

Binary logic consists of binary variables and a set of logical operations. The variables are designated by letters of the alphabet, such as A, B, C, X, Y, Z etc., with each variable having two and only two distinct possible values : 1 and 0.

there are three basic logical operations : AND, OR and NOT. Each operation produces a binary result, denoted by 'Z'.

① AND :

This operation is represented by a dot or product of input variables.

Example : X and Y are the input variables

Z is the output variable

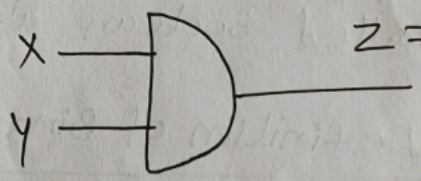
The output 'Z' = X · Y

(or)

Z = X AND Y

Truth Table : (AND Gate)

Input's		output
X	Y	Z = X · Y
0	0	0
0	1	0
1	0	0
1	1	1



Symbol of AND Gate.

② OR :

This operation is represented by a plus or addition of input variables.

Example: X and Y are the input variables
Z is the output variable

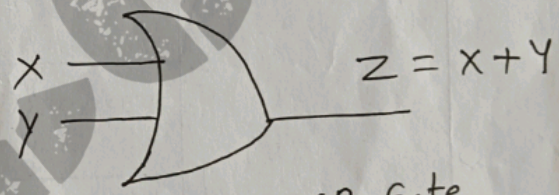
The output $Z = X + Y$

(or)

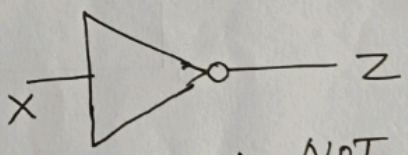
Z = X OR Y

Truth Table : (OR Gate)

Input's		output
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1



Symbol of OR Gate



Symbol of NOT Gate

③ NOT : This operation is represented by a prime or complement of input variable.

Example: X is the input & Z is the output

Truth Table:

I/P	O/P
X	Z
0	1
1	0

$Z = X'$ (or) $Z = \overline{X}$ (or) Z = NOT X.

Logic Gates :-

Logic gates are electronic circuits that operate on one or more input signals to produce an output signal. Electrical signals such as voltage or currents exist as analog signals having values over a given continuous range, say 0 to 3V, but in a digital system those voltages are interpreted to be either of two recognizable values, 0 or 1.

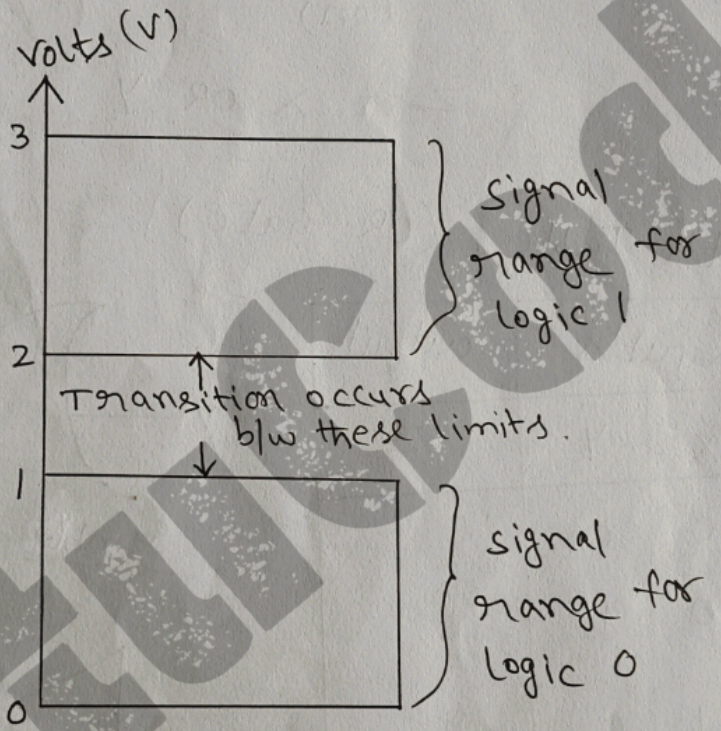


Fig. ① signal levels for Binary logic values.

In practice, each voltage level has an acceptable range, as shown in Fig ①. When the physical signal is in a particular range it is interpreted to be either a '0' or a '1'.

The input signals x and y in the AND and OR gates may exist in one of four possible states: 00, 10, 11, or 01. These input signals are shown in Fig. (2).

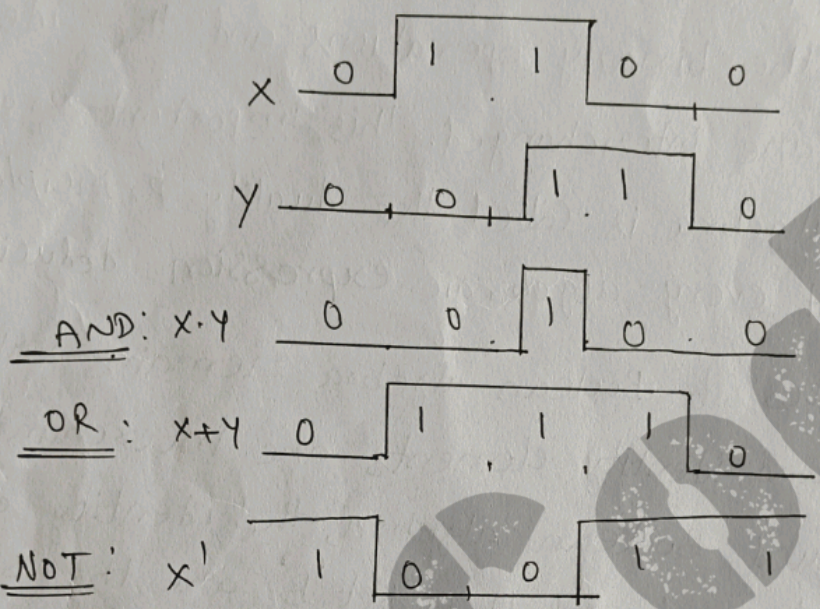


Fig. (2): Input-output signals for gates.

AND and OR gates may have more than two inputs. An AND gate with three inputs and an OR gate with four inputs are shown in Fig. (3).

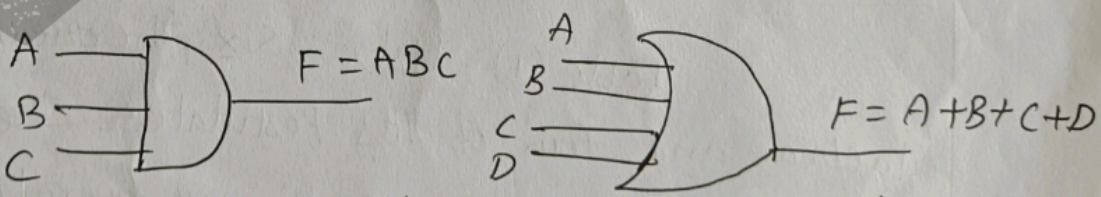


Fig. 3(a): 3-input AND gate (b) 4-input OR gate.

Basic Theorems and Properties of Boolean Algebra :-

Duality :-

One part may be obtained from the other if the binary operations and the identity elements are interchanged. This important property of Boolean algebra is called the duality principle and states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged. In a two-valued Boolean algebra, the identity elements and the elements of the set 'B' are the same: '1' and '0'. The duality principle has many applications. If the dual of an algebraic expression is desired, we simply interchange OR and AND operations and replace 1's by 0's and 0's by 1's.

Basic Theorems :

Table ① lists six theorems of Boolean algebra and four of its postulates. The notation is simplified by omitting the binary operator whenever doing so does not lead to confusion. The theorems and postulates listed are the most basic relationships

in Boolean algebra.

Postulate 1	(a) $x+0=x$	(b) $x \cdot 1 = x$
Postulate 5	(a) $x+x'=1$	(b) $x \cdot x' = 0$
Theorem 1	(a) $x+x=x$	(b) $x \cdot x = x$
Theorem 2	(a) $x+1=1$	(b) $x \cdot 0 = 0$
Theorem 3, involution	$(x')'=x$	—
Postulate 3, Commutative	(a) $x+y=y+x$	(b) $xy=yx$
Theorem 4, associative:	(a) $x+(y+z)=(x+y)+z$	(b) $x(yz)=(xy)z$
Postulate 4, Distributive	(a) $x(y+z)=xy+xz$	(b) $x+yz=(x+y)(x+z)$
Theorem 5, Demorgan	(a) $(x+y)'=x' \cdot y'$	(b) $(xy)'=x'+y'$
Theorem 6, Absorption	(a) $x+xy=x$	(b) $x(x+y)=x$

Table ①: Postulates and Theorems of Boolean Algebra.

The Postulates are basic axioms of the algebraic structure and need no proof. The Theorems must be proven from the Postulates. Proofs of the theorems with one variable are presented next. At the right is listed the number of the postulates which justifies that particular step of the proof.

THEOREM 1 (a): $X + X = X$.

Statement	Justification
$X+X = (X+X) \cdot 1$	Postulate 2 (b)
$= (X+X)(X+X')$	5 (a)
$= X + XX'$	4 (b)
$= X + 0$	5 (b)
$= X$.	2 (a)

Theorem 1 (b): $X \cdot X = X$.

Statement	Justification
$X.X = XX + 0$	P 2 (a)
$= XX + XX'$	5(b)
$= X(X+X')$	4(a)
$= X \cdot 1$	5(a)
$= X$	2 (b)

Theorem 2 (a): $X+1=1$.

Statement	Justification
$X+1 = 1\cdot(X+1)$	P 2(b)
$= (X+X')(X+1)$	5(a)
$= X+X'\cdot 1$	4(b)
$= X+X'$	2(b)
$= 1$	5(a)

Theorem 2(b):

$$X \cdot 0 = 0$$

Theorem 3:

$$(X')' = X.$$

The complement of x is x' and the complement of x' is x and is also $(x')' = x$.

Theorem 6(a): $X + XY = X$.

Statement	Justification
$X+XY = X \cdot 1 + XY$	P 2(b)
$= X(1+Y)$	4(a)
$= X \cdot (Y+1)$	3(a)
$= X \cdot 1$	2(a)
$= X$	2(b)

Theorem 6(b):

$$X(X+Y) = X$$

Theorem 5: De-morgan's

$$(X+Y)' = X'Y'$$

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

Equal

Boolean Functions :

Boolean Algebra is an algebra that deals with binary variables and logic operations. A Boolean Functions decribed by an algebric expression consists of binary variables, the consists '0' and '1', and the logic operation symbols.

Example: $F_1 = X + Y'Z$.

Complement of a Function :-

The Complement of a function F is F'. this is obtained from an interchange of 0's for 1's and 1's for 0's in the value of F.

Example: $(A+B+C)' = (A+X)'$ let $B+C=X$

$= A'X'$ by Theorem 5(a)

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

$= A'(B'C')$ by The. 5(a)

$= A'B'C'$ by The. 4(b).

Digital Logic Gates :

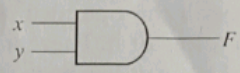
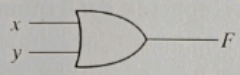
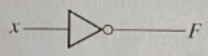
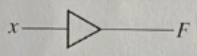
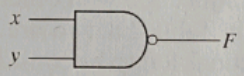
Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = x + y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$F = x'$	x | F 0 | 1 1 | 0
Buffer		$F = x$	x | F 0 | 0 1 | 1
NAND		$F = (xy)'$	x y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = (x + y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

FIGURE 2.5
Digital logic gates

Positive and Negative Logic:

The binary signal at the inputs and outputs of any gate has one of two values, except during transition. one signal value represents logic '1' and the other logic '0'. since two signal values are assigned to two logic values, there exist different assignments of signal level to logic value, as shown in the fig.

