2

LOGIC SIMPLIFICATION

2.1 INTRODUCTION

Logic simplification means the way of reducing the equation which involves many variables. A variable is a symbol used to represent the logic quantity. For example, '*A*' is a variable which can take either '1' or '0' value. In order to simplify any equations which is represented by many variables the following methods are used:

- By Boolean algebra
- By K-map (Karnaugh Map)
- By Quine-Mc clusky (Tabular)

— The first method, i.e. Boolean Algebra deals with various Boolean rules or laws to reduce the number of variables.

— K-map is the graphical representation of the equation which is to be solved.

Quine-McClusky Method is more systematic reduction technique and used for solving large number of variables.

The merit of using these methods are:

- The circuit is going to be reduced means less number of gates are required to design the circuit means reducing hardware required.
- Time saving is also a merit of using logic simplification techniques.
- The cost of circuit is reduced.

In This Chapter

- Introduction
- Formation of Truth table & Boolean equation for simple Problem
- Standard Representation of Boolean Function
- Sum of Product form (SOP)
- Product of Sum form (POS)
- Minterm/Maxterms
- Simplification of of Boolean Expressions with the Help of Rules & Laws of Boolean Algebra
- K-maps
- Simplification of Boolean Function using K-maps

2.2 FORMATION OF TRUTH TABLE AND BOOLEAN EQUATION FOR SIMPLE PROBLEM

In order to find the truth table, we first need the Boolean equation. So in order to find the Boolean

equation or expression for any logic circuit or problem following steps are required:

(*i*) Start from the left most side of the logic circuit.

(*ii*) Then from the left most side, write the Boolean expressions and they may be input to any other circuit and do the operation until the final output is reached.

Statement: Start with left most side, i.e., input signals and develop the Boolean equation until the output is reached for finding Boolean equation.

Let us take an example:

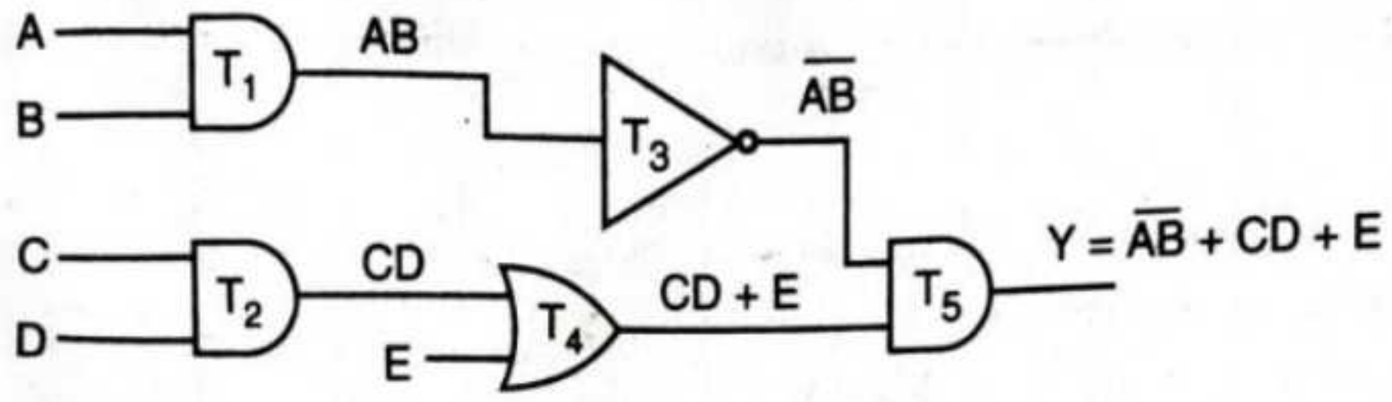


Fig. 2.1.

Let us represent each gate by certain symbol, i.e., T_1, T_2, T_3, T_4 and T_5.

Start from left most side, so inputs A and B is fed to AND gate T_1, so the output is AB. Then this output AB acts as the input of NOT gate i.e., T_3. The output generated at T_3 is $\overline{AB}$.

Then at AND gate T_2 inputs are C and D so, the output is CD which is further acts as the input to OR gate T_4 the another input to OR gate T_4 is E. The output which is generated at OR gate T_4 is $CD + E$. This output at T_4 acts as input to OR gate T_5 and the another input at OR gate T_5 is $\overline{AB}$. So the final output, i.e., Y is given as $\overline{AB} + CD + E$.

Formulation of Truth Table for Boolean Expression

In order to find the truth table, the first requirement is the Boolean expressions, then truth table is build Truth Table shows the output for all possible values of input variables. If 4 variables are involved in the Boolean expression or equation, then 16 (2^4) combinations are used starting from 0000 to 1111.

Let us take the previous example i.e.,

$$\boxed{Y = \overline{AB} + CD + E}$$

Here five variables are present, so we make 32 (2^5) combinations of inputs starting from 00000 to 11111.

To evaluate the expression $Y = \overline{AB} + CD + E$, we find the values of variables that make the expression equal to 1 using the rules of Boolean addition and multiplication. In this case the output 'Y' is '1' when $\overline{AB} = 0$, $CD = 1$ or 0, $E = 1$ or 0.

When $\overline{AB} = 0$, the values of A and B are

$A = 0, \quad B = 1$

$A = 1, \quad B = 0$

$A = B = 0$

$\Rightarrow$ $CD = 1$, when $C = D = 1$

$\Rightarrow$ $E = 1$

when $CD = 1$, then E may take '0' value and when $E = 1$, CD may take 0 value for CD = 0 when

$C = 0$ and $D = 1$

$C = 1$ and $D = 0$

$C = D = 0$

So, using these values, we can construct the truth Table as follows:

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

EXAMPLE 2.1. Find the truth Table of the logic circuit

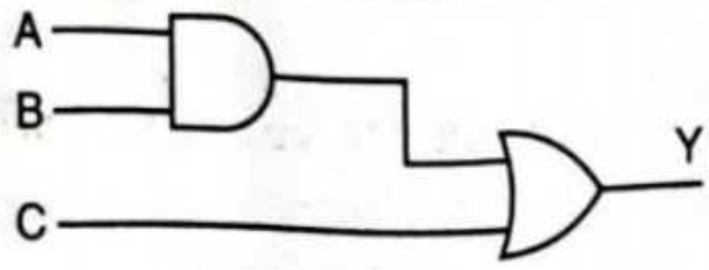


Fig. 2.2.

Solution: Steps for finding the truth Table:

(1) First step is to find the Boolean equation of the logic circuit.

(2) Then applying certain Boolean addition and multiplication rules to find out the truth Table.

In order to make the truth Table, label the gates by T_1, T_2, i.e.,

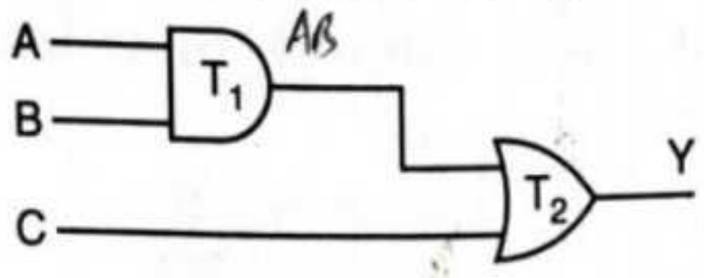


Fig. 2.3.

Here T_1 is the AND gate and the inputs of this T_1 is A and B and the output is AB and this output acts as the input to OR gate T_2. The another input the OR gate T_2 is C. Hence, the result of this OR gate is $AB + C$ which is equal to Y.

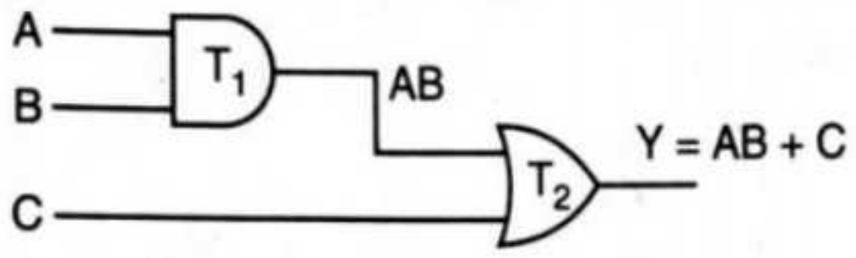


Fig. 2.4.

Formulation of Truth Table

To find the truth Table, the output term contain '3' variables ie., A, B and C, so 8 ($2^3 = 8$) combinations are formed from "000 to 111".

To evaluate this logic equation, i.e.

$$Y = AB + C$$

The output is '1' when either $AB = 1$ or $C = 1$ or both $AB = C = 1$

AB is '1' when $A = B = 1$

So, using these values, we can construct the truth Table, i.e.

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	1	—	*
1	1	1	1	—	*

Here according to the rules of Boolean algebra we place '1' at $A = B = 1$ or $C = 1$

or $AB = C = 1$.

As shown by (*) and in rest of the terms, i.e. when

A	B	C	
0	0	0	
0	1	0	→ Combinations
1	0	0	

Place Zero's.

EXAMPLE 2.2. Find the truth Table of the given logic circuit

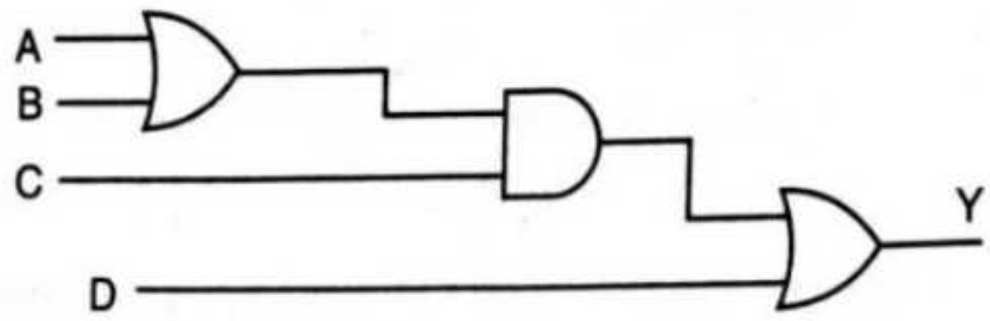


Fig. 2.5.

Solution: In order to make the truth Table first step is to find the Boolean equation of the logic circuit.

Label all gates by T_1, T_2 and T_3

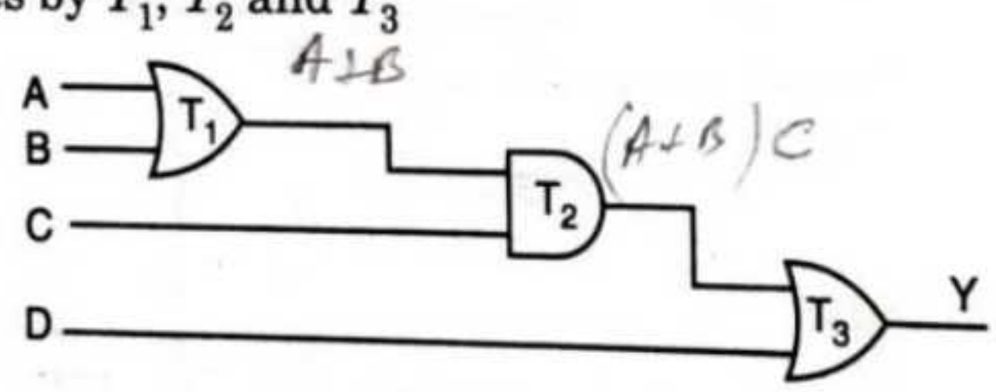


Fig. 2.6.

For gate T_1 i.e., OR gate the output is $(A + B)$. This $(A + B)$ acts as the input of AND gate T_2 and other input is C, so the output of AND gate T_2 is given as $(A + B)\ C$. This output acts as the input to OR gate T_3 and other input is D and the final output is given as $Y = (A + B)C + D$ as shown in diagram.

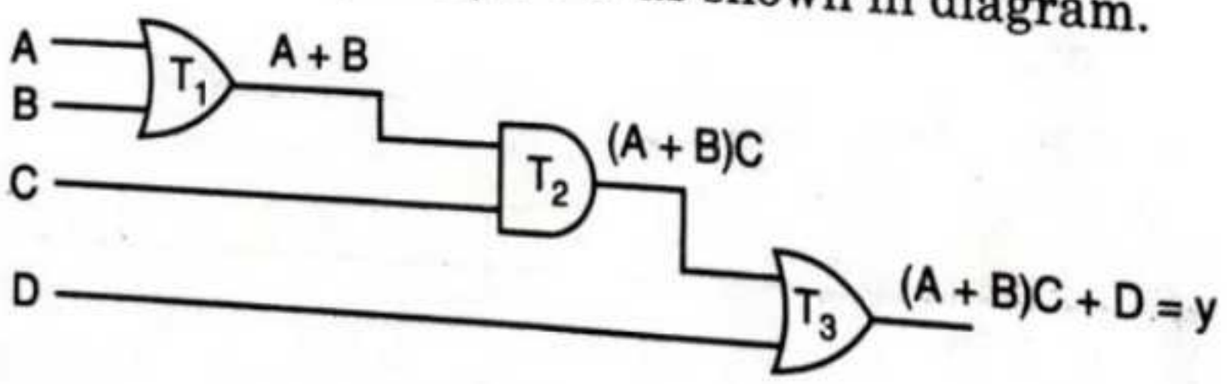


Fig. 2.7.

Formulation of Truth Table

Since the output equation (Y) contain four variables, i.e. A, B, C and D, so 16 ($2^4 = 16$) combinations are formed starting from "0000 to 1111".

In order to evaluate the logic expression, we apply Boolean addition and multiplication rules.

i.e.

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

The output is '1' when $(A+B)C=1$ or $D=1$ or both $(A+B)C=D=1$

For $(A+B)C=1$, $C=$ '1' and $A+B=$ '1'

$A+B=1$, when either $A=0$ and $B=1$

$A=1$ and $B=0$

$A=1=B$

so using these values, we can construct the truth Table as follows:

A	B	C	D	Y	
0	0	0	0	0	
0	0	0	1	1	— *
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	0	
1	1	0	1	1	— *
1	1	1	0	1	— *
1	1	1	1	1	— *

'*' values are generated by using the Boolean addition and multiplication rules and place zero's in rest of the terms, i.e.

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

EXAMPLE 2.3. Write the Boolean equation for the logic diagrams.

(1)

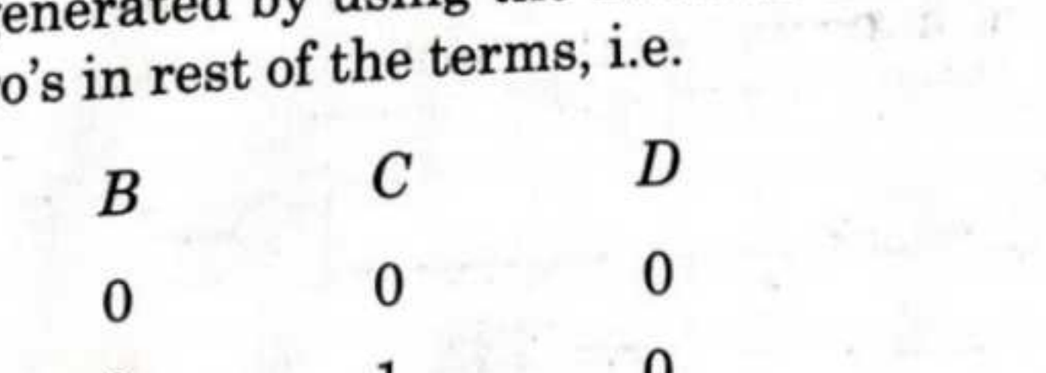


Fig. 2.8.

(2)

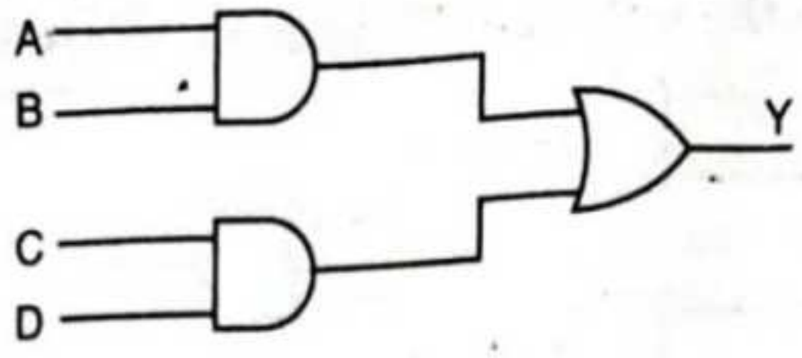


Fig. 2.9.

(3)

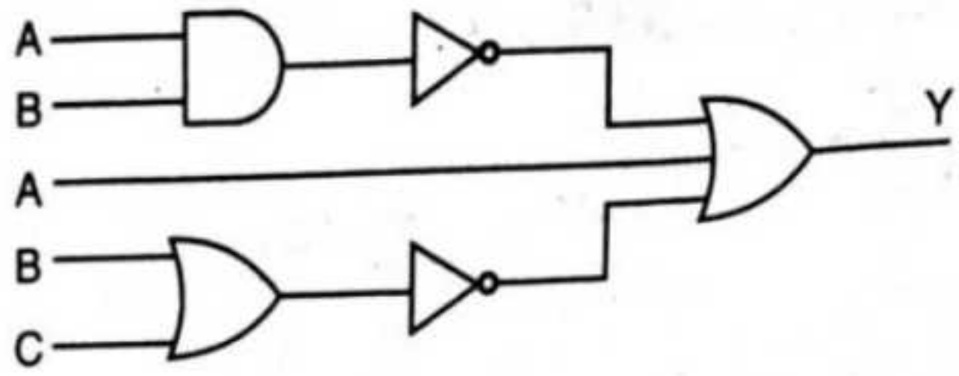


Fig. 2.10.

(4)

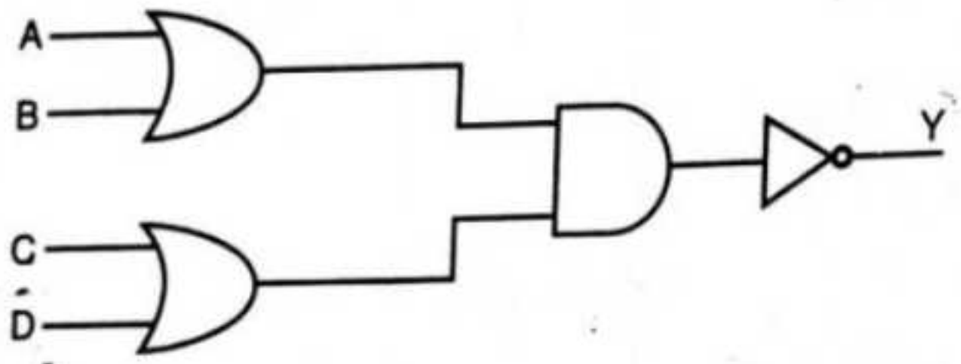


Fig. 2.11.

