Digital Electronics

Syllabus

Number systems; Binary Addition & Subtraction;1’s and 2’s complement, Subtraction using 2’s complement; Boolean algebra; Logic gates; Implementation of basic gates using universal gates; Realization of Boolean functions using basic & universal gates; Canonical forms(SOP & POS); Simplification of Boolean functions using Boolean postulates & K-map up to 4 variables with don’t care condition.

NUMBER SYSTEM

- It consist of set of Symbol used to perform arithmetic operations.
- No. System is defined by its base or Radix
- Radix or Base is no. of Symbols used, most commonly used no system are

Decimal → Base-10 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Binary → Base-2 {0, 1}

Octal → Base-8 {0, 1, 2, 3, 4, 6, 7}

Hexadecimal → Base-16 {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

Similarly any Base can be created, example

Base 5 → {0, 1, 2, 3, 4}

Base 7 → {0, 1, 2, 3, 4, 5, 6}

Base Conversion

Decimal to base-r (any Base)

Decimal Consist of 2 parts (integer + fraction)

for Integer	for fraction
→ divide continously by base-r	→ Multiply continously by base-r
→ Note reminder in reverse direction	→ Note carry in Same direction

Ex $(40.25)_{10} \longrightarrow (\ )_2$

2	10	0
2	5	1
2	2	0
	1	

↑

$.25 \times 2 = 0.5$ 0 ↓

$.5 \times 2 = 1.0$ 1 ↓

$(1010.01)_2$ Ans.

(Sweta)

Ex $(108.025)_{10} \longrightarrow (\ )_{12}$

$12 \mid 108 \quad 0 \uparrow$
$\quad\ 9 \quad\ \leftarrow$

$.025 \times 12 = 0.3 \quad 0$
$0.3 \times 12 = 3.6 \quad 3$
$.6 \times 12 = 7.2 \quad 7$
$.2 \times 12 = 2.4 \quad 2$ ↓

$(90.0372)_{12}$

⇒ To Convert no. from Base-r to Decimal.

Decimal is Summation of product of Symbol and base raise to its power.

Ex $(1011.01)_2 \longrightarrow (\ )_{10}$

$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$

$(11.25)_{10}$

Interconversion between 2 bases where non of them is Decimal.

⇒ for Such Condition Decimal is the intermediate base.

Ex $(\ )_m \longrightarrow (\ )_n$
$\quad\ \hookrightarrow (\ )_{10} \uparrow$

Ex $(125)_9 \longrightarrow (\ )_{12}$
$\quad\ \downarrow \hookrightarrow (104)_{10} \uparrow$

$1 \times 9^2 + 2 \times 9^1 + 5 \times 9^0$
$= 104$

$12 \mid 104 \quad 8$
$\quad\ 8$

$(125)_9 \longrightarrow (88)_{12}$.

An alternate Method is there for interconversion between base which comes under 2^{Power}

Ex Base-4, Base-8, Base-16

we can directly convert these base into binary and vice-versa.

Base-4 to Binary (direct Conversion)

$(231.3)_4 \rightarrow (\ )_2$

10 11 01 11

$(101101.11)_2$

$(101.01)_4 \rightarrow (\ )_2$

01 00 01 00 01

$(010001.0001)_2$

Split individual digit int 2 bit binary

To remember

base-4 $4 = 2^2$

∴ 2 bit binary.

Since binary use either 0 or 1

∴ you can easily Convert Symbol into binary by writing 1 at place whose positional weight is to be used.

	2^1	2^0
Positional weight	2	1
Now to write 3	1	1
2	1	0
1	0	1
0	0	0

(Sweta)

Similarly for Base-8 Split individual digit into 3 bit binary. ($8 = 2^3$)

→ by writing 1 inplace of positional weight to be used.

for example

	2^2	2^1	2^0
	4	2	1
7	1	1	1
6	1	1	0
5	1	0	1
2	0	1	0

$(743.125)_8 \longrightarrow (\quad)_2$

111 100 011 001 010 101

$(111100011.001010101)_2$.

Base-16 to Binary

Split individual digit into 4 bit binary ($16 = 2^4$) ✓

$(36B.2D)_{16} \longrightarrow (\quad)_2$

0011 0110 1011 0010 1101

2^3	2^2	2^1	2^0
8	4	2	1

$(001101101011.001011101)_2$ To write

A (10)	1	0	1	0
B (11)	1	0	1	1
3	0	0	1	1

Binary to Base-4 to convert no. from binary to base-4 make group of 2 bit starting from decimal point / Least Significant bit and write its equivalent Value.

$(0\,1\,1\,0\,1\,1\,0\,1\,0\,0 \cdot 0\,0\,1\,0\,1\,0)_2$

$\downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow\ \downarrow$

$1\ 2\ 3\ 1\ 0\ 0\ 2\ 2$ $\longrightarrow (12310 \cdot 022)_4$

Note ⇒ add zero at LSB for fraction and at MSB for Integer if group is not possible.

Binary to Base-8 (octal)

Make group of 3 bit and write its equivalent Value.

