Table of Contents

3.1 Introduction

- Inventor of Boolean algebra was George Boole (1815 - 1864).

- Designing of any digital system there are three main objectives;
  1) Build a system which operates within given specifications
  2) Build a reliable system
  3) Minimize resources

- Boolean algebra is a system of mathematical logic.
- Any complex logic can be expressed by Boolean function.
- Boolean algebra is governed by certain rules and laws.
- Boolean algebra is different from ordinary algebra & binary number system.
  In ordinary algebra; A + A = 2A and AA = A^2, here A is numeric value.
- In Boolean algebra;
  A + A = A and AA = A, here A has logical significance, but no numeric significance.

Table: Difference between Binary, Ordinary and Boolean system

Binary number system	Ordinary no. system	Boolean algebra
1 + 1 = 1 0	1 + 1 = 2	1 + 1 = 1
-	A + A = 2A and AA = A^2	A + A = A and AA = A

- In Boolean algebra, nothing like subtracting or division, no negative or fractional numbers.
- Boolean algebra represent logical operation only. Logical multiplication is same as AND operation and logical addition is same as OR operation.
- Boolean algebra has only two values 0 & 1.
- In Boolean algebra; If A = 0 then A ≠ 1. & If A = 1 then A ≠ 0.

3.1.1 Advantages of Boolean Algebra

1. Minimize the no. of gates used in circuit.
2. Decrease the cost of circuit.
3. Minimize the resources.
4. Less fabrication area is required to design a circuit.
5. Minimize the designer's time.
6. Reducing to a simple form. Simpler the expression more simple will be hardware.
7. Reduce the complexity.

3.2 Boolean Algebra Terminology

1. **Variable** : The symbol which represent an arbitrary elements of a Boolean algebra is known as variable.
   e.g. F = A + BC, here A, B and C are variable and it can have value either 1 or 0.

2. **Constant** : In expression F = A + 1, the first term A is variable and second term 1 is known as constant. Constant may be 1 or 0.
3. **Complement** : A complement of any variable is represented by a "$\overline{\ \ }$" (BAR) over any variable.
e.g. Complement of A is $\overline{A}$.
4. **Literal** : Each occurrence of a variable in Boolean function either in a non-complemented or complemented form is called literal.
5. **Boolean Function** : Boolean expressions are constructed by connecting the Boolean constants and variable with the Boolean operations. This Boolean expressions are also known as Boolean Formula.
e.g. F(A, B, C) = $(A+\overline{B})$ C OR F = $(A+\overline{B})$ C

3.3 Logic Operators

1. **AND** : Denoted by · (e.g. A AND B = A · B)
2. **OR** : Denoted by + (e.g. A OR B = A + B)
3. **NOT** OR **Complement** : Denoted by "$\overline{\ \ }$" (BAR) or $(\ \ )'$ (e.g. $\overline{A}$ or $(A)'$)

3.4 Axioms or Postulates

Axioms 1	0 · 0 = 0	**Axioms 5**	0 + 0 = 0	**Axioms 9**	1' = 0
Axioms 2	0 · 1 = 0	**Axioms 6**	0 + 1 = 1	**Axioms 10**	0' = 1
Axioms 3	1 · 0 = 0	**Axioms 7**	1 + 0 = 1		
Axioms 4	1 · 1 = 1	**Axioms 8**	1 · 1 = 1		

3.5 Boolean Algebra's Laws and Theorems

1. Complementation Laws:

- The term complement simply means to invert, i.e. to change 0's to 1's and 1's to 0's.

Law 1: 0' = 1
Law 2: 1' = 0
Law 3: If A = 0 then A' = 1
Law 4: If A = 1 then A' = 0
Law 5: A'' = A

2. AND Laws:

Law 1: A · 0 = 0
Law 2: A · 1 = A
Law 3: A · A = A
Law 4: A · A' = 0

3. OR Laws:

Law 1: A + 0 = A
Law 2: A + 1 = 1
Law 3: A + A = A
Law 4: A + A' = 1

4. Commutative Laws:

- Commutative laws allow change in position of AND or OR variables.

Law 1: A + B = B + A

Proof:

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

=

B	A	B + A
0	0	0
0	1	1
1	0	1
1	1	1

- This law can be extended to any numbers of variables for e.g.

$$A + B + C = B + A + C = C + B + A = C + A + B$$

Law 2: A · B = B · A

Proof:

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

=

B	A	B · A
0	0	0
0	1	0
1	0	0
1	1	1

- This law can be extended to any numbers of variables for e.g.

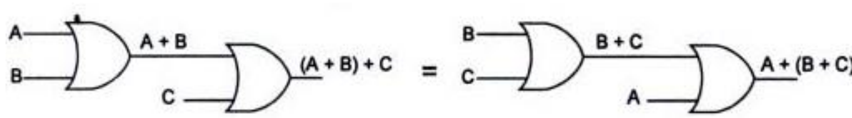
$$A \cdot B \cdot C = B \cdot A \cdot C = C \cdot B \cdot A = C \cdot A \cdot B$$

5. Associative Laws:

- The associative laws allow grouping of variables.

Law 1: (A + B) + C = A + (B + C)

Proof:



A	B	C	A + B	(A + B) + C
0	0	0	0	0
0	0	1	0	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	1	1

=

A	B	C	B + C	A + (B + C)
0	0	0	0	0
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	1

- This law can be extended to any no. of variables for e.g.

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

Law 2: (A · B) · C = A · (B · C)

Proof:

A	B	C	AB	(AB)C
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	1	0
1	1	1	1	1

=

A	B	C	BC	A(BC)
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	0	0
1	1	0	0	0
1	1	1	1	1

- This law can be extended to any no. of variables for e.g.

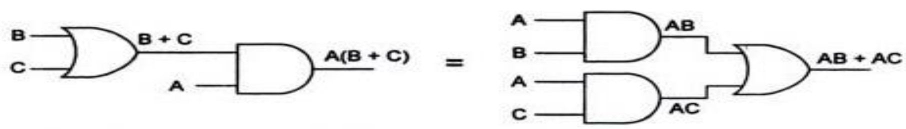
A · (B · C · D) = (A · B · C) · D = (A · B) · (C · D)

6. ***Distributive Laws:***

- The distributive laws allow factoring or multiplying out of expressions.

Law 1: A (B + C) = AB + AC

Proof:



A	B	C	B + C	A(B + C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

=

A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Law 2: A + BC = (A + B) (A + C)

Proof: R.H.S. = (A + B) (A + C)
= AA + AC + BA + BC
= A + AC + BA + BC
= A + BC (∵ A(1 + C + B) = A)
= L.H.S.

Law 3: A + A'B = A + B

Proof: L.H.S. = A + A'B
= (A + A') (A + B)
= A + B
= R.H.S.

7. ***Idempotence Laws:***

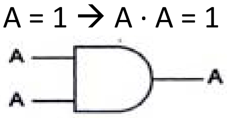
- Idempotence means the same value.

Law 1: A · A = A

Proof:

Case 1: If A = 0 → A · A = 0 · 0 = 0 = A

Case 2: If A = 1 → A · A = 1 · 1 = 1 = A

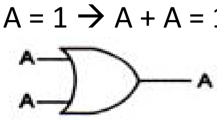


Law 2: A + A = A

Proof:

Case 1: If A = 0 → A + A = 0 + 0 = 0 = A

Case 2: If A = 1 → A + A = 1 + 1 = 1 = A



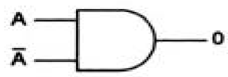
8. Complementation Law / Negation Law:

Law 1: A · A' = 0

Proof:

Case 1: If A = 0 → A · A' = 0 · 1 = 0

Case 2: If A = 1 → A · A' = 1 · 0 = 0

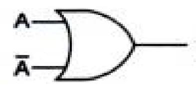


Law 2: A + A' = 1

Proof:

Case 1: If A = 0 → A + A' = 0 + 1 = 1

Case 2: If A = 1 → A + A' = 1 + 0 = 1



9. Double Negation / Involution Law:

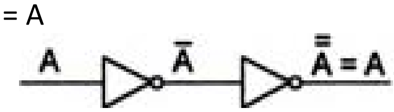
- This law states that double negation of a variables is equal to the variable itself.

  Law: A'' = A

  Proof:

  Case 1: If A = 0 → A'' = 0'' = 0 = A

  Case 2: If A = 1 → A'' = 1'' = 1 = A



- Any odd no. of inversion is equivalent to single inversion.
- Any even no. of inversion is equivalent to no inversion at all.

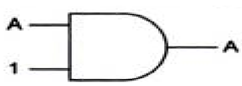
10. Identity Law:

Law 1: A · 1 = A

Proof:

Case 1: If A= 1 → A · 1 = 1 · 1 = 1 = A

Case 2: If A= 0 → A · 0 = 0 · 0 = 0 = A

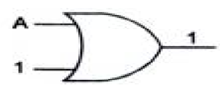


Law 2: A + 1 = 1

Proof:

Case 1: If A= 1 → A + 1 = 1 + 1 = 1 = A

Case 2: If A= 0 → A + 0 = 0 + 0 = 0 = A



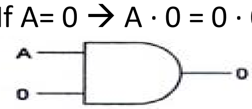
11. Null Law:

Law 1: A · 0 = 0

Proof:

Case 1: If A= 1 → A · 0 = 1 · 0 = 0 = 0

Case 2: If A= 0 → A · 0 = 0 · 0 = 0 = 0



Law 2: A + 0 = A

Proof:

Case 1: If A= 1 → A + 0 = 1 + 0 = 1 = A

Case 2: If A= 0 → A + 0 = 0 + 0 = 0 = A

12. Absorption Law:

Law 1: A + AB = A

Proof: L.H.S. = A + AB
= A (1 + B)
= A (1)
= A
= R.H.S.

Law 2: A (A + B) = A

Proof: L.H.S. = A (A + B)
= A · A + AB
= A + AB
= A (1 + B)
= A
= R.H.S.

13. Consensus Theorem:

Theorem 1:

A · B + A’C + BC = AB + A’C

Proof: L.H.S. = AB + A’C + BC
= AB + A’C + BC (A +A’)
= AB + A’C + BCA + BCA’
= AB (1 + C) + A’C (1 + B)
= AB + A’C
= R.H.S.

