Unit-1

Number system

Number systems are ordered set of symbols with rules ~~defining~~ defined for performing arithmetic operations like addition, multiplication. A collection of these symbols makes a number which in general has two parts - integer & fractional.

$$(N)_b = \underbrace{d_{n-1}\, d_{n-2} \cdots d_1\, d_0}_{\text{Integer}} \cdot \underbrace{d_{-1}\, d_{-2} \cdots d_{-m}}_{\text{fractional part}}$$

N = a number

b = radix or base of number system

n = no. of digits in integer portion

m = no. of digits in fractional portion

d_{n-1} = most significant digit (MSD)

d_{-m} = least significant digit (LSD)

where, $0 \leq (d_{n-1} \cdots d_{-m}) \leq b-1$

Positional weights

$$(d_{n-1} \cdots d_1\, d_0 \cdot d_{-1}\, d_{-2} \cdots d_{-m})_b$$

$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$

$$b^{n-1} \quad b^1 \quad b^0 \quad b^{-1} \quad b^{-2} \quad b^{-m}$$

Commonly used number system :-

No. system	Base	Symbols used
Binary	2	0, 1
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9 A, B, C, D, E, F

Binary number system :

The no. system ~~with~~ with base 2 is known as binary no. system. only two symbols 0 & 1 are used to represent numbers in this system. These are known as bits.

eg- 1101, 1101.11, 1010.11

* The left most bit is known as most significant bit (MSB) and the right most bits are known as least significant bits (LSB).
* A group of four bits is known as nibble and a group of eight bits is known as byte.

Scanned by CamScanner

Binary to decimal conversion :-

Any binary no. canbe converted into its equivalent ~~binary~~ decimal no. using the weights assigned to each bit position.

eg- $(1110)_2 = (\ \ )_{10}$

$$\begin{array}{cccc} 1 & 1 & 1 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

Decimal No. $= 1\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$

$= 8+4+2 = (14)_{10}$

eg- $(110.011)_2 = (\ \ \ )_{10}$

$$\begin{array}{ccccccc} 1 & 1 & 0 & . & 0 & 1 & 1 \\ \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow & \downarrow \\ 2^2 & 2^1 & 2^0 & & 2^{-1} & 2^{-2} & 2^{-3} \end{array}$$

Decimal No. $= 1\times2^2 + 1\times2^1 + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}$

$= 4+2+\frac{1}{4}+\frac{1}{8}$

$= 6+0.25+0.125$

$= (6.375)_{10}$

Decimal to binary conversion:-

For Integers, the conversion is obtained by continuous division by 2 and keeping track of the remainders, while for fractional parts, the conversion is affected by continuous multiplication by 2 and keeping track of the integers generated.

eg:- $(10.625)_{10} = (\quad)_2$

For Integer

2	10	
2	5	0
2	2	1
2	1	0
	0	1

$(10)_{10} = (1010)_2$

For fractional part

$0.625 \times 2 = 1.250$

$0.250 \times 2 = 0.500$

$0.50 \times 2 = 1.00$

i.e. $(0.625)_{10} = (.101)_2$

Hence $(10.625)_{10} = (1010.101)_2$

Octal Number system

The number system with base (or radix) eight is known as the octal number system. In this system, eight symbols 0, 1, 2, 3, 4, 5, 6 & 7 are used to represent numbers.

eg:- $(237)_8$ – valid octal no.

$(238)_8$, $(219)_8$ – invalid octal no.

Octal to Decimal conversion:-

Any octal no. canbe converted into its equivalent decimal number using the weights assigned to each octal digit position

eg:-(a) $(237)_8 = (\ \ )_{10}$

$$(237)_8 = 7\times8^0 + 3\times8^1 + 2\times8^2 = (159)_{10}$$

(b) $(0.54)_8 = (\ \ )_{10}$

$$(0.54)_8 = 5\times8^{-1} + 4\times8^{-2} = (0.6875)_{10}$$

(c.) $(120.4)_8 = (\ \ )_{10}$

$$(120.4)_8 = 0\times8^0 + 2\times8^1 + 1\times8^2 + 4\times8^{-1} = (80.5)_{10}$$

Decimal to octal conversion:-

The conversion from decimal to octal is similar to the conversion from decimal to binary. The only difference is that number 8 is used in place of 2 for division in the case of integers & for multiplication in the case of fractional numbers.