$(1\,1\,0\,0\,1\,0 \cdot 1\,0\,1\,1\,0\,0)_2 \longrightarrow (\ )_8$

$1\,1\,0\,0\,1\,0 \cdot 1\,0\,1\,1\,0\,0$

$6\quad 2\quad 5\quad 4 \longrightarrow (62 \cdot 54)_8$

Binary to Base-16 (Hexadecimal)

Make group of 4 bit and write its equivalent Value

$(1\,1\,1\,1\,0\,1\,1\,1\,0\,1 \cdot 1\,0\,1\,1\,1)_2 \longrightarrow (\ )_{16}$

$0\,0\,1\,1\,1\,1\,0\,1\,1\,1\,0\,1 \cdot 1\,0\,1\,1\,1\,0\,0\,0$

$3\quad D\quad D\quad B\quad 8$

$(3DD \cdot B8)_{16}$

(Sweta)

Arithmetic operation in any base :

Addition : 1 carry = base value, ie carry is generated when added value ≥ base value

Ex $(1011)_2 + (1101)_2 = (\ )_2$

```
  1 1 1 1
  0 1 0 1 1
  0 1 1 0 1
 ---------
 1 1 0 0 0
```

1 + 1 + 1 = 3
3 ≥ 2
∴ 3 − 2 = 1 with carry 1

Same again
1 + 1 = 2
where 2 ≥ 2
∴ carry is generated

1 + 1 = 2
Base-2 ≤ 2 → with carry 1
2 − 2 = 0

Since addition is in Base-2 and added value is ≥ Base
∴ carry will pass to next stage and this value will reduce by Base

∴ $(11000)_2$ Answer.

* Same logic is applied in all Base.

Ex $(746)_8 + (243)_8 = (\ )_8$

```
  1  1
  0  7  4  6
  0  2  4  3
 ------------
```

1 (10) ≥ 8 → −8 (1 carry)
9 ≥ 8 → −8 (1 carry)
9 ≥ 8 → −8
2 | 1 | 1 (with carry 1)

$(1211)_8$ final Answer

Subtraction : Borrow is required whenever we subtract large value from smaller.

1 Borrow = Base value.

Ex $(746)_8 - (274)_8 = (\ )_8$

1 Borro = Base ie $+8 = 12$

$$\begin{array}{r} 6\ \not{7}\ {}^{4}6 \\ -\ 2\ \ 7\ \ 4 \\ \hline (4\ \ 5\ \ 2)_8 \end{array}$$

Ex $(10001)_2 - (1110)_2 = (\ )_2$

1B=2 1B=2 1B=2

$$\begin{array}{r} 1\ 0\ 0\ 0\ 1 \\ -\ 0\ 1\ 1\ 1\ 0 \\ \hline (0\ 0\ 0\ 1\ 1)_2 \end{array}$$

Ex $(2AC)_{16} - (DE \cdot 3)_{16} = (\ )_{16}$

1B=16 +9 = 25 ; 1B=16 +11 = 27 ; 1B=16

$$\begin{array}{r} \not{2}\ \not{A} \quad {}^{1}\not{2}\ {}^{9}\not{A}\ \not{C} \cdot 0 \\ -\ \ D\ \ E \cdot 3 \\ \hline (1\ B\ D \cdot D)_{16} \end{array}$$

$27 - 14(E) = 13(D)$

$16 - 3 = 13 \Downarrow D.$

$25 - 13 = 12(B)$

(Sweta)

Multiplication : Same rule are applied in multiplication.

Ex $(111)_2 \times (101)_2 = (\quad)_2$

$$\begin{array}{r} 1\ 1\ 1 \\ \times\ 1\ 0\ 1 \\ \hline \end{array}$$

Carry: (1) 1 1 1

1 (1) 0 0 0 ×

0 1 1 1 × ×

$2 - 2 = 0$; $2 - 2(1\text{ carry}) = 0$; $2(1\text{ carry}) - 2 = 0$

$(1\text{ carry}) - 2 = 1\ 0$

$(100011)_2$ Answer.

✗ You can verify your answer's for all Arithmetic operation by simply converting them into Decimal.

Ex $(111)_2 = (7)_{10}$

$(101)_2 = \times(5)_{10}$

$(35)_{10}$

$(100011)_2$.

Answer.

LOGIC GATES

Q:-

Logic gates are the basic elements which form the building blocks for complex digital system.

Logic Gates

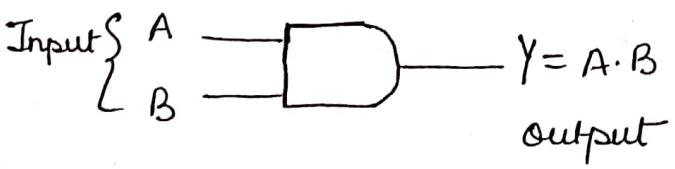
- Basic Gates
  - AND Gate
  - OR Gate
  - NOT Gate
- Universal Gates
  - NAND Gate
  - NOR Gate
- Derived Gates
  - Ex-OR Gate
  - Ex-NOR Gate

Boolean Expression: The relation between the inputs and outputs of a gate can be expressed mathematically by means of Boolean expression

Truth Table: it consist of all the possible combinations of the inputs and the corresponding state of output of a logic gate.

Fundamental Gates OR Basic Gates

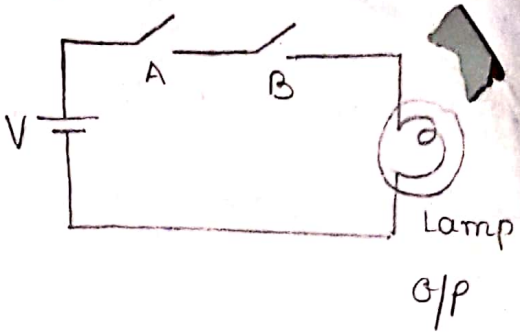
1) AND Gate (·) :- It peryorms logical multiplication
→ Two or more than two input, single output
→ Output is high only if all the inputs are high