- This theorem can be extended as,
  AB + A’C + BCD = AB + A’C

Theorem 2:

(A + B) (A’ + C) (B + C) = (A + B) (A’ + C)

Proof:
L.H.S. = (A + B) (A’ + C) (B + C)
= (AA’ + AC + A’B + BC) (B + C)
= (0 + AC + A’B + BC) (B + C)
= ACB+ACC+A’BB+A’BC+BCB+BCC
= ABC + AC + A’B + A’BC + BC + BC
= ABC + AC + A’B + A’BC + BC
= AC (1 + B) + A’B (1 + C) + BC
= AC + A’B + BC
= AC + A’B ……………………………(1)

R.H.S. = (A + B) (A’ + C)
= AA’ + AC + BA’ + BC
= 0 + AC + BA’ + BC
= AC + A’B + BC
= AC +A’B ……………………………(2)
Eq. (1) = Eq. (2); So, L.H.S = R.H.S.

- This theorem can be extended to any no. of variables.
  (A + B) (A’ + C) (B + C + D) = (A + B) (A’ + C)

14. Transposition theorem:

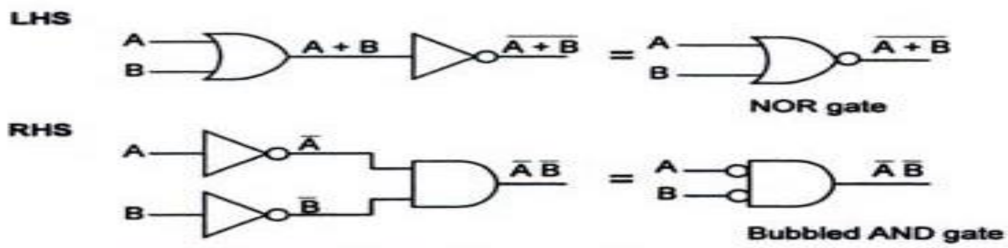
Theorem: AB + A’C = (A + C) (A’ +B)

Proof: R.H.S. = (A + C) (A’ +B)
= AA’ + AB + CA’ + CB
= 0 + AB + CA’ + CB
= AB + CA’ + CB
= AB + A’C (∵ AB + A’C + BC = AB + A’C)
=L.H.S.

15. De Morgan’s Theorem:

Law 1: (A + B)’ = A’ · B’ OR (A + B + C)’ = A’ · B’ · C’

Proof:



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

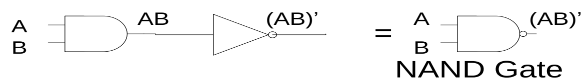
=

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

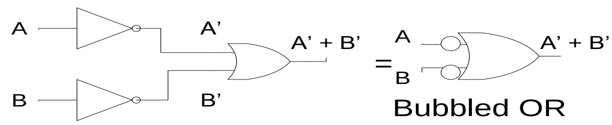
Law 2: (A· B)’ = A’ + B’ OR (A · B · C)’ = A’ + B’ + C’

Proof:

LHS



RHS



A	B	AB	(AB)’
0	0	0	1
0	1	0	1
1	0	0	1
0	1	1	0

=

A	B	A’	B’	A’+B’
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
0	1	0	0	0

16. Duality Theorem:

- Duality theorem arises as a result of presence of two logic system i.e. positive & negative logic system.
- This theorem helps to convert from one logic system to another.
- From changing one logic system to another following steps are taken:
  1) 0 becomes 1, 1 becomes 0.
  2) AND becomes OR, OR becomes AND.
  3) ‘+’ becomes ‘·’, ‘·’ becomes ‘+’.
  4) Variables are not complemented in the process.

Given Expression	*Dual*
1. $\overline{0} = 1$	$\overline{1} = 0$
2. $0 \cdot 1 = 0$	$1 + 0 = 1$
3. $0 \cdot 0 = 0$	$1 + 1 = 1$
4. $1 \cdot 1 = 1$	$0 + 0 = 0$
5. $A \cdot 0 = 0$	$A + 1 = 1$
6. $A \cdot 1 = A$	$A + 0 = A$
7. $A \cdot A = A$	$A + A = A$
8. $A \cdot \overline{A} = 0$	$A + \overline{A} = 1$
9. $A \cdot B = B \cdot A$	$A + B = B + A$
10. $A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$
11. $A \cdot (B + C) = AB + AC$	$A + BC = (A + B)(A + C)$
12. $A(A + B) = A$	$A + AB = A$
13. $A \cdot (A \cdot B) = A \cdot B$	$A + A + B = A + B$
14. $\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\ \overline{B}$

3.5.1 Reduction of Boolean Expression

3.5.1.1 De-Morganized the following functions

(1) $F = [(A + B')(C + D')]'$

Sol:

$F = [(A + B')(C + D')]'$

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

$\therefore F = A' B'' + C'D''$

$\therefore F = A'B + C'D$

(2) $F = [(AB)'(CD + E'F)((AB)' + (CD)')]'$

Sol:

$F = [(AB)'(CD + E'F)((AB)' + (CD)')]'$

$\therefore F = (AB)'' + (CD + E'F)' + ((AB)' + (CD)')'$

$\therefore F = AB + [(CD)'(E'F)'] + [(AB)''(CD)'']$

$\therefore F = AB + (C' + D')(E + F') + ABCD$

(3) $F = [(AB)' + A' + AB]'$

Sol:

$F = [(AB)' + A' + AB]'$

$\therefore F = (AB)'' \cdot A'' \cdot (AB)'$

$\therefore F = ABA(A' + B')$

$\therefore F = AB(A' + B')$

$\therefore F = ABA' + ABB'$

(4) $F = [(P + Q')(R' + S)]'$

Sol:

$F = [(P + Q')(R' + S)]'$

$\therefore F = (P + Q')' + (R' + S)'$

$\therefore F = P'Q'' + R''S'$

$\therefore F = P'Q + RS'$

(5) $F = [P(Q + R)]'$

Sol:

$F = [P(Q + R)]'$

$F = P' + (Q + R)'$

$F = P' + Q'R'$

(6) $F = [[(A + B)'(C + D)']'[(E + F)'(G + H)']']'$

Sol:

$F = [[(A + B)'(C + D)']'[(E + F)'(G + H)']']'$

$\therefore F = [(A + B)'(C + D)']'' + [(E + F)'(G + H)']''$

$\therefore F = [(A + B)'(C + D)'] + [(E + F)'(G + H)']$

$\therefore F = A'B'C'D' + E'F'G'H'$

3.5.1.2 Reduce the following functions using Boolean Algebra's Laws and Theorems

(1) $F = A + B[AC + (B + C')D]$

Sol:

$F = A + B[AC + (B + C')D]$

$\therefore F = A + B[AC + (BD + C'D)]$

$\therefore F = A + ABC + BBD + BC'D$

$\therefore F = A + ABC + BD + BC'D$

$\therefore F = A(1 + BC) + BD(1 + C')$

$\therefore F = A(1) + BD(1)$

$\therefore F = A + BD$

(2) $F = A[B + C'(AB + AC')']$

Sol:

$F = A[B + C'(AB + AC')']$

$\therefore F = A[B + C'(AB)'(AC')']$

$\therefore F = A[B + C'(A' + B')(A' + C)]$

$\therefore F = A[B + (A'C' + B'C')(A' + C)]$

$\therefore F = A[B + (A'C'A' + B'C'A')(A'C'C + B'C'C)]$

$\therefore F = A[B + (A'C' + B'C'A')(0 + 0)]$

$\therefore F = A[B + A'C'(1 + B')]$

$\therefore F = AB + A'AC'$

$\therefore F = AB$

(3) $F = (A + (BC)')' (AB' + ABC)$

Sol:

$F = (A + (BC)')' (AB' + ABC)$

$\therefore F = (A' (BC)'') (AB' + ABC)$

$\therefore F = (A'BC) (AB' + ABC)$

$\therefore F = A'BCAB' + A'BCABC$

$\therefore F = 0 + 0$

$\therefore F = 0$

(4) $F = [(A + B) (A' + B)] + [(A + B) (A + B')]$

Sol:

$F = [(A + B) (A' + B)] + [(A + B) (A + B')]$

$\therefore F = [AA' + AB + BA' + BB] + [AA + AB' + BA + BB'']$

$\therefore F = [0 + AB + A'B + B] + [A + AB' + AB + 0]$

$\therefore F = [B (A + A' + 1)] + [A (1 + B' + B)]$

$\therefore F = B + A$

$\therefore F = A + B$

(5) $F = [(A + B') (A' + B')] + [(A' + B') (A' + B')]$

Sol:

$F = [(A + B') (A' + B')] + [(A' + B') (A' + B')]$

$\therefore F = [AA' + AB' + B'A' + B'B'] + [A'A' + A'B' + B'A' + B'B']$

$\therefore F = [0 + AB' + A'B' + B'] + [A' + A'B' + B']$

$\therefore F = [B' (A + A' + 1)] + [A' + B'(1)]$

$\therefore F = B' + A' + B'$

$\therefore F = A' + B'$

(6) $F = (A + B) (A + B') (A' + B) (A' + B')$

Sol:

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

$\therefore F = (AA + AB' + BA + BB') (A'A' + A'B' + BA' + BB')$

$\therefore F = [A (1 + B' + B)] [A' (1 + B' + B)]$

$\therefore F = [A(1)] [A'(1)]$

$\therefore F = AA'$

$\therefore F = 0$

(7) $F = (B + BC) (B + B'C) (B + D)$

Sol:

$F = (B + BC) (B + B'C) (B + D)$

$\therefore F = (BB + BB'C + BBC + BCB'C) (B + D)$

$\therefore F = (B + 0 + BC + 0) (B + D)$

$\therefore F = B (B + D) \quad (\because B + BC = B(1 + C) = B)$

$\therefore F = B + BD \quad (\because B (B + D) = BB + BD)$

$\therefore F = B \quad (\because B + BD = B (1 + D))$

(8) $F = AB'C + B + BD' + ABD' + A'C$

Reduce the function to minimum no. of literals.

Sol:

$F = AB'C + B + BD' + ABD' + A'C$

$\therefore F = AB'C + B (1 + D' + AD') + A'C$

$\therefore F = AB'C + B + A'C$

$\therefore F = C (A' + AB') + B$

$\therefore F = C (A' + A) (A' + B') + B$

$\therefore F = C (1) (A' + B') + B$

$\therefore F = C (A' + B') + B$

$\therefore F = A'C + CB' + B$

$\therefore F = A'C + (C + B) (B' + B)$

$\therefore F = A'C + (B + C) (1)$

$\therefore F = A'C + B + C$

$\therefore F = C (1 + A') + B$

$\therefore F = B + C$

Here, 2 literals are present B & C.