Solution: Let us take (1) problem, i.e.

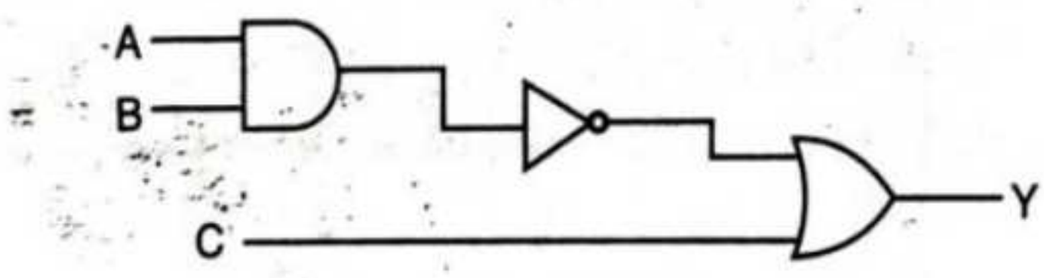


Fig. 2.12.

Label the logic circuit with T_1, T_2 and T_3.

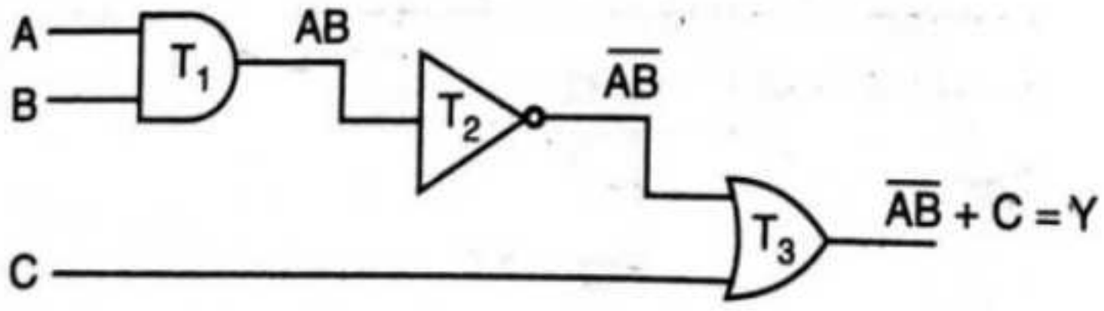


Fig. 2.13.

For AND gate T_1

Inputs are A, B

Output = AB

For NOT gate T_2 Input = AB

Output = $\overline{AB}$

For OR gate T_3 Inputs = $\overline{AB}$ and C

Output = $\overline{AB} + C$

So, the final output is equal to the output of T_3 gate, i.e. $\boxed{\overline{AB} + C = Y}$

(2)

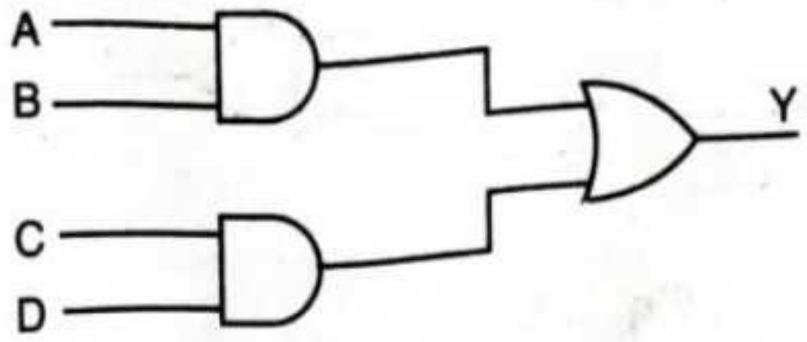


Fig. 2.14.

Level the logic circuit with T_1, T_2 and T_3, i.e.

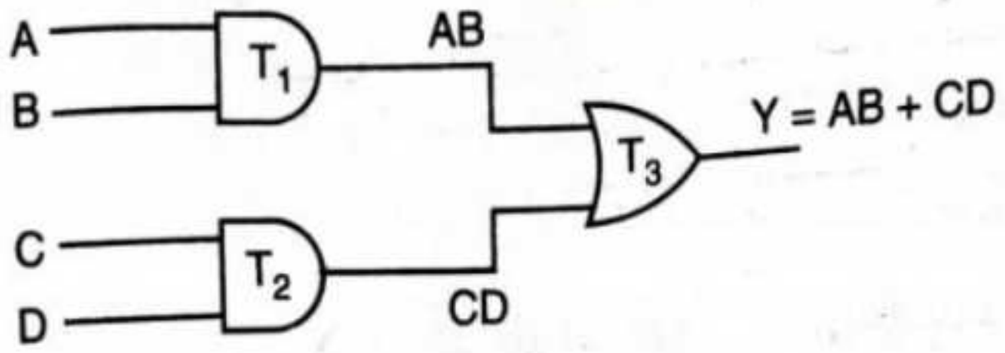


Fig. 2.15.

For AND gate T_1

Inputs = A & B

Output = AB

For AND gate T_2

Inputs = C & D

Output = CD

For OR gate T_3 Inputs = AB & CD

Output = $AB + CD$

So the final output 'Y' equal to the output of OR gate T_3, i.e. $\boxed{AB + CD}$

(3)

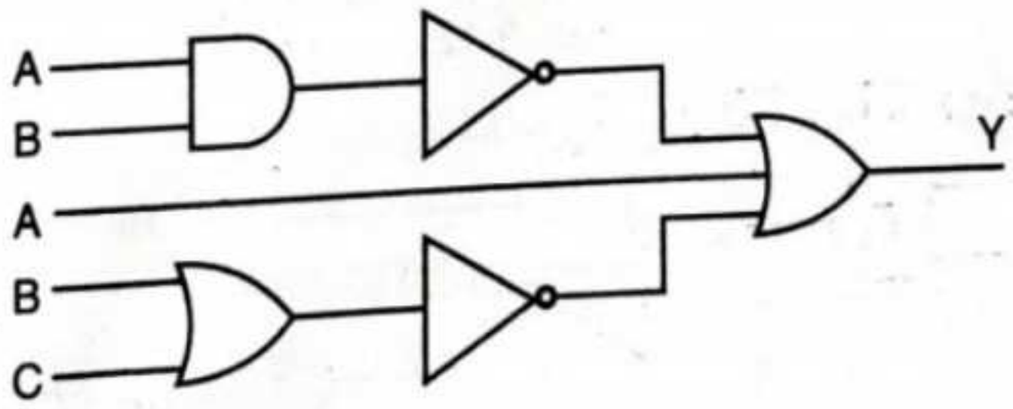


Fig. 2.16.

Level the logic circuit with T_1, T_2, T_3, T_4 and T_5, i.e.

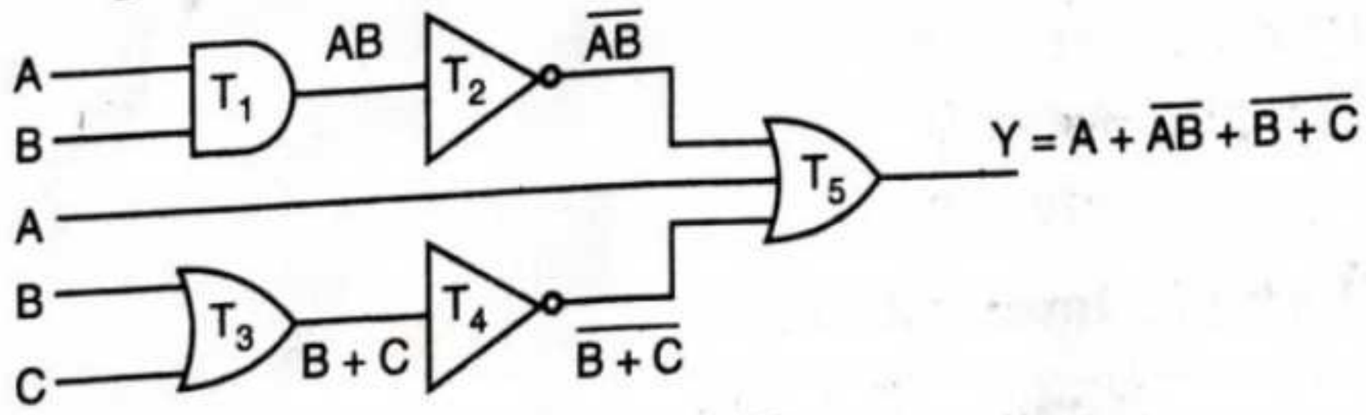


Fig. 2.17.

For AND gate T_1

Inputs = A & B

Output = AB

For NOT gate T_2

$$\text{Inputs} = AB$$

$$\text{Output} = \overline{AB}$$

For OR gate T_3

$$\text{Inputs} = B \,\&\, C$$

$$\text{Output} = B + C$$

For NOT gate T_4

$$\text{Input} = B + C$$

$$\text{Output} = \overline{B+C}$$

For OR gate T_5

$$\text{Inputs} = \overline{AB}, A \,\&\, \overline{B+C}$$

$$\text{Output} = \overline{AB} + A + \overline{B+C}$$

The final output 'Y' equal to OR gate T_5 and it is given as:

$$\boxed{\overline{AB} + A + \overline{B+C}}$$

(4)

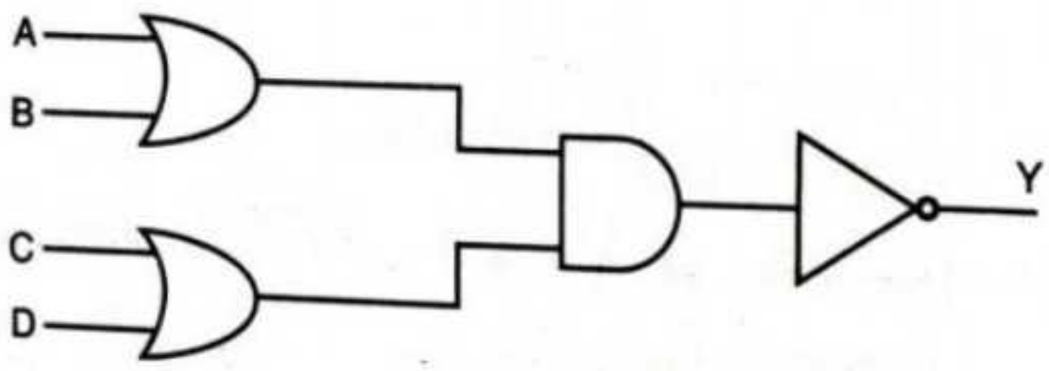


Fig. 2.18.

Label the logic circuit with T_1, T_2, T_3 & T_4

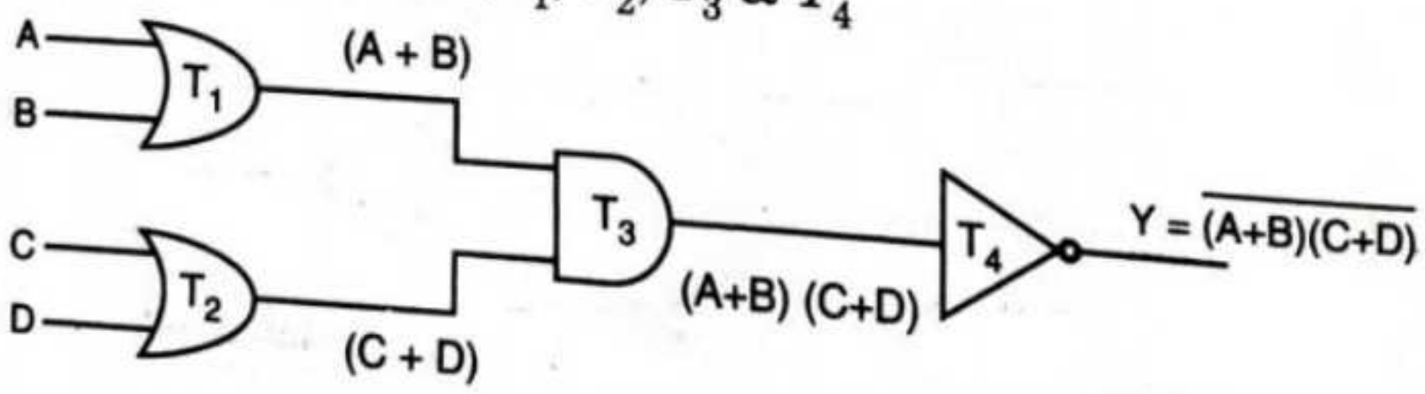


Fig. 2.19.

For T_1 OR gate

$$\text{Inputs} = A \,\&\, B$$

$$\text{Output} = A + B$$

For OR gate T_2

$$\text{Inputs} = C \,\&\, D$$

$$\text{Output} = C + D$$

For AND gate T_3

$$\text{Inputs} = (A + B) \,\&\, (C + D)$$

$$\text{Output} = (A + B)\,(C + D)$$

For NOT gate T_4

$$\text{Input} = (A + B)\,(C + D)$$

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

The output 'Y' is equal to NOT gate T_4 and it is given as:

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

2.3 IMPLEMENTATION OF BOOLEAN (LOGIC) EQUATION WITH GATES

In order to convert Boolean equation to gates following technique is used:

Start with the output and work towards the input. e.g., In order to realize this Boolean equation i.e., $\overline{ABC} + BC + \overline{A + C}$.

Here the final output is $\overline{ABC} + BC + \overline{A + C}$. Since it has three terms, i.e. $\overline{ABC}$, BC & $\overline{A + C}$ and all are sum together and hence for sum expression OR gate is used and it is of '3' input OR gate (because three output terms are there). It is represented as:

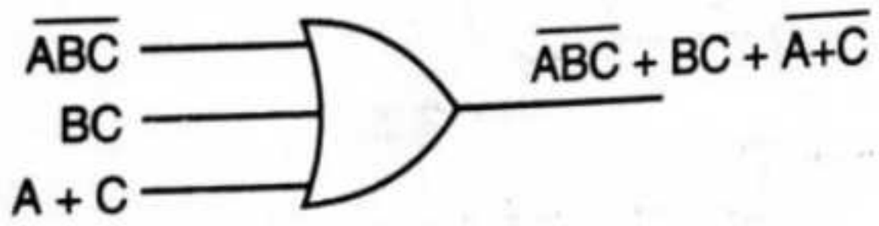


Fig. 2.20.

Then in order to realize $\overline{ABC} + BC + \overline{B + C}$ following gates are required.

The $\overline{ABC}$ is the output of the inverter whose input is ABC.

BC is the output of the AND gate with inputs B and C $\overline{A + C}$ is the output of the inverter where input is $A + C$. So, the gate representation is shown as follows:

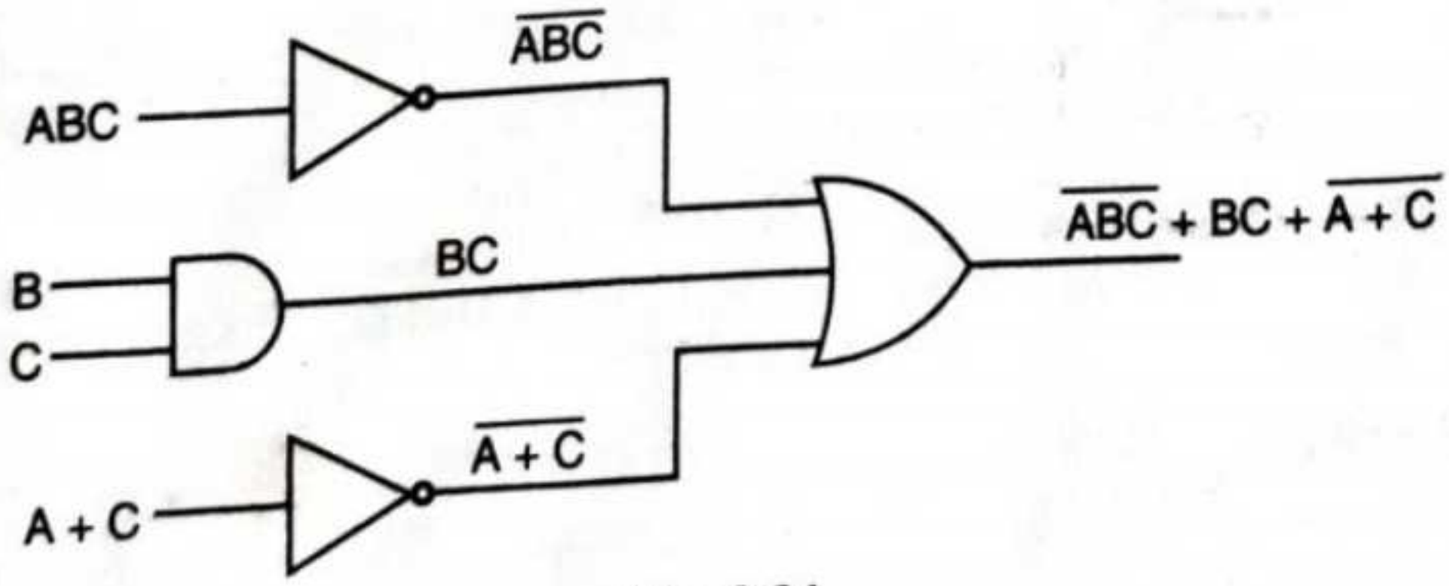


Fig. 2.21.