Logical Symbol

Input		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

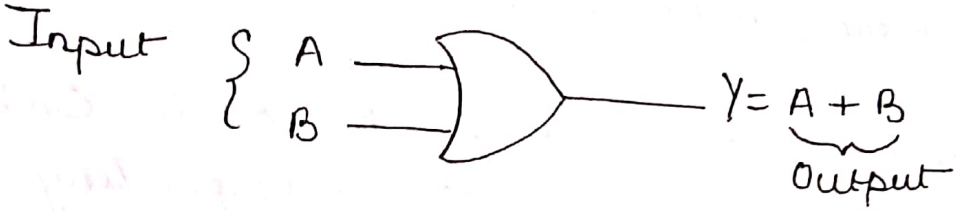
Truth table



OR Gate (+)

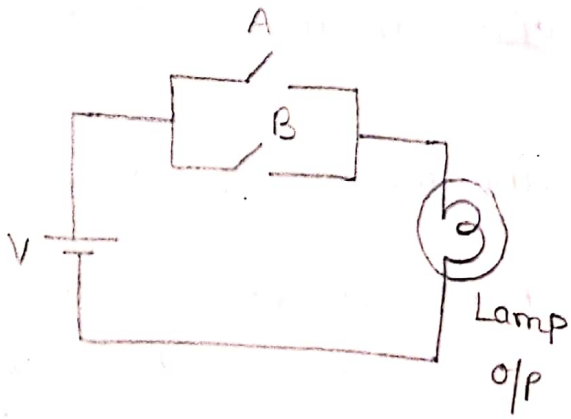
It performs logical addition

- Output is high if any one or ~~both~~ all inputs are high.
- Multiple input, single output gate



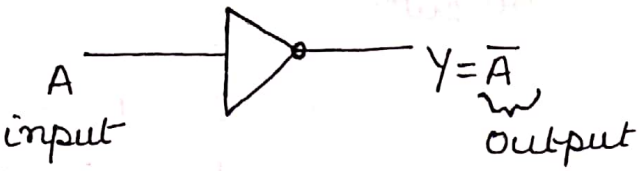
Input		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table



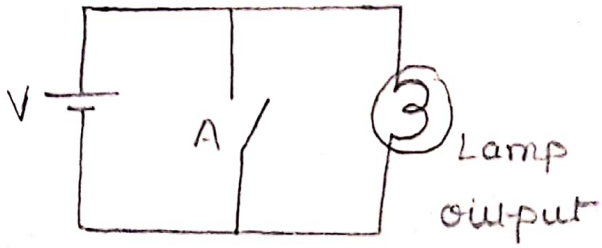
NOT Gate ($^-$) : It inverts or complements the input

- Single input, single output



Input	Output
A	Y
0	1
1	0

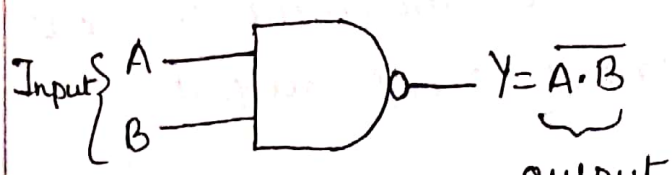
Truth Table



UNIVERSAL Gates : NAND and NOR gates are called 'universal gates' because any Boolean expression can be implemented by using NAND or NOR gates only.

NAND Gate : It implies an AND operation with an inverted output.

- Output is low if and only if all the inputs are high.
- Multiple input, single output
- AND, OR and NOT gates can be constructed using NAND gates only.

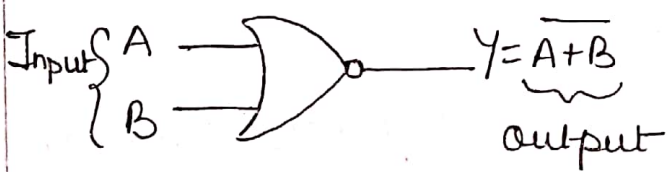


Input		Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table.

NOR Gate : It implies OR ~~function~~ operation with an inverted output

- Output of a NOR Gate is high if all its inputs are low
- Multiple inputs, Single output
- AND, OR, NOT gates can be constructed using NOR gates only.



Input		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

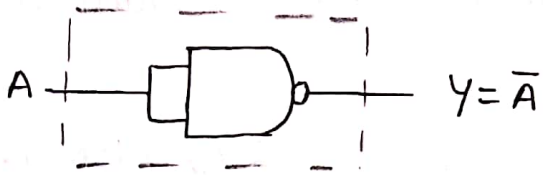
Implementation of Basic Gates using NAND Gates

1 NOT gate using NAND Gate

O/p of a NAND Gate (2 i/p), $Y = \overline{A \cdot B}$

if A = B, then $Y = \overline{A \cdot A} = \overline{A}$

∴



NOT gate using NAND

(20)

2. AND gate using NAND Gates

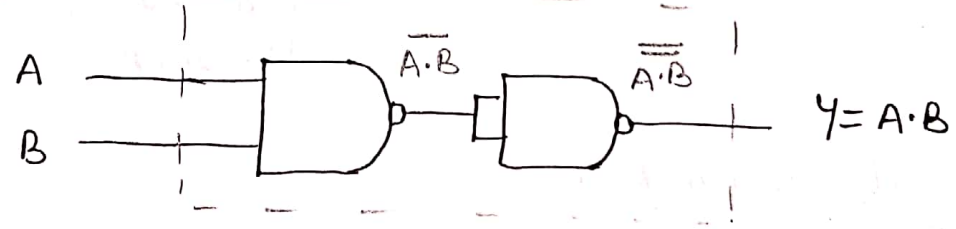
O/p of a 2 i/p NAND gate is $Y = \overline{A \cdot B}$

and that of a AND gate is $Y = A \cdot B$

if o/p of a NAND gate is inverted then

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

∴



AND Gate using NAND.