Examples:

a) $(247)_{10} = (\quad)_8$

8	247	
8	30	7
8	3	6
	0	3

i,e, $(247)_{10} = (367)_8$

b) $(0.6875)_{10} = (\quad)_8$

$0.6875 \times 8 = 5.50$

$0.50 \times 8 = 4.00$

i,e, $(0.6875)_{10} = (0.54)_8$

c) $(444.456)_{10} = (\quad)_8$

Integer part:-

8	444	
8	55	4
8	6	7
	0	6

i,e, $(444)_{10} = (674)_8$

Fractional part :-

$0.456 \times 8 = 3.648$

$0.648 \times 8 = 5.184$

$0.184 \times 8 = 1.472$

$0.472 \times 8 = 3.776$

$0.776 \times 8 = 6.208$

- - - -

- - - - -

The process canbe terminated when significant digits are obtained.

i.e, $(0.456)_{10} \approx (0.35136)_8$

Thus, the octal conversion of $(444.456)_{10} \approx (674.35136)_8$.

Octal to binary conversion:-

Octal numbers canbe converted into its equivalent binary numbers by replacing each octal digit by its 3 bit equivalent binary.

Octal digit	3 bit binary equivalent
0	000
1	001
2	010
3	011
4	100
5	101
6	110

Examples :-

a) $(736)_8 = (\quad)_2$

$7 \equiv 111$
$3 \equiv 011$
$6 \equiv 110$

thus, binary equivalent of $(736)_8$ is $(\underbrace{111}\,\underbrace{011}\,\underbrace{110})_2$

b) $(20.4)_8 = (\underbrace{010}\ \underbrace{000}.\underbrace{100})_2$

Binary to octal conversion:-

Binary numbers canbe converted into equivalent octal numbers by making groups of three bits starting from LSB & moving towards MSB for integer part of the number & then replacing each group of three bits by its octal representation. For fractional part, the grouping of three bits are made starting from the binary point.

Examples:

a) $(1001110)_2 = (\quad)_8$

$(1\underbrace{001}110)_2 = (001\ 001\ 110)_2$
$= (116)_8$

iii) $(1100111 0001 . 0001011110011)_2 = (\quad)_8$

$(11001110001.00010111100110)_2 =$

$= (011\ 001\ 110\ 001 . 000\ 101\ 111\ 001\ 100)_2$

$= (3161.05714)_8$

* Hexadecimal Number system *

The base for hexadecimal number system is 16 which requires 16 distinct symbols to represent the numbers. These are numerals 0 through 9 and alphabets A through F.

Since, numeric digits & alphabets both are used to represent the digits in the hexadecimal number system, therefore, this is an alphanumeric number system.

Decimal no.	Hexadecimal	Binary Equivalent	Decimal	Hexa decimal	Binary Equivalent
0	0	0000	8	8	1000
1	1	0001	9	9	1001
2	2	0010	10	A	1010
3	3	0011	11	B	1011
4	4	0100	12	C	1100
5	5	0101	13	D	1101
6	6	0110	14	E	1110
7	7	0111	15	F	1111

Hexadecimal to Decimal Equivalent :-

Any hexadecimal no. canbe converted into equivalent decimal no. using the weights assigned to each hexadecimal symbol.

In other words, conversion canbe carried out by multiplying each significant digit of the hexadecimal by its respective weight & adding the products.

Examples:

$(3A.2F)_{16} = (\quad)_{10}$

$(3A.2F)_{16} = 10\times16^{0} + 3\times16^{1} + 2\times16^{-1} + 15\times16^{-2}$

$= (58.1836)_{10}$

Decimal to hexadecimal conversion:-

For Integers: The hexadecimal equivalent of a decimal no. canbe obtained by dividing the given decimal no. by 16 continuously until a quotient of 0 is obtained. After that arrange the remainders obtained in each step in reverse order.

For fractional part, multiply by '16' & keep tracking of hexadecimal symbols generated. Afterthat arrange them in forward order.