Now, ABC must be generated by three input AND gate with inputs A, B and C. $(A + C)$ is the output of OR gate whose inputs are $A + C$. So the gate representation is shown as follows:

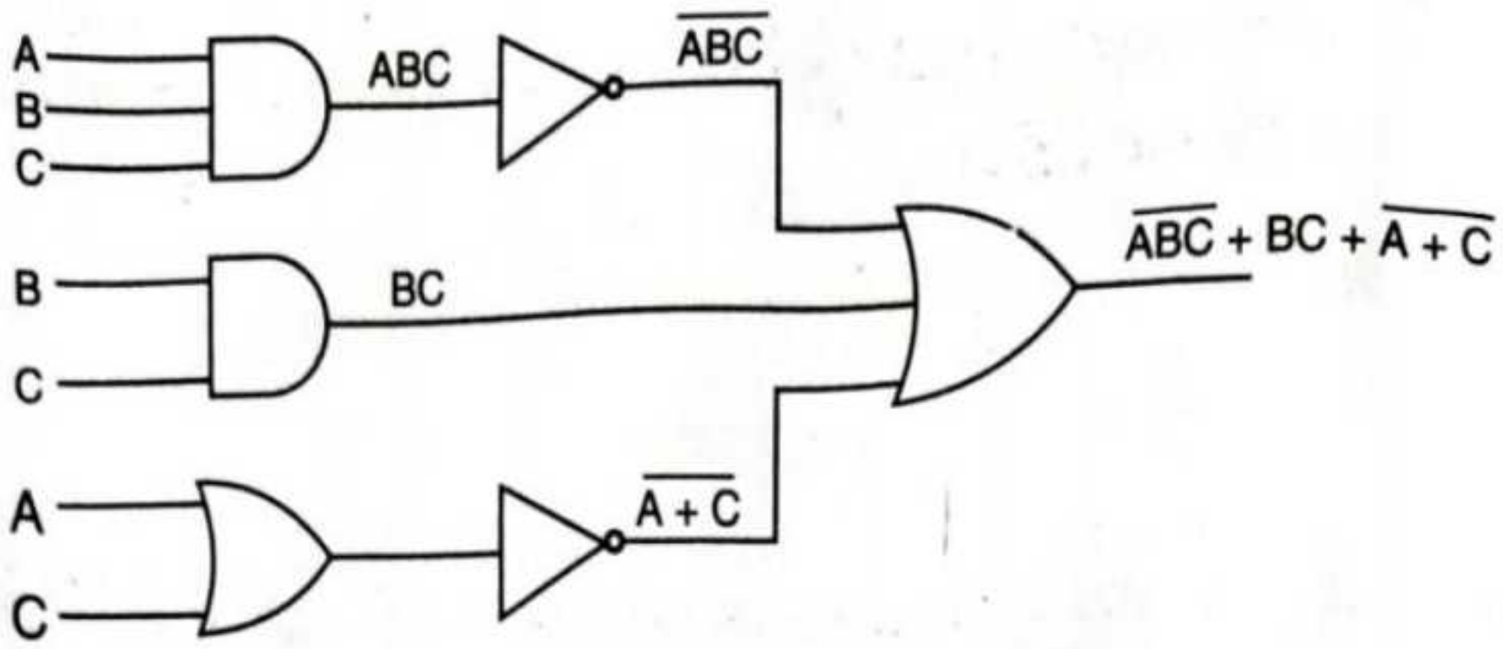


Fig. 2.22.

EXAMPLE 2.4. Implement the following Boolean equation with gates ($\overline{A+B}$) (AB) ($C+D$)

Solution: In order to implement the Boolean equation with gate start with the output, here three inputs are ($\overline{A+B}$), (AB) and ($C+D$). They required three input AND gate. It is represented as:

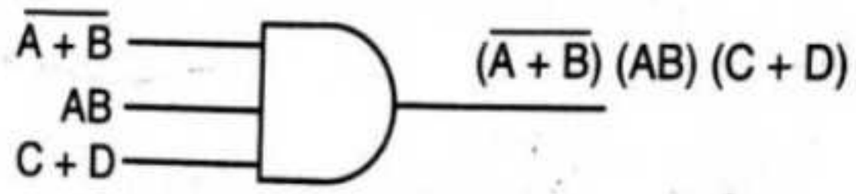


Fig. 2.23.

Then $\overline{A+B}$ is the output of the inverter with input $A+B$. AB is the output of AND gate with inputs A and B. ($C+D$) is the output of OR gate with inputs C and D. The realization is given as:

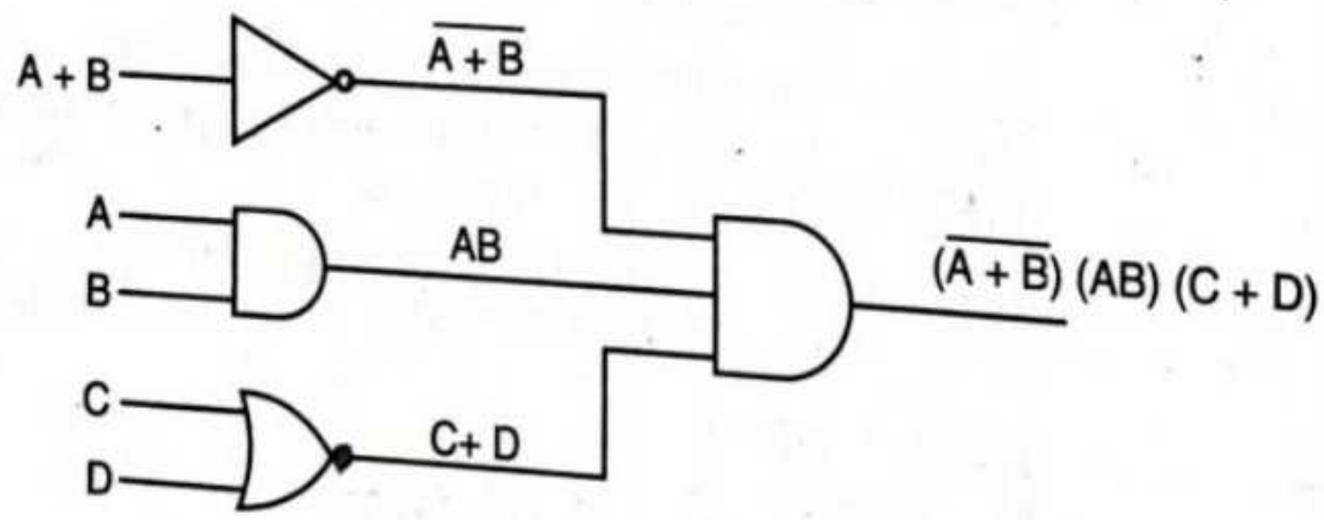


Fig. 2.24.

($A+B$) is the output of OR gate with input A and B. The gate representation is given as follows:

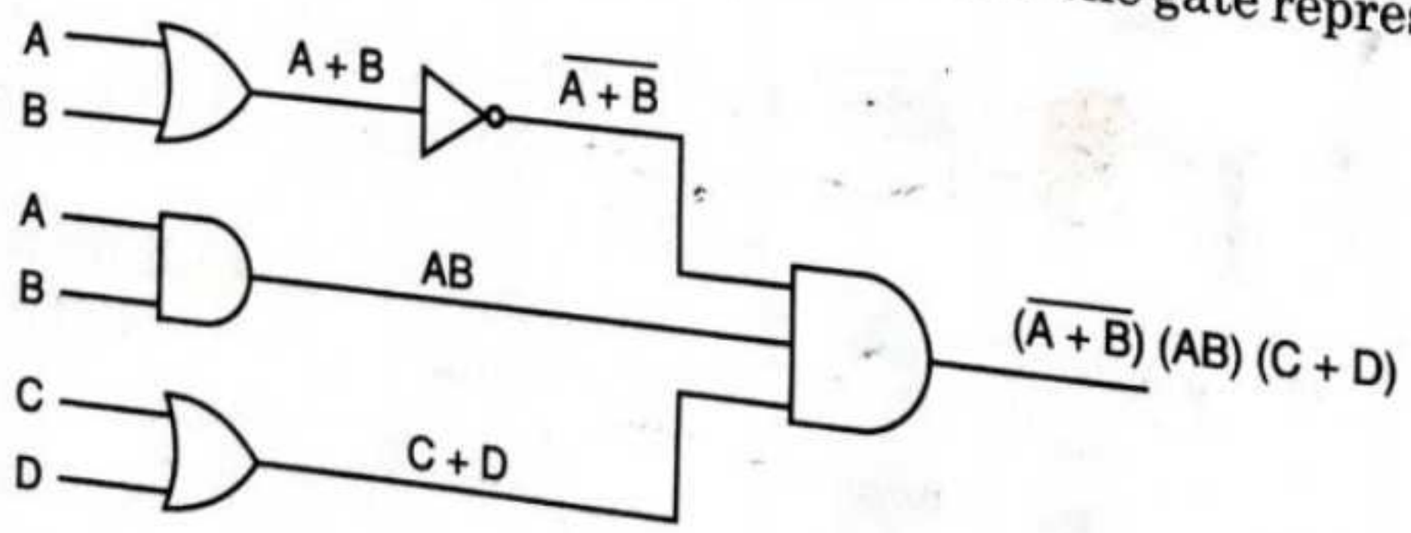


Fig. 2.25.

EXAMPLE 2.5. Implement the Boolean equation (AB) + $\overline{CD}$ using gates.

Solution: The given equation is $AB + \overline{CD}$. Here it requires two-input OR gate whose inputs are AB and $\overline{CD}$, i.e.

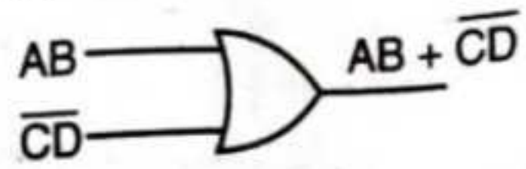


Fig. 2.26.

Then AB is the output of the two input AND gate with inputs A and B $\overline{CD}$ is the output of inverter with input CD, i.e.

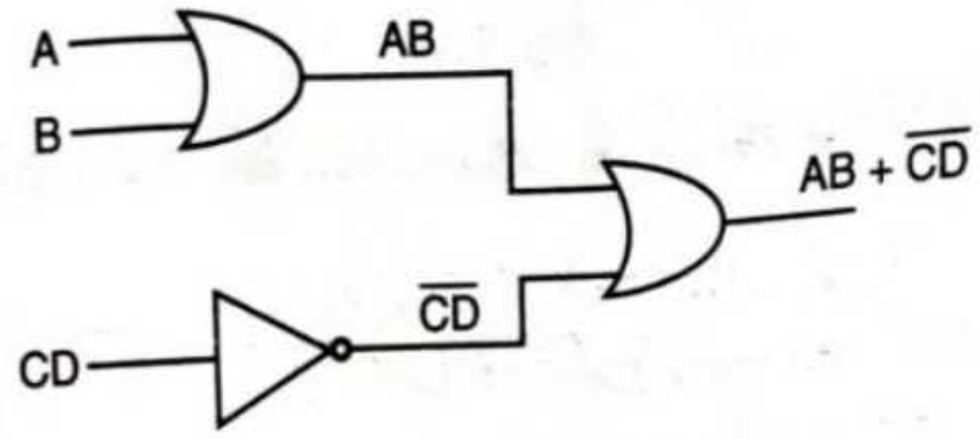


Fig. 2.27.

Then CD is the output of two-input AND gate with inputs C and D, i.e.

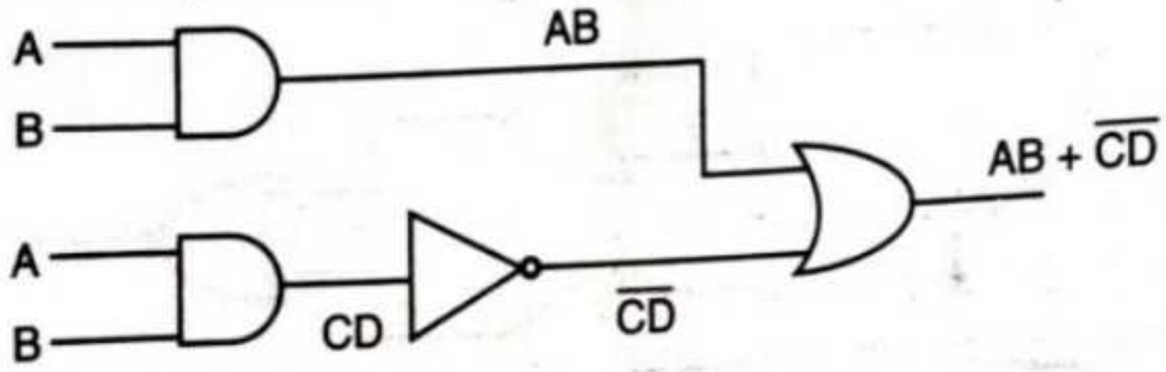


Fig. 2.28.

EXAMPLE 2.6. Implement the Boolean equation with logic gate, i.e. $(\overline{A})$ $(\overline{B+C})$ (CD).

Solution: Since the output expression requires three-input AND gate with inputs $\overline{A}$, $\overline{B+C}$ and CD. It is given as:

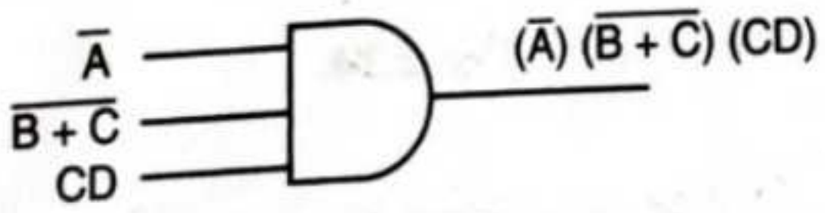


Fig. 2.29.

Then $\overline{A}$ is the output of inverter with input $A \cdot \overline{B+C}$ is the output of inverter with input $(B + C)$ CD is the output of AND gate with inputs C and D, i.e.

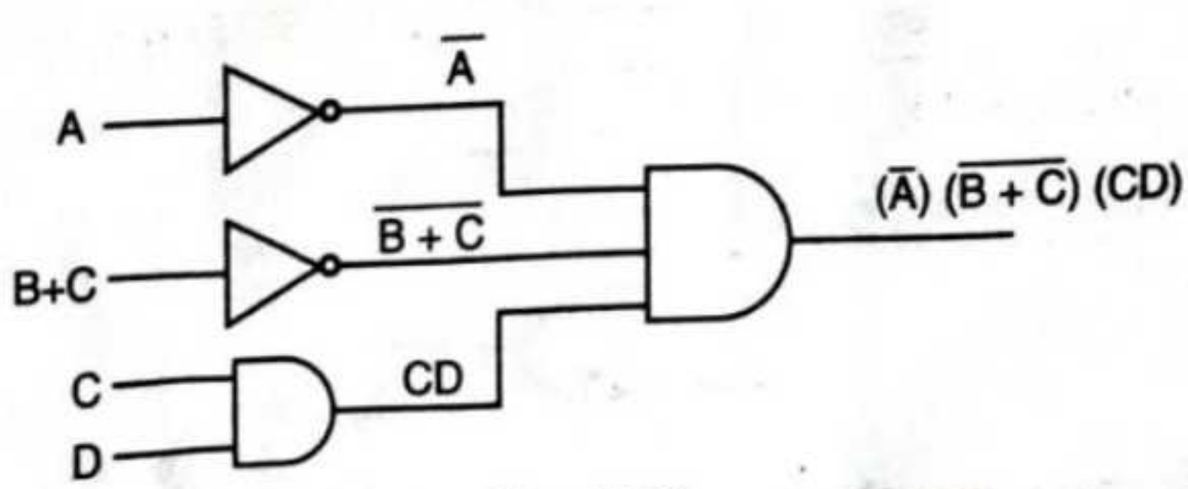


Fig. 2.30.

Then $B + C$ is the output of OR gate with inputs B and C. The gate realizati is given as:

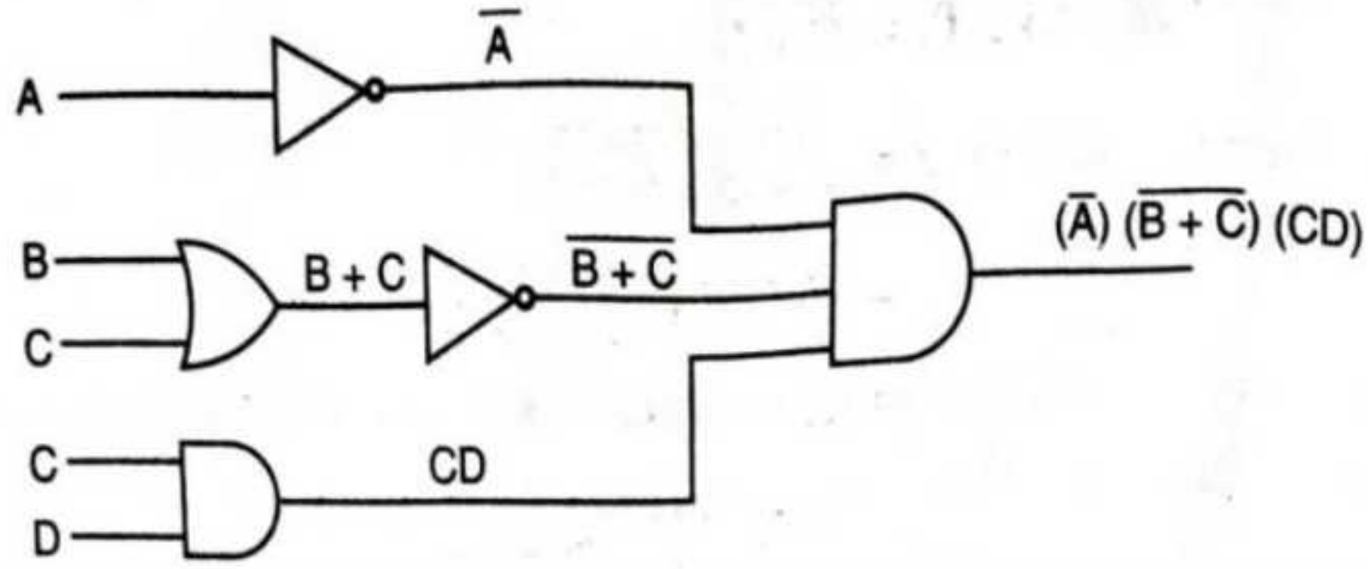


Fig. 2.31.

EXAMPLE 2.7. Write the Boolean expression for the logic circuits shown Figs:

(*i*)

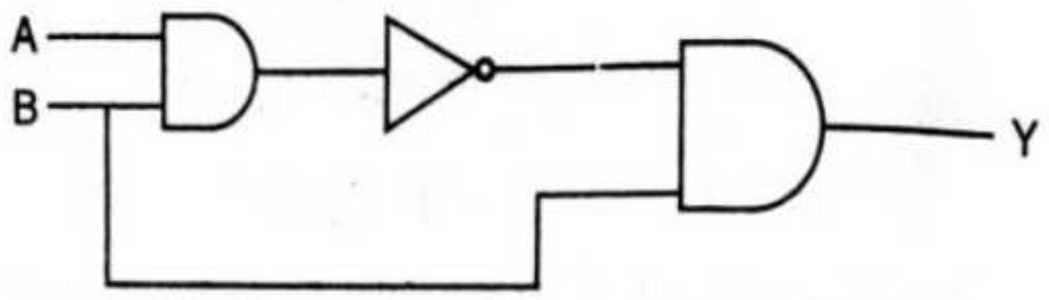


Fig. 2.32.