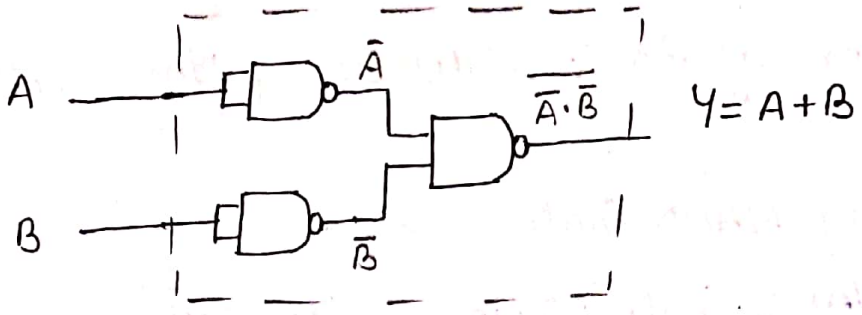
3. OR Gate using NAND Gate

O/p of a 2 i/p NAND gate is $Y = \overline{A \cdot B} = \bar{A} + \bar{B}$

and that of a OR gate $Y = A + B$

if $\bar{A}$ & $\bar{B}$ are given as i/p then

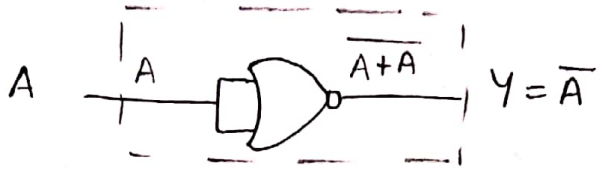
$$Y = \overline{\bar{A} \cdot \bar{B}} = \bar{\bar{A}} + \bar{\bar{B}} = A + B$$



OR gate using NAND.

Implement Basic Gates using NOR Gate only

a NOT gate using NOR



NOT Gate using NOR

2 i/p NOR gate

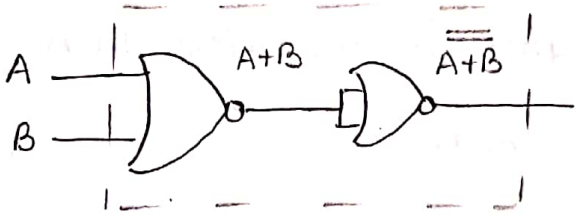
o/p $Y = \overline{A+B}$

if $A = B$

then $Y = \overline{A+A}$

$Y = \bar{A}$

b) OR gate using NOR.



OR gate using NOR

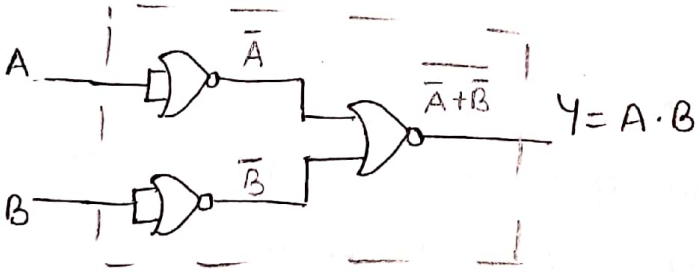
2 i/p NOR gate

o/p $Y = \overline{A+B}$

and that of OR gate

$Y = A+B$

c) AND gate using NOR



if Inverted i/p is given to a NOR gate

then $Y = \bar{A}+\bar{B}$

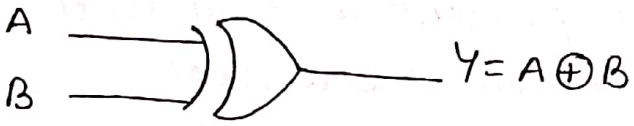
$= \overline{\bar{A}}\cdot\overline{\bar{B}}$

$Y = A\cdot B$

Derived Gates OR (Bubbled Gates)

1 Ex-OR Gate : O/p is high only if both inputs are different

- for more than two inputs, O/p is high for odd number of high inputs.



$Y = A \oplus B$

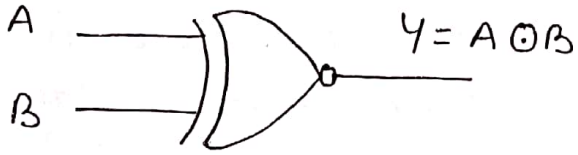
$Y = \bar{A}B + A\bar{B}$

Input		Output
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table

Ex-NOR Gate : output is high only for same i/p

- for more than two inputs, o/p is high for Even number of high inputs.



$Y = A \odot B$

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

Input		Output
A	B	$Y = A \odot B$
0	0	1
0	1	0
1	D	0
1	1	1

BOOLEAN ALGEBRA

- Boolean algebra is developed by George Boole.

In Boolean algebra, the binary digits are utilized to represent the two levels that occur with in digital logic circuit.

A binary '1' represent a HIGH level and a binary '0' represent a LOW level.

In Boolean Algebra, logic variables and logic statements are combined with the help of logical operators.

there are 3 logical operators to form a logical function.

1 > AND operator (·) logical Multiplication

2 > OR operator (+) logical Addition

3 > NOT operator ($^-$) inversion.

Laws of Boolean Algebra: Boolean Algebra is Mathematics of logic.

There are certain laws and theorems of Boolean algebra which can be used for simplification of complex boolean expressions and are a very useful tool when it comes to building a logic circuit for given logic function.