(9) $F = AB + AB'C + BC'$

Sol:

$F = AB + AB'C + BC'$

$\therefore F = A (B + B'C) + BC'$

$\therefore F = A (B + B') (B + C) + BC'$

$\therefore F = AB + AC + BC' \quad \{\because B + B' = 1\}$

$\therefore F = CA + C'B + AB$

$\therefore F = CA + C'B \quad \{\because \textit{consensus theorem}\}$

3.6 Different forms of Boolean Algebra

- There are two types of Boolean form
  1) Standard form
  2) Canonical form

3.6.1 Standard Form

- ***Definition:*** The terms that form the function may contain one, two, or any number of literals. i.e. each term need not to contain all literals. So standard form is simplified form of canonical form.
- A Boolean expression function may be expressed in a nonstandard form. For example the function:

$$F = (A + C)(AB' + D')$$

- Above function is neither sum of product nor in product of sums. It can be changed to a standard form by using distributive law as below;

$$F = AB' + AD' + AB'C + CD'$$

- There are two types of standard forms: (i) Sum of Product (SOP) (ii) Product of Sum (POS).

3.6.1.1 Standard Sum of Product (SOP)

- SOP is a Boolean expression containing AND terms, called product terms, of one or more literals each. The sum denote the ORing of these terms.
- An example of a function expressed in sum of product is:

$$F = Y' + XY + X'YZ'$$

3.6.1.2 Standard Product of Sum (POS)

- The OPS is a Boolean expression containing OR terms, called sum terms. Each terms may have any no. of literals. The product denotes ANDing of these terms.
- An example of a function expressed in product of sum is:

$$F = X(Y' + Z)(X' + Y + Z' + W)$$

3.6.2 Canonical Form

- ***Definition:*** The terms that form the function contain all literals. i.e. each term need to contain all literals.
- There are two types of canonical forms: (i) Sum of Product (SOP) (ii) Product of Sum (POS).

3.6.2.1 Sum of Product (SOP)

- A canonical SOP form is one in which a no. of product terms, each one of which contains all the variables of the function either in complemented or non-complemented form, summed together.
- Each of the product term is called **"MINTERM"** and denoted as lower case 'm' or 'Σ'.
- For minterms, **Each non-complemented variable → 1 & Each complemented variable → 0**
- **For example,**
  1. $XYZ = 111 = m_7$
  2. $A'BC = 011 = m_3$
  3. $P'Q'R' = 000 = m_0$
  4. $T'S' = 00 = m_0$

3.6.2.1.1 Convert to MINTERM

(1) F = P'Q' + PQ

Sol:

F = 00 + 11

∴ F = $m_0 + m_3$

∴ F = Σm(0,3)

(2) F = X'Y'Z + XY'Z' + XYZ

Sol:

F = 001 + 100 + 111

∴ F = $m_1 + m_4 + m_7$

∴ F = Σm(1,4,7)

(3) F = XY'ZW + XYZ'W' + X'Y'Z'W'

Sol:

F = 1011 + 1100 + 0000

∴ F = $m_{11} + m_{12} + m_0$

∴ F = Σm(0,11,12)

3.6.2.2 Product of Sum (POS)

- A canonical POS form is one in which a no. of sum terms, each one of which contains all the variables of the function either in complemented or non-complemented form, are multiplied together.
- Each of the product term is called "**MAXTERM**" and denoted as upper case 'M' or 'Π'.
- For maxterms, **Each non-complemented variable → 0 & Each complemented variable → 1**
- **For example,**
  1. X'+Y'+Z = 110 = M_6
  2. A'+B+C'+D = 1010 = M_{10}

3.6.2.2.1 Convert to MAXTERM

(1) F = (P'+Q)(P+Q')

Sol:

F = (10)(01)

∴ F = $M_2 \cdot M_1$

∴ F = ΠM(1,2)

(2) F = (A'+B+C)(A+B'+C)(A+B+C')

Sol:

F = (100) (010) (001)

∴ F = $M_4\ M_2\ M_1$

∴ F = ΠM(1,2,4)

(3) F = (X'+Y'+Z'+W)(X'+Y+Z+W')(X+Y'+Z+W')

Sol:

F = (1110)(1001)(0101)

∴ F = $M_{14} \cdot M_9 \cdot M_5$

∴ F = ΠM(5,9,14)

3.6.2 MINTERMS & MAXTERMS for 3 Variables

Table: Representation of Minterms and Maxterms for 3 variables

Row No.	A B C	Minterms	Maxterms
0	0 0 0	$A'B'C' = m_0$	$A + B + C = M_0$
1	0 0 1	$A'B'C = m_1$	$A + B + C' = M_1$
2	0 1 0	$A'BC' = m_2$	$A + B' + C = M_2$
3	0 1 1	$A'BC = m_3$	$A + B' + C' = M_3$
4	1 0 0	$AB'C' = m_4$	$A' + B + C = M_4$
5	1 0 1	$AB'C = m_5$	$A' + B + C' = M_5$
6	1 1 0	$ABC' = m_6$	$A' + B' + C = M_6$
7	1 1 1	$ABC = m_7$	$A' + B' + C' = M_7$

3.6.3 Conversion between Canonical Forms

3.6.3.1 Convert to MINTERMS

1. F(A,B,C,D) = $\Pi M(0,3,7,10,14,15)$

Solution:

Take complement of the given function;

$\therefore F'(A,B,C,D) = \Pi M(1,2,4,5,6,8,9,11,12,13)$

$\therefore F'(A,B,C,D) = (M_1\, M_2\, M_4\, M_5\, M_6\, M_8\, M_9\, M_{11}\, M_{12}\, M_{13})'$

Put value of MAXTERM in form of variables;

$\therefore F'(A,B,C,D) = [(A+B+C+D')(A+B+C'+D)(A+B'+C+D)(A+B'+C+D')(A+B'+C'+D)$
$(A'+B+C+D)(A'+B+C+D')(A'+B+C'+D')(A'+B'+C+D)(A'+B'+C+D')]'$

$\therefore F'(A,B,C,D) = (A'B'C'D) + (A'B'CD') + (A'BC'D') + (A'BC'D) + (A'BCD')$
$+ (AB'C'D') + (AB'C'D) + (AB'CD) + (ABC'D') + (ABC'D)$

$\therefore F'(A,B,C,D) = m_1 + m_2 + m_4 + m_5 + m_6 + m_8 + m_9 + m_{11} + m_{12} + m_{13}$

$\therefore F'(A,B,C,D) = \Sigma m(1,2,4,5,6,8,9,11,12,13)$

In general, $M_j' = m_j$

2. F = A + B'C

Solution:

A → B & C is missing. So multiply with (B + B') & (C + C')

B'C → A is missing. So multiply with (A + A').

∴ A = A (B + B') (C + C')

∴ A = (AB + AB') (C + C')

∴ A = ABC + AB'C + ABC' + AB'C'

And, B'C = B'C (A + A')

∴ B'C = AB'C + A'B'C

So, F = ABC + AB'C + ABC' + AB'C' + AB'C + A'B'C

∴ F = ABC + AB'C + ABC' + AB'C' + A'B'C

∴ F = 111 + 101 + 110 + 100 + 001

∴ F = $m_7 + m_6 + m_5 + m_4 + m_1$

∴ F = Σm(1,4,5,6,7)

3.6.3.2 Convert to MAXTERMS

1. F = Σ(1,4,5,6,7)

Solution:

Take complement of the given function;

∴ F'(A,B,C) = Σ(0,2,3)

∴ F'(A,B,C) = $(m_0 + m_2 + m_3)$

Put value of MINTERM in form of variables;

∴ F'(A,B,C) = (A'B'C' + A'BC' + A'BC)'

∴ F'(A,B,C) = (A+B+C)(A+B'+C)(A+B'+C')

∴ F'(A,B,C) = $M_0 \cdot M_2 \cdot M_3$

∴ F'(A,B,C) = ΠM(0,2,3)

In general, $m_j' = M_j$

2. F = A (B + C')

Solution:

A → B & C is missing. So add BB' & CC'

B + C' → A is missing. So add AA'

∴ A = A + BB' + CC'

∴ A = (A + B + CC’) (A + B’ + CC’)
∴ A = (A + B + C) (A + B + C’) (A + B’ + C) (A + B’ + C’)

And, B + C’ = B + C’ + AA’
∴ B + C’ = (A + B + C’) (A’ + B + C’)

So, F = (A + B + C) (A + B + C’) (A + B’ + C) (A + B’ + C’) (A + B + C’) (A’ + B + C’)
∴ F = (A + B + C) (A + B + C’) (A + B’ + C) (A + B’ + C’) (A’ + B + C’)
∴ F = (000) (001) (010) (011) (101)
∴ F = ΠM(0,1,2,3,5)

3. F = XY + X’Z

Solution:

∴ F = XY + X’Z
∴ F = (XY + X’) (XY + Z)
∴ F = (X +X’) (Y + X’) (X + Z) (Y + Z)
∴ F = (X’ + Y) (X + Z) (Y + Z)

X’ + Y → Z is missing. So add ZZ’
X + Z → Y is missing. So add YY’
Y + Z → X is missing. So add XX’

∴ X’ + Y = X’ + Y + ZZ’
∴ X’ + Y = (X’ + Y + Z) (X’ + Y + Z’)

And, X + Z = X’ + Z + YY’
∴ X + Z = (X + Y + Z) (X + Y’ + Z)

And, Y + Z = Y + Z + XX’
∴ Y + Z = (X + Y + Z) (X’ + Y + Z)

So, F = (X’ + Y + Z) (X’ + Y + Z’) (X + Y + Z) (X + Y’ + Z) (X + Y + Z) (X’ + Y + Z)
∴ F = (100) (101) (000) (010)
∴ F = $M_4 \, M_5 \, M_0 \, M_2$
∴ F = ΠM(0,2,4,5)

3.7 Karnaugh Map (K-Map)

- A Boolean expression may have many different forms.
- With the use of K-map, the complexity of reducing expression becomes easy and Boolean expression obtained is simplified.
- K-map is a pictorial form of truth table and it is alternative way of simplifying Boolean function.
- Instead of using Boolean algebra simplification techniques, you can transfer logic values from a Boolean statement or a truth table into a Karnaugh map (k-map)
- Tool for representing Boolean functions of up to six variables then after it becomes complex.
- K-maps are tables of rows and columns with entries represent 1's or 0's of SOP and POS representations.
- K-map cells are arranged such that adjacent cells correspond to truth table rows that differ in only one bit position (*logical adjacency*)
- K-Map are often used to simplify logic problems with up to 6 variables

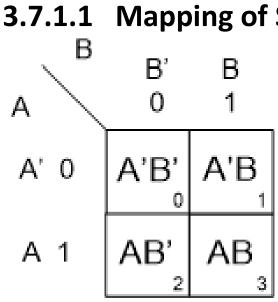
No. of Cells = 2^n, where n is a number of variables.