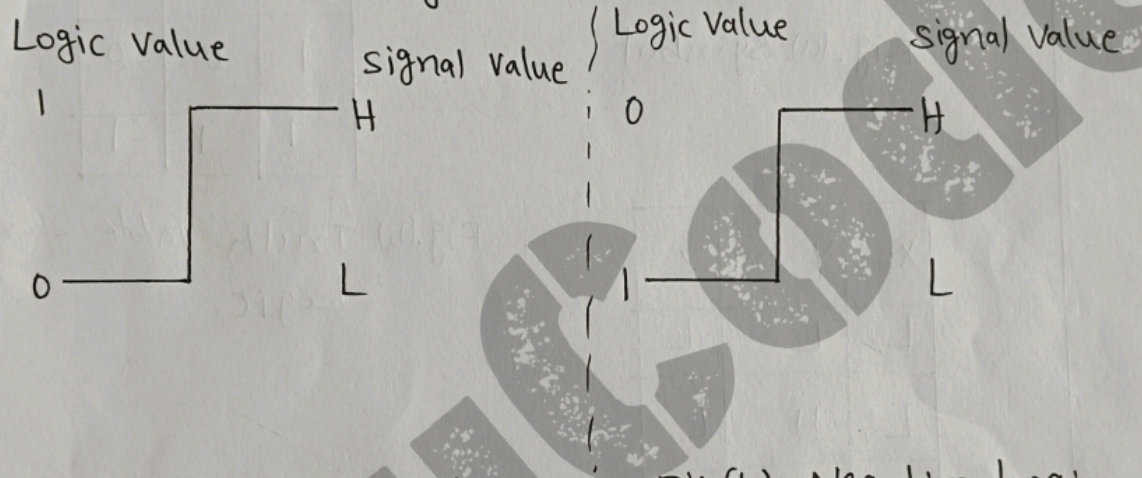


Fig (a) Positive Logic. Fig(b). Negative Logic.

Fig: Signal assignment and Logic polarity.

Choosing the high-level 'H' to represent logic '1' defines a positive logic system.

Choosing the low-level 'L' to represent logic '1' defines a negative logic - system.

Hardware digital gates are defined in terms of signal values such as H and L.

For example :-

(i) Positive Logic AND Gate :

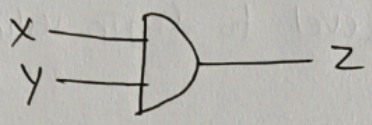


Fig. (a) Symbol

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Fig. (b) Truth Table for +ve Logic.

X	Y	Z
L	L	L
L	H	L
H	L	L
H	H	H

Fig. (c). Truth Table for +ve logic with 'H' and 'L'.

(ii) Negative Logic OR Gate :-

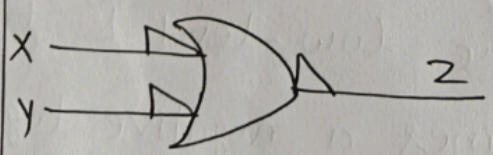


Fig. (a) Symbol

X	Y	Z
1	1	1
1	0	1
0	1	1
0	0	0

Fig. (b) TruthTable.

(or)

X	Y	Z
H	H	H
H	L	L
L	H	L
L	L	L

Fig. (c). Truth Table

NAND and NOR Implementation :

Digital circuits are frequently constructed with NAND or NOR gates rather than with AND and OR gates. NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families.

NAND Circuits :-

The NAND gate is said to be a universal gate because any logic circuit can be implemented with it.

A convenient way to implement a Boolean function with NAND gates is to obtain the simplified Boolean function in terms of Boolean operators and then convert the function to NAND-logic.

The conversion of an algebraic expression from AND, OR, and Complement to NAND can be done by simple circuit manipulation techniques that change AND-OR diagrams to NAND diagrams.

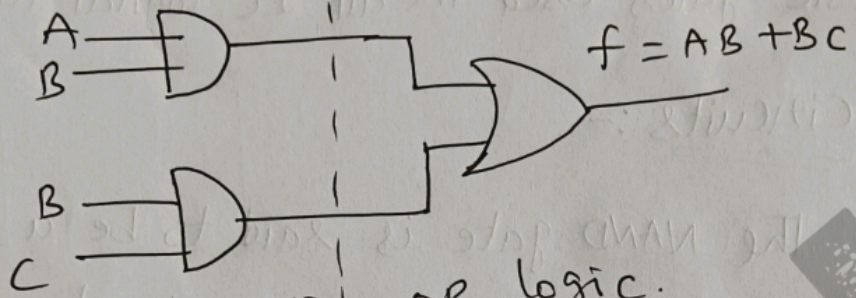
NOR Circuits :-

The conversion of an algebric expression from OR, AND, and Complement to NOR can be done by simple circuit manipulation techniques that change OR-AND diagrams to NOR diagrams.

For example :-

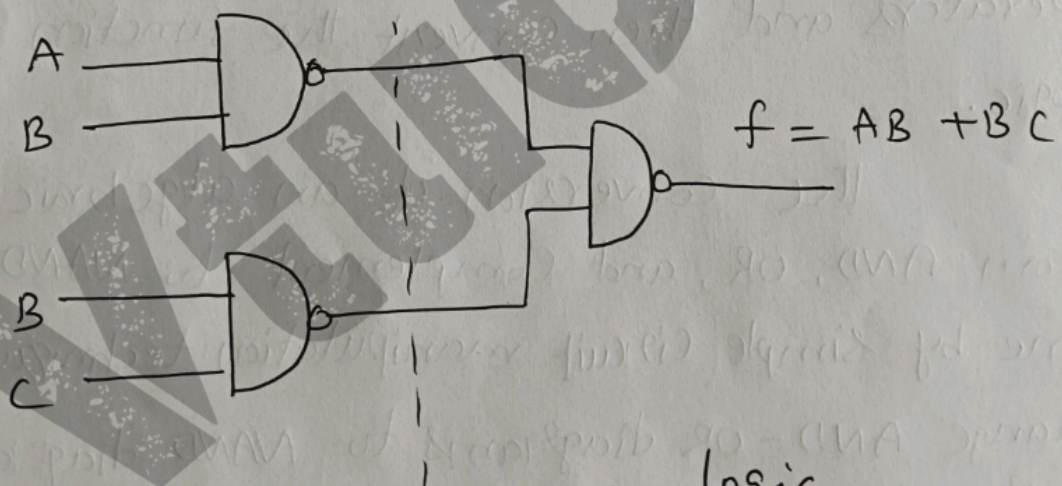
NAND Implementation :- (Use K-map (min term)) (m)

ex:- $f = AB + BC$



AND - OR logic.

We can replace AND - OR logic with NAND - NAND logic circuits.

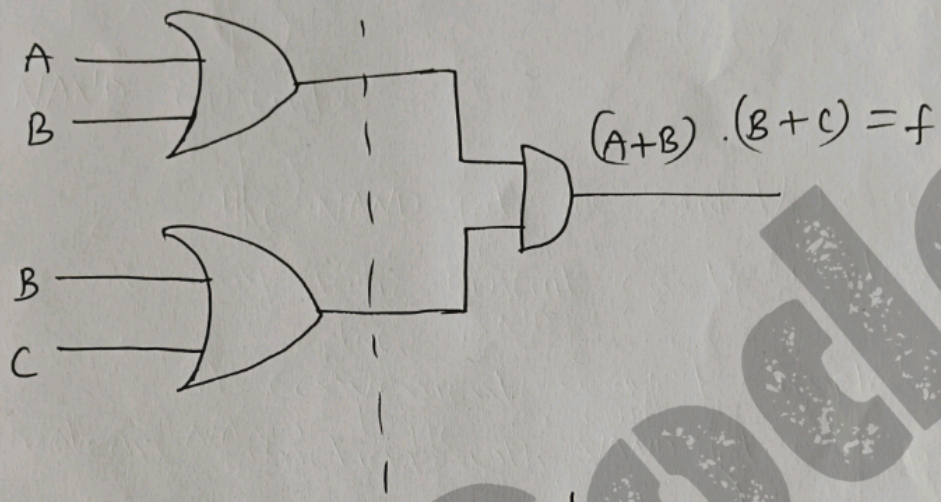


NAND - NAND logic.

For example :-

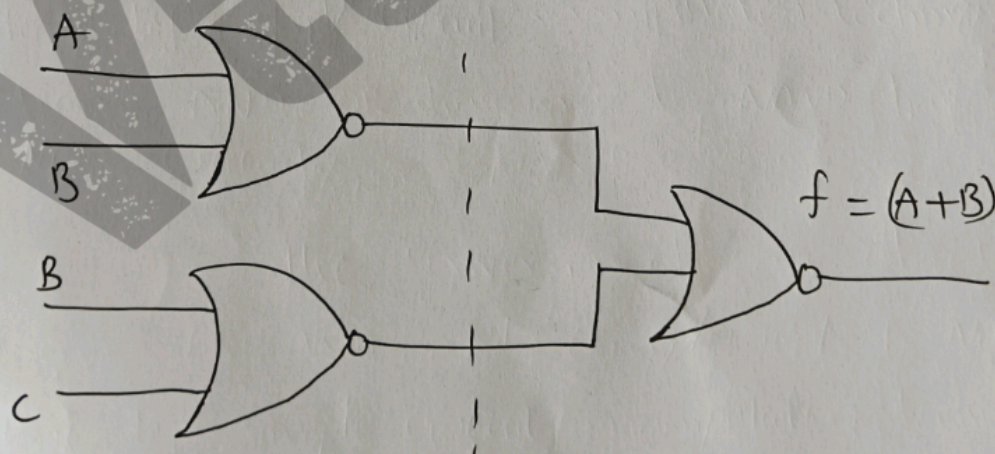
NOR Implementation :- (use K-map (maxterm) (M))

Ex:- $f = (A+B) \cdot (B+C)$



OR – AND logic.

We can replace OR-AND logic with NOR – NOR logic circuits.



NOR – NOR logic.

Principles of Combinational Logic :

Definition of Combinational logic :- when logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved, the resulting circuit is called Combinational logic.

A combinational circuit consists of input variables, logic gates and output variables. The logic gates accepts signals from the input variables and generates output signal.

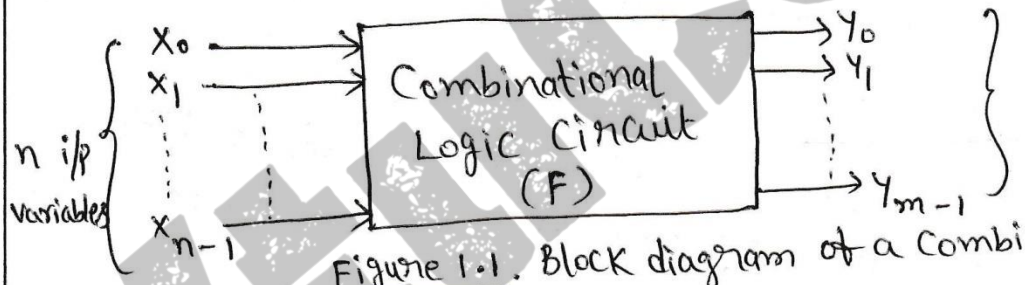


Figure 1.1. Block diagram of a combinational circuit.

In the combinational circuit, let 'x' be the set of 'n' i/p variables $\{X_0, X_1, X_2 \cdots X_{n-1}\}$ and 'Y' be the set of 'm' o/p variables $\{Y_0, Y_1, Y_2 \cdots Y_{m-1}\}$. The combinational function F, operate on the set 'X', to produce the output variable set 'Y'.

The output is thus related to input as $Y = F(X)$.

Switching Functions / Boolean Function :

Boolean expressions are constructed by connecting the Boolean constants and variables with the

Boolean operations. These Boolean expressions are also known as Boolean formulae. we use Boolean expressions to describe Boolean functions.

For example :

If the Boolean expression $(A+\bar{B})C$ is used to describe the function f, then Boolean function is written as $f(A,B,C) = (A+\bar{B})C$ (or) $f = (A+\bar{B})C$.

Definitions :

(i) Literal : The Boolean variable is appeared either in a complemented or uncomplemented is called.

For example, 'A' is the Boolean variabe then $\bar{A}$ or A are literals.

(ii) Product term :

It is defined as either a Literal or a product of literals. (or) It is any group of literals that are ANDed together.

Ex: A, $A\bar{B}$, ABC.

(iii) Sum term :

It is defined as either a literal or a sum of literals. (or) It is any group of literals that are ORed together.

Ex: A, $A+B$, $\bar{A}+B$.

These literals and terms are arranged in one of the two forms.

(1) Sum of product form (SoP form)

(2) Product of sum form (PoS form)

① Sum of Product form (SoP):

A sum of product is any group of product terms ORed together is called SoP.

Ex: $f(A,B,C) = ABC + A\bar{B}\bar{C} + \bar{B}C$

② Product of sum form (PoS):

It is defined as any groups of sum terms ANDed together is called PoS.

Ex: $f(P,Q,R,S) = (P+\bar{Q})(R+\bar{S})(P+Q+R)$.

Canonical Form (Standard SoP and PoS forms):

The Canonical forms are the special cases of SoP and PoS forms. These are also known as Standard SoP and PoS forms.