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

Proof: Taking RHS

$\therefore (A+B)(A+C)$

$\Rightarrow A \cdot A + AC + AB + BC$

$\Rightarrow A(1+C) + AB + BC$

$\Rightarrow A(1+B) + BC$

$\Rightarrow A + BC$ proved

LHS = RHS

7. ABSORPTIVE LAWS:

i) $A + AB = A$

ii) $A \cdot (A+B) = A$

iii) $A + \bar{A}B = A + B$

iv) $A \cdot (\bar{A} + B) = AB$

8. INVOLUTION LAW

$\overline{\overline{A}} = A$

9. IDEMPOTENT LAW

i) $A + A + A + A + \text{- - - - } + A = A$

ii) $A \cdot A \cdot A \cdot A \cdot \text{- - - - - - - } \cdot A = A$

10. DUALS: The Duals of a given boolean expression can be found in following way:-

i) Replace (+) OR Sign by (·) AND Sign

ii) Replace (·) AND Sign by (+) OR Sign

iii) Complement 0's and 1's

1. AND LAWS:

i) $A \cdot 0 = 0$

ii) $A \cdot 1 = A$

iii) $A \cdot A = A$

iv) $A \cdot \bar{A} = 0$

2. OR LAWS:

i) $A + 0 = A$

ii) $A + A = A$

iii) $A + 1 = 1$

iv) $A + \bar{A} = 1$

3. NOT LAWS:

i) $\bar{0} = 1$

ii) $\bar{1} = 0$

iii) $\bar{\bar{A}} = A$

4. COMMUTATIVE LAWS:

i) $A + B = B + A$

ii) $A \cdot B = B \cdot A$

5. ASSOCIATIVE LAWS:

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

ii) $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

6. DISTRIBUTIVE LAWS:

i) $A \cdot (B + C) = A \cdot B + A \cdot C$

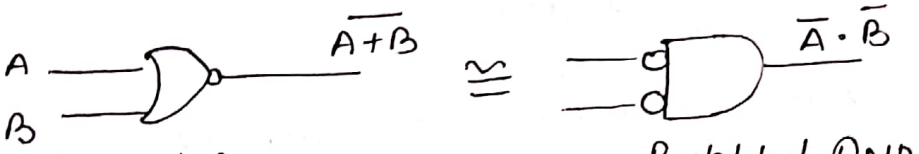
ii) $A + (B \cdot C) = (A + B)(A + C)$

Expression	Dual Relation
$A+0 = A$	$A \cdot 1 = A$
$A+\bar{A} = 1$	$A \cdot \bar{A} = 0$
$A+B = B+A$	$A \cdot B = B \cdot A$
$A(B+C) = AB+AC$	$A+(B \cdot C) = A \cdot (B+C)$

DE-MORGAN'S THEOREM

a) "The complement of a Sum equals the product of the complements"

$$\overline{A+B+C+---n} = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot ---- \bar{n}$$

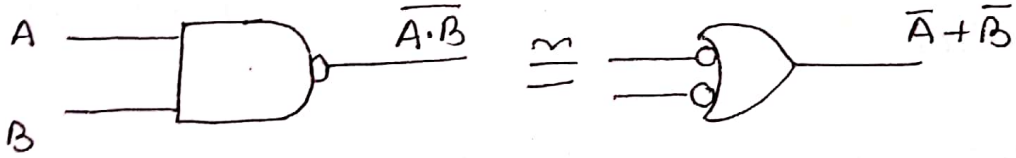


NOR — Bubbled AND

b) 2nd Theorem

"The Complement of a product equals the Sum of Complements"

$$\overline{A \cdot B \cdot C \cdot ---n} = \bar{A} + \bar{B} + \bar{C} + --+ \bar{n}$$



NAND — Bubbled OR

LOGICAL FUNCTIONS

Logical functions are expressed in terms of logical variables. The values taken by logical variables and logical functions are in binary form.

Any logic expression can be expressed in the following two standard forms.

1. Sum of products (SOP) form
2. Product of sums (POS) form

1 **Sum of Products (SOP) form**: Consist of product terms logically added.

Example: $AB + \bar{A}B + A\bar{B}$ or $ABC + A\bar{C} + A\bar{B}C$

- A product term is a logical product of many variables.

2. **Product of Sums (POS) form**: Consist of Sum terms logically multiplied.

Example: $(A+B)(\bar{A}+B)(A+\bar{C})$ OR $(A+\bar{B})(A+B+C)(A+C)$

- A sum term is logical addition of several variables.

CANONICAL form of a Logic Expression: When each term of a logic expression contains all variables, it is called in a canonical form.

a) **Minterm Canonical form** :- When a sum of products form of logic expression is in canonical form, it is called Minterm canonical form

- Each product term is called a Minterm
- Each minterm contains all variables whether in original or in complimented form.

Example: $Y = A\bar{B}\bar{C} + \underbrace{\bar{A}BC}_{\text{Minterm}} + \bar{A}B\bar{C}$

Minterm Designation :

Variable without Complement → 1

Variable with Complement → 0

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

$Y = m_0 + m_1 + m_2 + m_3$

$m_0 \Rightarrow \bar{A}\bar{B}C \rightarrow 001 \Rightarrow 1$

$m_1 \Rightarrow \bar{A}BC \rightarrow 011 = 3$

$m_2 \rightarrow A\bar{B}C \rightarrow 101 = 5$

$m_3 \rightarrow ABC \rightarrow 111 = 7$

$$Y = \overset{\text{Sum}}{\Sigma} \underset{\text{minterm}}{m} (1, 3, 5, 7)$$

b) Maxterm Canonical form: When a Product of Sums form of logic expression is in canonical form it is called Maxterm Canonical form.