(*ii*)

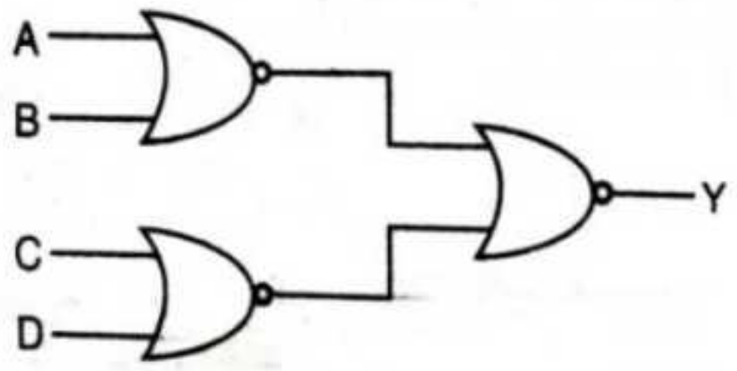


Fig. 2.33.

(*iii*)

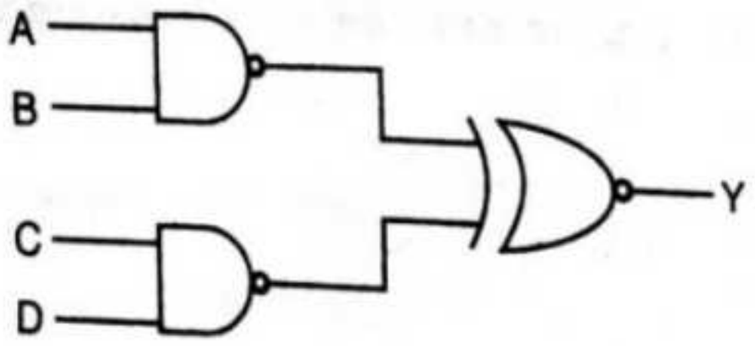


Fig. 2.34.

Solution:

(*i*)

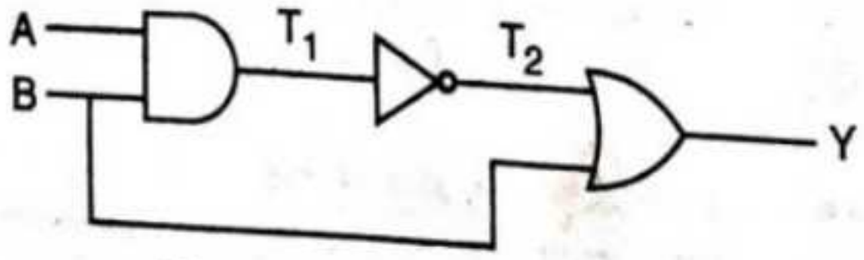


Fig. 2.35.

The output at $T_1 = AB$

Output at $T_2 = \overline{AB}$

Then the final output $Y = \overline{AB} + B$

(*ii*)

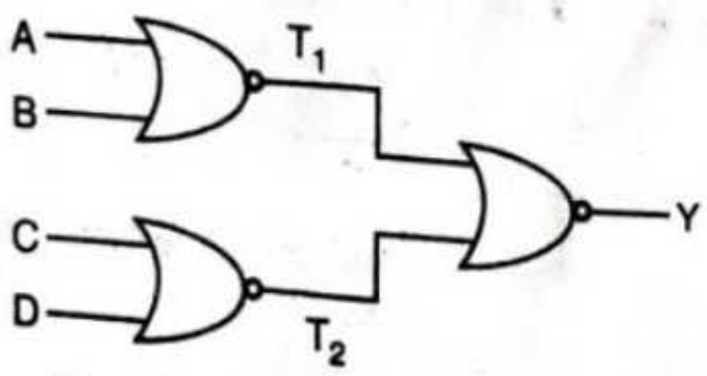


Fig. 2.36 (a).

The output at $T_1 = \overline{A+B}$

Output at $T_2 = \overline{C+D}$

Then the final output $Y = \overline{\overline{A+B}+\overline{C+D}}$

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

$$= (A+B)\,(C+D)$$

(*iii*)

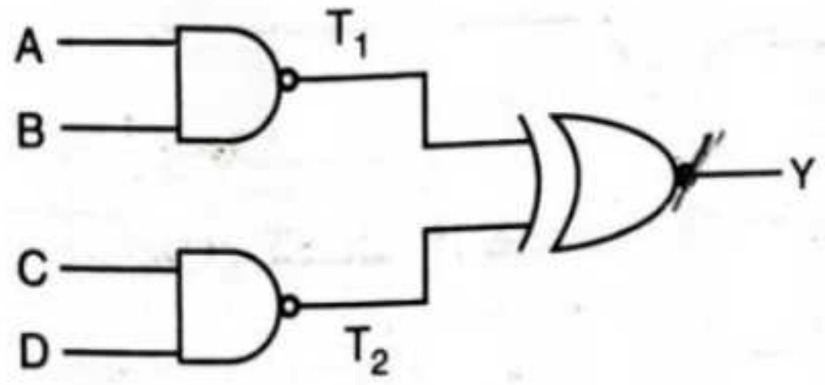


Fig. 2.36 (b).

Output at $T_1 = \overline{AB}$

Output at $T_2 = \overline{CD}$

Then the final output $Y = \overline{AB} \oplus \overline{CD}$

$$= \overline{\overline{AB}} \cdot \overline{CD} + \overline{AB} \cdot \overline{\overline{CD}}$$

$$= AB\,(\overline{CD}) + \overline{AB}\,(CD)$$

2.4 STANDARD REPRESENTATION OF BOOLEAN FUNCTION

Boolean function consist of Boolean variables (i.e., *A*, *B*, ...). Each variable have only two value either it is zero or it is one. The Boolean function can be represented into two standard forms as

(*i*) Sum of Product (SOP)

(*ii*) Product of Sum (POS)

2.4.1. Need of Standards

It makes the Boolean function or expression in more systematic and easier form by doing evaluation, simplification and implementation using these standards.

2.5 SUM OF PRODUCT FORM (SOP)

In sum of product form it requires two gates i.e., for sum 'OR' gate is required and for product 'AND' gate is required. In this form all variables (i.e., *A*, *B*, *C* ...) are combined to form various product (Boolean multiplication) which are known as minterms and then sum of all minterms or product terms gives us SOP form.

Let us take an **example**

$$Y = \underbrace{ABC}_{\text{Product term}} + \underbrace{A\,B\,\overline{C}}_{\text{Product term}} + \underbrace{A\,\overline{B}\,C}_{\text{Product term}} + \underbrace{\overline{A}\,\overline{B}\,C}_{\text{Product term}}$$

These product terms combined (means summed) together to form SOP form. So this expression is realized using AND-OR logic 'AND' is used for product term and 'OR' is used for sum term.

The implementation of SOP expression using gate :

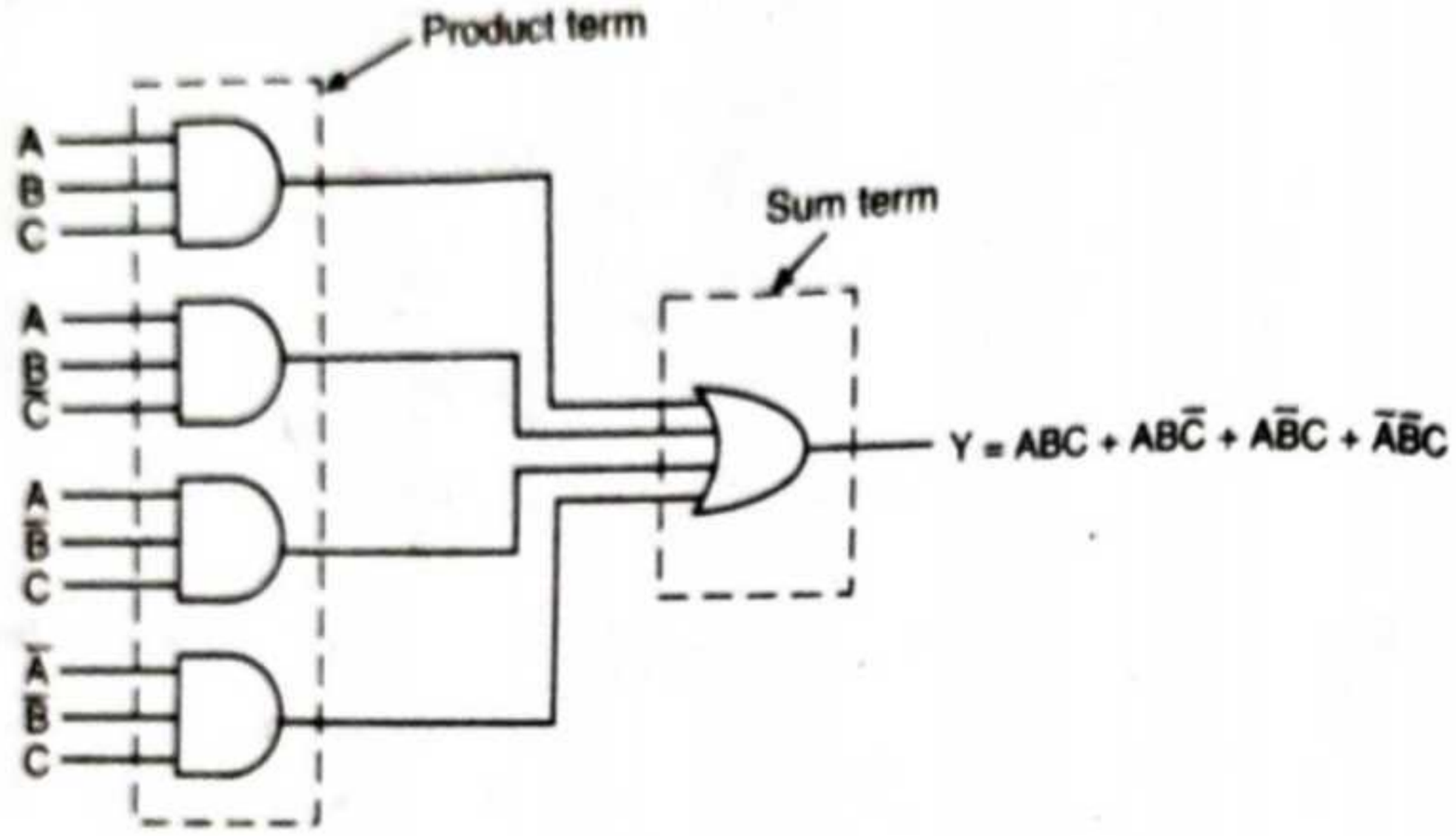


Fig. 2.37.

The SOP expression can be implemented using NAND gates (Universal gates)

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

Applying DeMorgan Theorem

$$\bar{Y} = \overline{ABC + AB\bar{C} + A\bar{B}C + \bar{A}\bar{B}C}$$

$$\bar{Y} = \overline{ABC} \cdot \overline{AB\bar{C}} \cdot \overline{A\bar{B}C} \cdot \overline{\bar{A}\bar{B}C} \quad [\overline{A+B} = \bar{A} \cdot \bar{B}]$$

Applying again DeMorgan Theorem

$$\bar{\bar{Y}} = Y = \overline{\overline{ABC} \cdot \overline{AB\bar{C}} \cdot \overline{A\bar{B}C} \cdot \overline{\bar{A}\bar{B}C}}$$

ABC is the output of AND and

$\overline{ABC}$ is the output of NAND

Similarly, $\overline{AB\bar{C}}$, $\overline{A\bar{B}C}$, $\overline{\bar{A}\bar{B}C}$ are the outputs of NAND gates.

So we require five NAND gates to realize this equation.

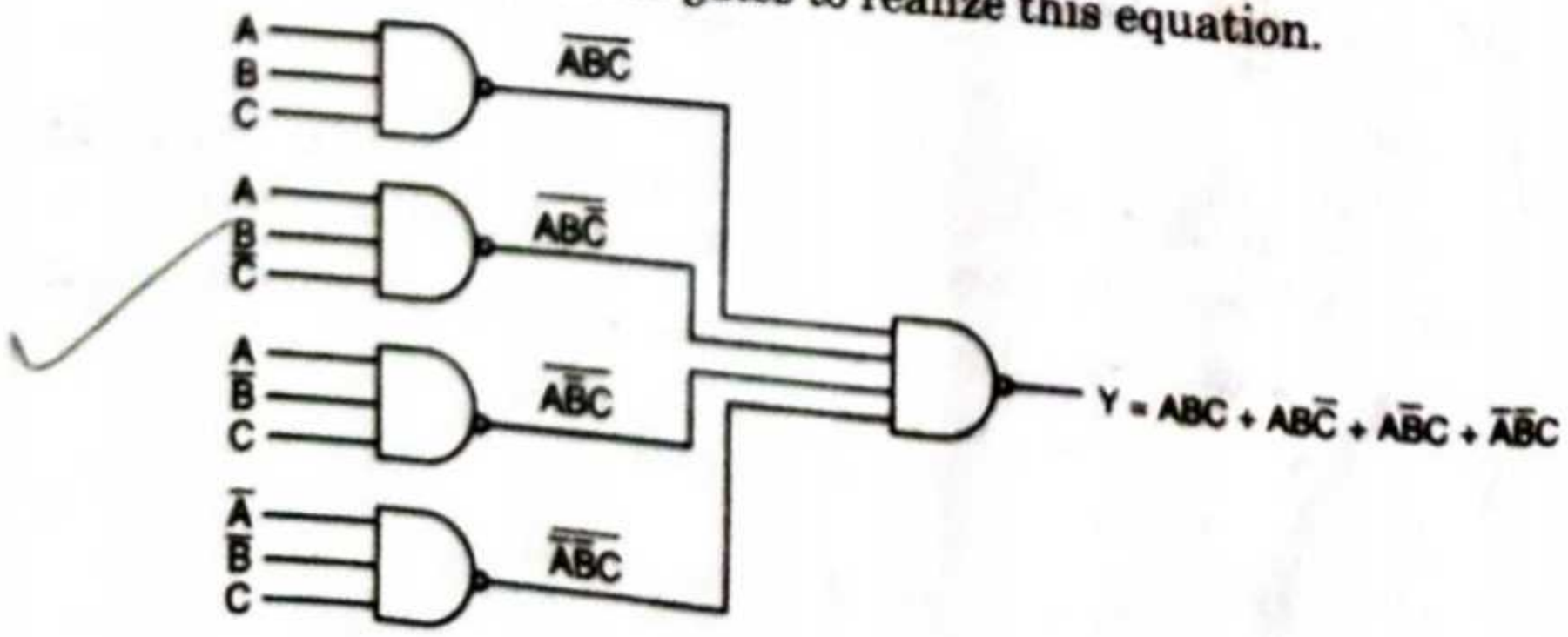


Fig. 2.38.

2.5.1. Minterm

The product term (i.e. ABC or $\overline{A}BC$) or minterm consist of product of all variable in complemented form or in true form.

Minterm is defined as the product term in SOP form.

They are represented by '1' in any truth table in SOP form.

Let us take an e.g.,

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

Here in table if the variable have zero value then it comes in it complemented form i.e. if $A = 0$, then it is represented as $\overline{A}$ or if the variable have one value, then it comes in its true form i.e., If $A = 1$, then it is represented as A. So $\overline{A}\,\overline{B}$, $\overline{A}B$, $A\overline{B}$ and AB are known as minterms.

SOP expression is given as:

$$Y = \underset{\text{Minterm}}{\overline{A}BC} + \underset{\text{Minterm}}{\overline{A}\,\overline{B}\,C} + \underset{\text{Minterm}}{ABC}$$

Or

$$Y = \underset{\text{Minterm}}{\overline{A}BC} + \underset{\text{Minterm}}{\overline{A}\,\overline{B}\,C} + \underset{\text{Minterm}}{ABC}$$

In numerical SOP representation, minterms can take the decimal notation rather than variable, i.e.

$$Y(A, B, C) = \underset{\text{Minterm symbol}}{\Sigma m}\ (3, 4, 5, 7)$$

Or

$$Y = \Sigma m\ (3, 4, 5, 7)$$

Or

$$Y = \Sigma\ (3, 4, 5, 7)$$

All the above equations have the same meaning and have minterms. In minterms, i.e. 3, 4, 5 and 7 we have to put '1' in these decimal numbers.

EXAMPLE 2.8. Convert the following expression in SOP form and design it by using gates.

(i) $A + BD\,(C + \overline{C})$

(ii) $(A + B)\,(C + D)$

(iii) $A\overline{B} + C\,(D + EF)$

(iv) $A\overline{B}\,C + (A + \overline{C})\,(B + D)$

Solution: (*i*) $A + BD\,(C + \overline{C}\,)$

In order to c vert into SOP form, remove the brackets by multiplying th variables, *i.e.*,

$$\Rightarrow \qquad A + BDC + BD\overline{C}$$

Gate Implementation:

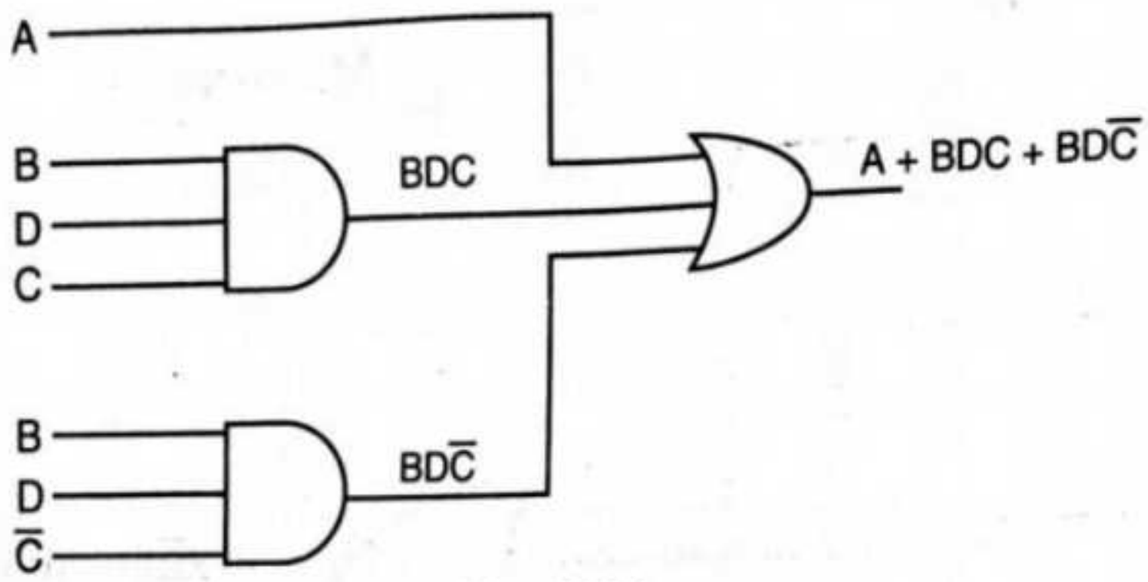


Fig. 2.39.