(i) Standard SoP form (or) minterm Canonical form :-

If each term in SoP form contains all the literals then the SoP form is known as the standard or Canonical SoP form.

Each individual term in the standard SoP-form is called minterm. Therefore, Canonical SoP form is also known as minterm Canonical form.

Ex: $f(A,B,C) = A\bar{B}\bar{C} + ABC + \bar{A}B\bar{C}$.

(ii) Standard POS form (or) Maxterm Canonical form:

If each term in POS form contains all the literals then the POS form is known as standard POS-form (or) Canonical POS form.

Each individual term in the standard POS form is called Maxterm. Therefore, Canonical POS form is also known as maxterm Canonical form.

Ex: $f(A,B,C) = (A+B+C).(A+\bar{B}+\bar{C})(\bar{A}+\bar{B}+\bar{C})$

Converting Expressions in standard SOP or POS forms:

SOP form can be converted to standard SOP by ANDing the terms in the expression with terms formed by ORing the literal and its complement which are not present in that term. For example, a three variable expression with variables A, B and C, if there is a term AB, where 'C' is missing, then we form term $(C+\bar{C})$ and AND it with AB.

$f = AB$ (or) $f(A,B,C) = AB$

$\therefore f = AB(C+\bar{C})$

$$f = ABC + AB\bar{C}$$

Steps to Convert SOP to standard SOP form :

① Finding the missing literal in each Product term if any.

② AND each Product term having missing literal/s with term/s form by ORing the literal and its Complement.

③ Expand the terms by applying distributive law and reorder the literals in the product terms.

④ Reduce the expression by omitting repeated product terms if any. Because $A + A = A$.

Example : Convert the given expression in standard SOP form

$$f(A, B, C) = AC + AB + BC$$

Solution:

step ① :- Find the missing literal in each product term

$$f(A, B, C) = AC + AB + BC$$

AC → 'B' missing, AB → 'C' missing, BC → 'A' missing.

step ② : $f = AC(B + \bar{B}) + AB(C + \bar{C}) + BC(A + \bar{A})$

step ③ : Expand the terms and reorder literals.

$$f(A, B, C) = \underline{ABC} + A\bar{B}C + \underline{ABC} + AB\bar{C} + \underline{ABC} + \bar{A}BC$$

step ④ : omit repeated product terms.

$$\therefore f(A, B, C) = ABC + A\bar{B}C + AB\bar{C} + \bar{A}BC$$

steps to Convert Pos to Standard Pos form:

① Find the missing literals in each sum term if any.

② OR each sum term having missing literal/s with term/s form by ANDing the literal and its complement.

③ Expand the terms by applying distributive law and reorder the literals in the sum terms.

④ Reduce the expression by omitting repeated sum terms if any. Because $A \cdot A = A$.

Example: Convert the given expression in Standard Pos form

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

Solution.

Step①:- Find the missing literal/s in each Sum term.

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

↳ Literal B
↳ Literal A
↳ Literal C
} missing

Step②:-

OR sum term with (missing literal . its Complement).

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

Step③:- Expand the terms and reorder literals.

Since, $A+BC = (A+B) \cdot (A+C)$ we have,

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

step④:- omit repeated sum terms.

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

Generation of Switching Equations from Truth Tables:

Each individual term in standard SoP form is called minterm and each individual term in standard PoS form is called Maxterm.

Table ①, gives the minterms and max terms for a three literals/variable logical functions where the number of minterms as well as maxterms is $2^3 = 8$. In general, for an n-variable logical function there are 2^n minterms and an equal number of max terms.

Variables A	B	C	minterms m_i	max terms M_i
0	0	0	$\bar{A}\bar{B}\bar{C} = m_0$	$A+B+C = M_0$
0	0	1	$\bar{A}\bar{B}C = m_1$	$A+B+\bar{C} = M_1$
0	1	0	$\bar{A}B\bar{C} = m_2$	$A+\bar{B}+C = M_2$
0	1	1	$\bar{A}BC = m_3$	$A+\bar{B}+\bar{C} = M_3$
1	0	0	$A\bar{B}\bar{C} = m_4$	$\bar{A}+B+C = M_4$
1	0	1	$A\bar{B}C = m_5$	$\bar{A}+B+\bar{C} = M_5$
1	1	0	$AB\bar{C} = m_6$	$\bar{A}+\bar{B}+C = M_6$
1	1	1	$ABC = m_7$	$\bar{A}+\bar{B}+\bar{C} = M_7$

Table①: Minterms and Maxterms for three variables.

Each minterm is represented by m_i and each maxterm is represented by M_i, where the subscript 'i' is the decimal number equivalent of the natural binary-number.

Examples: (i) $f(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}BC + AB\bar{C}$

$= m_0 + m_1 + m_3 + m_6$

$= \sum m(0,1,3,6)$.

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

$= M_1 . M_3 . M_6$

$= \Pi M(1,3,6)$.

where Σ denotes Sum of Product

and Π denotes Product of Sum.

we know that logical expression can be represented in the truth table form. It is possible to write logical expression in standard SOP or POS form corresponding to a given truth table.

For example :- ①.

Inputs			output	
A	B	C	Y	
0	0	0	0	
0	0	1	0	
0	1	0	1	→ $\bar{A}B\bar{C}$
0	1	1	1	→ $\bar{A}BC$
1	0	0	0	
1	0	1	0	
1	1	0	1	→ $AB\bar{C}$
1	1	1	0	

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

$= m_2 + m_3 + m_6 = \Sigma m(2,3,6)$

example ② :

A	B	Y	
0	0	1	
0	1	1	
1	0	0	→ $\bar{A}+B$
1	1	0	→ $\bar{A}+\bar{B}$

$\therefore f(A,B) = (\bar{A}+B).(\bar{A}+\bar{B})$.

$= M_2 . M_3$

$= \Pi M(2,3)$.

Complements of Canonical Formulae:

A standard POS form derived from a truth table is logically equivalent to a standard SOP form.

for example: $f(A,B,C) = (A+\bar{B}+C).(\bar{A}+B+\bar{C})$

$= M_2 . M_5$

$= \pi M(2,5)$

$f(A,B,C) = \Sigma m(0,1,3,4,6,7)$.

Karnaugh Map Minimization:

For simplication of Boolean expressions by Boolean algebra we need better understanding of Boolean laws, rules and theorems. During the process of simplication we have to predict each successive step. For these reasons, we can never be absolutely certain that an expression simplified by Boolean algebra alone is the simplest possible expression. on the otherhand, the map method gives us a systematic approach for simplifying a Boolean expression. The map method, first proposed by Veitch and modified by Karnaugh, hence it is known as the Veitch diagram or the Karnaugh map.

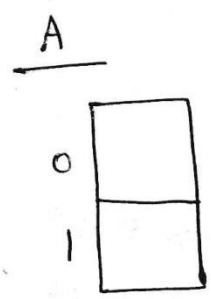
→ A ~~map~~ map is a diagram made up of squares. Each square represents either a minterm or a maxterms.

→ The number of squares in the K-map is given by 2^n. where n = number of variables.

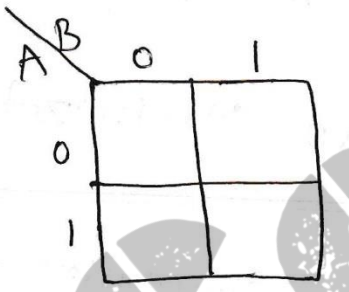
→ To maintain adjacency property Gray Code sequence is used in K-maps. (Any two adjacent cells will differ by only one bit).

Karnaugh maps - 3, 4, 5 Variables :

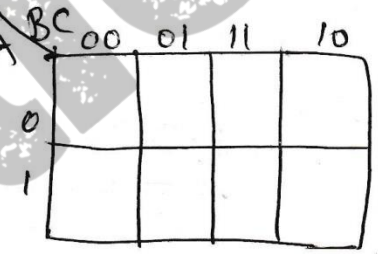
The basis of this method is a graphical chart known as Karnaugh map (K-map). It contains boxes called cells.



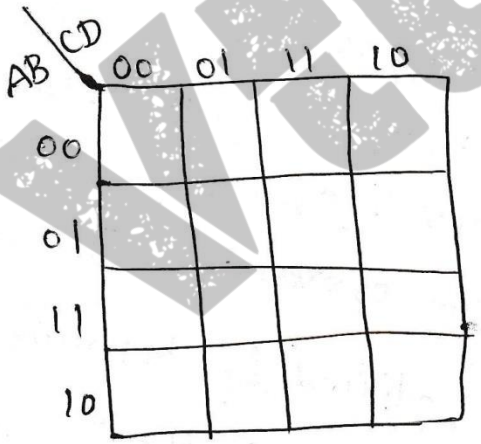
1-variable K-map
(2 cells)



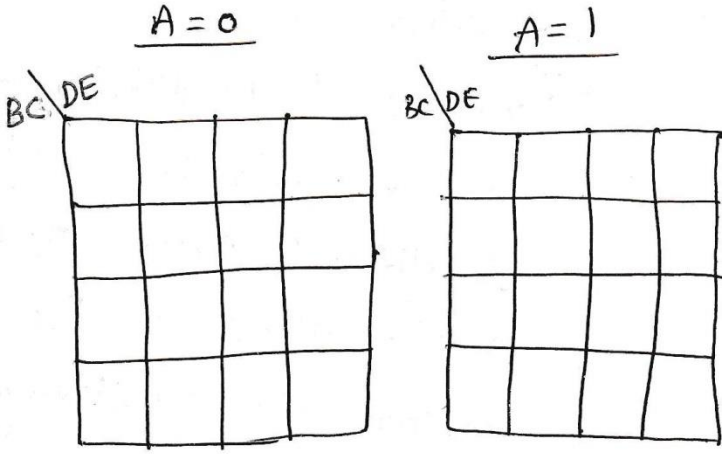
2-variable K-map
(4 cells)



3-variable K-map
(8-cells)



4-Variable K-map
(16-cells)



5-variable K-map
(32-cells)

Figure : outlines of 1, 2, 3, 4 and 5 variable K-maps.

Plotting a Karnaugh Map:

We know that logic function can be represented in various forms such as truth table, standard SOP and standard POS Boolean expression.

(i) Representation of Truth Table on K-map :-

Cell : The smallest unit of a K-map, corresponding to one line of a truth table. The input variables are the cells co-ordinates and the output variable is the cell's contents.

Fig①, shows K-maps plotted from truth tables with 2, 3 and 4 variables. Looking at the fig①, we can easily notice that the terms which are having output 1, have the corresponding cells marked with 1's. The other cells are marked with 0's.

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

A \ B	0	1
0	0	1
1	1	0

(or)

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

Fig 1(a): Representation of 2-variable truth table on K-map

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

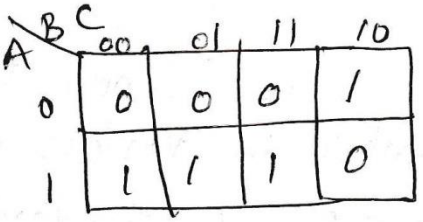


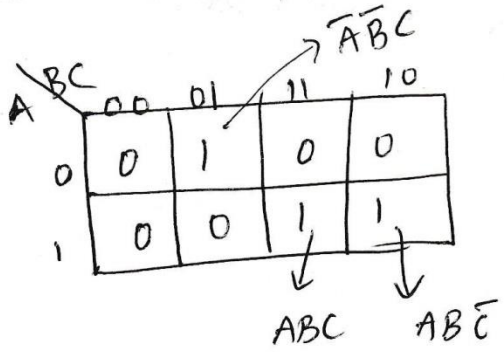
Fig 1(b):- Representation of 3-variable truth table on K-map.

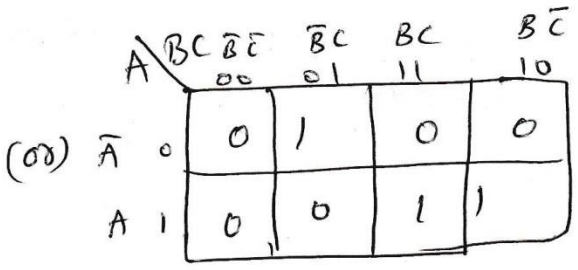
(ii) Representing standard SOP on K-map:

A Boolean expression in the standard SOP form can be plotted on the K-map by placing a '1' in each cell corresponding to a term (minterm) in the SOP expression. Remaining cells are filled with 0's.