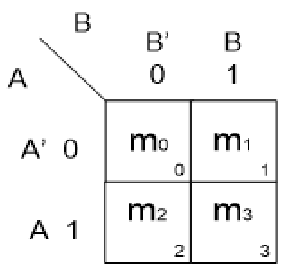
- The Karnaugh map is completed by entering a '1' (or '0') in each of the appropriate cells.
- Within the map, adjacent cells containing 1's (or 0's) are grouped together in twos, fours, or eights and so on.

3.7.1 2 Variable K-Map

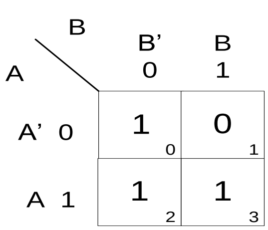
- For 2 variable k-map, there are 2^2 = 4 cells.
- If A & B are two variables then;
  SOP → Minterms → A'B' (m_0, 00) ; A'B (m_1, 01) ; AB' (m_2, 10) ; AB (m_3, 11)
  POS → Maxterms → A + B (M_0, 00) ; A + B' (M_1, 01) ; A' + B (M_2, 10) ; A' + B' (M_3, 11)

3.7.1.1 Mapping of SOP Expression

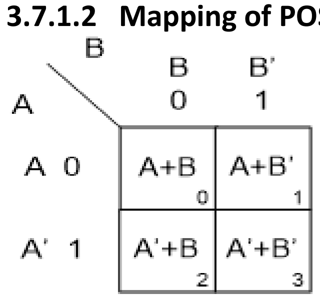


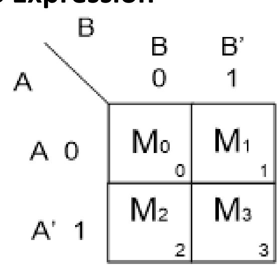


- 1 in a cell indicates that the minterm is included in Boolean expression.
- For e.g. if F = ∑m(0,2,3), then 1 is put in cell no. 0,2,3 as shown below.

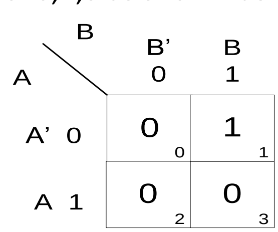


3.7.1.2 Mapping of POS Expression





- 0 in a cell indicates that the maxterm is included in Boolean expression.
- For e.g. if F = ΠM(0,2,3), then 0 is put in cell no. 0,2,3 as shown below.



3.7.1.3 Reduce Sum of Product (SOP) Expression using K-Map

(1) $F = m_0 + m_1$

Sol:

A \ B	B' 0	B 1
A' 0	1_{0}	1_{1}
A 1	0_{2}	0_{3}

F = A'

(2) F = A'B' + AB'

Sol:

A \ B	B' 0	B 1
A' 0	1_{0}	0_{1}
A 1	1_{2}	0_{3}

F = B'

(3) $F = \Sigma(1,3)$

Sol:

A \ B	B' 0	B 1
A' 0	0_{0}	1_{1}
A 1	0_{2}	1_{3}

F = B

(4) $F = m_2 + m_3$

Sol:

A \ B	B' 0	B 1
A' 0	0_{0}	0_{1}
A 1	1_{2}	1_{3}

F = A

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

Sol:

A \ B	B' 0	B 1
A' 0	1_{0}	1_{1}
A 1	0_{2}	1_{3}

F = A' + AB

(6) $\sum m(0,1,2,3)$

Sol:

A \ B	B' 0	B 1
A' 0	1_{0}	1_{1}
A 1	1_{2}	1_{3}

F = 1

3.7.1.4 Reduce Product of Sum (POS) Expression using K-Map

(1) $F = \Pi M(0,2,3,1)$

Sol:

A \ B	B 0	B' 1
A 0	0_{0}	0_{1}
A' 1	0_{2}	0_{3}

F = 0

(2) F = (A+B) (A'+B) (A+B')

Sol:

A \ B	B 0	B' 1
A 0	0_{0}	0_{1}
A' 1	0_{2}	1_{3}

F = A B

(3) $F = M_3 \cdot M_1 \cdot M_2$

Sol:

A \ B	B 0	B' 1
A 0	1_{0}	0_{1}
A' 1	0_{2}	0_{3}

F = A' B'

3.7.2 3 Variable K-Map

- For 3 variable k-map, there are $2^3 = 8$ cells.

3.7.2.1 Mapping of SOP Expression

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	$\bar{A}\bar{B}\bar{C}$ 0	$\bar{A}\bar{B}C$ 1	$\bar{A}BC$ 3	$\bar{A}B\bar{C}$ 2
A 1	$A\bar{B}\bar{C}$ 4	$A\bar{B}C$ 5	ABC 7	$AB\bar{C}$ 6

3.7.2.2 Mapping of POS Expression

A \ BC	$B+C$ 00	$B+\bar{C}$ 01	$\bar{B}+\bar{C}$ 11	$\bar{B}+C$ 10
A 0	$A+B+C$ 0	$A+B+\bar{C}$ 1	$A+\bar{B}+\bar{C}$ 3	$A+\bar{B}+C$ 2
$\bar{A}$ 1	$\bar{A}+B+C$ 4	$\bar{A}+B+\bar{C}$ 5	$\bar{A}+\bar{B}+\bar{C}$ 7	$\bar{A}+\bar{B}+C$ 6

3.7.2.3 Reduce Sum of Product (SOP) Expression using K-Map

(1) F = A’B’C + ABC + A’BC’

Sol:

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	0 $_{0}$	1 $_{1}$	0 $_{3}$	1 $_{2}$
A 1	0 $_{4}$	0 $_{5}$	1 $_{7}$	0 $_{6}$

F = A’B’C + ABC + A’BC’

(2) F = Σ(1,6,7)

Sol:

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	0 $_{0}$	1 $_{1}$	0 $_{3}$	0 $_{2}$
A 1	0 $_{4}$	0 $_{5}$	1 $_{7}$	1 $_{6}$

F = A’B’C + AB

(3) F = A’B’C’ + ABC’ + AB’C’ + A’BC

Sol:

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	1 $_{0}$	0 $_{1}$	1 $_{3}$	0 $_{2}$
A 1	1 $_{4}$	0 $_{5}$	0 $_{7}$	1 $_{6}$

F = B’C’ + AC’ + A’BC

(4) F = Σm(0,1,2,4,5,6)

Sol:

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	1 $_{0}$	1 $_{1}$	0 $_{3}$	1 $_{2}$
A 1	1 $_{4}$	1 $_{5}$	0 $_{7}$	1 $_{6}$

F = B’ + C’

(5) F = $m_3 + m_4 + m_6 + m_7$

Sol:

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	0 $_{0}$	0 $_{1}$	1 $_{3}$	0 $_{2}$
A 1	1 $_{4}$	0 $_{5}$	1 $_{7}$	1 $_{6}$

F = BC + AC’

(6) F = Σm(3,7,1,6,0,2,5,4)

Sol:

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0	1 $_{0}$	1 $_{1}$	1 $_{3}$	1 $_{2}$
A 1	1 $_{4}$	1 $_{5}$	1 $_{7}$	1 $_{6}$

F = 1

3.7.2.4 Reduce Product of Sum (POS) Expression using K-Map

(1) F = (A’+B’+C’) (A’+B+C’)

Sol:

A \ BC	$B+C$ 00	$B+\bar{C}$ 01	$\bar{B}+\bar{C}$ 11	$\bar{B}+C$ 10
A 0	1 $_{0}$	1 $_{1}$	1 $_{3}$	1 $_{2}$
$\bar{A}$ 1	1 $_{4}$	0 $_{5}$	0 $_{7}$	1 $_{6}$

F = (A’ + C’)

(2) F = $M_0 \cdot M_3 \cdot M_7$

Sol:

A \ BC	$B+C$ 00	$B+\bar{C}$ 01	$\bar{B}+\bar{C}$ 11	$\bar{B}+C$ 10
A 0	0 $_{0}$	1 $_{1}$	0 $_{3}$	1 $_{2}$
$\bar{A}$ 1	1 $_{4}$	1 $_{5}$	0 $_{7}$	1 $_{6}$

F = (A + B + C) (B’ + C’)

(3) F = ΠM(1,2,5)

Sol:

A \ BC	$B+C$ 00	$B+\bar{C}$ 01	$\bar{B}+\bar{C}$ 11	$\bar{B}+C$ 10
A 0	1_{0}	0_{1}	1_{3}	0_{2}
$\bar{A}$ 1	1_{4}	0_{5}	1_{7}	1_{6}

F = (B + C’) (A + B + C)

(4) F = ΠM(0,4,1,5,7,3)

Sol:

A \ BC	$B+C$ 00	$B+\bar{C}$ 01	$\bar{B}+\bar{C}$ 11	$\bar{B}+C$ 10
A 0	0_{0}	0_{1}	0_{3}	1_{2}
$\bar{A}$ 1	0_{4}	0_{5}	0_{7}	1_{6}

F = (B) (C’)

(5) F = (A+B+C)(A+B’+C’)(A’+B+C)

Sol:

A \ BC	$B+C$ 00	$B+\bar{C}$ 01	$\bar{B}+\bar{C}$ 11	$\bar{B}+C$ 10
A 0	0_{0}	1_{1}	0_{3}	1_{2}
$\bar{A}$ 1	0_{4}	1_{5}	1_{7}	1_{6}

F = (B + C) (A + B’ + C’)

(6) ΠM(5,7,0,3,2,4,6,1)

Sol:

A \ BC	$B+C$ 00	$B+\bar{C}$ 01	$\bar{B}+\bar{C}$ 11	$\bar{B}+C$ 10
A 0	0_{0}	0_{1}	0_{3}	0_{2}
$\bar{A}$ 1	0_{4}	0_{5}	0_{7}	0_{6}

F = 0

3.7.3 4 Variable K-Map

- For 4 variable k-map, there are 2^4 = 16 cells.