(*ii*) $(A + B)\,(C + D)$

Remove the brackets by applying Boolean multiplication

$$A(C + D) + B(C + D)$$

$$\Rightarrow \qquad AC + AD + BC + BD$$

Gate Implementation:

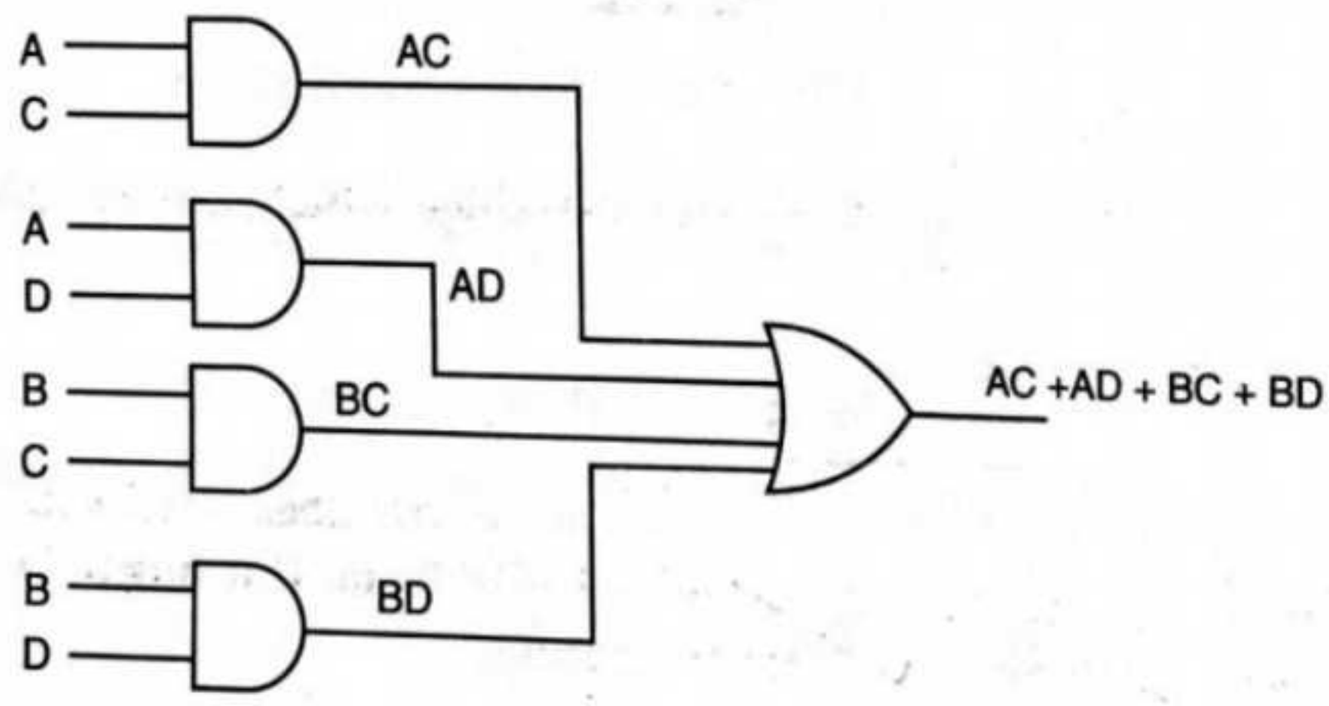


Fig. 2.40.

(*iii*) $A\overline{B} + C\,(D + EF)$

Remove the brackets by applying Boolean multiplication, i.e.

$$\Rightarrow \qquad A\overline{B} + CD + CEF$$

Gate Implementation:

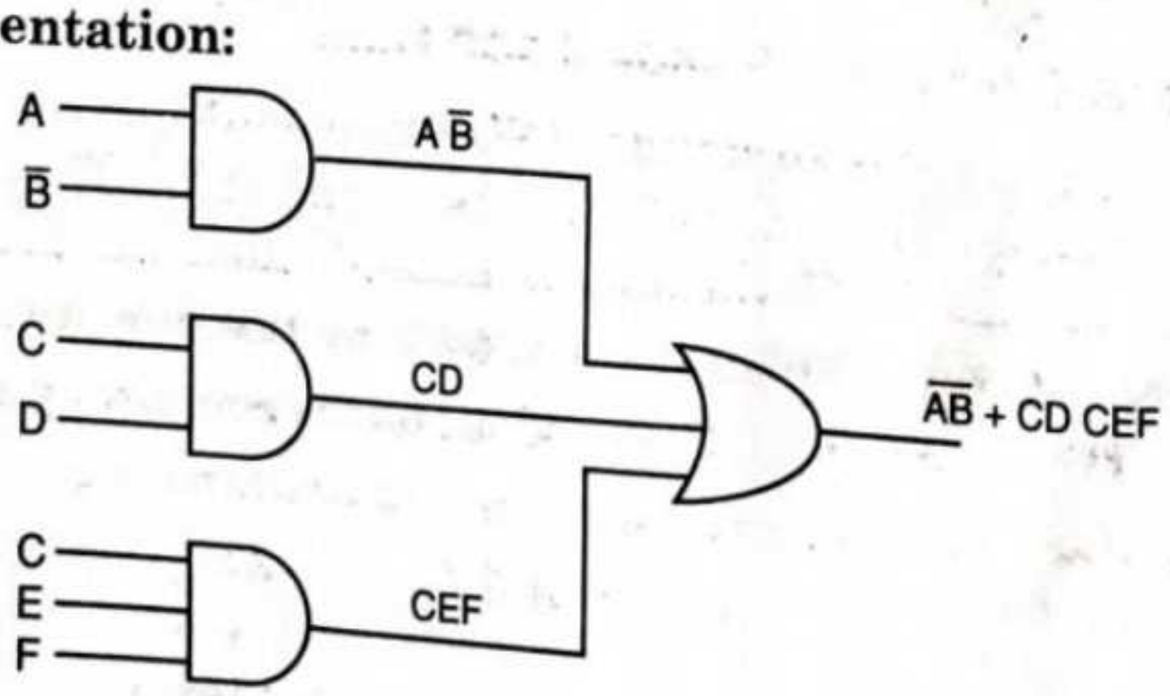


Fig. 2.41.

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

Remove the brackets by applying Boolean multiplication

$$\Rightarrow \quad A\overline{B}C + A(B + D) + \overline{C}(B + D)$$

$$\Rightarrow \quad A\overline{B}C + AB + AD + \overline{C}B + \overline{C}D$$

Gate Implementation :

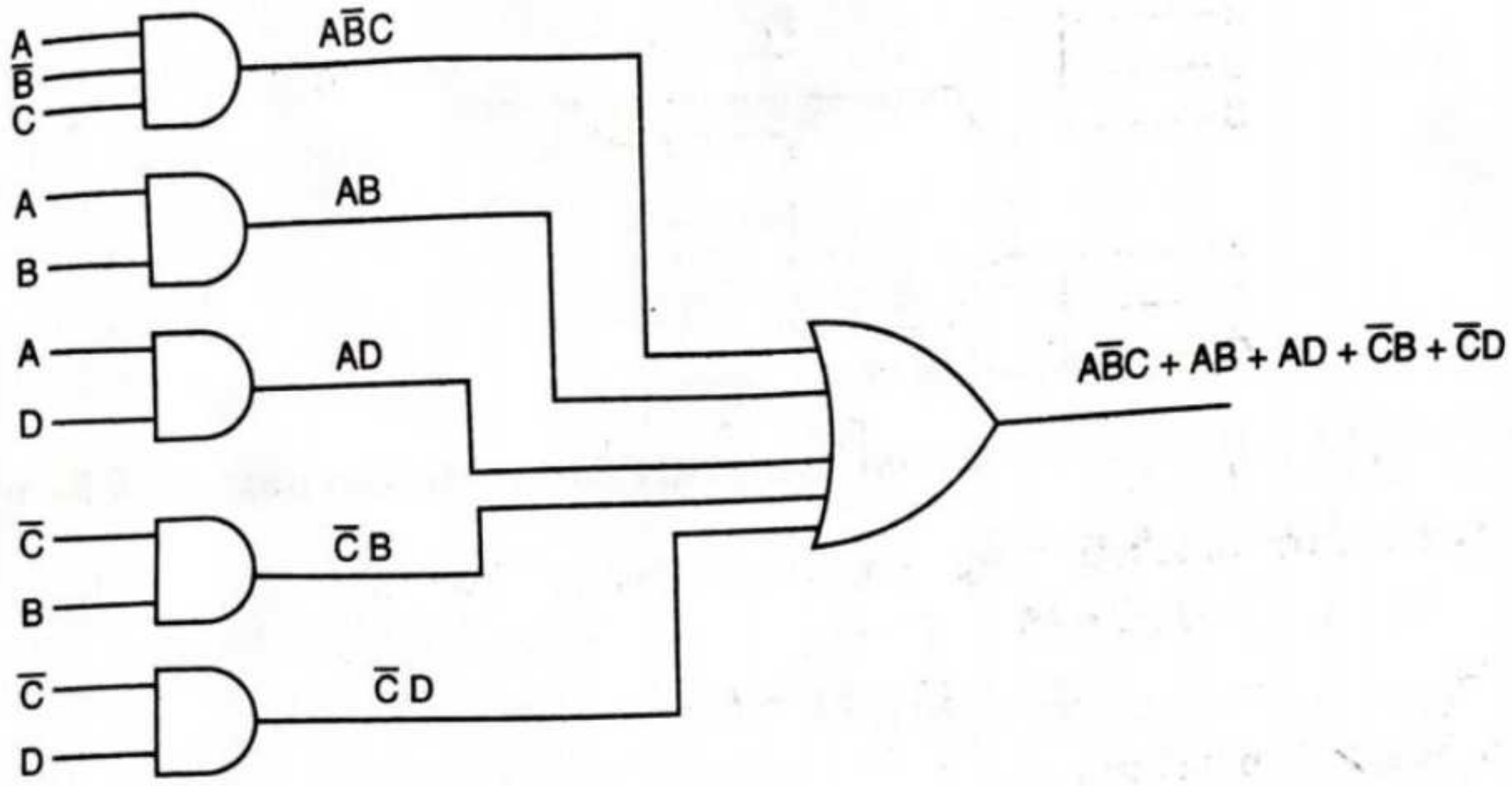


Fig. 2.42.

2.5.2. Standard SOP Form

Standard SOP form contain all the variables which are included in the expression.

Let us take a SOP form i.e.,

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

Here in this form, the Ist term i.e. term i.e. AB does not include the 3rd variable 'C' and hence this is not a standard SOP form. If it has to be standard SOP form, then it must have 'C' term (variable).

2.5.3. Need of Standard SOP Form

- Used in determining truth tables
- Used in K-map simplification which is a graphical method for reducing the number of gates.

2.5.4. Conversion SOP form into Standard SOP Form

(i) In order to convert SOP to standard SOP form multiply the product term by the sum of a missing variable and its complement. This produces two product term. e.g., if 'C' variable is missed, then the product term is multiplied by $(C + \overline{C})$ which is equal to '1' by Boolean law.

(ii) Then repeat (i) to obtain all the product terms containing all variables.

(iii) If some product term is common, then it is written once.

$$\begin{array}{ccccc} AB & + & A\overline{B}C & + & AC \\ | & & | & & | \\ \text{Ist term} & & \text{2nd term} & & \text{3rd term} \end{array}$$

Here

The Ist term i.e., AB, the missing variable is C and multiply it by $(C + \overline{C})$

2nd term $A\,\overline{B}\,C$ contain all the variable so no need of doing any operati

3rd term AC, the missed term is B and hence $(B + \overline{B})$ is multiplied.

So,

$$\Rightarrow \qquad AB(C + \overline{C}) + A\,\overline{B}\,C + AC(B + \overline{B})$$

[As $B + \overline{B} = 1$, $C + \overline{C} = 1$]

$$\Rightarrow \qquad ABC + AB\overline{C} + A\,\overline{B}\,C + ABC + A\,\overline{B}\,C$$

Here ABC and $A\,\overline{B}\,C$ repeats, so it is written once.

So,

$$\Rightarrow \qquad \boxed{ABC + AB\overline{C} + A\,\overline{B}\,C}$$

↓

Standard SOP output

EXAMPLE 2.9. Convert the following SOP form into standard SOP for

(i) $A\,\overline{B}\,C + \overline{A}\,\overline{B} + \overline{B}$

(ii) $A + \overline{B}\,C$

(iii) $\overline{A}\,B\overline{C}\,D + ABC + BC\overline{D}$

(iv) $A\overline{C} + B\overline{A} + \overline{A}\,\overline{B}\,\overline{C}$

Solution: *(i)* $A\,\overline{B}\,C + \overline{A}\,\overline{B} + \overline{B}$

Take the Ist term $A\,\overline{B}\,C$, here no term is missed so no operation is performe

The second term, i.e. $\overline{A}\,\overline{B}$, C is missing term so multiply this product ter by $(C + \overline{C})$ $\overline{A}\,\overline{B}\,(C + \overline{C})$

$$\Rightarrow \qquad \overline{A}\,\overline{B}\,C + \overline{A}\,\overline{B}\,\overline{C}$$

The third term, i.e. $\overline{B}$, two variables, i.e. A and C are missed, so multip product term by $(A + \overline{A})\,(C + \overline{C})$.

i.e., $\qquad \overline{B}\,(A + \overline{A})\,(C + \overline{C})$

$$\Rightarrow \qquad (\overline{B}A + \overline{B}\,\overline{A})\,(C + \overline{C})$$

$$\Rightarrow \qquad \overline{B}AC + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

$$\Rightarrow \qquad A\overline{B}C + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C}$$

The final output is given as :

$$\Rightarrow \qquad \underline{A\overline{B}C} + \underline{\overline{A}\,\overline{B}C} + \underline{\overline{A}\,\overline{B}\,\overline{C}} + \underline{A\overline{B}C} + A\overline{B}\,\overline{C} + \underline{\overline{A}\,\overline{B}C} + \underline{\overline{A}\,\overline{B}\overline{C}}$$

The underline terms are repeat, so written only once i.e.,

$$\boxed{A\overline{B}C + \overline{A}\,\overline{B}C + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{C}}$$

(ii) $A + \overline{B}\,C$

Standard SOP form is given as:

$$\Rightarrow \quad A(B+\overline{B})(C+\overline{C})+B\overline{C}(A+\overline{A})$$

$$\Rightarrow \quad (AB+A\overline{B})(C+\overline{C})+AB\overline{C}+\overline{A}B\overline{C}$$

$$\Rightarrow \quad ABC+AB\overline{C}+A\overline{B}C+A\overline{B}\,\overline{C}+AB\overline{C}+\overline{A}B\overline{C}$$

[$AB\overline{C}$ repeats so taken once]

$$=\boxed{ABC+AB\overline{C}+A\overline{B}C+A\overline{B}\,\overline{C}+\overline{A}B\overline{C}}$$

(iii) $\overline{A}B\overline{C}D+ABC+BC\overline{D}$

Standard SOP form is given as:

$$\Rightarrow \quad \overline{A}B\overline{C}D+ABC(D+\overline{D})+BC\overline{D}(A+\overline{A})$$

$$\Rightarrow \quad \overline{A}B\overline{C}D+ABCD+ABC\overline{D}+BC\overline{D}A+BC\overline{D}\,\overline{A}$$

$$\Rightarrow \quad \overline{A}B\overline{C}D+ABCD+ABC\overline{D}+\overline{A}BC\overline{D}$$

[$ABC\overline{D}$ repeats so it is taken once]

(iv) $A\overline{C}+B\overline{A}+\overline{A}\,\overline{B}\,\overline{C}$

Standard SOP form is given as :

$$\Rightarrow \quad A\overline{C}+B\overline{A}+\overline{A}\,\overline{B}\,\overline{C}$$

$$\Rightarrow A\overline{C}(B+\overline{B})+\overline{A}B(C+\overline{C})+AB\overline{C}$$

$$\Rightarrow \quad AB\overline{C}+A\overline{B}\,\overline{C}+\overline{A}BC+\overline{A}B\overline{C}+\overline{A}\,\overline{B}\,\overline{C}$$

2.5.5. Binary Representation of Standard SOP Form

A standard product term or minterm is filled with '1', so the binary representation is given as if

$A\overline{B}CD$ is the minterm, then its value is '1' and it is only possible when the product of A, $\overline{B}$, C and D are '1'.

Means if A is present, then it is taken as '1', if $\overline{B}$ is present then it is taken as '0', if C is present, then it is taken as '1', if D is present, then it is taken as '1'.

$$A\overline{B}CD=1\,.\,0\,.\,1\,.\,1=1\,.\,1\,.\,1\,.\,1=1$$

So the binary representation is given as 1 0 1 1 (decimal 11)

EXAMPLE 2.10. Determine the binary value for the following standard SOP form:

(i) $A\overline{B}\,\overline{C}\,\overline{D}+A\overline{B}CD+\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$

(ii) $ABCD+A\overline{B}\,\overline{C}D$

(iii) $ABC+A\overline{B}\,\overline{C}+\overline{A}BC$

Solution: (i) $A\overline{B}\,\overline{C}\,\overline{D}+A\overline{B}CD+\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$

$A\overline{B}\,\overline{C}\,\overline{D}$ is '1' when $A=1,\ \overline{B}=0,\ \overline{C}=0,\ \overline{D}=0$

$A\overline{B}CD$ is '1' when $A=1,\ \overline{B}=0,\ C=1,\ D=1$

$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ is '1' when $\overline{A}=0,\ \overline{B}=0,\ \overline{C}=0,\ \overline{D}=0$

So, $A\overline{B}\,\overline{C}\,\overline{D} = 1000$

[If '1' is there then it is taken as A(true form)

$A\overline{B}CD = 1011$

and if '0' is there; taken as $\overline{A}$)]

$\overline{A}\,\overline{B}\,\overline{C}\,\overline{D} = 0000$

(*ii*) $ABCD + A\overline{B}\,\overline{C}\,D$

$ABCD$ is '1' when $A = B = C = D = 1$

$A\overline{B}\,\overline{C}\,D$ is '1' when $A = 1$, $\overline{B} = 0$, $\overline{C} = 0$ and $D = 1$

So, $ABCD = 1111$

$A\overline{B}\,\overline{C}\,D = 1001$

(*iii*) $ABC + A\overline{B}\,\overline{C} + \overline{A}BC$

ABC is '1' when $A = B = C = 1$

$A\overline{B}\,\overline{C}$ is '1' when $A = 1$, $\overline{B} = 0$, $\overline{C} = 0$

$\overline{A}BC$ is '1', when $\overline{A} = 0, B = 1, C = 1$

So, $ABC = 111$

$A\overline{B}\,\overline{C} = 100$

$\overline{A}BC = 011$

2.6 PRODUCT OF SUM (POS) FORM

In this form variable are summed (Boolean addition) in their true form or complemented form and this sum term is known as Maxterm (M) and then these maxterms are multiplied together to form product of sum form.