Example: plot Boolean expression $Y = AB\bar{C} + ABC + \bar{A}\bar{B}C$ on the K-map.

Sol) The expression has 3-variables and hence it can be plotted using 3-variable K-map as in fig(1).





Representing standard POS on K-map:

A Boolean expression in the SPOS can be plotted on the K-map by placing a "0" in each cell corresponding to a term (maxterm) in the expression. Remaining cells are filled with "1"s.

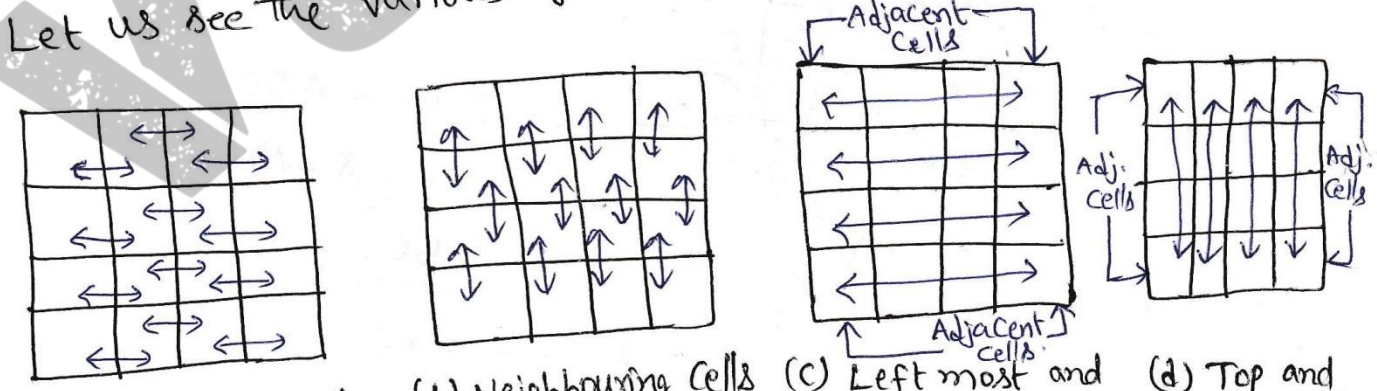
Example: Plot Boolean expression $Y=(\bar{A}+\bar{B}+C)(A+\bar{B}+C)(A+\bar{B}+\bar{C})$ on the K-map.

Sol).

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

Grouping cells for simplification:

The grouping is nothing but combining terms in adjacent cells. Two cells are said to be adjacent if they conform the single change rule. i.e. there is only one variable difference between co-ordinates of two cells. Let us see the various grouping rules.



Fig(1) a) Neighbouring Cells in the row are adjacent.

(b) Neighbouring Cells in the column are adjacent.

(c) Left most and corresponding rightmost cells are adjacent.

(d) Top and corresponding Bottom cells are adjacent

Fig (1): Adjacent Cells.

Rules to Simplify K-maps:

(1) At the time of grouping the adjacent cells containing 1's always use maximum possible group.

(2) All the cells containing 1's must be covered at least once in any group.

(3) At the time of grouping don't care (X) values can be taken as 1's.

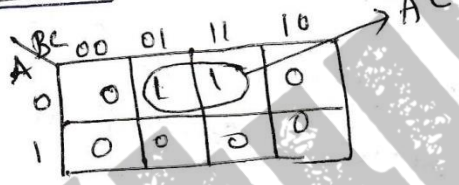
(4) All don't care values need not be covered.

Grouping Two Adjacent ones (Pairs):

A pair is a group of two adjacent cells in a K-map. It cancels one variable in a K-map simplification.

examples:

(i)

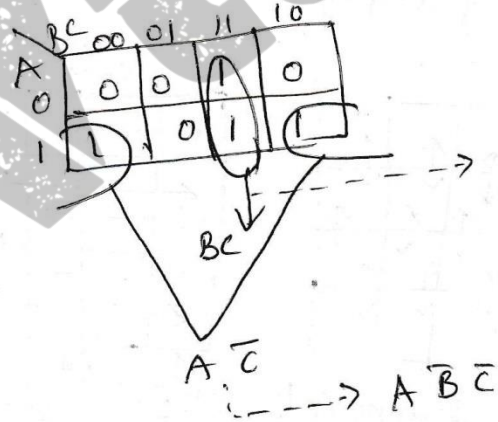


$Y = \bar{A}\bar{B}C + \bar{A}BC$

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

$Y = \underline{\bar{A}C}$ $(\because \bar{B}+B=1)$

(ii)



$\bar{A}BC + ABC$

$BC(\bar{A}+A)$

$\underline{BC}$

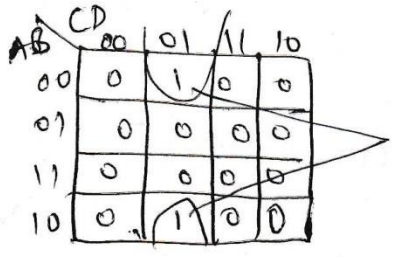
$A\bar{B}\bar{C} + AB\bar{C}$

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

$\underline{A\bar{C}}$

$\therefore Y = \underline{BC + A\bar{C}}$

(iii)

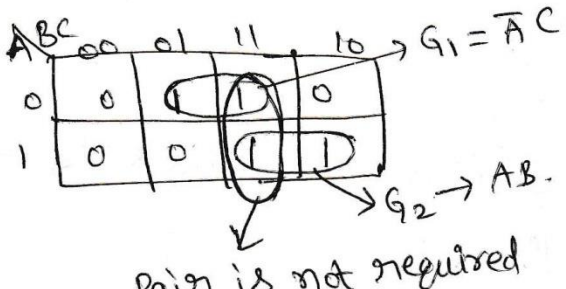


$Y = \bar{A}\bar{B}\bar{C}D + A\bar{B}\bar{C}D$

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

$Y = \bar{B}\bar{C}D$ //

(iv)

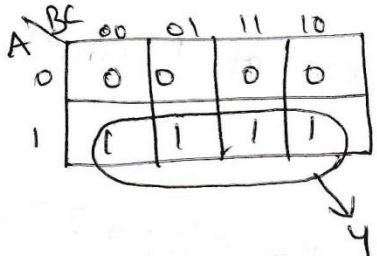


$Y = \bar{A}C + AB$

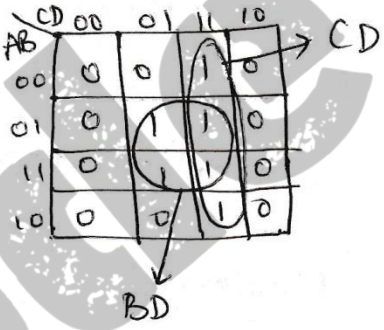
Grouping Four Adjacent ones (Quad):

A Quad is a group of four adjacent cells in a K-map. It cancels 2 variables in a K-map simplification.

examples:- (i)

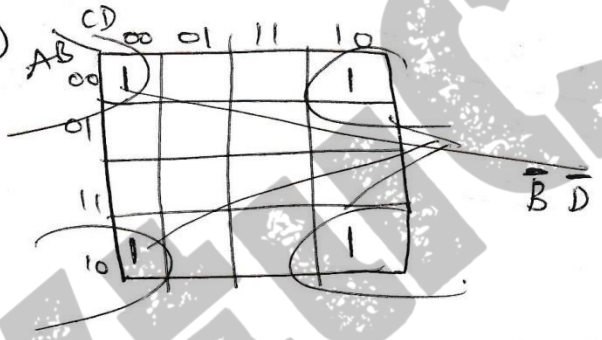


(ii)



$Y = BD + CD$.

(iii)

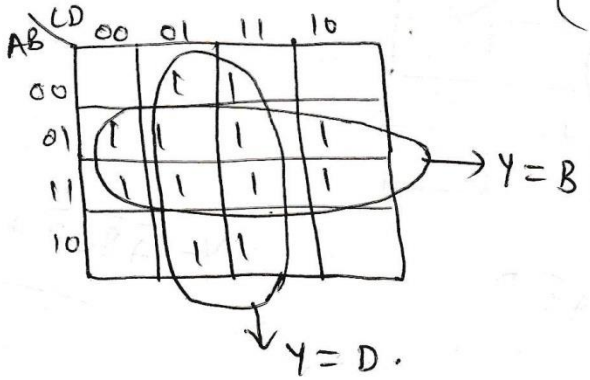


Grouping Eight Adjacent ones (octet):

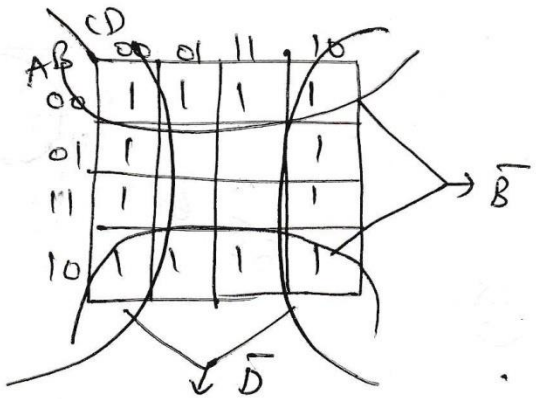
A octet is a group of eight adjacent cells in a K-map. It Cancels 3 variables in a K-map simplification.

examples:-

(i)



(ii)



Illegal Grouping:

The below figures shows the examples of Illegal grouping of cells.

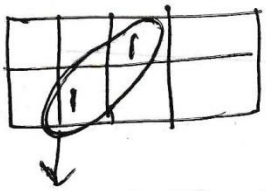


Fig (a): Diagonal grouping is illegal.

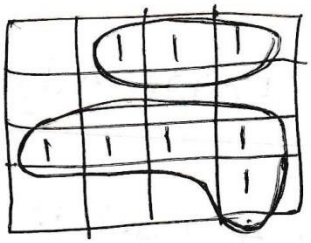


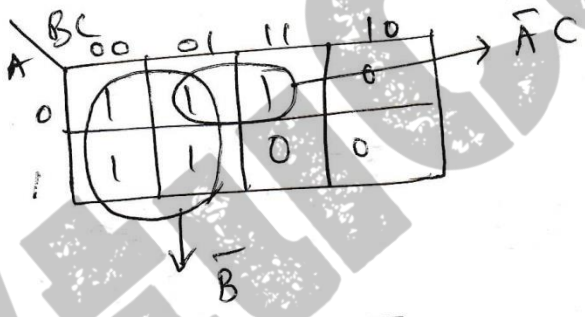
Fig (b): Grouping of odd no. of cells is illegal.

Simplification of Sum of Products Expressions (minimal sums):

Problems:

① Minimize the expression $Y = A\bar{B}C + \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C}$.

Sol)

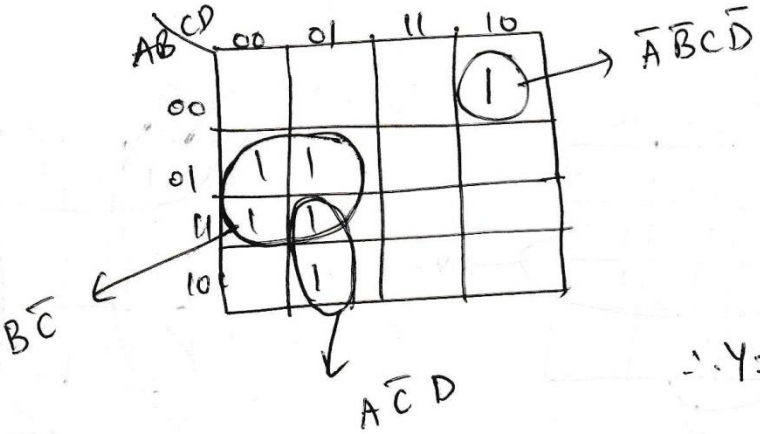


$\therefore Y = \bar{A}C + \bar{B}$

② Minimize the expression

$Y = \bar{A}B\bar{C}\bar{D} + \bar{A}BC\bar{D} + AB\bar{C}\bar{D} + AB\bar{C}D + A\bar{B}\bar{C}D + \bar{A}\bar{B}CD$.

Sol)



$\therefore Y = \bar{A}\bar{B}C\bar{D} + A\bar{C}D + B\bar{C}$

③ Reduce the following function using K-map technique

$f(A,B,C,D) = \Sigma m(0,1,4,8,9,10)$.

