



Unit -1

Discrete Structures

Syllabus:-

Boolean algebra: Introduction Of Boolean Algebra, Truth Table, Basic Logic Gate, Basic Postulates Of Boolean Algebra, Principle Of Duality, Canonical Form, Karnaugh Map.

Introduction Of Boolean Algebra

- Boolean Algebra is used to analyze and simplify the digital (logic) circuits.
- It uses only the binary numbers i.e. 0 and 1.
- It is also called as Binary Algebra or logical Algebra.
- Boolean algebra was invented by George Boole in 1854.
- In 1938, Claude E. Shannon proposed using Boolean algebra in design of relay switching circuits

Fundamental Concepts of Boolean Algebra

- **Use of Binary Digit:-**

Boolean equations can have either of two possible values, 0 and 1

- **Logical Addition**

Symbol '+', also known as 'OR' operator, used for logical addition. Follows law of binary addition.

- **Logical Multiplication**

Symbol '.', also known as 'AND' operator, used for logical multiplication. Follows law of binary multiplication.

- **Complementation**

Symbol '-', also known as '**NOT**' operator, used for complementation. Follows law of binary compliment

Operator Precedence

- Each operator has a precedence level
- Higher the operator's precedence level, earlier it is evaluated
- Expression is scanned from left to right
- First, expressions enclosed within parentheses are evaluated
- Then, all complement (NOT) operations are performed
- Then, all '.' (AND) operations are performed
- Finally, all '+' (OR) operations are performed

Basic Postulates and Theorem Of Boolean Algebra

Postulate 1	(a) $x + 0 = x$	(b) $x \cdot 1 = x$
Postulate 2	(a) $x + x' = 1$	(b) $x \cdot x' = 0$
Postulate 3, commutative	(a) $x + y = y + x$	(b) $x \cdot y = y \cdot x$
Postulate 4, distributive	(a) $x \cdot (y + z) = x \cdot y + x \cdot z$	(b) $x + (y \cdot z) = (x + y) \cdot (x + z)$
Theorem 1	(a) $x + x = x$	(b) $x \cdot x = x$
Theorem 2	(a) $x + 1 = 1$	(b) $x \cdot 0 = 0$
Theorem 3, involution	$(x')' = x$	
Theorem 4, associative	(a) $x + (y + z) = (x + y) + z$	(b) $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Theorem 5, deMorgan	(a) $(x + y)' = x' \cdot y'$	(b) $(x \cdot y)' = x' + y'$
Theorem 6, absorption	(a) $x + (x \cdot y) = x$	(b) $x \cdot (x + y) = x$

Principle of duality

- According to the duality principle, if we have postulates or if we have theorems of Boolean Algebra for any one type of operation then the operation can be converted into another type of operation.
- In other words AND can be converted to OR and OR can be converted into AND
- We can interchange '0 with 1', '1 with 0', '(+) sign with (.) sign' and '(.) sign with (+) sign' to perform dual operation. T
- This principle ensures if a theorem is proved using postulates of Boolean algebra, then the dual of this theorem automatically holds and there is no requirement of proving it separately.
- The dual of a Boolean expression can easily be obtained by interchanging sums and products and interchanging 0 as well as 1. Let's know how to find the dual of any expression.
- For example, the dual of $x\bar{y} + 1$ is equal to $(x + y) \cdot 0$

- Two formulas A_1 and A_2 are said to be duals of each other if either one can be obtained from the other by replacing \wedge (AND) by \vee (OR) by \wedge (AND). Also if the formula contains T (True) or F (False), then we replace T by F and F by T to obtain the dual.
- Duality principle states that for any true statement, the dual statement obtained by interchanging unions into intersections (and vice versa) and interchanging Universal set into Null set (and vice versa) is also true. If dual of any statement is the statement itself, it is said **self-dual** statement.