Examples :-

a) $(236)_{10} = (\ )_{16}$

16	236	
16	14	C
	0	E

i.e $(236)_{10} = \cancel{(CE)}_{16}\ (EC)_{16}$

b) $(675.625)_{10} = (\ )_{16}$

Integral part

16	675	
16	42	3
16	2	A
	0	2

$(675)_{10} = (2A3)_{16}$

Fractional part :-

$0.625 \times 16 = 10.000 \equiv A.000$

i.e $(0.625)_{10} = (0.A)_{16}$

Hence, $(675.625)_{10} = (2A3.A)_{16}$

Hexadecimal to binary conversion:-

Hexadecimal numbers canbe converted into its equivalent binary numbers by replacing each hex digit by its equivalent 4 bit binary numbers.

Examples:-

$(2F9A.5)_{16} = (\ \ )_2$

$= (0010\,1111\,1001\,1010 . 0101)_2$

Binary to Hexadecimal conversion:-

Binary numbers canbe converted into its equivalent hexadecimal numbers by making group of four bits starting from LSB & moving towards MSB for integer part & then replacing each group of four bits by its hexadecimal representation.

For fractional part, the above procedure is repeated starting from the bit next to the binary point & moving towards the right.

Examples:

$(110\ 0111\ 0001 . 1001\ 1001\ 101)_2$

$= (0110\ 0111\ 0001 . 1001\ 1001\ 1010)_2$

$= (671.99A)_{16}$

Conversion from Hex to octal & vice versa

$$\text{Hexadecimal} \rightleftarrows \text{Binary} \rightleftarrows \text{Octal}$$

Examples :-

a) $(A72.BF8)_{16} = (\quad)_8$

$(A72.BF8)_{16} = (1010\ 0111\ 0010 . 1011\ 1111\ 1000)_2$

~~= 101 100 111~~

$= (101\ 001\ 110\ 010 . 101\ 111\ 111)_2$

$= (5162.577)_8$

b) $(247.36)_8 = (\quad)_{16}$

$(247.36)_8 = (010\ 100\ 111 . 011\ 110)_2$

$= (1010\ 0111 . 0111\ 1000)_2$

$= (A7.78)_{16}$

* Binary Arithmetic :-

[A] Binary Addition :-

Addition Rule

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 0$ & carry = 1

Ex - Add $(15)_{10} + (10)_{10}$ by binary addition method.

$15 \equiv 1111$

$10 \equiv 1010$

$$\begin{array}{r} 1111 \\ +1010 \\ \hline 11001 \end{array} \qquad \begin{array}{r} 15 \\ +10 \\ \hline 25 \end{array}$$

[B] Binary subtraction :-

Subtraction rule

$0 - 0 = 0$

$0 - 1 = 1$ with Borrow = 1

$1 - 0 = 1$

$1 - 1 = 0$

Ex $(13)_{10} - (9)_{10}$

$$\begin{array}{rr} 13 & 1101 \\ -9 & 1001 \\ \hline 4 & 0100 \end{array}$$

Ex $(1011)_2 - (0110)_2$

$$\begin{array}{r} 1011 \\ 0110 \\ \hline 0101 \end{array}$$

1's & 2's complements

Digital circuits are used for performing binary arithmetic operations. It is possible to use the circuits designed for binary addition to perform the binary subtraction also if we can change the problem of subtraction to that of an addition. This concept eliminates the need of additional circuit for subtraction. This makes design of arithmetic circuits very convenient & cheaper.

For this purpose, 1's & 2's complement representation of binary no. is discussed.

1's complement subtraction:-

The 1's complement of a binary no. canbe obtained by changing all 1s to 0s & all 0s to 1s.

eg- 1's complement of (1011) ≡ (0100)

Steps:-

a) Determine the 1's complement of subtrahend.

b) Add this to the ~~a~~ second number by binary addition method.

c) If any carry is generated at MSB, add it to the result.

NOTE:- Carry generation reveals that final answer is positive. i.e, we have subtracted a smaller number from larger number.

d) If carry is not generated at MSB, This means results is in 1's complement form& final result is a negative number.

In other words, we have subtracted a larger number from smaller number.

Ex:-

a) $(1111)_2 - (1010)_2$ by 1's complement method.