Sol)

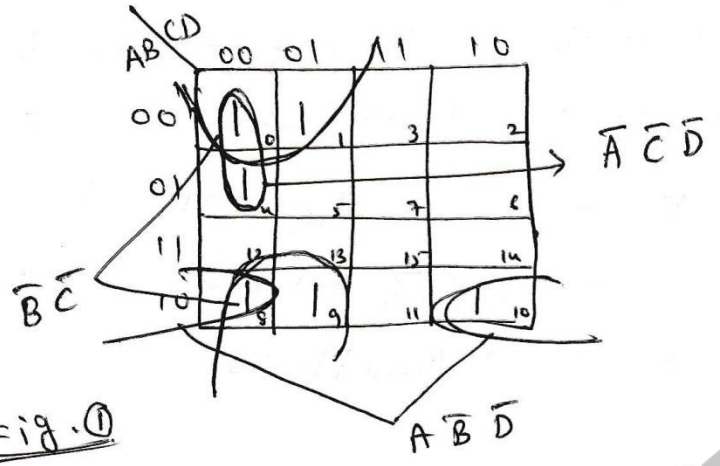


Fig. ①

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

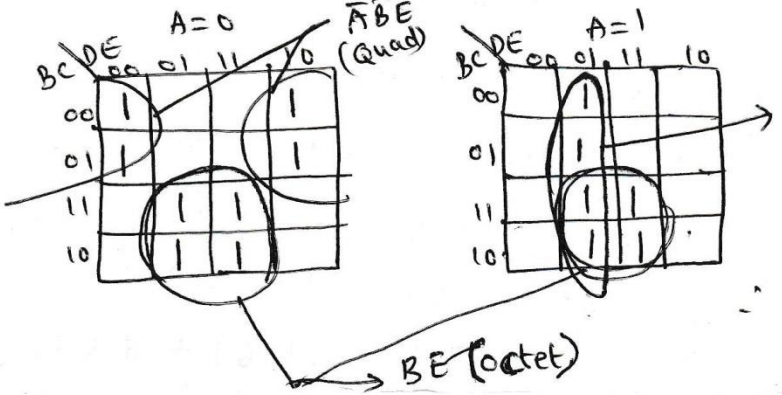
Essential prime Implicants:

After grouping the cells, the sum terms which appear in the K-map are called prime implicant groups. It is observed that some cells may appear in only one prime implicant group. while other cells may appear in more than one prime implicant group. In above figure ①, cells 1, 4, 9 and 10 appear in only one prime implicant group. These cells are called Essential cells and corresponding prime implicants are called Essential prime implicants.

④ Simplify the boolean function

$f(A,B,C,D,E) = \Sigma m(0,2,4,6,9,11,13,15,17,21,25,27,29,31)$.

Sol)



$\therefore f = \bar{A}\bar{B}\bar{E} + A\bar{D}E + BE$

Incompletely specified Functions (Don't Care terms):

In some logic circuits, certain input conditions never occur, therefore the corresponding output never appears. In such cases the output level is not defined, it can be either HIGH or LOW. These output levels are indicated by 'x' or 'd' in the truth tables and are called don't care outputs (or) don't care conditions (or) Incompletely specified functions.

For example:-

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

Here outputs are defined for input conditions from 000 to 101. For remaining conditions of input, output is not defined, hence these are called don't care conditions for this truth table.

A circuit designer is free to make the output for any "don't care" conditions either a "0" or "1" in order to produce the simplest output expression.

Describing Incomplete Boolean Function:

In order to obtain for Incomplete - Boolean functions we use additional term to specify don't care conditions in the original expression.

for example: (i) $f(A,B,C) = \Sigma m(0,2,4) + d(1,5)$

minterms are 0, 2 & 4. The additional term d(1,5) are don't care terms.

Similarly, (ii) $f(A,B,C) = \Pi M(0,1,6) + d(2,3)$.

Don't Care Conditions in Logic design:

In this section, we see the example of Incompletely specified Boolean function. Let us see the logic circuit for an even parity generator for 4-bit BCD number.

A	B	C	D	P
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	—
1	0	1	1	—
1	1	0	0	—
1	1	0	1	—
1	1	1	0	—
1	1	1	1	—

Table (1) ↑

the Truth Table (1) shows the for even Parity-generator. the output for last six input Conditions Cannot be specified, because such input conditions does not occur when input is in the BCD form.

The Boolean function for even-Parity generator with 4-bit BCD input Can be expressed in minterm Canonical formula as,

$$f(A,B,C,D) = \Sigma m(1,2,4,7,8) + d(10,11,12,13,14,15).$$

Minimization of Incompletely specified Functions:

A circuit desiner is free to make the output for any don't Care Condition either a '0' or '1' in order to produce the simplest output expression.

for example:

(i)

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

Truth Table ↑

K-map →

A \ BC	00	01	11	10
0	0	1	1	0
1	0	1	X	X

fig 1(a)

It is not always advisable to put don't cares as 1's. This is shown in fig 1(a).

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

C

Here the don't care '7' taken as '1' and the don't care '6' taken as '0'.

$Y = C$

(ii) Find the reduced SOP form of the following function.

$f(A,B,C,D) = \Sigma m(1,3,7,11,15) + \Sigma d(0,2,4)$.

Sol)

AB \ CD	00	01	11	10
00	X	1	1	X
01	X	0	1	0
11	0	0	1	0
10	0	0	1	0

Fig (a)

⇒

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

$\bar{A}\bar{B}$

CD.

Fig (b).

$\therefore f(A,B,C,D) = \bar{A}\bar{B} + CD$.

Simplifying Max term equations : (POS)

Problems : ① Minimize the expression

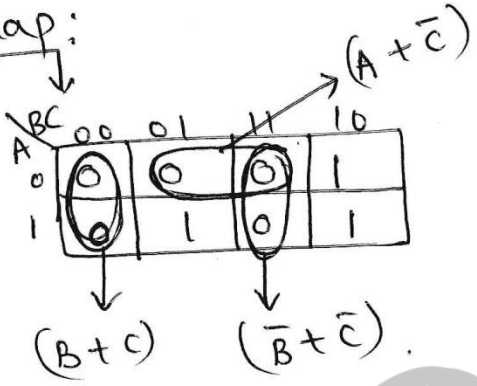
$$Y = (A+B+\bar{C})(A+\bar{B}+\bar{C})(\bar{A}+\bar{B}+\bar{C})(\bar{A}+B+C)(A+B+C)$$

Sol)

$$Y = M_1 \cdot M_3 \cdot M_7 \cdot M_4 \cdot M_0$$

$$Y = \pi M(0,1,3,4,7).$$

K-map :

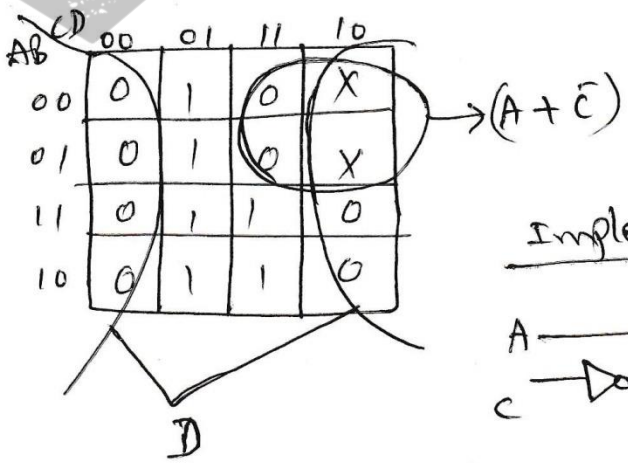


$$\therefore Y = (A+\bar{C}) \cdot (B+C) \cdot (\bar{B}+\bar{C}).$$

② Reduce the following function using K-map technique and implement using basic gates.

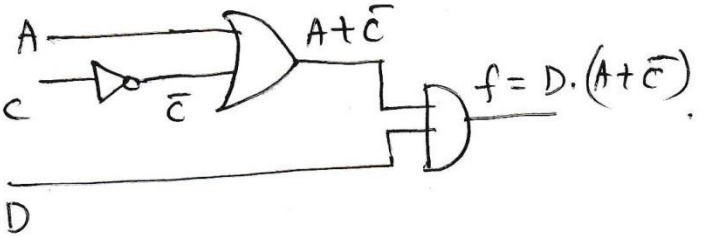
$$f(A,B,C,D) = \pi M(0,3,4,7,8,10,12,14) + d(2,6).$$

Sol) K-map :



$$\therefore f = D \cdot (A+\bar{C})$$

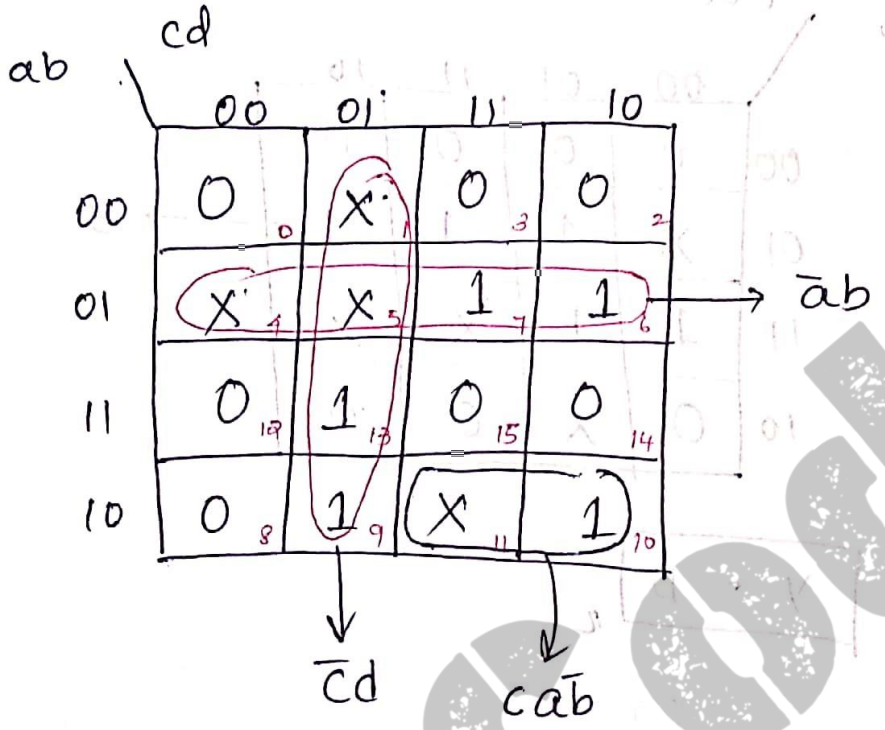
Implementation,



✓ Find the minimal SOP(Sum of Product) for the following Boolean functions using K-map. (8m) June-17

i) $f(a,b,c,d) = \sum m(6,7,9,10,13) + d(1,4,5,11)$

Soln

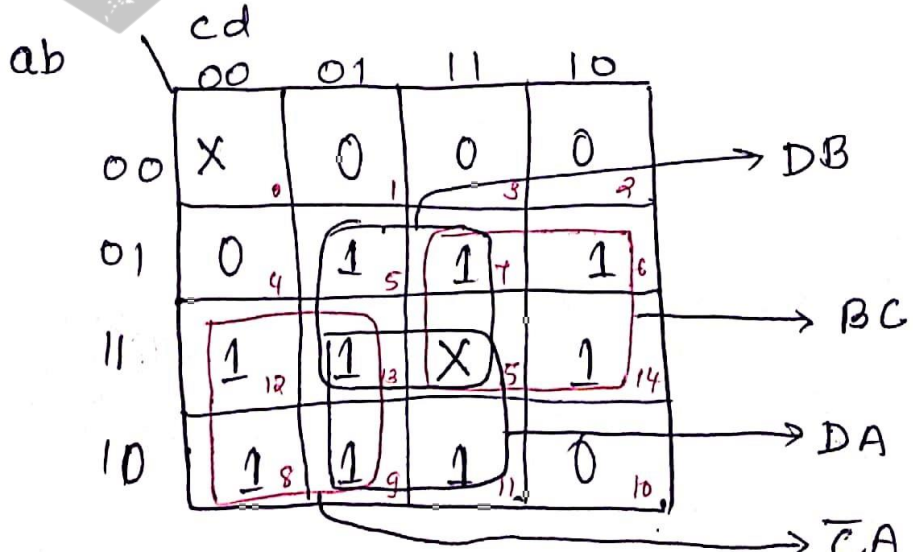


$$Y = \bar{a}b + \bar{c}d + c\bar{a}\bar{b}$$

ii) $f(a,b,c,d) = \pi M(1,2,3,4,10) + d(0,15)$

Converting above Boolean fun^n to minterms.