Here *first variables are summed and then they are multiplied to form POS form.*

It requires OR-AND gate. The OR gate is used to form the sum term and AND gate is used to form the product term.

e.g., $Y = (A + B + C)(A + B + \overline{C})(A + \overline{B} + C)(\overline{A} + \overline{B} + C)$

The POS term can be implemented using OR-AND gates as shown below

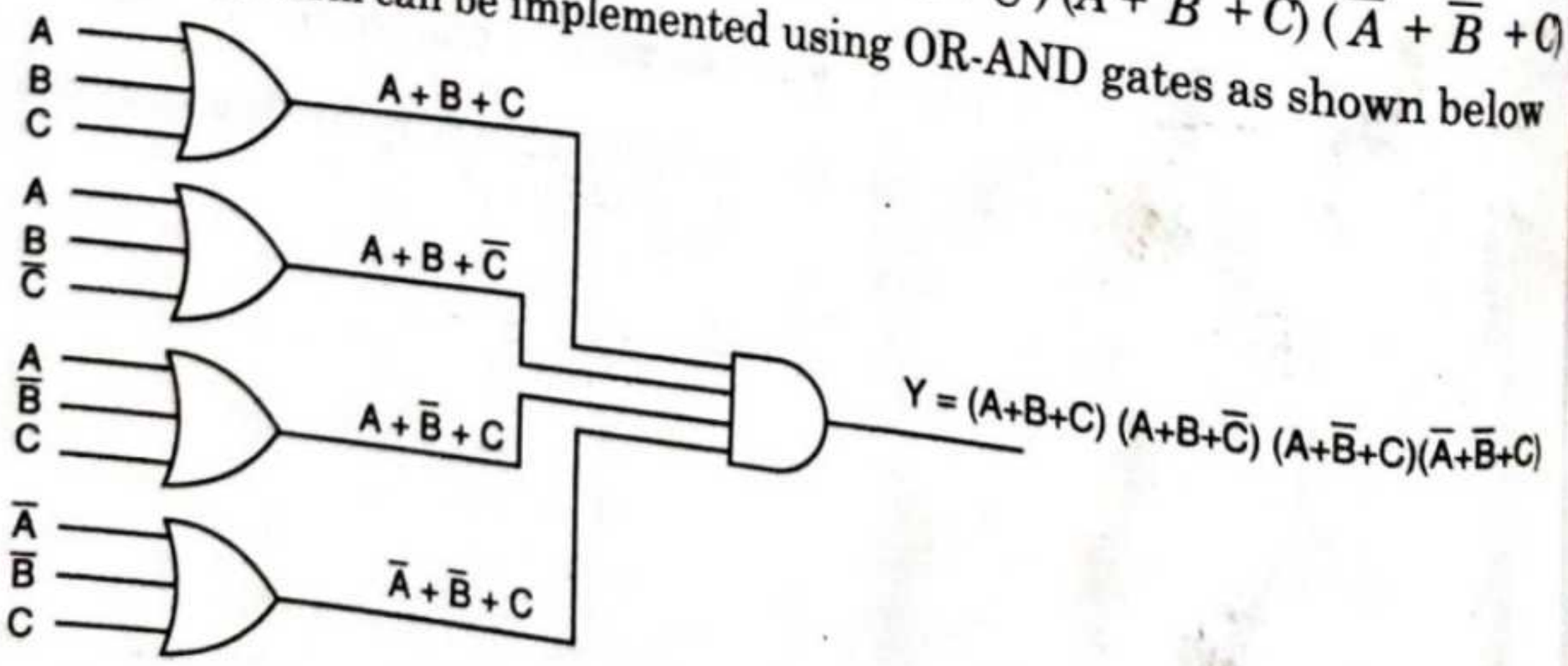


Fig. 2.43.

The POS form can be implemented using NOR gate (Universal gates).

i.e., $$Y = (A + B + C)\,(A + B + \overline{C})\,(A + \overline{B} + C)\,(\overline{A} + \overline{B} + C)$$

Applying DeMorgan's Theorem :

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

[As $\overline{A \,.\, B} = \overline{A} + \overline{B}$]

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

↓ ↓ ↓ ↓

Output of NOR gate with inputs A, B, C — Output of NOR gate with inputs $A, B, \overline{C}$ — Output of NOR gate with inputs $A, \overline{B}, C$ — Output of NOR gate with inputs $\overline{A}, \overline{B}, C$

Again applying DeMorgan's Theorem

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

So, designing is as follows:

[As $\overline{A+B} = \overline{A} \,.\, \overline{B}$]

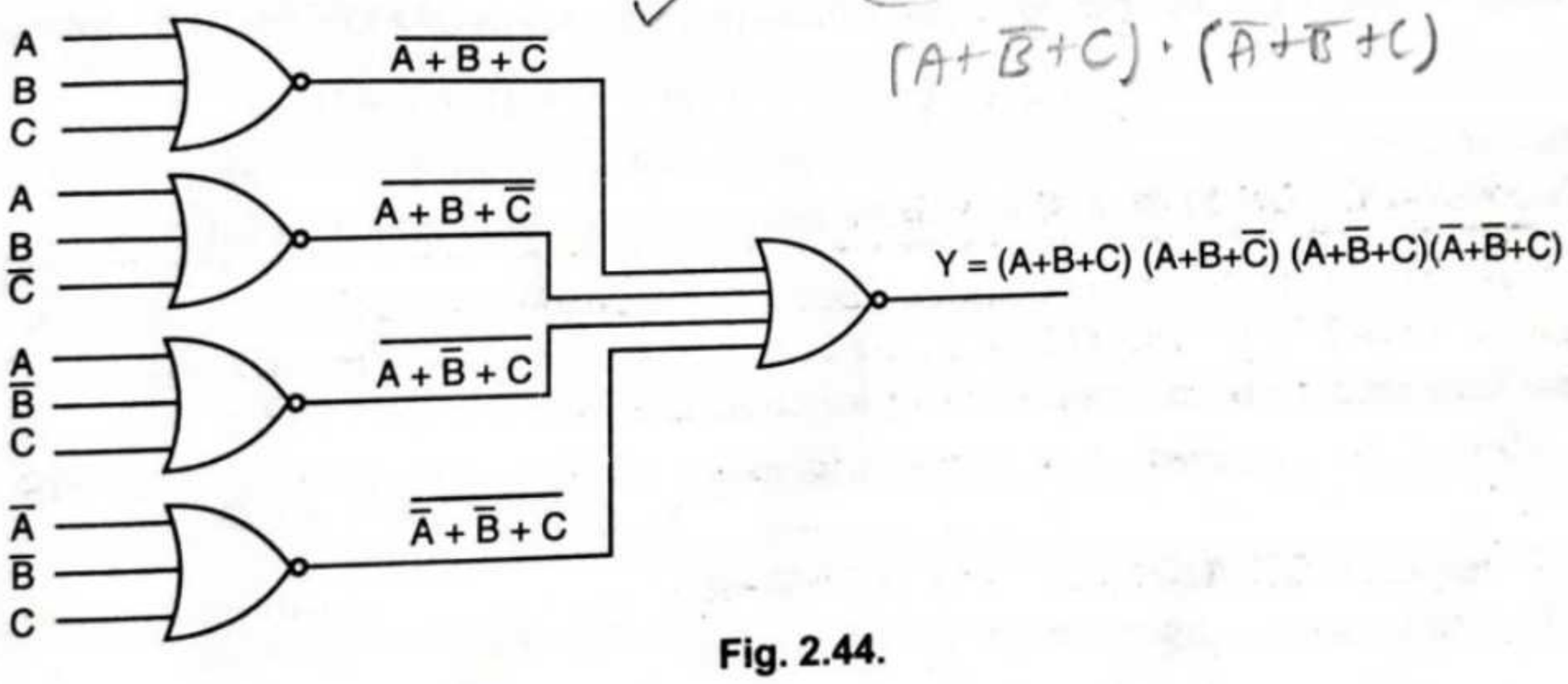


Fig. 2.44.

2.6.1. Maxterm

The summed term is known as maxterm : It may be in true form or complement form let us take an e.g.,

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

In maxterms, if variable have zero value, then it comes in true form i.e. here $A = 0$ and $B = 0$, then it is represented as $A + B$. If the variable have one value then it comes in complemented form i.e. when $A = 0$ and $B = 1$, then it is represented as $A + B$ (as B value is '1', so it is in complemented form)

Maxterm is represented by zeros in truth table. The representation of P equation is as follows

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

Or

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

The numerical POS form is given as :

$$Y = \pi M \quad (0, 1, 2, 6)$$

Symbol Maxterm

for POS form

Or

$$Y(A, B, C) = \pi M \ (0, 1, 2, 6)$$

Or

$$Y = \pi(0, 1, 2, 6)$$

Or

$$Y = M(0, 1, 2, 6)$$

EXAMPLE 2.11. Implement the following equation of POS form by gate

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

Solution: For sum terms, it require three OR gates and for product term, require single AND gate. So the gate implementation is given as follows:

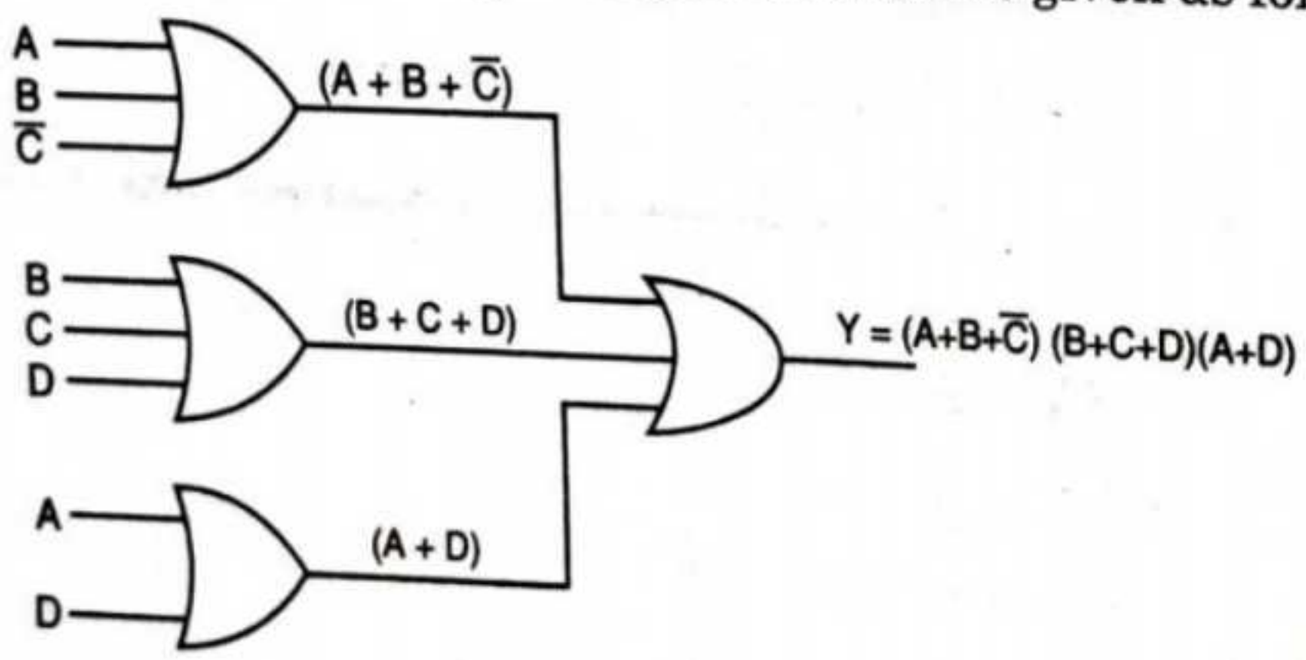


Fig. 2.45.

2.6.2. Standard POS Form

In this standard POS form it includes all the variable in the sum expressio

e.g., $(A + B + \overline{C})(A + \overline{B})$

Here in the Ist term i.e., $(A + B + \overline{C})$ all the three variables are present bu in the 2nd term i.e., $(A + \overline{B})$, variable C or $\overline{C}$ is absent.

The standard form is given as

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

2.6.3. Converting POS Form to Standard Form

(1) Add the product of the missing and complement of missing term to the sum term.

e.g., $(A + \overline{B} + \overline{C})(A + \overline{B})$ here C or $\overline{C}$ term is missing in 2nd term, so we can apply rule '1' as $(A + \overline{B} + C\overline{C})$

↓

Adding product of complement and missed term as $C\overline{C}$ and by Boolean law

$C\overline{C} = 0$ means no change in the expression.

(2) Then applying the rule, i.e. Boolean law, i.e., $(A + CD) = (A + C)(A + D)$

(3) Repeat step 1 until all resulting sum terms contain all variables in the expression whether in true or complemented form.

e.g., convert it into standard POS form

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

Solution: Here the Ist term variable C or $\overline{C}$ and term applying rule '2', i.e.

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

$$\Rightarrow \quad (A + \overline{B} + C)(A + \overline{B} + \overline{C})$$

In 2nd term i.e., $(A + \overline{B} + \overline{C})$ no variable is missing.

So the standard POS form is given as

$$(A + \overline{B} + C) \quad (A + \overline{B} + \overline{C}) \quad (A + \overline{B} + \overline{C})$$

$(A + \overline{B} + \overline{C})$ repeats twice, so written once

$$(A + \overline{B} + C)(A + \overline{B} + \overline{C}). \textbf{ Ans.}$$

EXAMPLE 2.12. Convert the following expressions into standard POS form.

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

(ii) $(\overline{A} + \overline{B} + \overline{C})(A + B)$

(iii) $(A + B)(A + B + C)(\overline{A} + \overline{B})$

Solution: (*i*) $(A + \overline{B} + C)(A + C + \overline{D})$

In Ist term, i.e. $(A + \overline{B} + C)$, the D or $\overline{D}$ term is missing, so add $D\overline{D}$ and applying rule '2' of standard POS conversion method, i.e.

$$\Rightarrow \quad (A + \overline{B} + C + D\overline{D})$$

$$\Rightarrow \quad (A + \overline{B} + C + D)(A + \overline{B} + C + \overline{D})$$

For 2nd term $(A + C + \overline{D})$, the missed term is B or B, so add $B\overline{B}$ and term it is given as

$$\Rightarrow \quad (A + B\overline{B} + C + \overline{D})$$

$$\Rightarrow \quad (A + B + C + \overline{D})(A + \overline{B} + C + \overline{D})$$

So, final standard POS form is given as

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

$(A + \overline{B} + C + \overline{D})$ repeats so taken once

$$\Rightarrow \quad (A + \overline{B} + C + D)\,(A + \overline{B} + C + \overline{D})\,(A + B + C + \overline{D})$$

(*ii*) $(\overline{A} + \overline{B} + \overline{C})\,(A + B)$

The standard POS term is given as:

$$\Rightarrow \quad (\overline{A} + \overline{B} + \overline{C})\,(\overline{A} + \overline{B} + C\overline{C})$$

↑

Missed Term

$$\Rightarrow \quad \boxed{(\overline{A} + \overline{B} + \overline{C})\,(A + B + C)\,(A + B + \overline{C})}$$

(*iii*) $(A + B)\,(A + B + C)\,(\overline{A} + \overline{B})$

Applying rules of conversion of standard POS

$$\Rightarrow \quad (A + B + C\overline{C})\;(A + B + C)\;(\overline{A} + \overline{B} + C\overline{C})$$

↑ missed term — ↑ no missing term — ↑ missed term

$$\Rightarrow \quad (A + B + C)\,(A + B + \overline{C})\,(A + B + C)\,(\overline{A} + \overline{B} + C)\,(\overline{A} + \overline{B} + \overline{C})$$

The $(A + B + C)$ term repeats twice, so written once

$$\Rightarrow \quad \boxed{(A + B + C)\,(A + B + \overline{C})\,(\overline{A} + \overline{B} + C)\,(\overline{A} + \overline{B} + \overline{C})}$$

2.6.4. Binary Representation of Standard POS Form

On POS form '0' value is used for sum term. e.g., if A is there, then '0' and if $\overline{A}$ is there, then its value is '1'.

e.g., $(\overline{A} + \overline{B} + \overline{C} + D)$

Here A is there, so '0' value is used

$\overline{B}$ is there, so '1' value is used

C is there, so '0' value is used

D is there, so '0' value is used.

i.e.,

$$\overline{A} + \overline{B} + \overline{C} + D$$

$$\downarrow$$

$$0 + \overline{1} + 0 + 0 = 0$$

So, the value is given as "0 1 0 0" (decimal 4)

EXAMPLE 2.13. Represented the binary notation for the following standard POS expression:

(*i*) $(\overline{A} + \overline{B} + C)\,(\overline{A} + \overline{B} + \overline{C})$

(*ii*) $(\overline{A} + \overline{B} + \overline{C} + D)\;\;(A + B + C + \overline{D})$

(*iii*) $(\overline{A} + \overline{B})\,(A + B)\,(\overline{A} + B)$

Solution: **(*i*)** $(\overline{A} + \overline{B} + C)(\overline{A} + \overline{B} + \overline{C})$

Sum term value is equal to zero, i.e.

$$\overline{A} + \overline{B} + C = 0$$

$$\downarrow$$

$$0 + \bar{1} + 0 = 0$$

So $\overline{A} + \overline{B} + C$ is given as "0 1 0"

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

$\downarrow$

Here $\overline{A}$ is there so its value is '1'

$\overline{B}$ is there so its value is '1'

$\overline{C}$ is there so its value is '1'

So, $\overline{A} + \overline{B} + \overline{C}$ = "1 1 1"

(*ii*) $(\overline{A} + \overline{B} + \overline{C} + D)\ (A + B + C + \overline{D})$

$\overline{A} + \overline{B} + \overline{C} + D$ is equal to zero only if

$\downarrow$

$\bar{1} + \bar{1} + \bar{1} + 0 \quad \Rightarrow$ "1 1 1 0"

$A + B + C + \overline{D}$ is equal to zero only if

$\downarrow$

$0 + 0 + 0 + \bar{1}$ i.e. "0 0 0 1"

(*iii*) $(\overline{A} + \overline{B})(A + B)(\overline{A} + B)$

$\overline{A} + \overline{B}$ is equal to zero only if $\overline{A} + \overline{B}$ is zero i.e., "1 1" $A + B$ is equal to 0 only if $A = B = 0$, i.e. "0 0"

$\overline{A} + B$ is equal to zero only if $\overline{A} = 1$ and $B = 0$, i.e. "1 0"

2.7 DIFFERENCE BETWEEN MINTERMS AND MAXTERM

Minterm	Maxterm
— It is denoted by Σm — Here the output is represented into Sum of Product form (SOP form) and realized by AND-OR gate. — In this form '1' is used for representing the minterm. — Minterms expression is given as e.g. Σm (0, 1, 2, 3) Here 0, 1, 2, 3 minterms are put with '1' value while solving the k-map. — The output expression of minterms is given as e.g., $f = \overline{A}\,\overline{B} + \overline{B}\,\overline{C}\,\overline{D}$	— It is denoted by πM — Here the output is represented into Product of Sum form (POS form) and realized by OR-AND gate. — In this form '0' represent the maxterm. — Maxterms expression is given as e.g., πM (1, 3, 4) Here 1, 3 and 4 maxterms are put with '0' value while solving the k-map. — The output expression is given as e.g., $(A + \overline{B})(B + C + D)$

2.8 SIMPLIFICATION OF BOOLEAN EXPRESSIONS WITH THE HELP OF RULES LAWS OF BOOLEAN ALGEBRA

Simplification means reducing the expression. The need of simplification o Boolean expression is to *reduce the number of gates and hence the circuit become simple and easy to implement.*

Simplification is done using various Boolean Laws and rules which i discussed earlier. Let us take an example.

$$AB + B(C + D) + A(A + D)$$

Here following rules are applied :

Step 1 : Apply distributive law to second and third terms, i.e.

$$AB + BC + BD + A \cdot A + AD$$

Step 2 : Then apply this rule i.e., $A \cdot A = A$ to the fourth term, i.e.

$$AB + BC + BD + A + AD$$

Step 3 : Then apply this rule $A + AD = A$ $[A\,[1 + D] = A]$ to last two terms

$$AB + BC + BD + A$$

Step 4 : Then apply this rule i.e., $A + AB = A$ to first and last term, i.e.

$$A + BC + BD$$

$$\downarrow$$

So, this is the simplified expression using Boolean rules and laws.