3.7.3.1 Mapping of SOP Expression

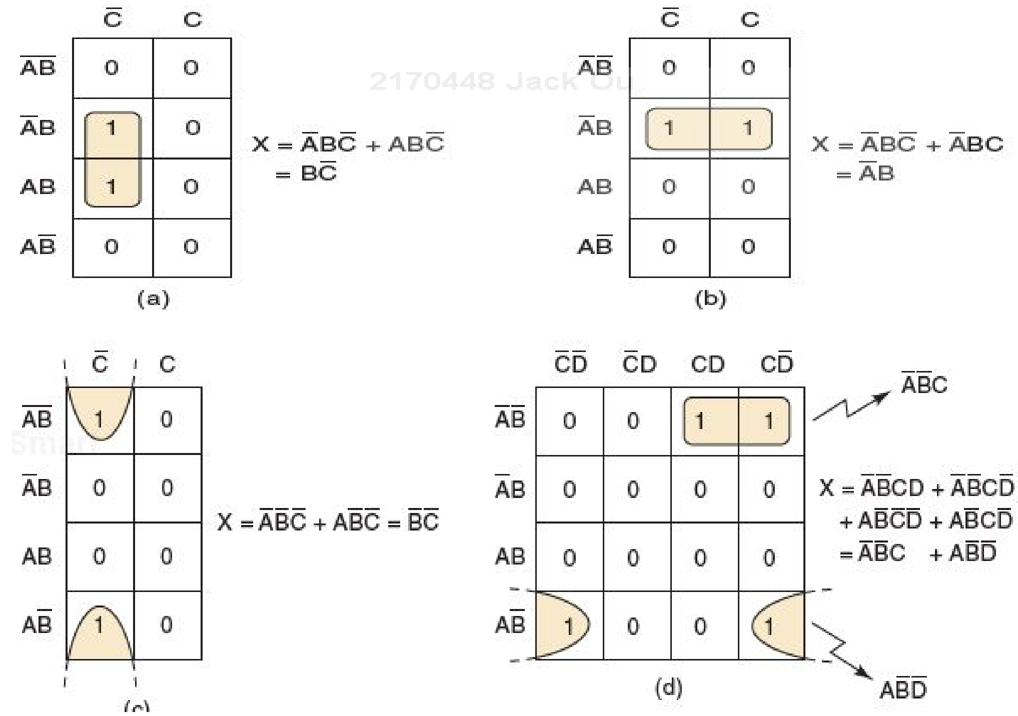
AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

3.7.3.2 Mapping of POS Expression

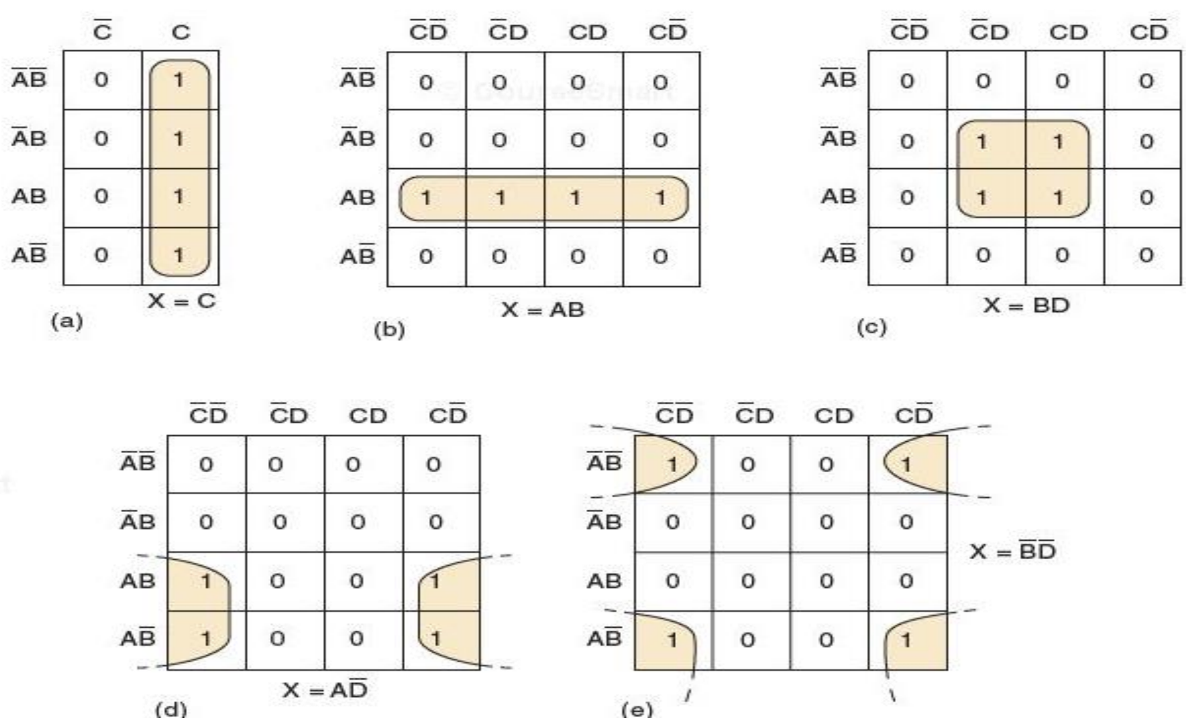
AB \ CD	C + D 00	C + D’ 01	C’ + D’ 11	C’ + D 10
A + B 00	0	1	3	2
A + B’ 01	4	5	7	6
A’ + B’ 11	12	13	15	14
A’ + B 10	8	9	11	10

3.7.3.3 Looping of POS Expression

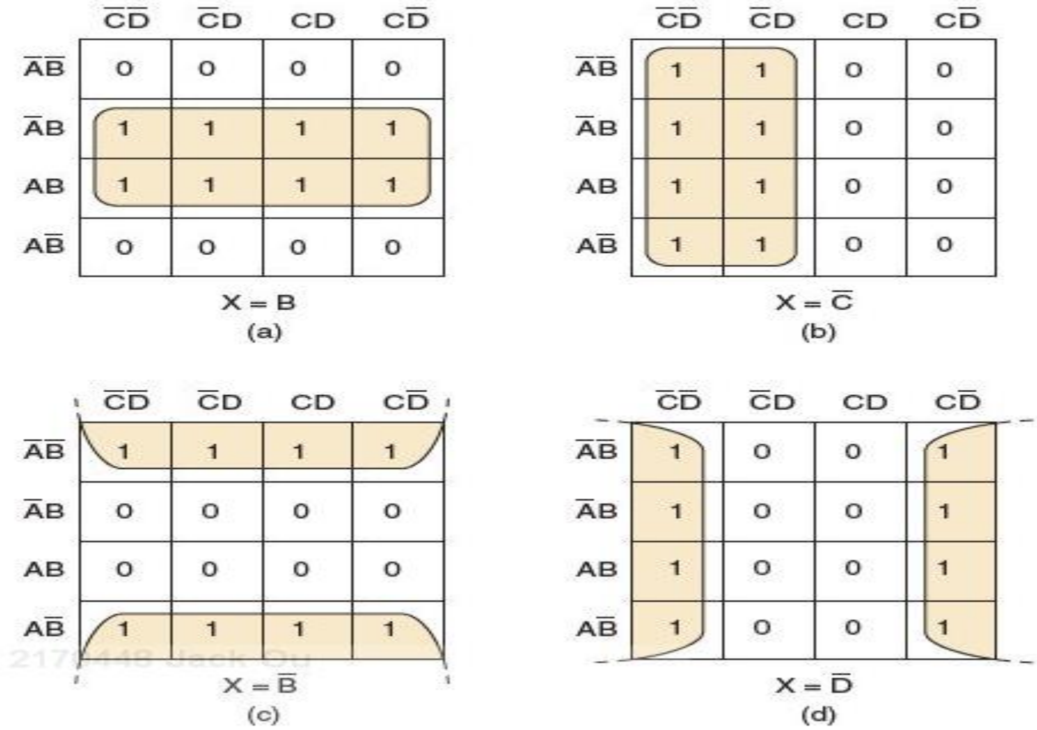
- Looping Groups of Two:



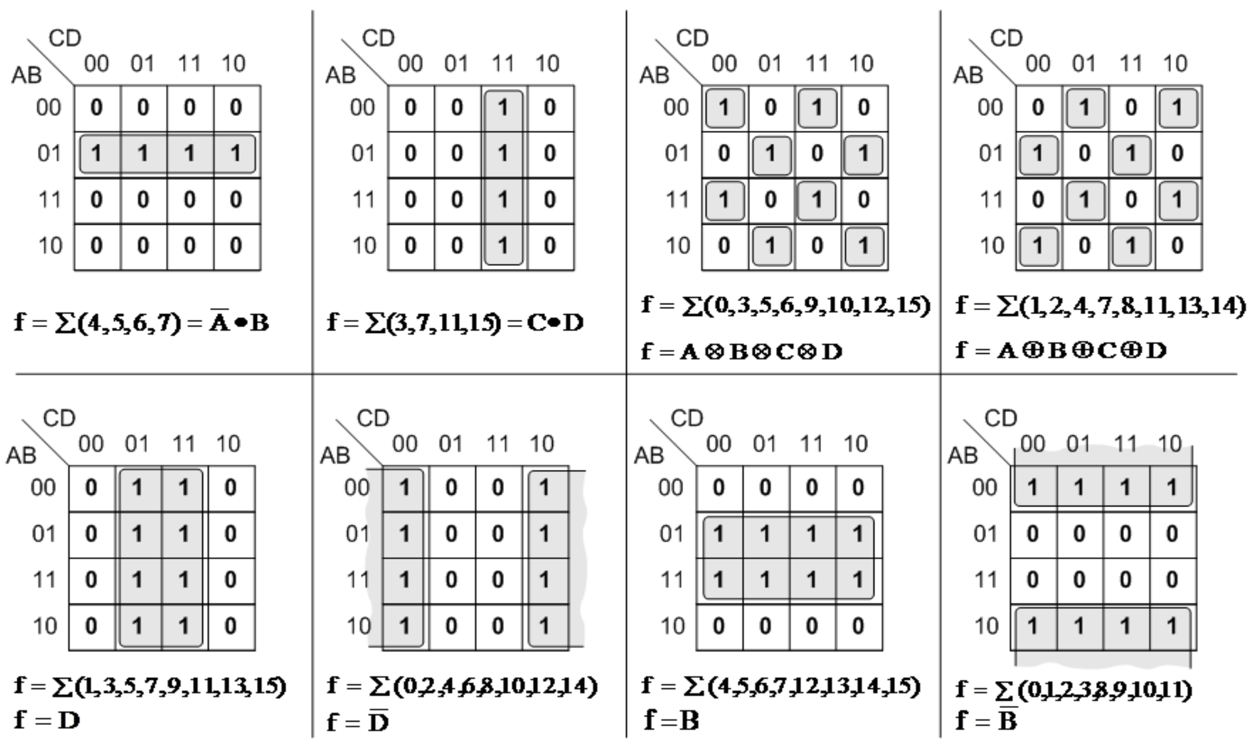
- Looping Groups of Four:



- Looping Groups of Eight:



- Examples



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

$f=\sum(0,8)=\bar{B}\bullet\bar{C}\bullet\bar{D}$

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

$f=\sum(5,13)=B\bullet\bar{C}\bullet D$

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

$f=\sum(13,15)=A\bullet B\bullet D$

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

$f=\sum(4,6)=\bar{A}\bullet B\bullet\bar{D}$

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

$f=\sum(2,3,6,7)=\bar{A}\bullet C$

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

$f=\sum(4,6,12,14)=B\bullet\bar{D}$

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

$f=\sum(2,3,10,11)=\bar{B}\bullet C$

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

$f=\sum(0,2,8,10)=\bar{B}\bullet\bar{D}$

3.7.3.4 Reduce Sum of Product (SOP) Expression using K-Map

(1) F = ∑ (0,1,2,4,5,6,8,9,12,13,14)

Sol:

AB \ CD	00	01	11	10
00	1_{0}	1_{1}	0_{3}	1_{2}
01	1_{4}	1_{5}	0_{7}	1_{6}
11	1_{12}	1_{13}	0_{15}	1_{14}
10	1_{8}	1_{9}	0_{11}	0_{10}

F = C’ + A’D’ + BD’

(2) F = A’B’C’ + B’CD’ + A’BCD’ + AB’C’

Sol:

AB \ CD	00	01	11	10
00	1_{0}	1_{1}	0_{3}	1_{2}
01	0_{4}	0_{5}	0_{7}	1_{6}
11	0_{12}	0_{13}	0_{15}	0_{14}
10	1_{8}	1_{9}	0_{11}	1_{10}

F = B’C’ + B’D’ + A’CD’

(3) F = ∑ (0,1,2,3,5,7,8,9,12,13)

Sol:

AB \ CD	00	01	11	10
00	1_{0}	1_{1}	1_{3}	1_{2}
01	0_{4}	1_{5}	1_{7}	0_{6}
11	1_{12}	1_{13}	0_{15}	0_{14}
10	1_{8}	1_{9}	0_{11}	0_{10}

F = A’B’ + AC’ + A’D

(4) F = ∑ (0,1,3,4,5,6,7,13,15)

Sol:

AB \ CD	00	01	11	10
00	1_{0}	1_{1}	1_{3}	0_{2}
01	1_{4}	1_{5}	1_{7}	1_{6}
11	0_{12}	1_{13}	1_{15}	0_{14}
10	0_{8}	0_{9}	0_{11}	0_{10}

F = A’C’ + A’D + BD + A’B

(5) F = ∑m (5,6,7,9,10,11,13,14,15)

Sol:

AB \ CD	00	01	11	10
00	0 (0)	0 (1)	0 (3)	0 (2)
01	0 (4)	1 (5)	1 (7)	1 (6)
11	0 (12)	1 (13)	1 (15)	1 (14)
10	0 (8)	1 (9)	1 (11)	1 (10)

F = BD + BC + AD + AC

3.7.3.5 Reduce Product of Sum (POS) Expression using K-Map

(1) F = ΠM (0,1,2,5,7,8,9,10,14,15)

Sol:

AB \ CD	C + D 00	C + D' 01	C' + D' 11	C' + D 10
A + B 00	0 (0)	0 (1)	1 (3)	0 (2)
A + B' 01	1 (4)	0 (5)	0 (7)	1 (6)
A' + B' 11	1 (12)	1 (13)	0 (15)	0 (14)
A' + B 10	0 (8)	0 (9)	1 (11)	0 (10)

F = (B + D) (B + C) (A + B’ + D’) (A’ + B’ + C’)

(2) F = $M_1 M_3 M_4 M_7 M_6 M_9 M_{11} M_{12} M_{14} M_{15}$

Sol:

AB \ CD	C + D 00	C + D' 01	C' + D' 11	C' + D 10
A + B 00	1 (0)	0 (1)	0 (3)	1 (2)
A + B' 01	0 (4)	1 (5)	0 (7)	0 (6)
A' + B' 11	0 (12)	1 (13)	0 (15)	0 (14)
A' + B 10	1 (8)	0 (9)	0 (11)	1 (10)

F = (B’ + D) (C’ + B’) (D’ + B)

3.7.3.6 Reduce SOP & POS Expression with Don’t Care Combination using K-Map

(1) F = ∑m (1,5,6,12,13,14) + d (2,4)

Sol:

AB \ CD	00	01	11	10
00	0 (0)	1 (1)	0 (3)	X (2)
01	X (4)	1 (5)	0 (7)	1 (6)
11	1 (12)	1 (13)	0 (15)	1 (14)
10	0 (8)	0 (9)	0 (11)	0 (10)

F = BC’ + BD’ + A’C’D

(2) F = ΠM (4,7,10,11,12,15) · d (6,8)

Sol:

AB \ CD	C + D 00	C + D' 01	C' + D' 11	C' + D 10
A + B 00	1 (0)	1 (1)	1 (3)	1 (2)
A + B' 01	0 (4)	1 (5)	0 (7)	X (6)
A' + B' 11	0 (12)	1 (13)	0 (15)	1 (14)
A' + B 10	X (8)	1 (9)	0 (11)	0 (10)

F = (B’ + C + D) (B’ + C’ + D’) (A’ + B + C’)

3.7.4 5 Variable K-Map

- For 5 variable k-map, there are $2^5 = 32$ cells.

3.7.4.1 Mapping of SOP Expression

AB \ CDE	000	001	011	010	110	111	101	100
00	0	1	3	2	6	7	5	4
01	8	9	11	10	14	15	13	12
11	24	25	27	26	30	31	29	28
10	16	17	19	18	22	23	21	20

3.7.4.2 Reduce Sum of Product (SOP) Expression using K-Map

(1) F = ∑m (0,2,3,10,11,12,13,16,17,18,19,20,21,26,27)
Sol:

AB \ CDE	000	001	011	010	110	111	101	100
00	1	0	1	1	0	0	0	0
01	0	0	1	1	0	0	1	1
11	0	0	1	1	0	0	0	0
10	1	1	1	1	0	0	1	1

F = C'D + B'C'E' + AB'D' + A'BCD'

(2) F = ∑m (0,2,4,6,9,11,13,15,17,21,25,27,29,31)
Sol:

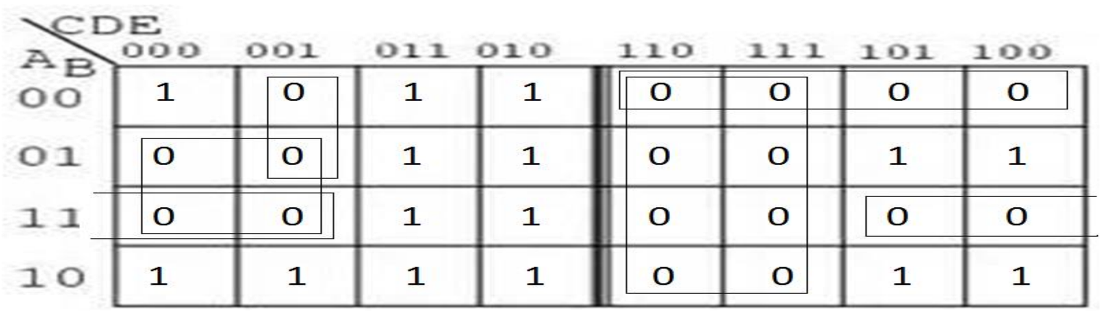
AB \ CDE	000	001	011	010	110	111	101	100
00	1	0	0	1	1	0	0	1
01	0	1	1	0	0	1	1	0
11	0	1	1	0	0	1	1	0
10	0	1	0	0	0	0	1	0

F = BE + AD'E + A'B'E'

3.7.4.3 Reduce Product of Sum (POS) Expression using K-Map

(1) F = ΠM (1,4,5,6,7,8,9,14,15,22,23,24,25,28,29,30,31)

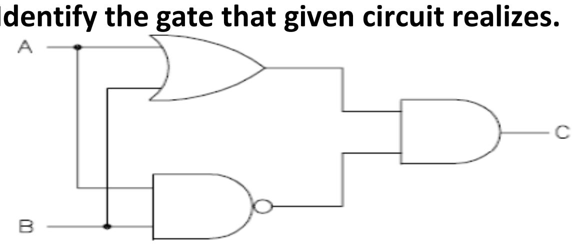
Sol:



F = (B' + C + D) (C' +D') (A + B + C') (A' + B' + D) (A + C + D + E')

3.8 Converting Boolean Expression to Logic Circuit and Vice-Versa

(1) For the logic circuit shown fig., find the Boolean expression and the truth table. Identify the gate that given circuit realizes.



Sol:

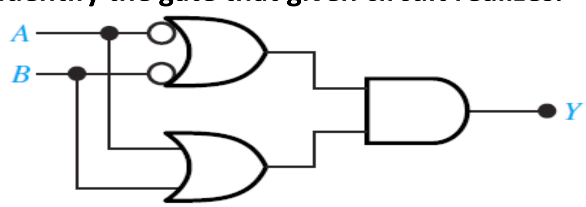
- Here,
  Output of OR gate will be (A+B)
  Output of NAND gate will be (AB)'
  So, C will be AND of these two outputs
  ∴ C = (A+B) · (A·B)'

- Truth table for the same can be given below;

Input					**Output**
A	**B**	**A+B**	**A·B**	**(A·B)'**	**(A+B)·(A·B)'**
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0

- From the truth table it is clear that the circuit realizes Ex-OR gate.

(2) For the logic circuit shown fig., find the Boolean expression and the truth table. Identify the gate that given circuit realizes.



Sol:

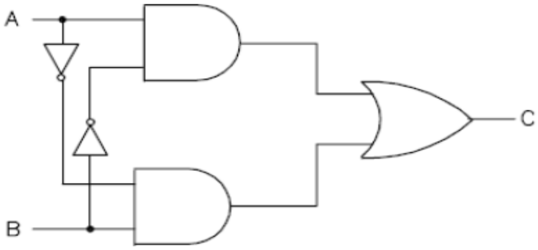
- Here, bubble indicates inversion.
  Hence input of top OR gate is A' and B' and hence its output will be A'+B'
  Output of bottom OR gate will be A+B
  ∴ Y = (A'+B')(A+B)

- Truth table for the same can be given below;

Input						**Output**
A	**B**	**A'**	**B'**	**A'+B'**	**A+B**	**(A'+B')(A+B)**
0	0	1	1	1	0	0
0	1	1	0	1	1	1
1	0	0	1	1	1	1
1	1	0	0	0	1	0

- From the truth table it is clear that the circuit realizes Ex-OR gate.
- **NOTE: NAND = Bubbled OR**

(3) For the logic circuit shown fig., find the Boolean expression.