- Each Sum term is called a Maxterm.
- Each Maxterm contains all variables. Whether in Original or in complimented form

Example: $Y = (\bar{A}+B+\bar{C})\underbrace{(\bar{A}+B+C)}_{\text{Maxterm}}(\bar{A}+\bar{B}+\bar{C})$

Maxterm Designation

Variable with bar → 1

Variable without bar → 0

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

$Y = M_0\ M_1\ M_2\ M_3$

$M_0 \rightarrow A+B+C \rightarrow 000 = 0$

$M_1 \rightarrow A+B+\bar{C} \rightarrow 001 = 1$

$M_2 \rightarrow A+\bar{B}+\bar{C} \rightarrow 011 = 3$

$M_3 \rightarrow \bar{A}+B+C \rightarrow 100 = 4$

$Y = \underset{\text{Maxterm}}{\overset{\text{Product}}{\Pi M}}(0, 1, 3, 4)$

Conversion of Sum of Products form into Canonical form:

Example: $f = A + A\bar{B}$

$A\cdot(B+\bar{B}) + A\bar{B}$ $\quad [B+\bar{B}=1]$

$AB + A\bar{B} + A\bar{B}$ $\quad [A\cdot 1 = A]$

$AB + A\bar{B}$ $\quad [A\bar{B}+A\bar{B}=A\bar{B}]$

$\underset{3}{1\ 1}\quad \underset{2}{1\ 0}$

$f = \Sigma m(2, 3)$

Conversion of product of sums form into canonical form :-

Example: $(A+B)\cdot(B+C)$

$(A+B+C\bar{C})\ (A\bar{A}+B+C)$ $\quad [C\bar{C}=0]$

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

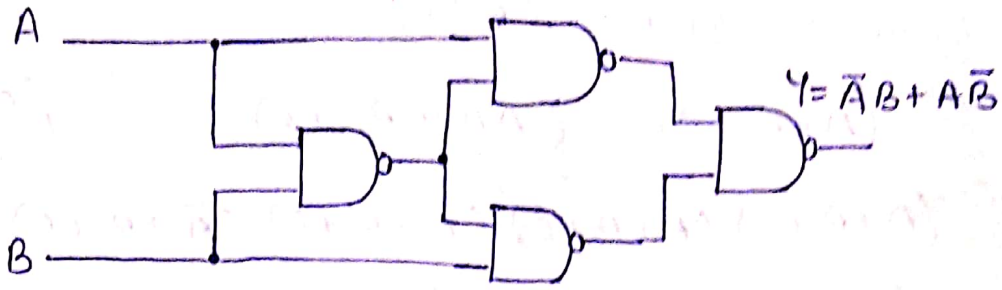
$(A+B+C)(A+B+\bar{C})(\bar{A}+B+C)$

$(0\ 0\ 0)\ (0\ 0\ 1)\ (1\ 0\ 0)$

$f = \Pi M(0, 1, 4)$

III

1) Implement of Ex-OR using 4 NAND Gate only



$$Y = \overline{[A \cdot (\overline{AB})]} \cdot \overline{[B(\overline{AB})]}$$

$$= \overline{[A \cdot (\overline{AB})]} + \overline{[B \cdot (\overline{AB})]}$$

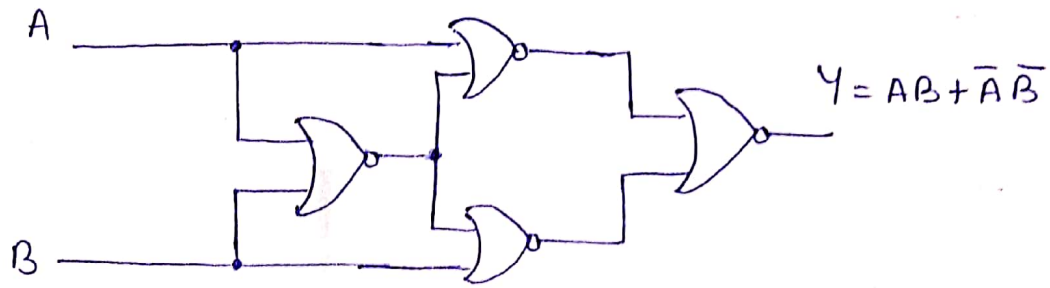
$$= A(\bar{A} + \bar{B}) + B(\bar{A} + \bar{B})$$

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

$[A\bar{A} = 0,\ B\bar{B} = 0]$

$$\boxed{Y = \bar{A}B + A\bar{B}}$$

2) Implement Ex-NOR Gate using 4 NOR Gate only



$$\overline{[A + (\overline{A+B})]} + \overline{[B + (\overline{A+B})]}$$

$$= \overline{[A + (\overline{A+B})]} \cdot \overline{[B + (\overline{A+B})]}$$

$$= [A + (\bar{A} \cdot \bar{B})] \cdot [B + (\bar{A} \cdot \bar{B})]$$

$$= [(A+\bar{A})(A+\bar{B})] \cdot [(B+\bar{A})(B+\bar{B})]$$

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

$$Y = A\bar{A} + AB + \bar{A}\bar{B} + \bar{B}B \Rightarrow \boxed{Y = AB + \bar{A}\bar{B}}$$

$\{A+\bar{A} = 1,\ B+\bar{B} = 1,\ A\bar{A} = 0,\ B\bar{B} = 0\}$

KARNAUGH MAP

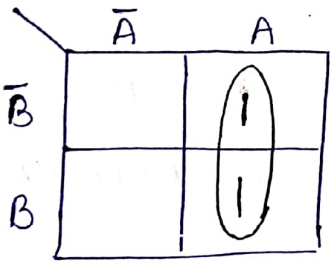
Karnaugh map (K-Map): is a graphical procedure for simplification of Boolean expression.