Gate Implementation

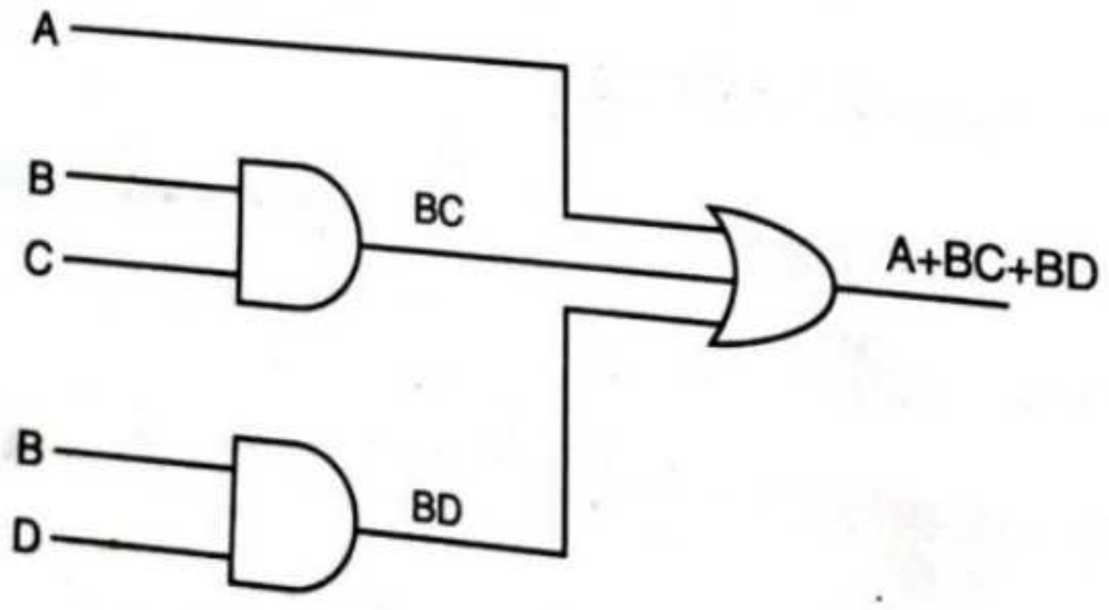


Fig. 2.46.

So, here in this expression the number of gates are reduced to three but in the previous expression means without simplification '6' gates are required, i.e.

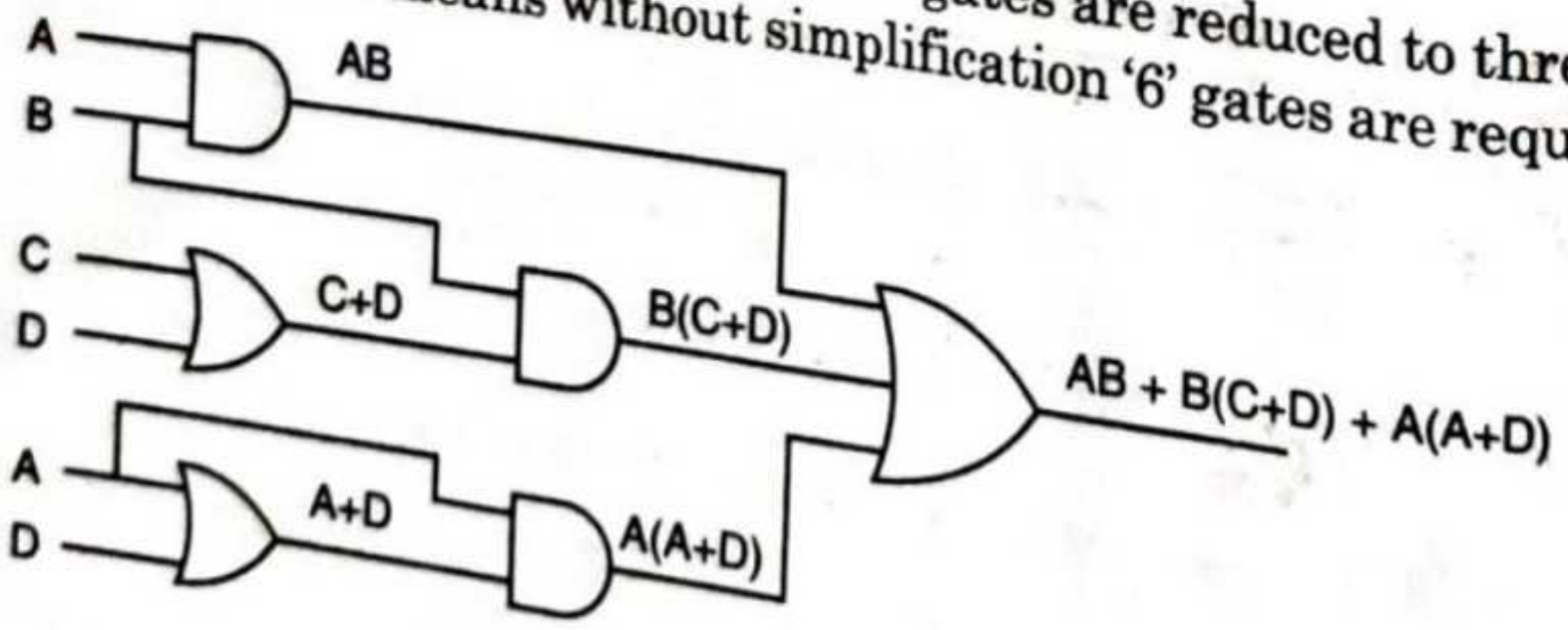


Fig. 2.47.

EXAMPLE 2.14. Using Boolean algebra simply the given expression:

$$A + B\,[CD + (B + \overline{C})D]$$

Solution: Step 1: Applying distributive law to inside bracket term, i.e.

$$A + B\,[CD + BD + \overline{C}\,D]$$

Step 2: Applying distributive law to outside bracket, i.e.

$$A + BCD + B\,.\,BD + B\overline{C}\,D$$

Step 3: Applying this rule $B\,.\,B = B$ to the third term and is given as

$$A + BCD + BD + B\overline{C}\,D$$

Step 4: Take the BD common to 3rd and 4th term

$$A + BCD + BD\,(1 + \overline{C}\,)$$

Step 5: Apply the rule $1 + \overline{C} = 1$, to the 3rd term

$$A + BCD + BD$$

Step 6 : Take BD common to 2nd and 3rd term, i.e.

$$A + BD\,(1 + C)$$

Step 7 : Apply this rule $(1 + C) = 1$, to 2nd term

$$\boxed{A + BD}$$

Gate Implementation

$A + BD$ is the simplified expression, it require '2' gates i.e., OR and AND gate.

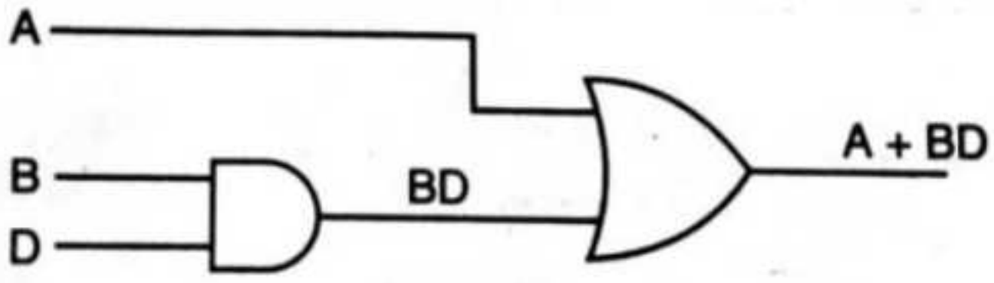


Fig. 2.48.

EXAMPLE 2.15. Simplify the given expression using Boolean law and rules

$$(B + BC)\,(B + \overline{B}\,C)\,(A + D)$$

Solution: Step 1: Applying distributive law to first and second term of the expression:

$$(B + BC)\,(B + \overline{B}\,C)\,(A + D)$$

$$\Rightarrow \quad B(B + \overline{B}\,C) + BC\,(B + \overline{B}\,C)\,(A + D)$$

$$\Rightarrow \quad (B\,.\,B + B\,.\,\overline{B}\,C + B\,.\,BC + BC\,.\,\overline{B}\,C)\,(A + D)$$

Step 2: Applying this rule $B\,.\,B = B$, $B\,.\,\overline{B} = 0$, i.e.,

$$(B + 0 + BC + 0)\,(A + D)$$

$$(B + BC)\,(A + D)$$

Step 3: Applying this rule $B + BC = B$, i.e.

$$B + (A + D)$$

Step 4: Again applying distributive law to the expression:

$$B\,(A + D)$$

$$\Rightarrow \quad AB + BD$$

↑

Simplified expression

Gate Implementation

$AB + BD$ is the required reduced expression and hence it require '3' gates, i.e., '2' AND gate and '1' OR gate.

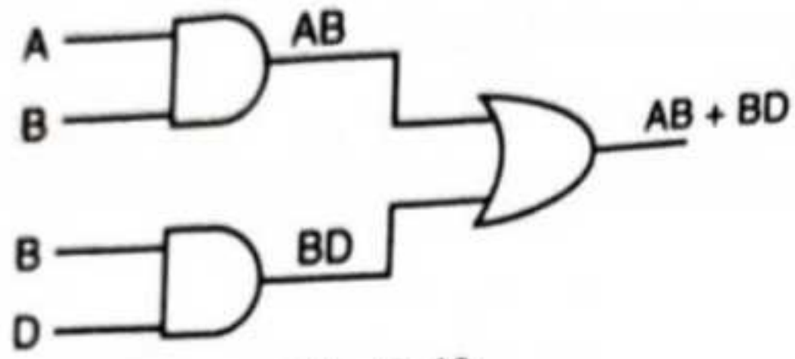


Fig. 2.49.

2.9 'K' MAP (KARNAUGH MAPING TECHNIQUE)

Karnaugh map is a graphical representation of the Boolean variables, i.e. ABC, $\overline{A}\,\overline{B}\,\overline{C}$ ... or $(A + B)$ and used to simplify the Boolean expressions. Earlier, we discussed Boolean laws and rules for simplification, but they are limited to few variables and if variables are more we can use 'K' mapping technique.

2.9.1. Features of K-maps

- It is graphical representation of variables either in POS form or SOP form.
- It is arranged in cells rather than rows or column in truth table.
- Here each cell represent a binary value of the variable.
- Gray coding is used in 'K' map because gray code has the property of one bit changing between previous and next decimal notation, so the pairing is easily possible i.e., if Gray code for decimal '1' is 001 and decimal '2' has gray code of 011 then only 2nd place it is differ and hence (0 and 1 combine) pairing forms and reduced the variable
- Cells depends upon the number of variable in the expression and used this value 2^n where n is the number of variable. e.g. if A, B and C variables are there, then $n = 3$ and hence cells are $2^{n=3}$ i.e., $2^3 = 8$.

  So eight cells are formed.

  For '4' variables, cells are $2^4 = 16$

  For '2' variable, cells are $2^2 = 4$
- 'K' map is suitable to solve $n \leq 6$ variables.

2.9.2. Types of K-map

According to number of variable, it is divided into following types :

(*i*) '2' variable K-map

(*ii*) '3' variable K-map

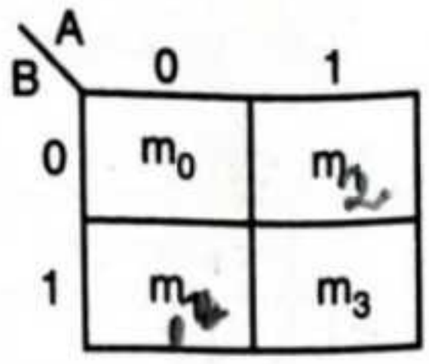
(*iii*) '4' variable K-map

(*iv*) '5' variable K-map

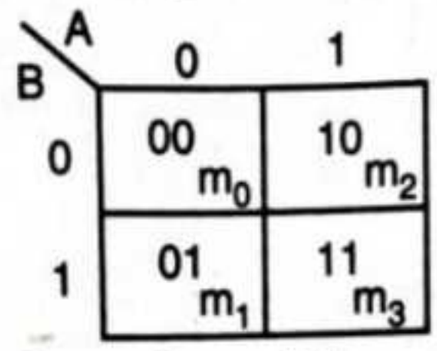
(*v*) '6' variable K-map

(1) Two Variable K-map

In two variable, the value of 'n' is '2', so $2^n = 4$ cells are formed. It is represented as :



Here m_0, m_1, m_2, m_3 represents minterms. Maxterms can be placed in these cells and is represented as M_0, M_1, M_2 and M_3.



Here for first cell i.e., m_0, the value of the minterm is read as first column and then row value i.e.,

m_0 = "0 0"

['0 0' is equal to decimal zero, so "m_0" is used]

↑ ↑

column value ('A') Row value ('B')

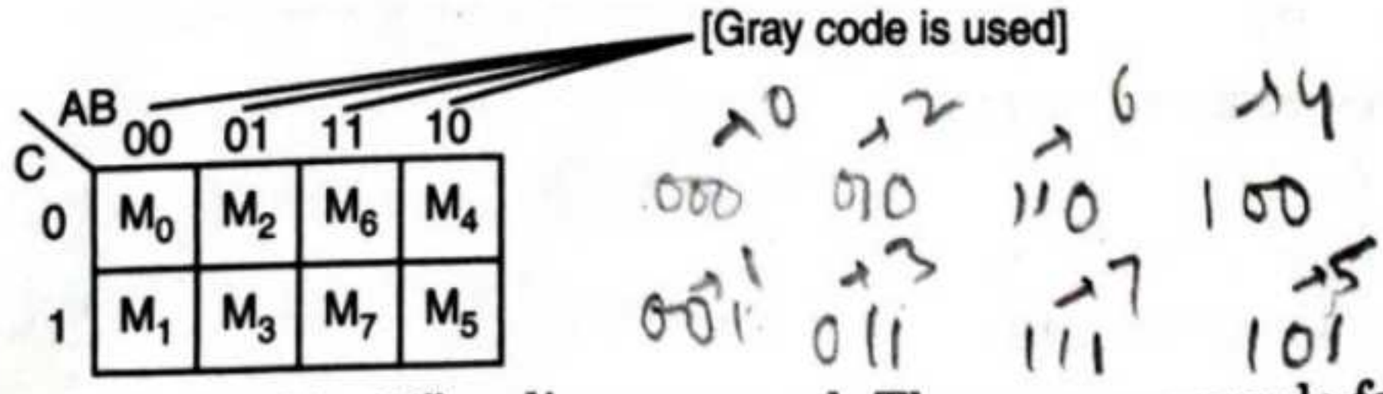
m_1 = "0 1"

m_2 = "1 0"

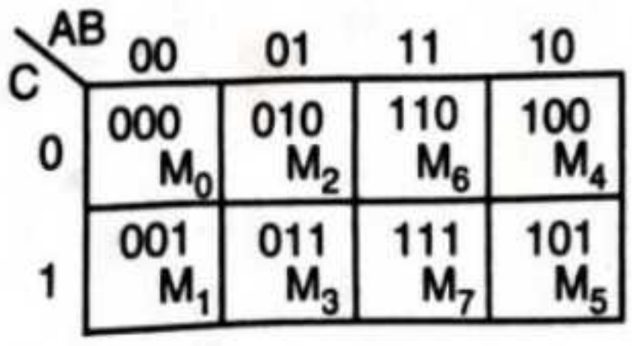
m_3 = "1 1"

(2) Three Variable K-map

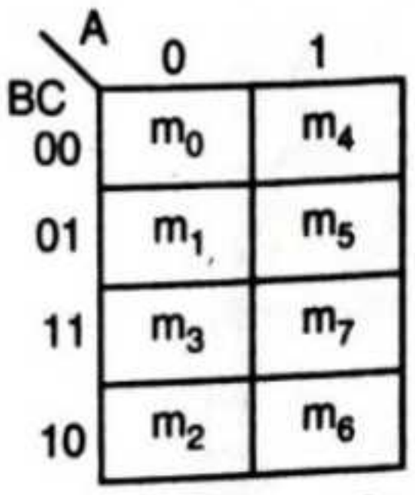
In three variable, the value of 'n' is '3', so $2^3 = 8$, cells are formed. It is represented as :



Here on the top "00, 01, 11, 10" coding are used. They are gray code for '0', '1', '2' and '3'. This order of notation is fixed because it is useful in making pairing.