Example—The dual of $(A \cap B) \cup C$ is $(A \cup B) \cap C$

Operator/Variable and Their Duality

Operator / Variable	Dual of the Operator
AND	OR
OR	AND
1	0
0	1
A	\bar{A}
\bar{A}	A

Some Boolean expressions and their corresponding duals are given in the table below:

Given Expression	Dual	Given Expression	Dual
$0 = 1$	$1 = 0$	$A \cdot (A+B) = A$	$A + A \cdot B = A$
$0 \cdot 1 = 0$	$1 + 0 = 1$	$AB = A + B$	$A+B = A \cdot B$
$A \cdot 0 = 0$	$A + 1 = 1$	$(A+C) (A+B) = AB + AC$	$AC + AB = (A+B) \cdot (A+C)$
$A \cdot B = B \cdot A$	$A + B = B + A$	$A+B = AB + AB + AB$	$AB = (A+B) \cdot (A+B) \cdot (A+B)$
$A \cdot A = 0$	$A + A = 1$	$AB + A + AB = 0$	$((A+B)) \cdot A \cdot (A+B) = 1$
$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A+(B+C) = (A+B) + C$		

Boolean Functions

- A Boolean function is an expression formed with:
 - Binary variables
 - Operators (OR, AND, and NOT)
 - Parentheses, and equal sign
- The value of a Boolean function can be either 0 or 1
- A Boolean function may be represented as:
 - An algebraic expression, or
 - A truth table

Representation as an Algebraic Expression

$$W = X + Y \cdot Z$$

- Variable W is a function of X , Y , and Z , can also be written as $W = f(X, Y, Z)$
- The RHS of the equation is called an **expression**
- The symbols X , Y , Z are the **literals** of the function
- For a given Boolean function, there may be more than one algebraic expressions








Representation as a Truth Table

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

Logic Gates

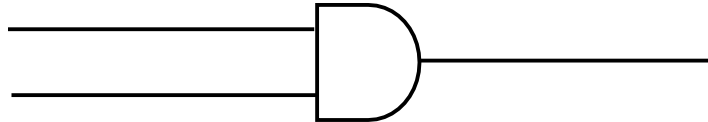
- Logic gates are electronic circuits that operate on one or more input signals to produce standard output signal
- Are the building blocks of all the circuits in a computer
- Some of the most basic and useful logic gates are AND, OR, NOT, NAND and NOR gates

Logic Gates

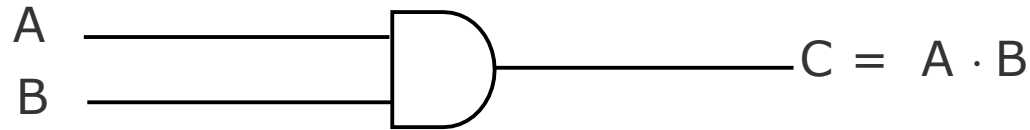
Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	\overline{A}	AB	\overline{AB}	$A+B$	$\overline{A+B}$	$A\oplus B$	$\overline{A\oplus B}$																																																																																																
Symbol																																																																																																							
Truth Table	<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1
A	X																																																																																																						
0	1																																																																																																						
1	0																																																																																																						
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					

AND Gate

- Physical realization of logical multiplication (AND) operation.
- Generates an output signal of 1 only if all input signals are also 1.



AND Gate (Block Diagram Symbol and Truth Table)

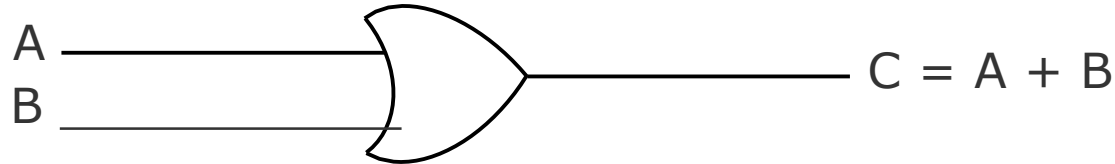


Inputs		Output
A	B	$C = A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate

- Physical realization of logical addition (OR) operation
- Generates an output signal of 1 if at least one of the input signals is also 1

OR Gate (Block Diagram Symbol and Truth Table)