$$\sum m(5,6,7,8,9,11,12,13,14) + d(0,15)$$

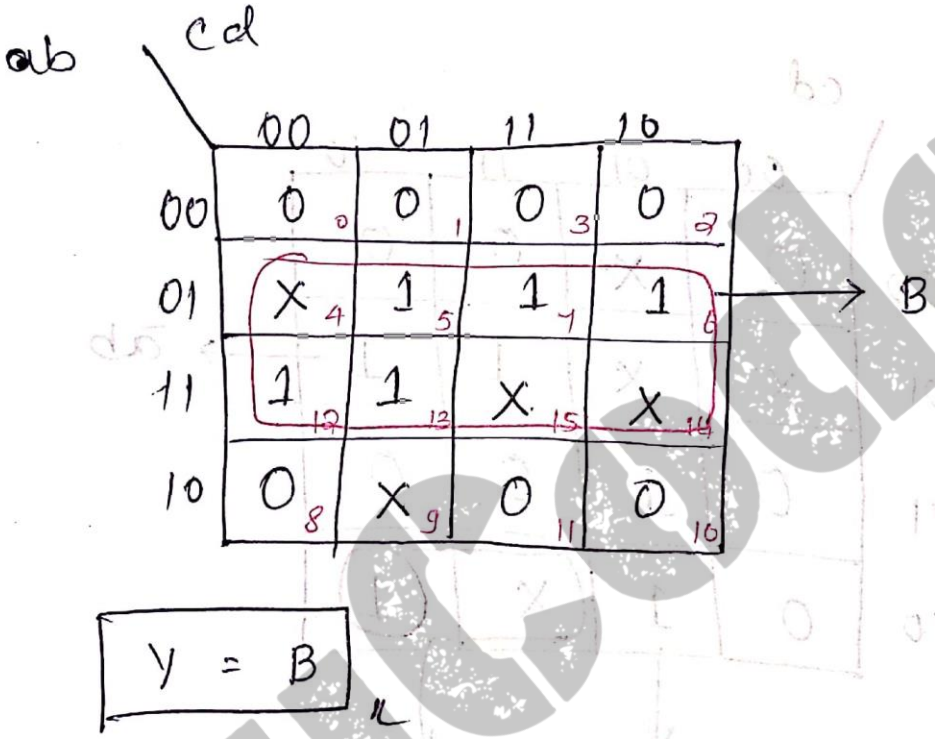


$\therefore Y = BD + BC + DA + \bar{C}A$

- Minimize the following Boolean functions using k-map method. (6m) Dec/Jan -2017.

$f(a, b, c, d) = \sum m(5, 6, 7, 12, 13) + \sum d(4, 9, 14, 15)$

Soln



ab \ cd	00	01	11	10
00	0 (0)	0 (1)	0 (3)	0 (2)
01	X (4)	1 (5)	1 (7)	1 (6) → B
11	1 (12)	1 (13)	X (15)	X (14)
10	0 (8)	X (9)	0 (11)	0 (10)

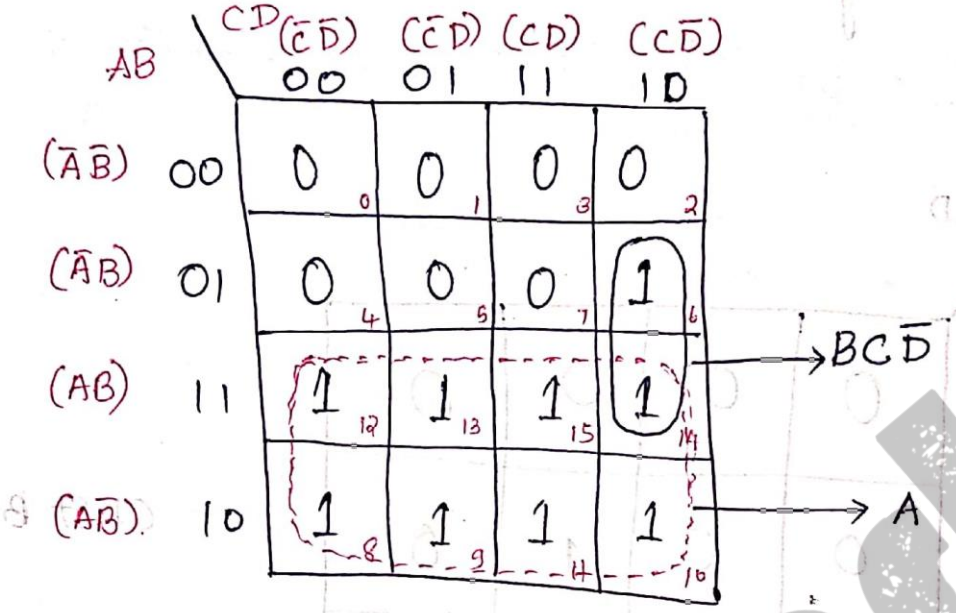
$Y = B$

✓ Give SOP & POS circuit for, (6m). June - 18.

$f(A, B, C, D) = \sum m(6, 8, 9, 10, 11, 12, 13, 14, 15)$

Soln

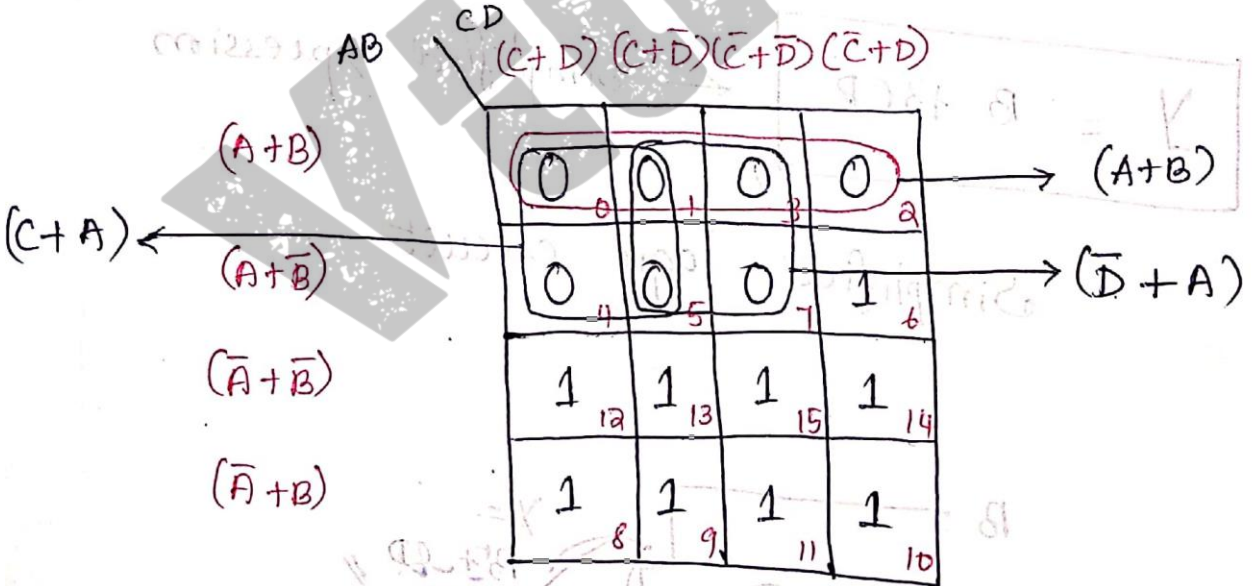
SOP



$$Y = A + BC\bar{D}$$

converting minterm to maxterm → $\sum m(6, 8, 9, 10, 11, 12, 13, 14, 15)$

$\Rightarrow \pi M(0, 1, 2, 3, 4, 5, 7)$

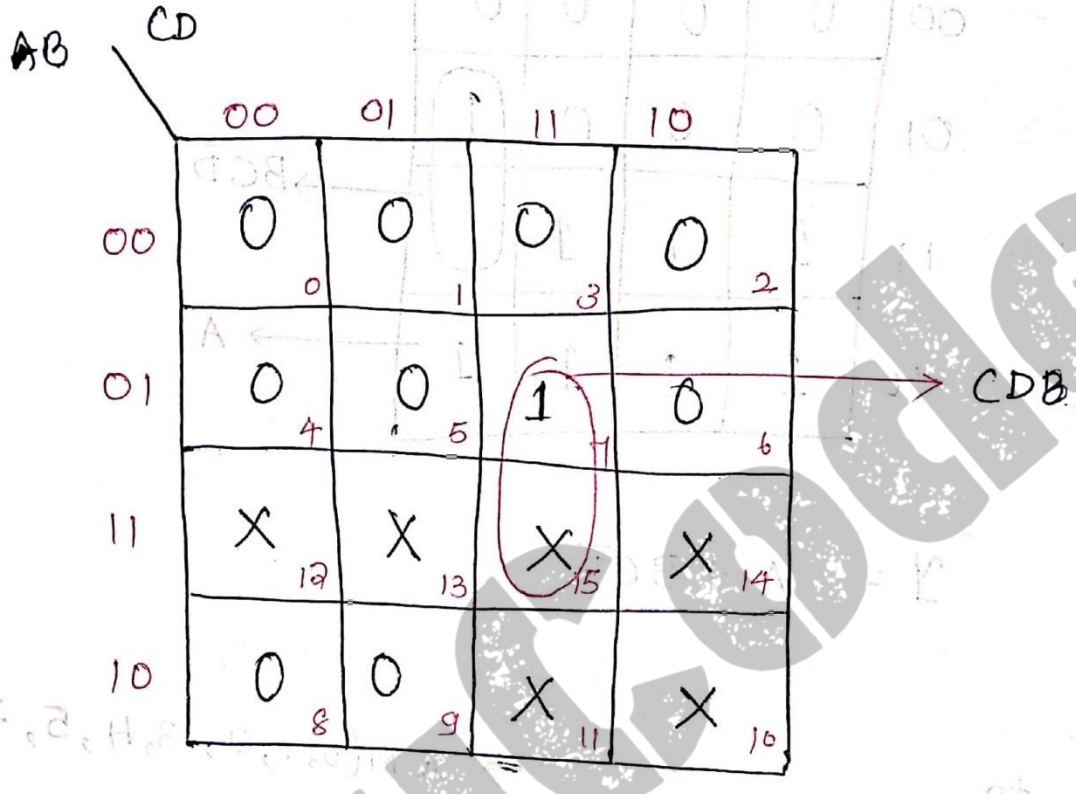


$$Y = (A+B)(A+C)(A+\bar{D})$$

(24)

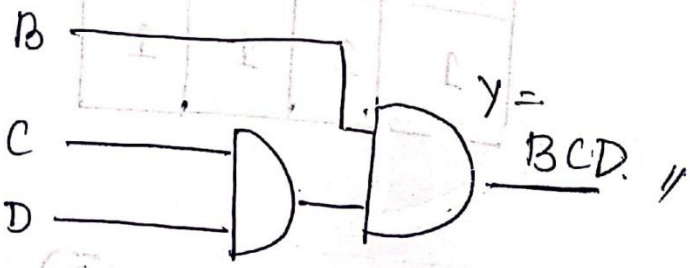
✓ Give a simplest logic circuit for following logic equation where d represents don't care condition for following locations: (6m) Jan-18

$F(A, B, C, D) = \sum m(7) + d(10, 11, 12, 13, 14, 15)$

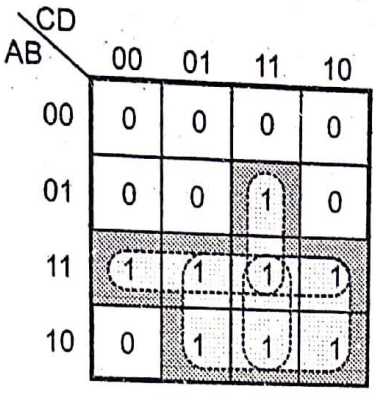


$Y = BCD$ ← Simplified expression

Simplified logic circuit



Solution : $f(A,B,C,D) = \sum m(7, 9, 10, 11, 12, 13, 14, 15)$



$= AB + AC + AD + BCD$

Logic diagram :