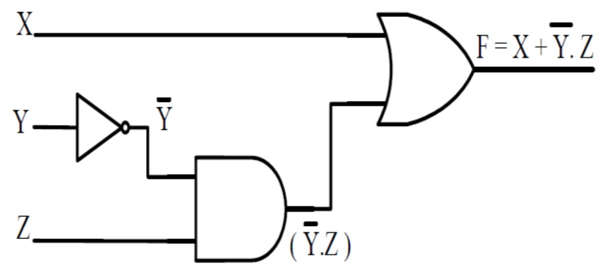


Sol:

- Here,
  Output of top AND gate will be AB’
  Output of bottom AND gate will be A’B
  So, C will be OR of these two outputs
  ∴ C = AB’ + A’B

(4) For the given Boolean expression draw the logic circuit. F = X + (Y’ + Z)

Sol: The expression primarily involves three logic gates i.e NOT, AND and OR.
To generate Y’ a NOT gate is required.
To generate Y’Z an AND gate is required.
To generate final output OR gate is required.



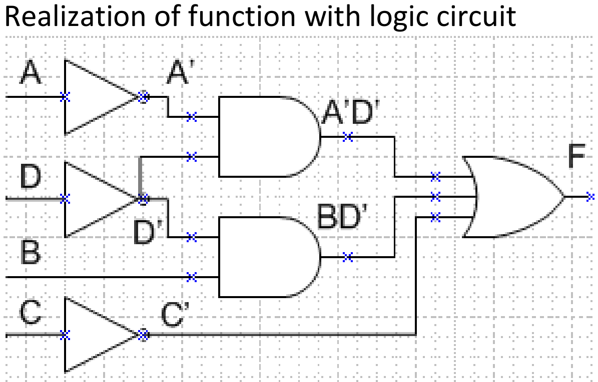
(5) F = ∑ (0,1,2,4,5,6,8,9,12,13,14) Reduce using k-map method. Also realize it with logic circuit or gates with minimum no. of gates.

Sol:

AB \ CD	00	01	11	10
00	1 $_{0}$	1 $_{1}$	0 $_{3}$	1 $_{2}$
01	1 $_{4}$	1 $_{5}$	0 $_{7}$	1 $_{6}$
11	1 $_{12}$	1 $_{13}$	0 $_{15}$	1 $_{14}$
10	1 $_{8}$	1 $_{9}$	0 $_{11}$	0 $_{10}$

F = C’ + A’D’ + BD’

- Realization of function with logic circuit



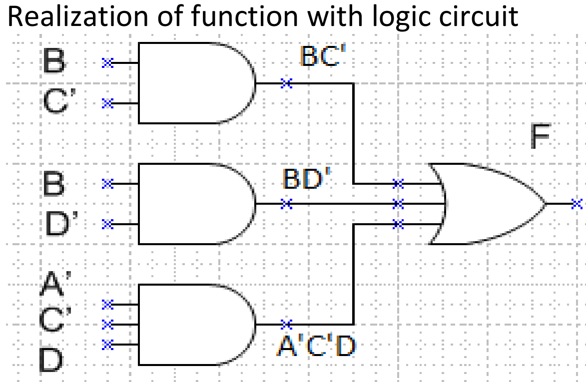
(6) F = ∑m (1,5,6,12,13,14) + d (2,4) Reduce using k-map method. Also realize it with logic circuit.

Sol:

AB \ CD	00	01	11	10
00	0 $_{0}$	1 $_{1}$	0 $_{3}$	X $_{2}$
01	X $_{4}$	1 $_{5}$	0 $_{7}$	1 $_{6}$
11	1 $_{12}$	1 $_{13}$	0 $_{15}$	1 $_{14}$
10	0 $_{8}$	0 $_{9}$	0 $_{11}$	0 $_{10}$

F = BC’ + BD’ + A’C’D

- Realization of function with logic circuit



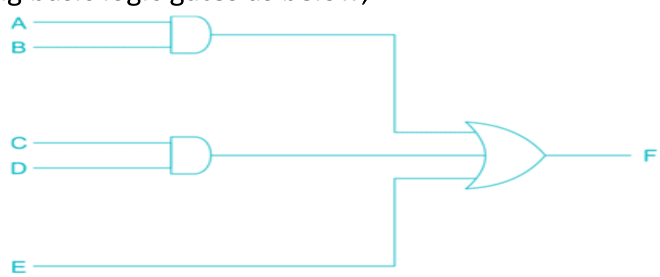
3.9 NAND and NOR Realization/Implementation

- Steps to implement any function using NAND or NOR gate only as below;
  1. Reduce the given function if necessary.
  2. For NAND, add Bubbles at the outputs of AND gates and at the inputs of OR gates.
  3. For NOR, add Bubbles at the outputs of OR gates and at the inputs of AND gates.
  4. Add an inverter symbol wherever you created a Bubble.
  5. Ignore cascading connection of two NOT gates, if any are present.
  6. Replace all gates with NAND gates or NOR gates depending on the type of implementation.

(1) F = AB + CD + E Implement given function using (i) NAND gates only (ii) NOR gates only

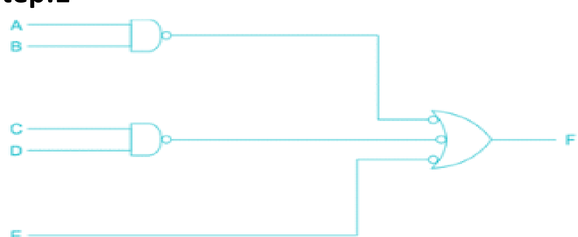
Sol:

- Function realization using basic logic gates as below;

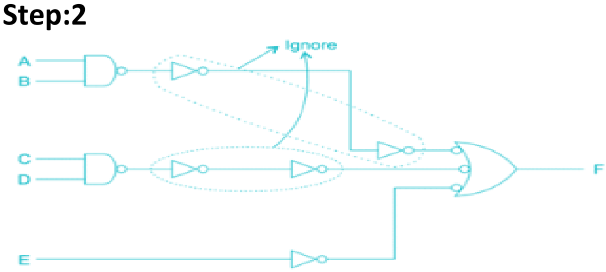


Using NAND gates

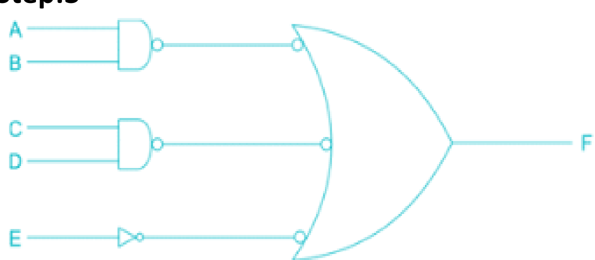
Step:1



Step:2

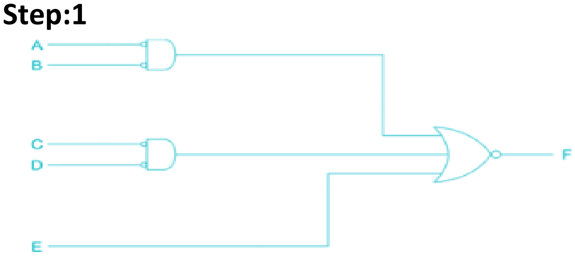


Step:3

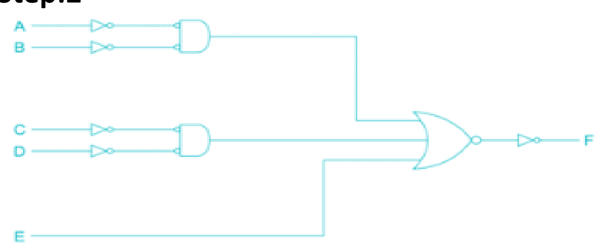


Using NOR gates

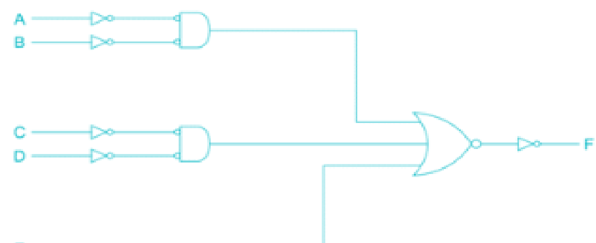
Step:1



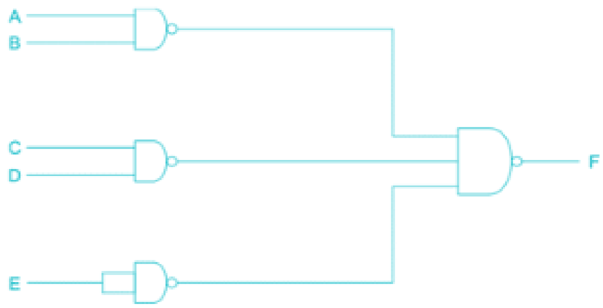
Step:2



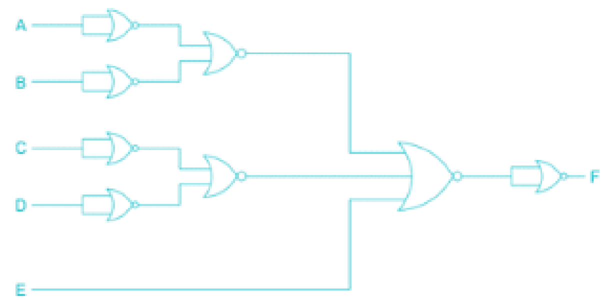
Step:3



Step:4



Step:4



3.10 Tabulation / Quine-McCluskey Method

- As we know that the Karnaugh map method is a very useful and convenient tool for simplification of Boolean functions as long as the number of variables does not exceed four.
- But for case of large number of variables, the visualization and selection of patterns of adjacent cells in the Karnaugh map becomes complicated and too much difficult. For those cases Quine McCluskey tabulation method takes vital role to simplify the Boolean expression.
- The Quine McCluskey tabulation method is a specific step-by-step procedure to achieve guaranteed, simplified standard form of expression for a function.
- Steps to solve function using tabulation method are as follow;

Step 1 − Arrange the given min terms in an **ascending order** and make the groups based on the number of ones present in their binary representations. So, there will be **at most 'n+1' groups** if there are 'n' Boolean variables in a Boolean function or 'n' bits in the binary equivalent of min terms.