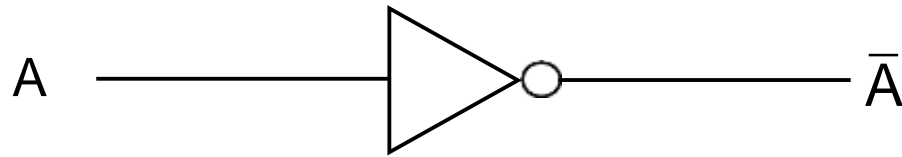


Inputs		Output
A	B	$C = A + B$
0	0	0
0	1	1
1	0	1
1	1	1

NOT Gate

- Physical realization of complementation operation
- Generates an output signal, which is the reverse of the input signal

NOT Gate (Block Diagram Symbol and Truth Table)

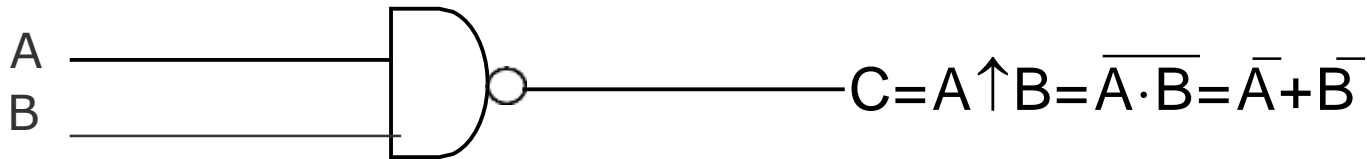


Input	Output
A	\bar{A}
0	1
1	0

NAND Gate

- Complemented AND gate
- Generates an output signal of:
 - 1 if any one of the inputs is a 0
 - 0 when all the inputs are 1

NAND Gate (Block Diagram Symbol and Truth Table)



Inputs		Output
A	B	$C = \overline{A} + \overline{B}$
0	0	1
0	1	1
1	0	1
1	1	0

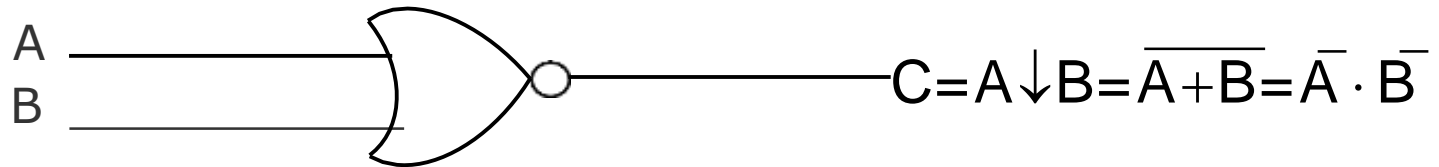
Logic Circuits

- When logic gates are interconnected to form a gating / logic network, it is known as a *combinational logic circuit*
- The Boolean algebra expression for a given logic circuit can be derived by systematically progressing from input to output on the gates
- The three logic gates (AND, OR, and NOT) are logically complete because any Boolean expression can be realized as a logic circuit using only these three gates