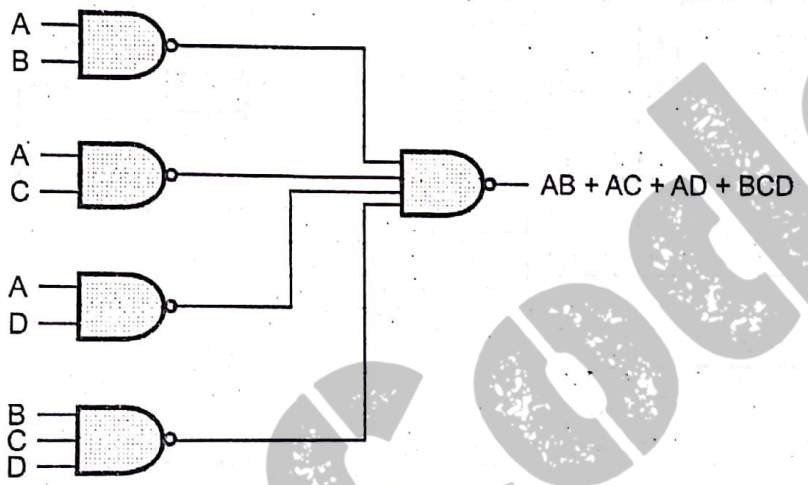
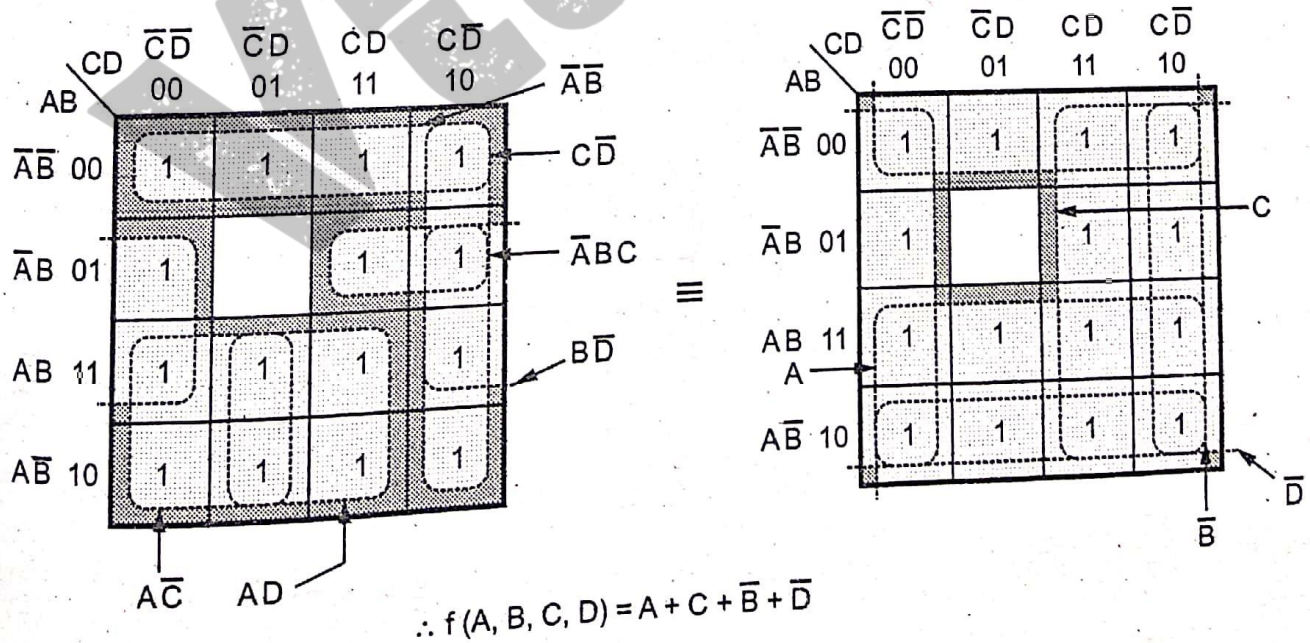


Fig. 7.3.25

Example 7.3.18 *Simplify the following using K-map*

$F(A, B, C, D) = \overline{A}\,B\,C + A\,D + B\,\overline{D} + C\,\overline{D} + A\,\overline{C} + \overline{A}\,\overline{B}$ **VTU : Aug.-08, Marks 6**

Solution :



$\therefore f(A, B, C, D) = A + C + \overline{B} + \overline{D}$

Fig. 7.3.26

Example 7.3.19 *Using Karnaugh map simplify the following Boolean expression and give the implementation of the same using :*

i) NAND gates only (SOP form)

$F(A, B, C, D) = \sum m\ (0, 1, 2, 4, 5, 12, 14) + dc\ (8, 10)$ **VTU : Feb.-08, Marks 4**

Solution : i) NAND gates only (SOP form)

$F(A, B, C, D) = \sum m(0, 1, 2, 4, 5, 12, 14) + dc\ (8, 10)$

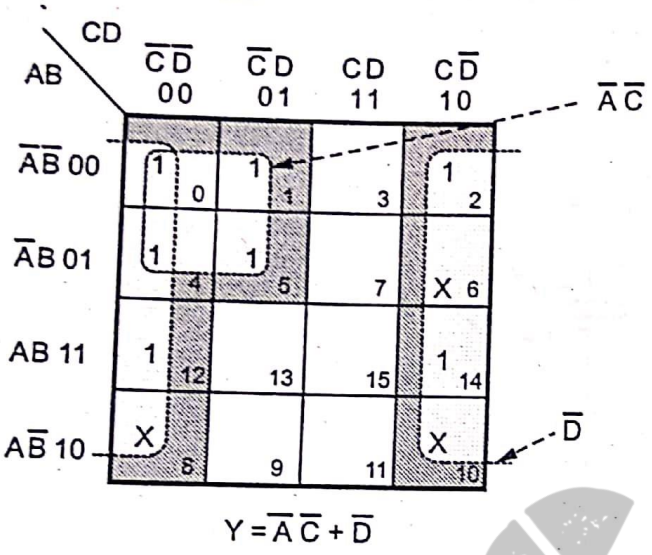


Fig. 7.3.27 (a)

Fig. 7.3.27 (b)

Example 7.3.20 *Simplify the following logic equation using Karnaugh map and give the implementation of the simplified expression.*

$F(A, B, C, D) = \Sigma m(7) + d\ (10, 11, 12, 13, 14, 15).$ **VTU : Jan.-18, Feb.-09, Marks 5**

Solution :

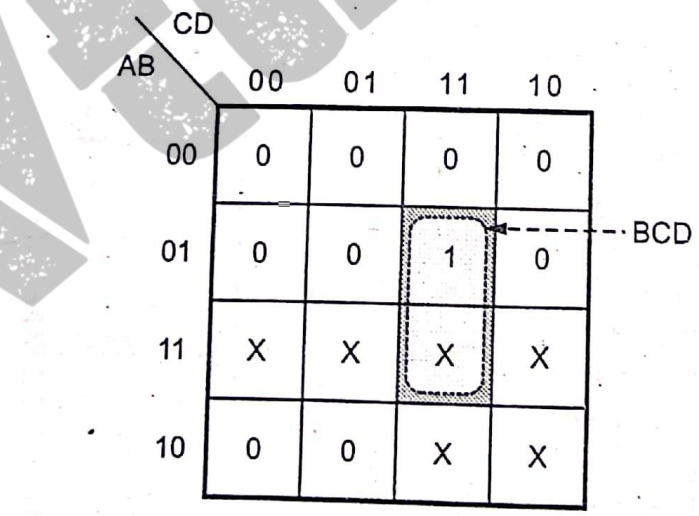


Fig. 7.3.28 (a)

$F(A, B, C, D) = BCD$

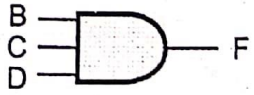
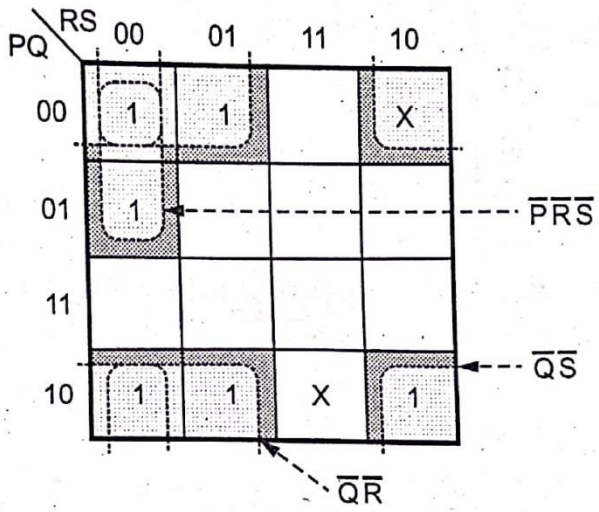


Fig. 7.3.28 (b)

Example 7.3.21 *Reduce the following functions using K-map techniques.*

i) $f(P,Q,R,S) = \sum m(0,1,4,8,9,10) + d(2,11)$ **VTU : Feb.-10, Marks 3**

Solution :



$f(P,Q,R,S) = \overline{Q}\,\overline{R} + \overline{Q}\,\overline{S} + \overline{P}\,\overline{R}\,\overline{S}$

Fig. 7.3.29

Example 7.3.22 *Find the minimal SOP form for the given min-terns using K-map.*

$F(A, B, C, D) = \sum m(4, 5, 6) + d(10, 12, 13, 14, 15)$. **VTU : Jan.-19, Marks 3**

Solution :

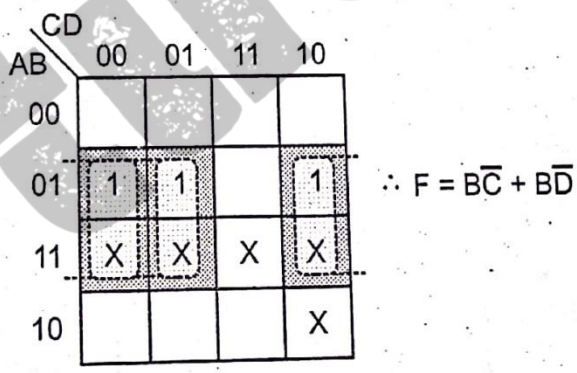


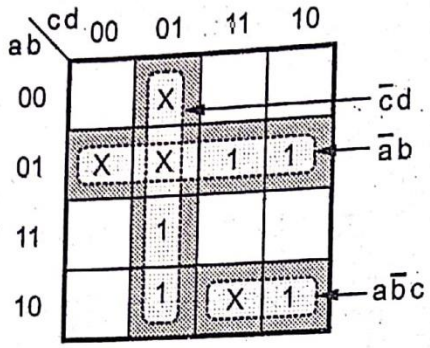
Fig. 7.3.30

Example 7.3.23 *Find the minimal SOP (sum of product) for the following Boolean functions using K - map*

i) $f(a, b, c, d) = \sum m(6,7,9,10,13) + d(1,4,5,11)$

ii) $f(a, b, c, d) = \pi M(1,2,3,4,10) + d(0,15)$. **VTU : July-17, Marks 8**

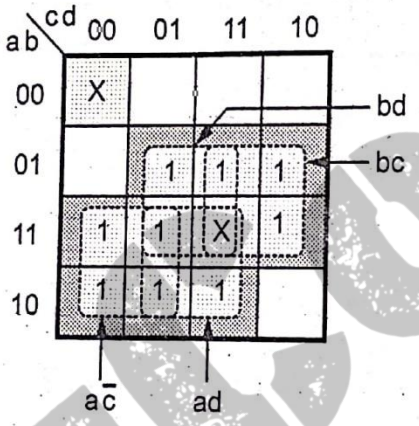
Solution : i)



$\therefore f = \bar{c}d + \bar{a}b + a\bar{b}c$

$$f = \pi M(1,2,3,4,10) + d(0,15) = \sum m(5,6,7,8,9,11,12,13,14) + d(0,15)$$

ii)