1's complement of 1010 ≡ 0101

$$\begin{array}{r} 1111 \\ +\ 0101 \\ \hline 1\,0100 \\ +1 \\ \hline 0101 \end{array}$$

Direct Subtraction -

$$\begin{array}{r} 1111 \\ 1010 \\ \hline 0101 \\ \hline \end{array}$$

b) $(1010)_2 - (1111)_2$

1's complement of 1111 ≡ 0000

$$\begin{array}{r} 1010 \\ +\ 0000 \\ \hline 1010 \\ \hline \end{array}$$

→ Here no carry is generated.

i.e answer is in 1's comp. form

2's complement subtraction :-

2's complement of a binary number =
(1's comp) + 1

Rules for binary subtraction using 2's comp. is same as 1's complement. except carry generated at MSB is ignored here.

eg:- $(1111)_2 - (1010)_2$

2's complemnt of 1010 = 0110

$$\begin{array}{r} 1111 \\ +\ 0110 \\ \hline 10101 \end{array}$$

Carry should ignored. so Ans = 0101.

eg:- $(1010) - (1111)_2$

2's comp. of 1111 = 0001

$$\begin{array}{r} 1010 \\ 0001 \\ \hline 1011 \end{array}$$

→ No carry at MSB, Hence Result is in 2's comp. form.

[C] Binary multiplication :-

Rules

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

$$
\begin{array}{r}
101 \\
\times\ 101 \\
\hline
101 \\
000 \\
101 \\
\hline
11001 \\
\hline
\end{array}
$$

[D] Binary Division :

Rules

$0 \div 1 = 0$

$1 \div 1 = 1$

Example.

Divide 111010l by 1001

1001) 1110101 (1101

1001

x1011

1001

xx1001

1001

0000

Logic Gates

A logic gate is an electronic circuit which makes logical decisions.

The most common logic gates used are OR, AND, NOT, NAND & NOR.

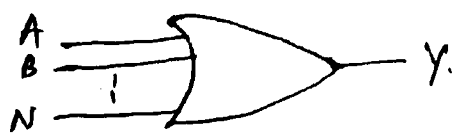
The exclusive-OR (EX-OR) and exclusive-NOR (EX-NOR) gates are another logic gates which can be constructed using basic gates such as AND, OR and NOT.

* The NAND & NOR gates are called as the universal gates, because either NAND & NOR are sufficient for the realization of any logical expression.

(a) OR Gate :-

→ multiple i/p single o/p logic gate

→ Symbol :-



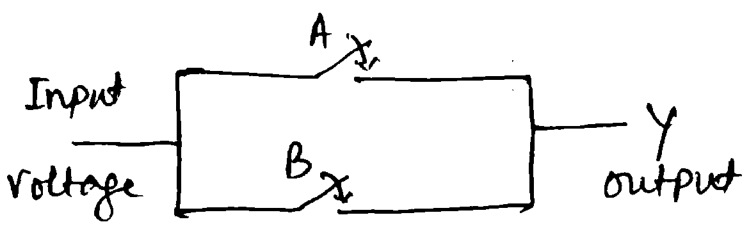
→ logical expression:-

$$Y = A \text{ OR } B \text{ OR } C \text{ --- OR } N$$
$$= A + B + C + \text{---} + N$$

→ Truth Table (for two i/p OR gate)

Inputs		output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

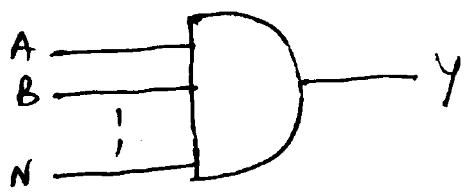
→ Electrical Equivalent circuit -



(b) AND gate

→ multiple i/p single o/p logic gate

→ Symbol



→ logical expression:-

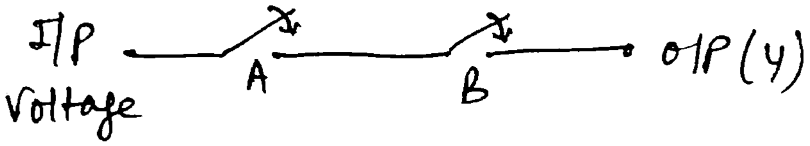
$Y = A$ AND B AND C --- AND N

$= A \cdot B \cdot C \cdot \ldots \cdot N$

→ Truth table for two i/p AND gate

Inputs		output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

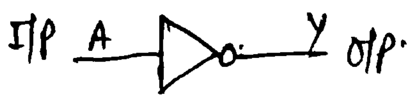
→ Electrical equivalent circuit -



(c.) NOT gate :-

* Single i/p single o/p logic gate

* Symbol -



* logical expression:-

$Y = \bar{A}$

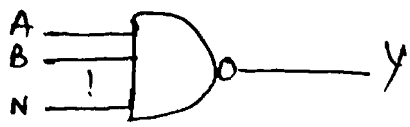
* Truth table

I/P	O/P
A	Y
0	1
1	0

* Also known as Inverter.