NOR Gate

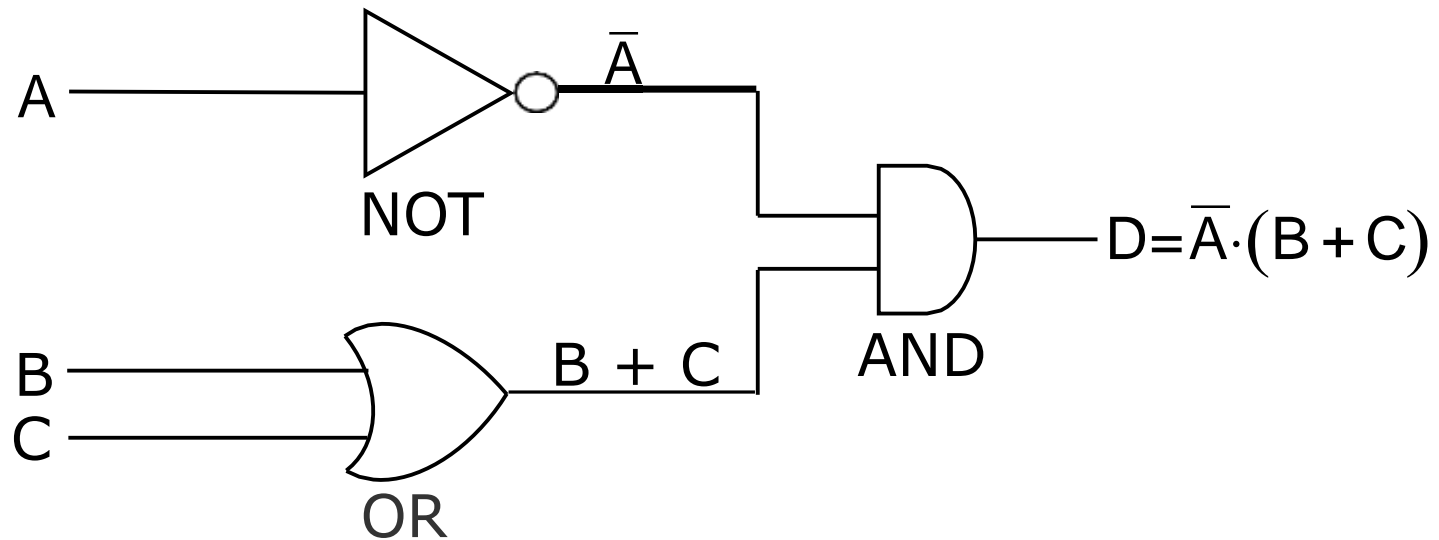
- Complemented OR gate
- Generates an output signal of:
 - 1 only when all inputs are 0
 - 0 if any one of inputs is a 1

NOR Gate (Block Diagram Symbol and Truth Table)



Inputs		Output
A	B	$C = \overline{A} \cdot \overline{B}$
0	0	1
0	1	0
1	0	0
1	1	0

Finding Boolean Expression of a Logic Circuit

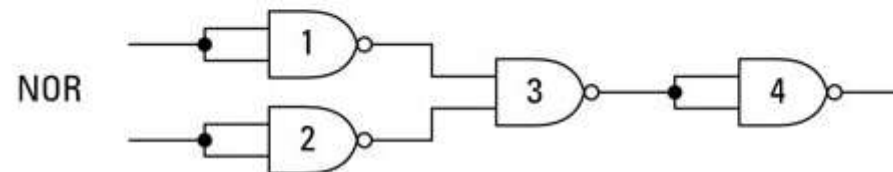
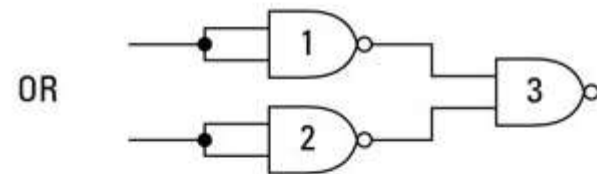
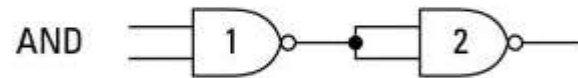
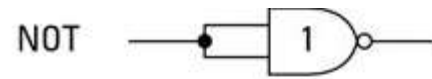


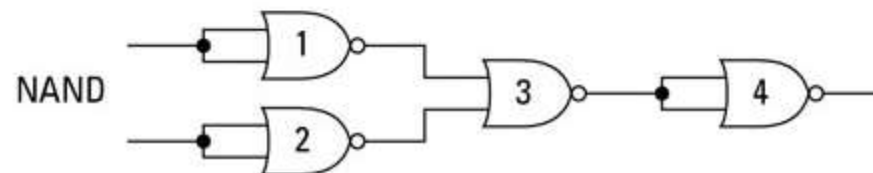
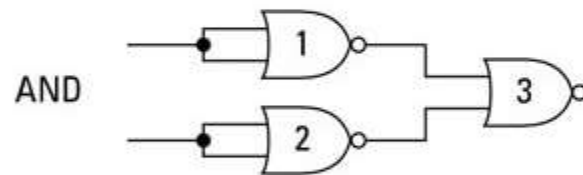
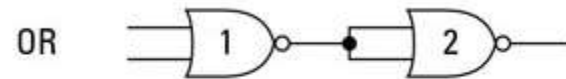
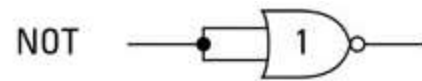
Universal NAND Gate