Or



Here first cell column value is for AB i.e., 00 and for row value i.e., $C = 0$

So $ABC \Rightarrow$ "0 0 0" = 0 = m_0 (decimal zero "0 0 0")

Similarly for other minterms.

(3) Four Variable K-map

In four variable, 'n' value is '4', so cells are $2^4 = 16$. It is given as :

CD \ AB	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

Here M_0 value in terms of binary notation is given as

A	B	C	D
↓	↓	↓	↓
0	0	0	0

M_1 is given as

A	B	C	D
0	0	0	1

Similarly for other minterms. The binary values in each minterm in cell format is given as

CD \ AB	00	01	11	10
00	0000 m_0	0100 m_4	1100 m_{12}	1000 m_8
01	0001 m_1	0101 m_5	1101 m_{13}	1001 m_9
11	0011 m_3	0111 m_7	1111 m_{15}	1011 m_{11}
10	0010 m_2	0110 m_6	1110 m_{14}	1010 m_{10}

(4) Five Variable K-map

In five variable K map, the number of cells are $2^5 = 32$ [2^n = no. of cells & n is the number].

It is represented as

A = 0

DE \ BC	00	01	11	10
00	m_0 0000	m_4 0100	m_{12} 1100	m_8 1000
01	m_1 0001	m_5 0101	m_{13} 1101	m_9 1001
11	m_3 0011	m_7 0111	m_{15} 1111	m_{11} 1011
10	m_2 0010	m_6 0110	m_{14} 1110	m_{10} 1010

A = 1

DE \ BC	00	01	11	10
00	m_{16} 0000	m_{20} 0100	m_{28} 1100	m_{24} 1000
01	m_{17} 0001	m_{21} 0101	m_{29} 1101	m_{25} 1001
11	m_{19} 0011	m_{23} 0111	m_{31} 1111	m_{27} 1011
10	m_{18} 0010	m_{22} 0110	m_{30} 1110	m_{26} 1010

Here A is most significant bit. The first block of 16 cells include $A = 1$.

If we have to read m_{24} value then it is given ABCDE = 1 1000

A BCDE
(Value) (Value)

Here, A is '1' because m_{24} lie in second block where A is '1' and BCDE value is 1000. So ABCDE is 11000.

The Maxterm representation of five variable is given as

A = 0

DE \ BC	00	01	11	10
00	M_0 0000 (B+C+D+E)	M_4 0100 (B+$\overline{C}$+D+E)	M_{12} 1100 ($\overline{B}$+$\overline{C}$+D+E)	M_8 1000 ($\overline{B}$+C+D+E)
01	M_1 0001 (B+C+D+$\overline{E}$)	M_5 0101 (B+$\overline{C}$+D+$\overline{E}$)	M_{13} 1101 ($\overline{B}$+$\overline{C}$+D+$\overline{E}$)	M_9 1001 ($\overline{B}$+C+D+$\overline{E}$)
11	M_3 0011 (B+C+$\overline{D}$+$\overline{E}$)	M_7 0111 (B+$\overline{C}$+$\overline{D}$+$\overline{E}$)	M_{15} 1191 ($\overline{B}$+$\overline{C}$+$\overline{D}$+$\overline{E}$)	M_{11} 1011 ($\overline{B}$+C+$\overline{D}$+$\overline{E}$)
10	M_2 0010 (B+C+$\overline{D}$+E)	M_6 0110 (B+$\overline{C}$+$\overline{D}$+E)	M_{14} 1110 ($\overline{B}$+$\overline{C}$+$\overline{D}$+E)	M_{10} 1010 ($\overline{B}$+C+$\overline{D}$+E)

A = 1

DE \ BC	00	01	11	10
00	M_{16} 0000 (B+C+D+E)	M_{20} 0100 (B+$\overline{C}$+D+E)	M_{28} 1100 ($\overline{B}$+$\overline{C}$+D+E)	M_{24} 1000 ($\overline{B}$+C+D+E)
01	M_{17} 0001 (B+C+D+$\overline{E}$)	M_{21} 0101 (B+$\overline{C}$+D+$\overline{E}$)	M_{29} 1101 ($\overline{B}$+$\overline{C}$+D+$\overline{E}$)	M_{25} 1001 ($\overline{B}$+C+D+$\overline{E}$)
11	M_{19} 0011 (B+C+$\overline{D}$+$\overline{E}$)	M_{23} 0111 (B+$\overline{C}$+$\overline{D}$+$\overline{E}$)	M_{31} 1111 ($\overline{B}$+$\overline{C}$+$\overline{D}$+$\overline{E}$)	M_{27} 1011 ($\overline{B}$+C+$\overline{D}$+$\overline{E}$)
10	M_{18} 0010 (B+C+$\overline{D}$+E)	M_{22} 0110 (B+$\overline{C}$+$\overline{D}$+E)	M_{30} 1110 ($\overline{B}$+$\overline{C}$+D+E)	M_{26} 1010 ($\overline{B}$+C+$\overline{D}$+E)

(5) Six Variable K-map

In six variable K-map, the number of cells are $2^6 = 64$. It is represented as

A = 0

EF \ CD	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

AB = 00

A = 1

EF \ CD	00	01	11	10
00	m_{32}	m_{36}	m_{44}	m_{40}
01	m_{33}	m_{37}	m_{45}	m_{41}
11	m_{35}	m_{39}	m_{47}	m_{43}
10	m_{34}	m_{38}	m_{46}	m_{42}

AB = 10

B = 1

EF \ CD	00	01	11	10
00	m_{16}	m_{20}	m_{28}	m_{24}
01	m_{17}	m_{21}	m_{29}	m_{25}
11	m_{19}	m_{23}	m_{31}	m_{27}
10	m_{18}	m_{22}	m_{30}	m_{26}

AB = 01

EF \ CD	00	01	11	10
00	m_{48}	m_{52}	m_{60}	m_{56}
01	m_{49}	m_{53}	m_{61}	m_{57}
11	m_{51}	m_{55}	m_{63}	m_{59}
10	m_{50}	m_{54}	m_{62}	m_{58}

AB = 11

2.9.3. Variable Notation (Standard Product Notation) in K-maps

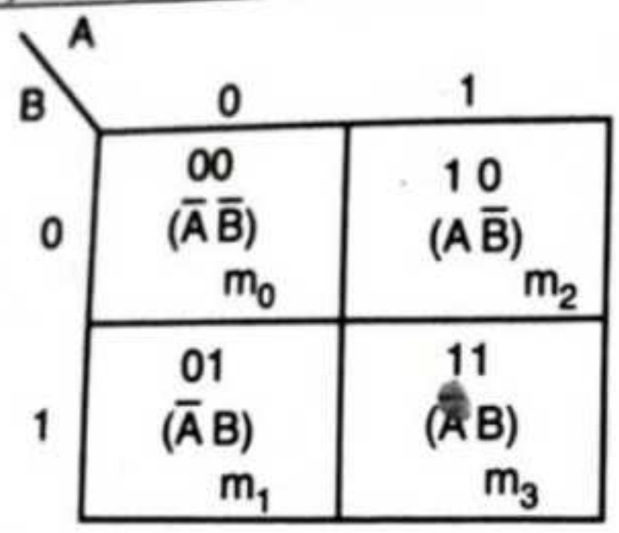
(i) '2' Variable K-map

In this '2' variable if the minterm is "0 0" or m_0, then it is represented as

$\overline{A}\,\overline{B}$

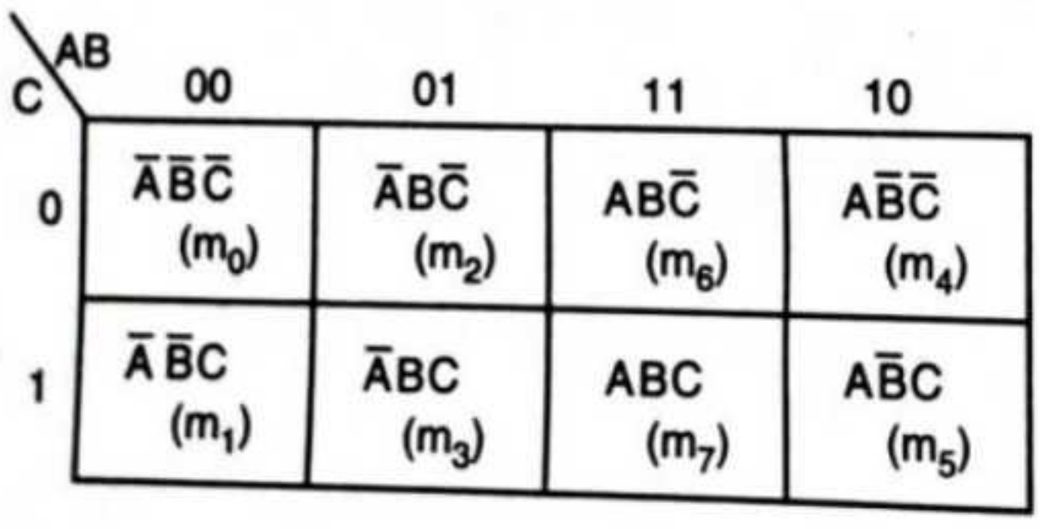
For

	A	B	Variable Notation (Product term)
m_0	0	0	$\Rightarrow \overline{A}\,\overline{B}$
m_1	0	1	$\Rightarrow \overline{A}B$
m_2	1	0	$\Rightarrow A\overline{B}$
m_3	1	1	$\Rightarrow AB$



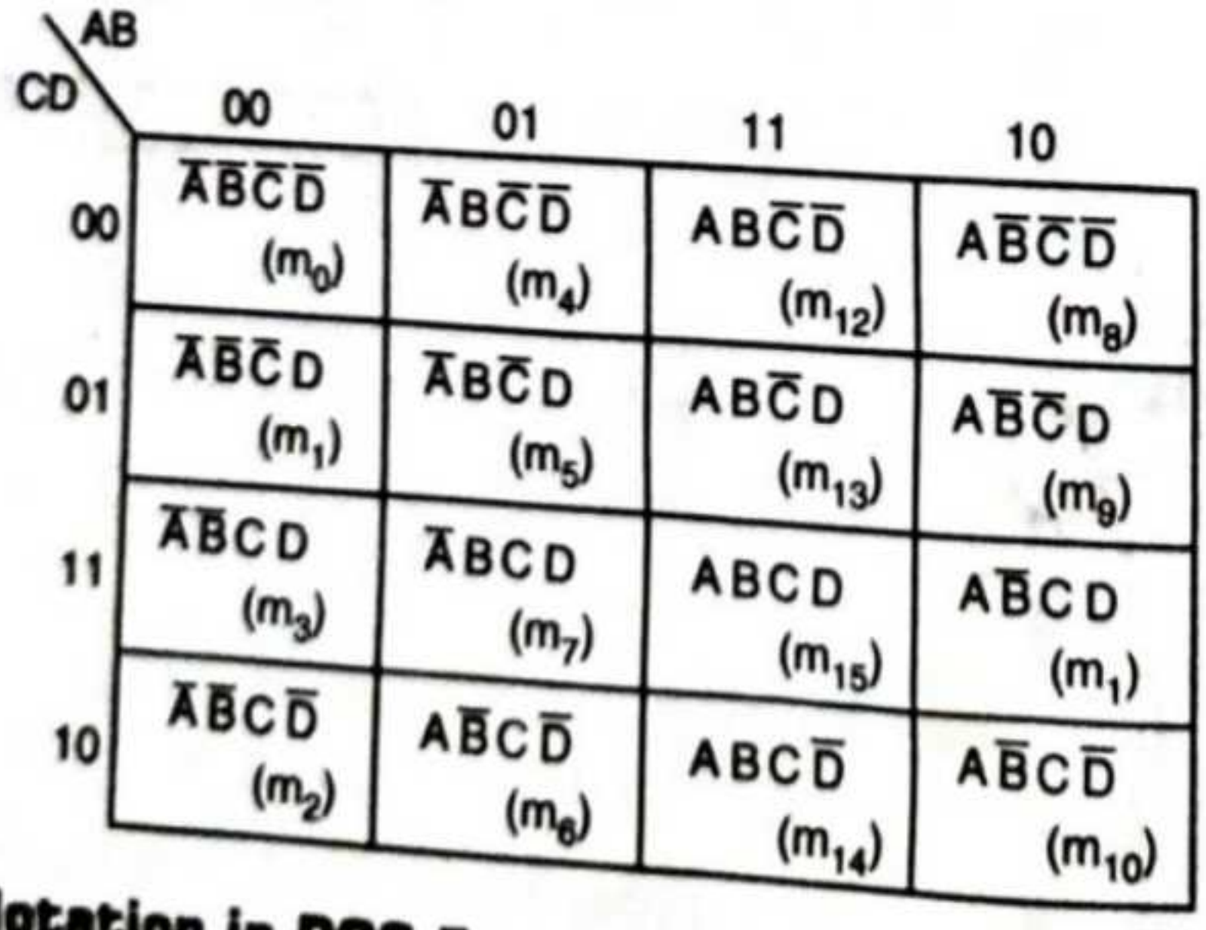
(ii) '3' Variable K-map

In this K-map if m_0 = "0 0 0", then it is represented as $\overline{A}\,\overline{B}\,\overline{C}$. The Produ notation is given as :



(iii) '4' Variable K-map

Here if m_0 = "0 0 0 0", then it is represented as $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$. The Produ notation of '4' variable K-map is given as :



2.9.4. K-map Notation in POS Form

The POS form has '0' value when A term is there and if $\overline{A}$ is there, then value is 1. The POS expression is realized using OR-AND gate.

The K-map is given as :

(i) '2' Variable K-map

Here maxterms are used instead of minterms. It is denoted by 'M'

A \ B	0	1
0	$A+B$ (00)	$\bar{A}+B$ (10)
1	$A+\bar{B}$ (01)	$\bar{A}+\bar{B}$ (11)

(ii) '3' Variable K-map

The sum notation is given as :

C \ AB	00	01	11	10
0	$A+B+C$ (M_0)	$A+\bar{B}+C$ (M_2)	$\bar{A}+\bar{B}+C$ (M_6)	$\bar{A}+B+C$ (M_4)
1	$A+B+\bar{C}$ (M_1)	$A+\bar{B}+\bar{C}$ (M_3)	$\bar{A}+\bar{B}+\bar{C}$ (M_7)	$\bar{A}+B+\bar{C}$ (M_5)

(iii) '4' Variable K-map

The sum notation is given as follows :

CD \ AB	00	01	11	10
00	$A+B+C+D$ M_0	$A+\bar{B}+C+D$ M_4	$\bar{A}+\bar{B}+C+D$ M_{12}	$\bar{A}+B+C+D$ M_8
01	$A+B+C+\bar{D}$ M_1	$A+\bar{B}+C+\bar{D}$ M_5	$\bar{A}+\bar{B}+C+\bar{D}$ M_{13}	$\bar{A}+B+C+\bar{D}$ M_9
11	$A+B+\bar{C}+\bar{D}$ M_3	$A+\bar{B}+\bar{C}+\bar{D}$ M_7	$\bar{A}+\bar{B}+\bar{C}+D$ M_{15}	$\bar{A}+B+\bar{C}+\bar{D}$ M_{11}
10	$A+B+\bar{C}+D$ M_2	$A+\bar{B}+\bar{C}+D$ M_6	$\bar{A}+\bar{B}+\bar{C}+D$ M_{14}	$\bar{A}+B+\bar{C}+D$ M_{10}

2.10 SIMPLIFICATION OF BOOLEAN FUNCTION USING K-MAP

Following steps are required :

(*i*) Fill all the minterms/maxterms in desired K-map.

(*ii*) Pairing of one's/zero's are to be made keeping in mind that pair always occurs in 2^n i.e., 2^0, 2^1, 2^2, 2^3 and 2^4.

(*iii*) Write down the value of prime implicant which are desired due to effective pair of one's (or zero's).

(*iv*) In, SOP form the variable where value is changed in a pair (i.e., 1 to 0 or 0 to 1) is neglected and write the value of variable which is not changed in pair (i.e., if $AB = 00$ and $AB = 01$, then B is neglected and $\bar{A}$ is taken) and write its complemented form of variable if its value is zero and write its value in true form if its value is one.

(v) In POS form, the variable whose value is changed in an pair is neglected and write the value of variable whose value is not changed in pair and write its true form if it has zero value otherwise it is written in complemented form if its value is one.

(vi) Sum all of the prime implicants if it is SOP form and product all prime implicant if it is POS form.

(vii) Now implement the given function using NAND gates if it is SOP form or implement by NOR gates if it is POS form.

Pairing of one's / zero's

(i) Pairing of adjacent one's always start with the largest number first, such as if we have '4' variable then we make $2^4 = 16$ one's pair first and then $2^3 = 8$ one's pair and then $2^2 = 4$ one's pairs and $2^1 = 2$ one's pair and then $2^0 = 1$ ones individual.

(ii) **Grouping of two ones :** Two one's can be paired if they are adjacent to each other and resulting terms due to their pair have one less variable than the original terms.

(iii) **Grouping of four ones :** Four adjacent ones can be grouped together if two of variables associated with maxterms/minterms are same and other two variable are different resulting terms have only two variable in true or in complementary form.

Don't Care Condition

It is represented as cross mark (×). They are used in minimizing the equation by pairing with some minterms because these value is considered as '1'. It is always comes with maxterms or minterms. It never comes alone.

e.g., $Y = \Sigma m(0, 1, 2, 3) + d\,(3, 4, 8)$

↑

don't care terms

$Y = \pi M(1, 2, 5, 8) + d\,(3, 4, 8)$

↑

don't care terms

2.11 QUINE-McCLUSKEY (QM) METHOD OR TABULAR METHOD

K-map is effective tool for minimizing the logic function. Since K-map is used when the variable are less than or equal to six. But when the variable increased from six, then the K-map is difficult to design. For this we use another method of reducing the logic function known as Quine McCluskey (QM) method. It is also known as tabular method. In this method a step by step procedure is used to simplify the logic function. The steps to be used in the QM method are

(1) A set of prime implicants are obtained first.

(2) From the set of all prime implicants, a set of essential prime implicants are obtained.

(3) The minterms/maxterms which are not covered by the essential prime implicants are taken into consideration and a minimum cover is obtained from the remaining prime implicants.