Step 2 − Compare the min terms present in **successive groups**. If there is a change in only one-bit position, then take the pair of those two min terms. Place this symbol '_' in the differed bit position and keep the remaining bits as it is.

Step 3 − Repeat step2 with newly formed terms till we get all **prime implicants**.

Step 4 − Formulate the **prime implicant table**. It consists of set of rows and columns. Prime implicants can be placed in row wise and min terms can be placed in column wise. Place '1' in the cells corresponding to the min terms that are covered in each prime implicant.

Step 5 − Find the essential prime implicants by observing each column. If the min term is covered only by one prime implicant, then it is **essential prime implicant**. Those essential prime implicants will be part of the simplified Boolean function.

Step 6 − Reduce the prime implicant table by removing the row of each essential prime implicant and the columns corresponding to the min terms that are covered in that essential prime implicant. Repeat step 5 for reduced prime implicant table. Stop this process when all min terms of given Boolean function are over.

(1) Simplify the following expression to sum of product using Tabulation Method

$$F(A,B,C,D) = \sum(0,1,2,3,4,6,7,11,12,15)$$

Sol:

- Determination of Prime Implicants

Group 0	m0: 0000	√	(0,1) 000-	√	**(0,1,2,3) 00--**	
			(0,2) 00-0	√	**(0,2,4,6) 0--0**	
			(0,4) 0-00	√	(0,2,1,3) 00--	redundant
					(0,4,2,6) 0--0	redundant
Group 1	m1: 0001	√	(1,3) 00-1	√	**(2,3,6,7) 0-1-**	
	m2: 0010	√	(2,3) 001-	√	(2,6,3,7) 0-1-	
	m4: 0100	√	(2,6) 0-10	√		
			(4,6) 01-0	√		
			(4,12) -100			
Group 2	m3: 0011	√	(3,7) 0-11	√	**(3,7,11,15) --11**	
	m6: 0110	√	(3,11) -011	√	(3,11,7,15) --11	redundant
	m12: 1100	√	(6,7) 011-	√		
Group 3	m7: 0111	√	(7,15) -111	√		
	m11: 1011	√	(11,15) 1-11	√		
Group 4	m15: 1111	√				

- Determination of Prime Implicants

	PI	m0	m1	m2	m3	m4	m6	m7	m11	m12	m15
EPI	4 12					x				x	
EPI	0 1 2 3	x	x	x	x						
	0 2 4 6	x		x		x	x				
	2 3 6 7			x	x		x	x			
EPI	3 7 11 15				x			x	x		x
		√	√	√	√	√	√	√	√	√	√

$$F(A,B,C,D) = BC'D' + A'B' + CD + A'D'$$

(2) Simplify the following expression to sum of product using Tabulation Method

$$F(A,B,C,D) = m(0,4,8,10,12,13,15) + d(1,2)$$

Sol:

- Determination of Prime Implicants

Group 0	m0: 0000	√	(0,4) 0-00	√	**(0,4,8,12) –00**
			(0,8) -000	√	(0,8,4,12) --00 redundant
			(0,d1) 000-		**(0,8,d2,10) -0-0**
			(0,d2) 00-0	√	(0,d2,8,10) -0-0 redundant
Group 1	m4: 0100	√	(4,12) -100	√	
	m8: 1000	√	(8,10) 10-0	√	
	d1: 0001	√	(8,12) 1-00	√	
	d2: 0010	√	(d2,10) -010	√	
Group 2	m10: 1010	√	**(12,13) 110-**		
	m12: 1100	√			
Group 3	m13: 1101	√	**(13,15) 11-1**		
		√			
Group 4	m15: 1111	√			

- Determination of Essential Prime Implicants

	PI	m0	m4	m8	m10	m12	m13	m15
	12 13					x	x	
EPI	13 15						x	x
EPI	0 4 8 12	x	x	x		x		
EPI	0 8 d2 10	x		x	x			
		√	√	√	√	√	√	√

$$F(A,B,C,D) = ABD + C'D' + B'D'$$

(3) Simplify the following expression to sum of product using Tabulation Method

$$F(A,B,C,D) = \Pi(1,3,5,7,13,15)$$

Sol:

- Determination of Prime Implicants

Group 0					
Group 1	M1: 0001	√	(1,3) 00-1	√	**(1,3,5,7) 0--1**
			(1,5) 0-01	√	(1,5,3,7) 0--1 redundant
Group 2	M3: 0011	√	(3,7) 0-11	√	**(5,7,13,15) -1-1**
	M5: 0101	√	(5,7) 01-1	√	(5,13,7,15) -1-1 redundant
			(5,13) -101	√	
Group 3	M7: 0111	√	(7,15) -111	√	
	M13: 1101	√	(13,15) 11-1	√	
Group 4	M15: 1111	√			

- Determination of Essential Prime Implicants

	PI	M1	M3	M5	M7	M13	M15
EPI	1 3 5 7	x	x	x	x		
EPI	5 7 13 15			x	x	x	x
		√	√	√	√	√	√

$$F(A,B,C,D) = (A + D')(B' + D')$$

(4) Simplify the following expression to sum of product using Tabulation Method

$$F(A,B,C,D) = M(0,8,10,12,13,15) \cdot d(1,2,3)$$

Sol:

- Determination of Prime Implicants

Group 0	M0: 0000	√	(0,8) -000	√	**(0,8,d2,10) -0-0**
			(0,d1) 000-	√	**(0,d1,d2,d3) 00--**
			(0,d2) 00-0	√	(0,d2,8,10) -0-0 redundant
					(0,d2,d1,d3) 00-- redundant
Group 1	M8: 1000	√	(8,10) 10-0	√	
	d1: 0001	√	**(8,12) 1-00**		
	d2: 0010	√	(d1,d3) 00-1	√	
			(d2,10) -010	√	
			(d2,d3) 001-	√	
Group 2	M10: 1010	√	**(12,13) 110-**		
	M12: 1100	√			
	d3: 0011	√			
Group 3	M13: 1101	√	**(13,15) 11-1**		
Group 4	M15: 1111	√			

- Determination of Essential Prime Implicants

	PI	M0	M8	M10	M12	M13	M15
	8 12		x		x		
	12 13				x	x	
EPI	13 15					x	x
EPI	0 8 d2 10	x	x	x			
	0 d1 d2 d3	x					
		√	√	√	√	√	√

$$F(A,B,C,D) = (A' + B' + D')(B + D)(A' + B' + C)$$

3.11 GTU Questions

Unit	Group	Questions	Summer-15	Winter-15	Summer-16	Winter-16	Summer-17	Winter-17	Summer-18	Winter-18	Answer Topic No_ Pg. No.
3	A	State and explain De Morgan’s theorems with truth tables.	7							7	3.5_8
	A	Apply De Morgan’s theorem to solve the following: (1) [A + (BC)']' [AB' + ABC] = 0 (2) A [B + C' (AB + AC')'] = AB	7								3.5_8
	A	Reduce the expression: (1) A + B (AC + (B+C’) D) (2) (A + (BC)’)’(AB’ + ABC)			4						3.5.1.2_10
	A	Demonstrate by means of truth tables the validity of the De Morgan’s theorems for three variables.				3					3.5_8
	A	Simplify the following Boolean functions to a minimum number of literals. (i) F(x,y,z)=xy+xyz+xyz’+x’yz (ii) F(p, q, r, s) = (p’+q) (p+q+s)s’				4					3.5.1.2_11
	A	State and prove De Morgan’s theorems					7				3.5_8
	A	Reduce the expression F = (B+BC) (B+B'C)(B+D)								3	3.5.1.2_11
	B	Explain minterm and maxterm					3	3			3.6.2.1_12 3.6.2.2_13
	B	Compare SOP and POS.						3			3.6.1.1_12
	B	Give examples of standard and nonstandard SOP and POS forms. Explain how a NON standard POS expression can be converted in to standard POS expression using example you have given.							7		3.6_12
	B	Express the Boolean function F=A+A'C in a sum of min-terms.								7	3.6.3.1_15
	B&C	Express A’B + A’C as sum of minterms and also plot K-map.		7							3.6.3.1._15 3.7.2_18
	C	What are SOP and POS forms of Boolean expressions? Minimize the following expression using K-map Y= Σm(4,5,7,12,14,15) + d(3,8,10)	7								3.6.1.1_12 3.7.3.6_24
	C	Write short note on K-map. OR Explain K-map simplification technique.		7				7			3.7_17

3	C	Simplify the following Boolean function using K-map F (w, x, y, z) = Σ m(1, 3, 7, 11, 15) with don’t care, d(w, x, y, z) = Σm(0,2,5)			3						3.7.3.6_24
	C	Simplify the Boolean function F = A’B’C’+AB’D+A’B’CD’ using don’t-care conditions d=ABC+AB’D’ in (i) sum of products and (ii) product of sums by means of Karnaugh map and implement it with no more than two NOR gates. Assume that both the normal and complement inputs are available.				7					3.7.3.6_24 3.9_28
	C	Reduce using K-map (i)Σm(5,6,7,9,10,11,13,14,15) (ii)ΠM(1,5,6,7,11,12,13,15)					4				3.7.3.4_23 3.7.3.5_24
	C	Minimize using K-map f(A,B,C,D) = Σ(1,3,4,6,8,11,15) +d(0,5,7) also draw MSI circuit for the output.						7			3.7.3.6_24
	C	Simplify equation using K-map : F(a,b,c,d) = Σm(3,7,11,12,13,14,15) Realize the expression with minimum number of gates.							4		3.7.3.4_24 3.8_26
	C	Minimize the following Boolean expression using K- Map and realize it using logic gates. F(A,B,C,D)=Σm(0,1,5,9,13,14,15)+d(3,4,7,10,11)								7	3.7.3.6_24
	C&D	Compare K-map and tabular method of minimization.								3	3.7_17 3.10_29
	D	Simplify following Boolean function using tabulation method: F(w, x, y, z) = Σ (0, 1, 2, 8, 10, 11, 14, 15)			7						3.10_29
	D	Simplify the Boolean function F(x1,x2,x3,x4)=Σm(0,5,7,8,9,10,11,14,15) using tabulation method.				7					3.10_29
	E	Simplify Y=A’BCD’ + BCD’ + BC’D’ + BC’D and implement using NAND gates only.		7							3.9_28
	E	Implement following Boolean function using only NAND gates. Y=ABC’+ABC+A’BC.						4			3.9_28