- NAND gate is an universal gate, it is alone sufficient to implement any Boolean expression
- **To understand this, consider:**
 - Basic logic gates (AND, OR, and NOT) are logically complete
 - Sufficient to show that AND, OR, and NOT gates can be implemented with NAND gates

Universal NOR Gate

- NOR gate is an universal gate, it is alone sufficient to implement any Boolean expression
- To understand this, consider:
 - Basic logic gates (AND, OR, and NOT) are logically complete
 - Sufficient to show that AND, OR, and NOT gates can be implemented with NOR gates





Karnaugh Map

- The **Karnaugh map (KM or K-map)** is a method of simplifying Boolean algebra expressions. Maurice Karnaugh introduced it in 1953.
- We can minimize Boolean expressions of 3, 4 variables very easily using K-map without using any Boolean algebra theorems. K-map can take two forms Sum of Product (SOP) and Product of Sum (POS) according to the need of problem. K-map is table like representation but it gives more information than TRUTH TABLE. We fill grid of K-map with 0's and 1's then solve it by making groups.
- Always remember **$POS \neq (SOP)'$**
The correct form is **$(POS \text{ of } F) = (SOP \text{ of } F')'$**

Steps to solve expression using K-map-

- Select K-map according to the number of variables.
- Identify minterms or maxterms as given in problem.
- For SOP put 1's in blocks of K-map respective to the minterms (0's elsewhere).
- For POS put 0's in blocks of K-map respective to the maxterms(1's elsewhere).
- Make rectangular groups containing total terms in power of two like 2,4,8 ..(except 1) and try to cover as many elements as you can in one group.
- From the groups made in step 5 find the product terms and sum them up for SOP form.