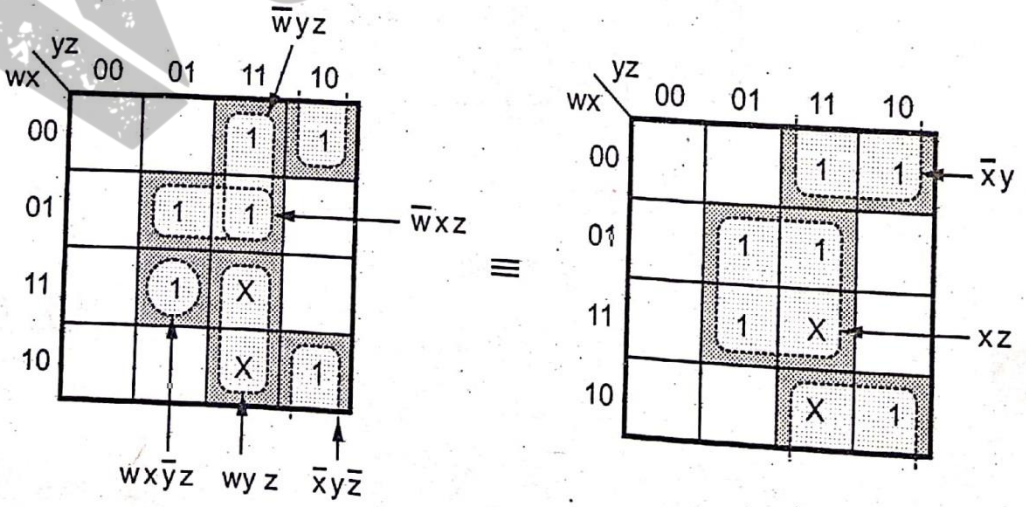
$\therefore f(a,b,c,d) = a\bar{c} + ad + bd + bc$

Example 7.3.24 *Simplify the following function using K-map and design it by using NAND gates (use only four gates) :*

$f = w'xz + w'yz + x'yz' + wxy'z \; ; d = wyz$

VTU : July-11, Marks 8

Solution : K-map simplification :



$\therefore \quad f = \bar{x}y + xz$

Fig. 7.3.31

Implementation

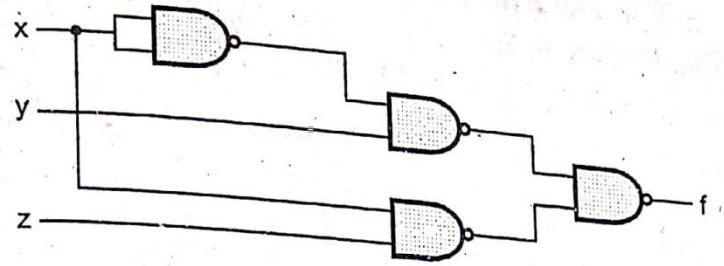


Fig. 7.3.32

Example 7.3.25 *A digital system is to be designed in which the months of the year is given as input in four bit form. The month January is represented as '0000', February as '0001' and so on. The output of the system should be '1' corresponding to the input of the month containing 31 days or otherwise it is '0'. Consider the excess numbers in the input beyond '1011' as don't care conditions. For this system of four variables (A, B, C, D), find the following :*

i) Boolean expression in $\sum m$ and πM form. ii) Write the truth table.

iii) Using K-map, simplify the Boolean expression of canonical minterm form.

iv) Implement the simplified equation using NAND-NAND gates.

VTU : Dec.-11, Jan.-14, 17, Marks 10

Solution : Truth table

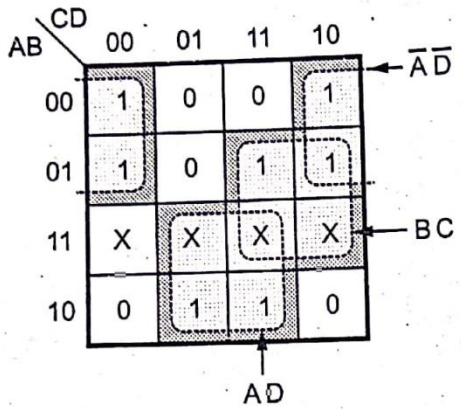
A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Boolean expression in $\sum$m and πM form

$$Y = \sum m\,(0, 2, 4, 6, 7, 9, 11) + d\,(12, 13, 14, 15)$$

$$Y = \pi M\,(1, 3, 5, 8, 10) + d\,(12, 13, 14, 15)$$

K - map simplification



$\therefore Y = \overline{A}\,\overline{D} + BC + AD$

Implementation

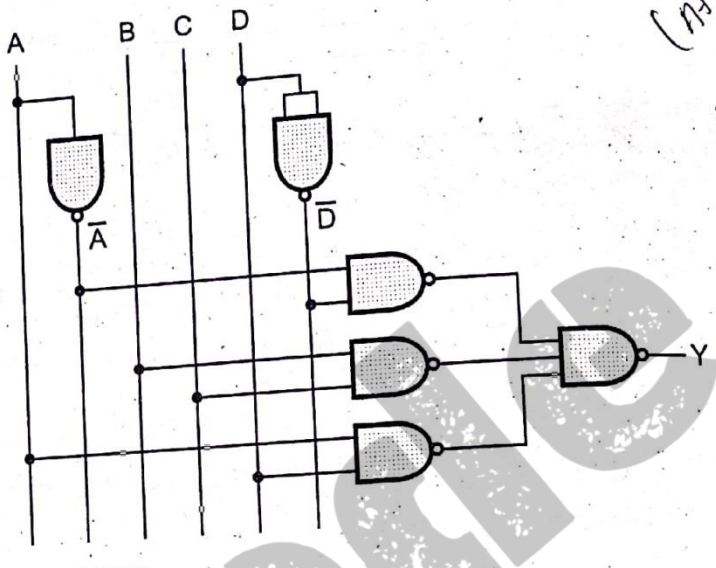


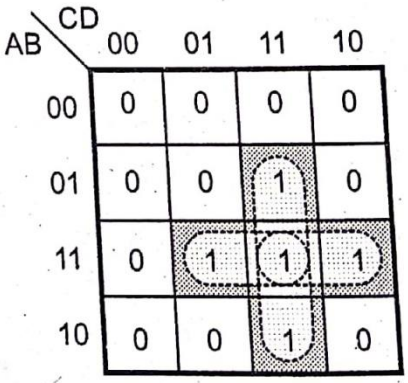
Fig. 7.3.33

Example 7.3.26 *The system has four inputs, the output will be high only when the majority of the inputs are high. Find the following. i) Give the truth table and simplify by using K - map. ii) Boolean expression in $\sum$ m and $\prod M$ form iii) Implement the simplified equation using NAND-NAND gates and NOR – NOR gates.* **VTU : Jan.-15, Marks 10**

Solution : Truthe Table

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

K- map simplification



$Y = ABD + ABC + BCD + ACD$

Fig. 7.3.34

$Y = \Sigma m\ (7, 11, 13, 14, 15)$

$Y = \pi M\ (0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 12)$

NAND - NAND implementation

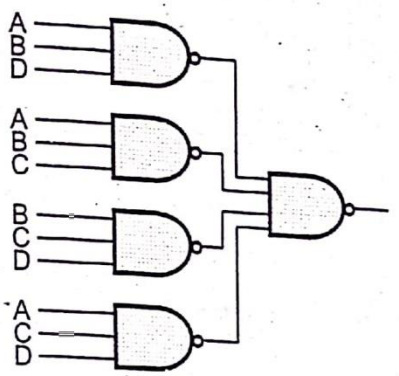


Fig. 7.3.35

NOR – NOR Implementation

$$Y = \overline{\overline{Y}} = \overline{\overline{ABD + ABC + BCD + ACD}}$$

$$= \overline{\overline{(ABD)} \cdot \overline{(ABC)} \cdot \overline{(BCD)} \cdot \overline{(ACD)}}$$

$$= \overline{(\overline{A} + \overline{B} + \overline{D}) \cdot (\overline{A} + \overline{B} + \overline{C}) \cdot (\overline{B} + \overline{C} + \overline{D}) \cdot (\overline{A} + \overline{C} + \overline{D})}$$

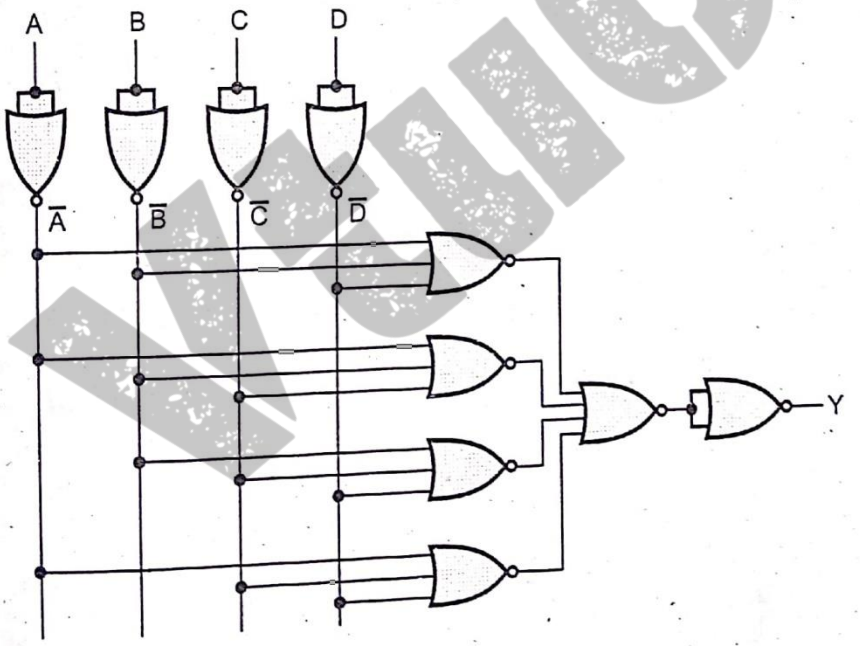
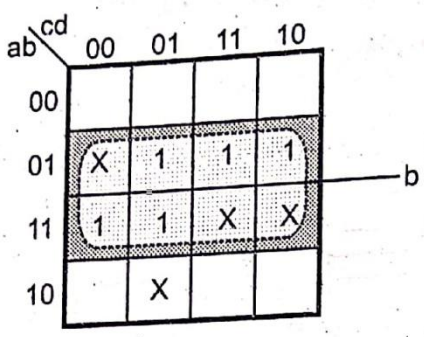


Fig. 7.3.36

Example 7.3.27 *Minimize the following Boolean function using K - map method*
$f(a,b,c,d) = \Sigma m\ (5, 6, 7, 12, 13) + \Sigma d\ (4, 9, 14, 15)$ **VTU : Jan.-17, Marks 6**

Solution :



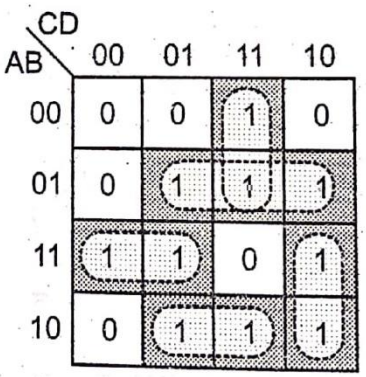
$\therefore f(a, b, c, d) = b$

Example 7.3.28 *Design a logic circuit to provide an output when any two or three of four switches are closed.* **VTU : July-18, Marks 5**

Solution : Step 1 : Derive the truth table for given problem statement. Assume that truth table indicates - 0 for switch open and 1 for switch close.

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

Step 2 : K-map simplification



$Y = \overline{A}CD + \overline{A}BC + \overline{A}BD + AB\overline{C} + A\overline{B}D + AC\overline{D}$

Fig. 7.3.37

Logic Diagram :

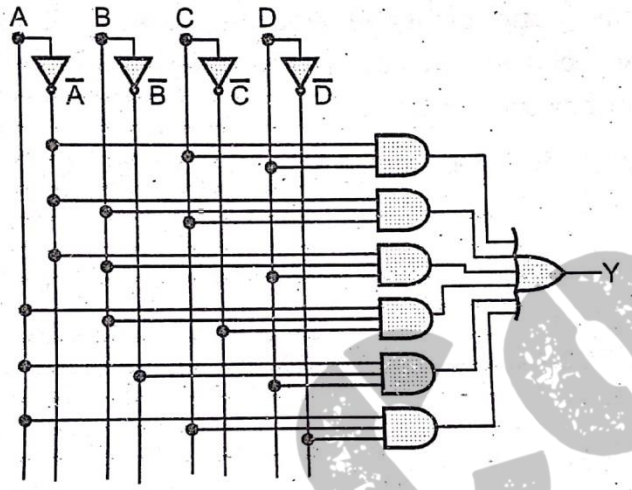


Fig. 7.3.38

Review Questions

1. *What is don't care condition.*
2. *Simplify the following switching function using Karnaugh map*
   $F(A,B,C,D) = \Sigma(0,5,7,8,9,10,11,14,15) + \phi(1,4,13)$. **[Ans. : $\overline{B}\,\overline{C} + BD + AC$]**
3. *Determine the minimal sum of product form of*
   $F(w,x,y,z) = \Sigma m(4,5,7,12,14,15) + d(3,8,10)$. **[Ans. : $\overline{w}\,x\,\overline{y} + xyz + w\overline{z}$]**