(d) NAND gate :-

* multiple i/p single o/p logic gate
* It is a combination of NOT + AND
* Symbol.



* logical expression:-

$$Y = \overline{ABC \cdots N}$$

* Truth table for two i/p NAND gate

Inputs		o/p
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(e) NOR gate :-

* multiple i/p single o/p logic gate.
* It is a combination of NOT & OR.
* Symbol -

* logical expression:-

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

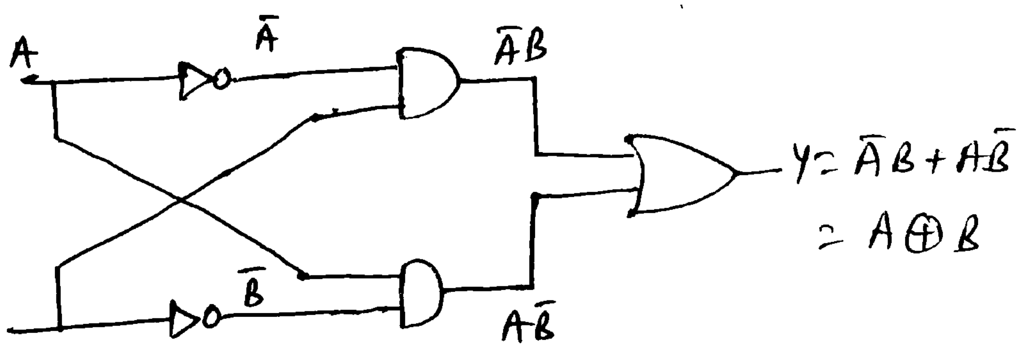
* Truth table for two i/p ~~NAND~~ NOR gate

Inputs		O/P
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

(f) EX-OR gate :

* multiple i/p single o/p logic gate

* EX-OR using basic gate



* Symbol -

* logical expression:-

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

Scanned by CamScanner

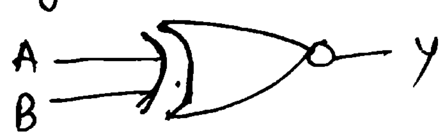
Truth table (for 2-input)

I/P		O/P
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

(g) EX-NOR gate

→ multiple i/p single o/p gate

→ Symbol



→ logical expressions

$$Y = \overline{A \oplus B} = A \odot B = AB + \bar{A}\bar{B}$$

→ Truth table -

Inputs		O/P
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

* Properties of Ex-OR

$A \oplus A = 0$

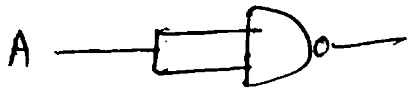
$A \oplus \bar{A} = 1$

$A \oplus 0 = A$

$A \oplus 1 = \bar{A}$

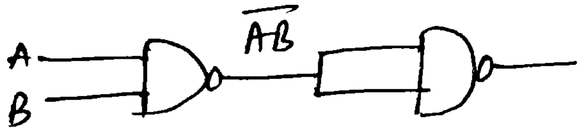
A Realization of different logic gates using NAND gate only

a) NOT



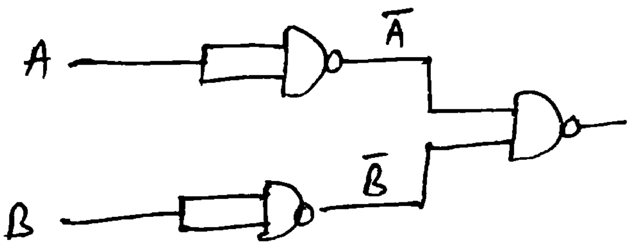
$Y = \overline{A \cdot A} = \overline{A}$

b) AND.