SOP FORM

K-map of 3 variables-

■ $Z = \sum A, B, C(1, 3, 6, 7)$

■ From **red** group we get product term—

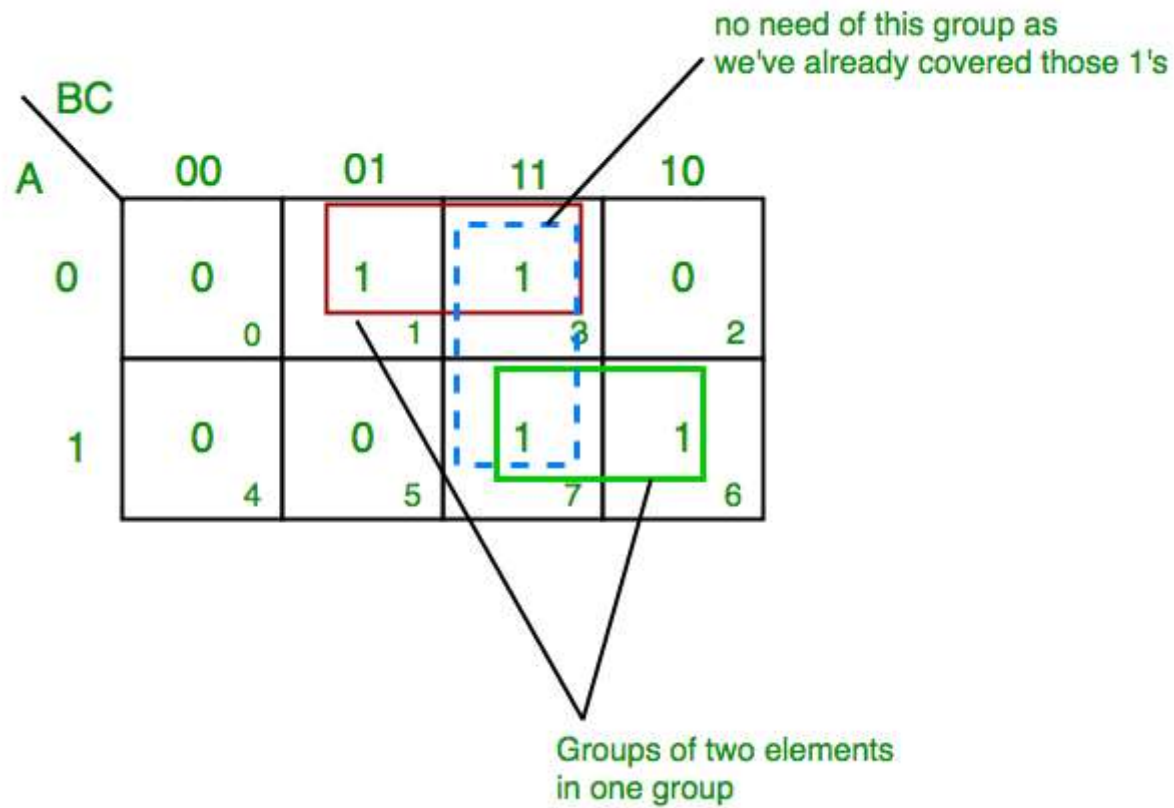
$$A'C$$

■ From **green** group we get product term—

$$AB$$

■ Summing these product terms we get- **Final expression**

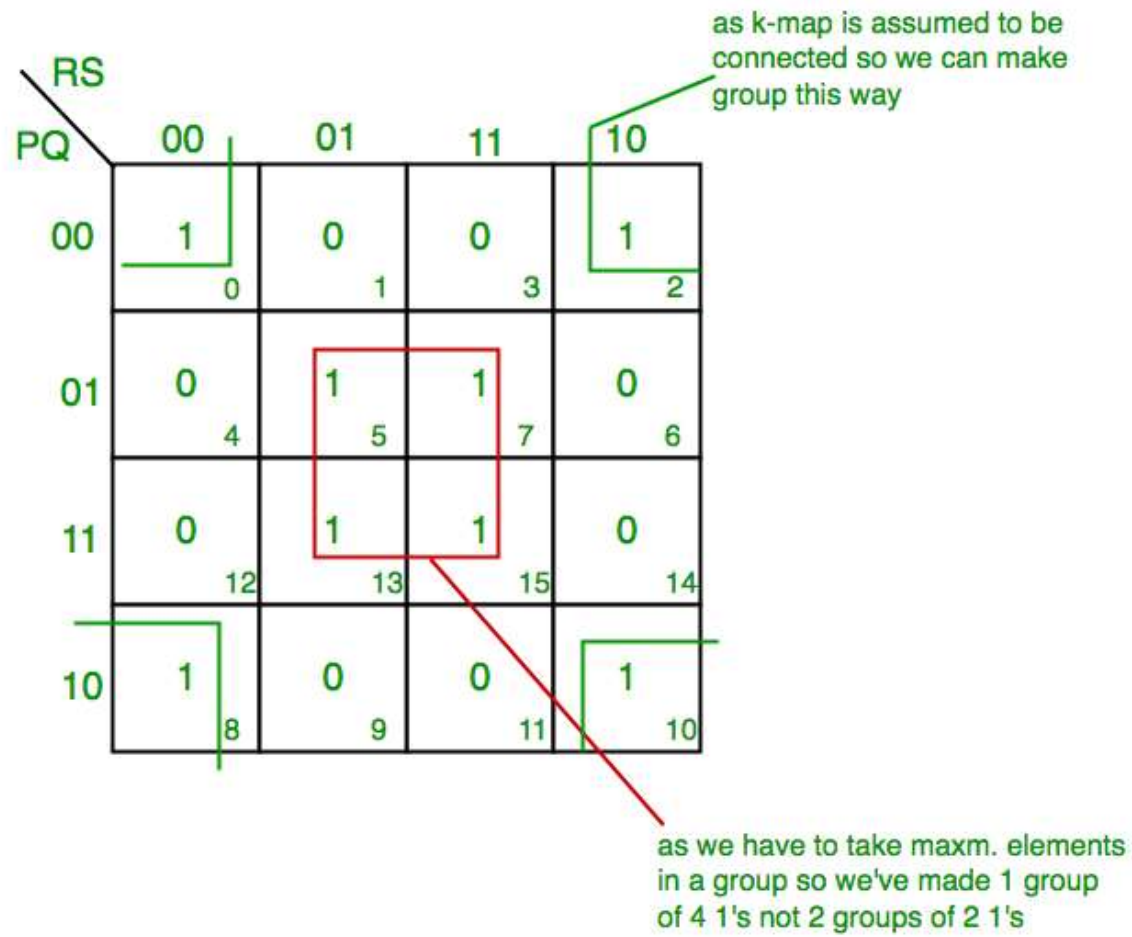
$$(A'C + AB)$$



SOP FORM

K-map for 4 variables

- $F(P,Q,R,S) = \sum(0,2,5,7,8,10,13,15)$
- From **red** group we get product term—
 QS
- From **green** group we get product term—
 $Q'S'$
- Summing these product terms we get- **Final expression**
 $(QS + Q'S')$



POS FORM

K-map of 3 variables-

■ $F(A,B,C)=\pi(0,3,6,7)$

■ From **red** group we find terms

$$A \quad B \quad C'$$

■ Taking complement of these two

$$A' \quad B' \quad C$$

■ Now **sum** up them

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

■ From **green** group we find terms

$$B \quad C$$

■ Taking complement of these two terms

$$B' \quad C'$$

■ Now sum up them

$$(B' + C')$$

■ From **brown** group we find terms

$$A' B' C'$$

■ Taking complement of these two

$$A B C$$

■ Now **sum** up them

$$(A + B + C)$$

■ **Final expression** $(A' + B' + C) (B' + C') (A + B + C)$

2 elements in one group

		BC			
A	0	00	01	11	10
	1				
	0	0 0	1 1	0 3	1 2
	1	1 4	1 5	0 7	0 6

K-map of 4 variables-

■ $F(A,B,C,D)=\pi(3,5,7,8,10,11,12,13)$

■ From **green** group we find terms

$$C' \quad D \quad B$$

■ Taking their complement and summing them

$$(C+D'+B')$$

■ From **red** group we find terms

$$C \quad D \quad A'$$

■ Taking their complement and summing them

$$(C'+D'+A)$$

■ From **blue** group we find terms

$$A \quad C' \quad D'$$

■ Taking their complement and summing them

$$(A'+C+D)$$