- It is a 2 dimensional representation of truth table
- It consist of squares. Each square represent a minterm. for n variables, 2^n squares are there
- for a minterm, 1's are written in corresponding squares.
- The two adjacent squares differ only by 1 variable. These can be grouped together for simplification.

TWO VARIABLE K-MAP

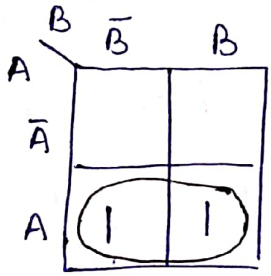
Q Solve using K-MAP

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



$Y = A$

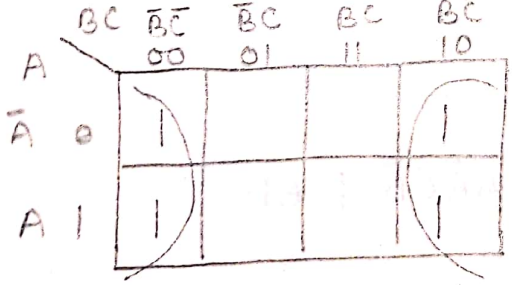
OR



$Y = A$

THREE VARIABLE K-MAP

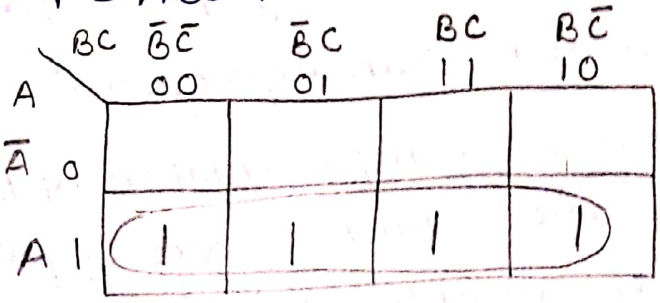
Q Solve $F = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C} + A\bar{B}\bar{C}$



$f = \bar{C}$

Q) Solve using K-Map

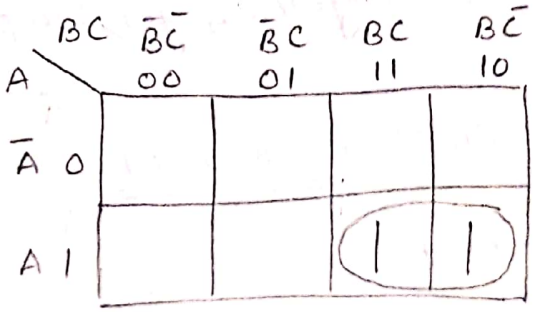
$$f = AB\bar{C} + A\bar{B}\bar{C} + ABC + A\bar{B}C$$



$$f = A$$

Q) Solve using K-map

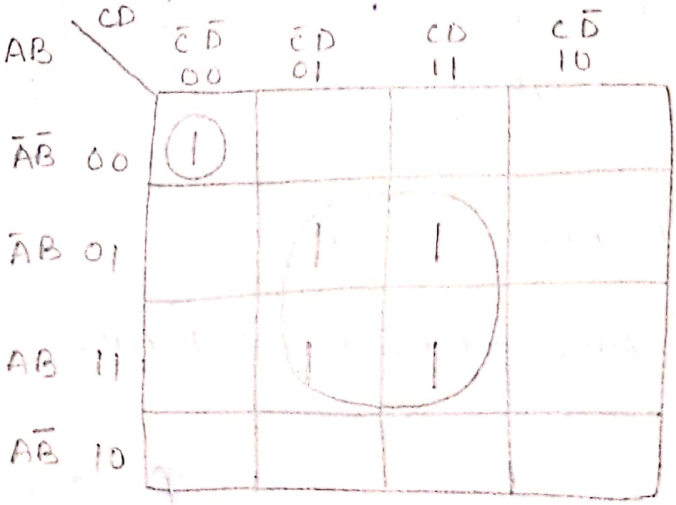
$$Y = AB\bar{C} + ABC$$



$$Y = AB$$

FOUR VARIABLE K-MAP

Q) Solve $Y = \bar{A}B\bar{C}D + AB\bar{C}D + \bar{A}BCD + ABCD + \bar{A}\bar{B}\bar{C}\bar{D}$



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

LABELLING OF K-MAP SQUARES

For SOP

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

OR

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

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

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

For POS

$(A+B)$ \ $C+D$	$C+D$ 00	$C+\bar{D}$ 01	$\bar{C}+\bar{D}$ 11	$\bar{C}+D$ 10
$(A+B)$ 00	0	1	3	2
$(A+\bar{B})$ 01	4	5	7	6
$(\bar{A}+\bar{B})$ 11	12	13	15	14
$(\bar{A}+B)$ 10	8	9	11	10