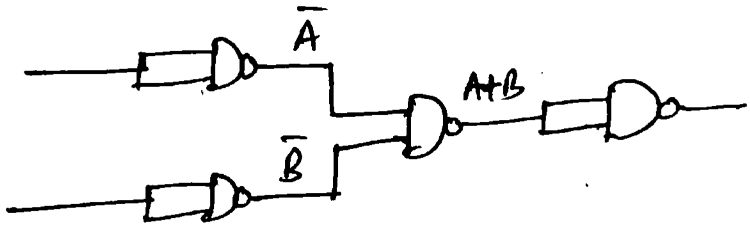
$Y = \overline{\overline{AB}} = AB$

c) OR



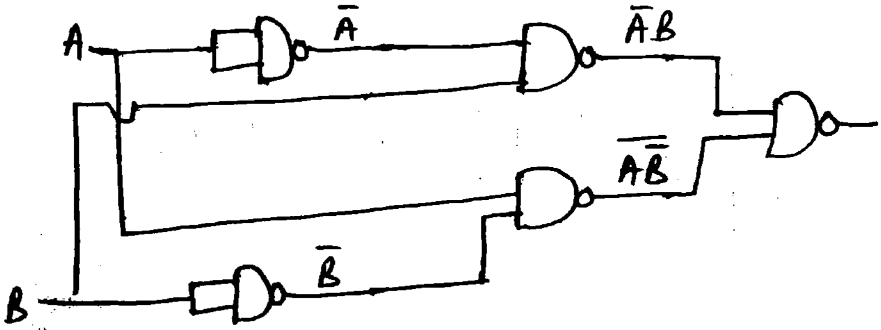
$Y = \overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A}} + \overline{\overline{B}} = A + B$

d) NOR



$Y = \overline{A+B}$

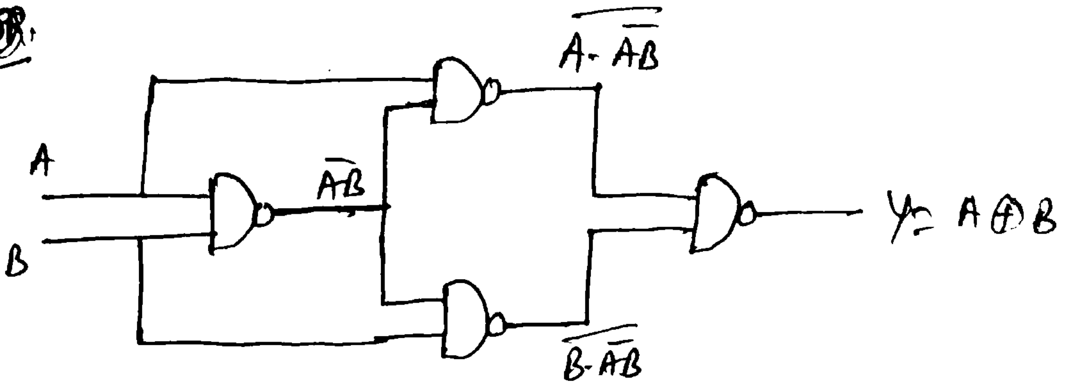
e) EX-OR



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

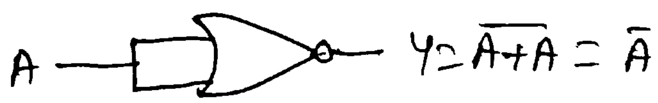
$= \overline{A}B + A\overline{B}$

8.