■ From **brown** group we find terms

$$A \quad B' \quad C$$

■ Taking their complement and summing them

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

Finally we express these as product

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

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

Canonical Forms:

■ The problem of finding whether a given statement is tautology or contradiction or satisfiable in a finite number of steps is called the Decision Problem. For Decision Problem, construction of truth table may not be practical always. We consider an alternate procedure known as the reduction to normal forms.

Normal Forms

There are two such forms:

Disjunctive Normal Form (DNF)

Conjunctive Normal Form (CNF)

Disjunctive Normal Form

A compound statement is in disjunctive normal form if it is obtained by operating OR among variables (negation of variables included) connected with ANDs. In terms of set operations, it is a compound statement obtained by Union among variables connected with Intersections.

Examples

$$(A \wedge B) \vee (A \wedge C) \vee (B \wedge C \wedge D)$$

$$(P \cap Q) \cup (Q \cap R)$$

Conjunctive Normal Form

A compound statement is in conjunctive normal form if it is obtained by operating AND among variables (negation of variables included) connected with ORs. In terms of set operations, it is a compound statement obtained by Intersection among variables connected with Unions.

Examples

$$(A \vee B) \wedge (A \vee C) \wedge (B \vee C \vee D)$$

$$(P \cup Q) \cap (Q \cup R)$$

Reference Books

- **Discrete Mathematics and Its Applications**
by **Kenneth H. Rosen**
- Elements of Discrete Mathematics, C. L. Liu
McGraw Hill, New Delhi