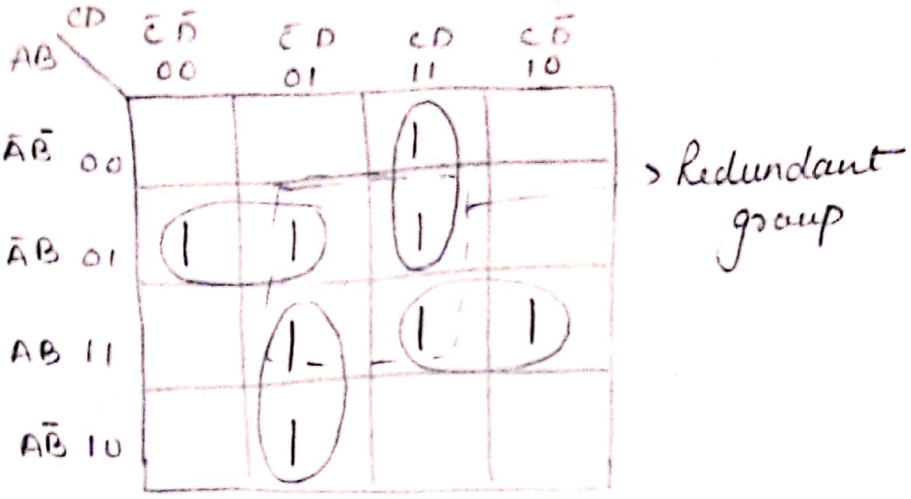
$C+D$ \ $A+B$	$A+B$ 00	$A+\bar{B}$ 01	$\bar{A}+\bar{B}$ 11	$\bar{A}+B$ 10
$C+D$ 00	0	4	12	8
$C+\bar{D}$ 01	1	5	13	9
$\bar{C}+\bar{D}$ 11	3	7	15	11
$C+\bar{D}$ 10	2	6	14	10

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

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

Q) Solve using K-Map

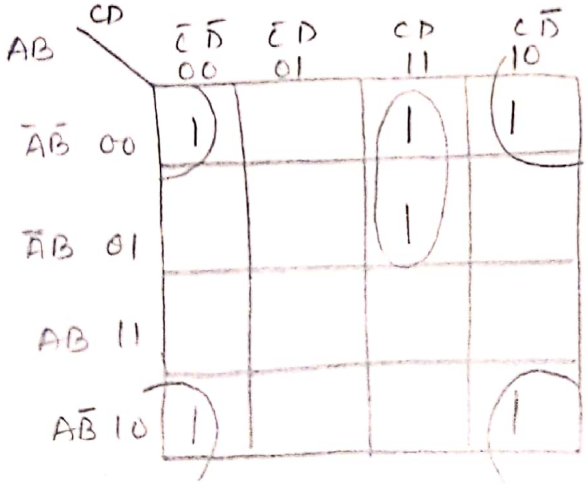
$F = \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}D + AB\bar{C}D + A\bar{B}\bar{C}\bar{D} + ABCD + AB\bar{C}\bar{D}$



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

Q) Solve using K-map

$Z = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}BCD + A\bar{B}C\bar{D} + \bar{A}\bar{B}CD$



$Z = \bar{B}\bar{D} + \bar{A}CD$

Q) Solve: $X = \bar{A}\bar{B}C + A\bar{C}\bar{D} + A\bar{B} + ABC\bar{D} + \bar{A}\bar{B}\bar{C}$

$X = \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D} + AB\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}\bar{C}\bar{D} + AB\bar{C}D + A\bar{B}C\bar{D} + ABC\bar{D} + \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D}$

$X = \bar{A}\bar{B}CD + \bar{A}\bar{B}C\bar{D} + AB\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + ABC\bar{D}$

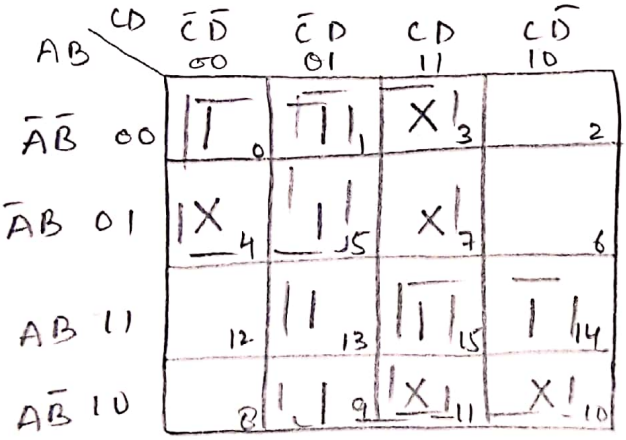
DON'T CARE CONDITIONS

For some functions, the outputs corresponding to certain combinations of i/p variable do not occur. In such situation, we have freedom to assume it '0' or '1' as o/p.

for each of these combinations these conditions are know as don't care conditions and in the K-map it is represented as (X) Crossmark in the corresponding cell.

Example: Solve using K-map

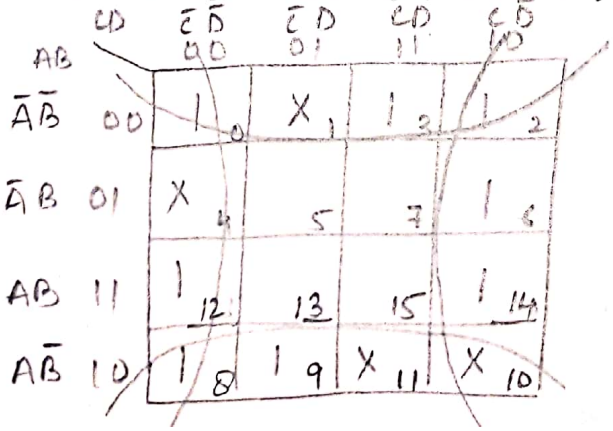
$$Y = \Sigma m(0,1,5,9,13,14,15) + d(3,4,7,10,11)$$



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

Q → Solve using K-map

$$F(ABCD) = \Sigma m(0,2,3,6,8,9,12,14) + d(1,4,10,11)$$



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