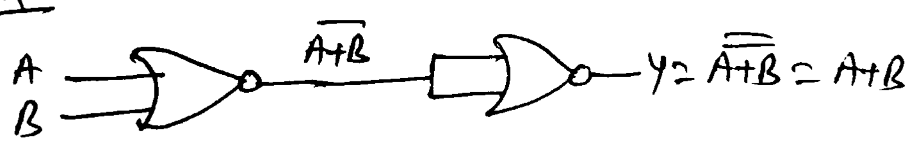


* Realization of different gate using NOR gate

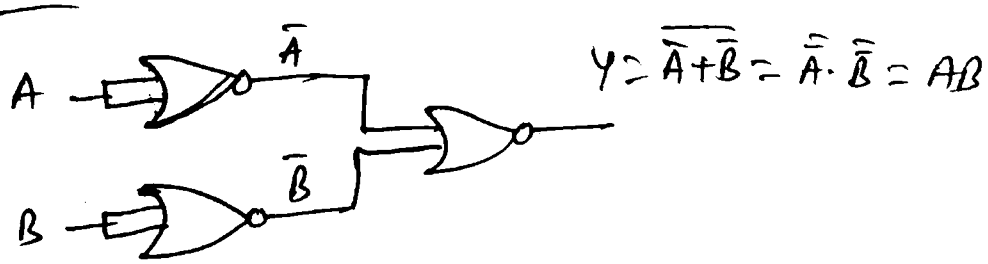
a) NOT



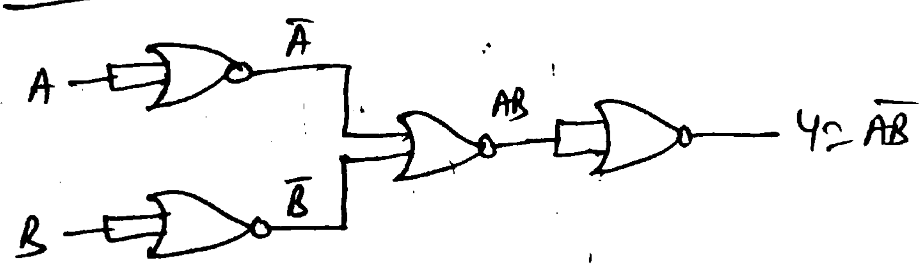
b) OR



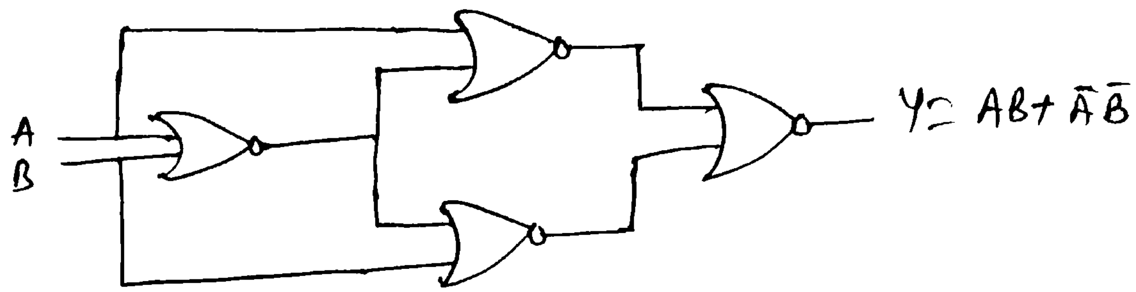
c) AND



d) NAND



e) EX-NOR –



* Examples :-

1) Realise the logic expression $Y = (A+B)(\bar{A}+C)(B+D)$ using basic gates.

2) Realise $Y = B\bar{C} + \bar{A}C + \bar{A}\bar{B}$.

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

Boolean Algebra & its simplification

In 1854, an English mathematician George Boole invented a new kind of algebra (the algebra of logic) popularly known as Boolean Algebra or Switching Algebra.

Boolean Algebra differs significantly from conventional Algebra. This algebra deals with the rules by which the logical operations are carried out. This is the basic of all digital systems like computers, calculators etc.

* **Boolean Addition :-**

$A + 1 = 1$

$A + A = A$

$A + 0 = A$

$A + \bar{A} = 1$

* **Boolean Multiplication :-**

$A \cdot 1 = A$

$A \cdot 0 = 0$

$A \cdot A = A$

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

* **Inversion :-**

$\bar{\bar{A}} = A$

$\bar{0} = 1$

$\bar{1} = 0$

Properties of Boolean Algebra

(1) Commutative Property :-

$$A+B = B+A$$
$$\& \; A\cdot B = B\cdot A$$

i.e, order of the AND & OR operation performed on the variables makes no difference.

(2.) Associative property :-

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

i.e It makes no difference in what order the variables are grouped during the AND & OR operation of several variables.

(3) Distributive property :-

a) $A\cdot(B+C) = AB+AC$

* $\boxed{\text{b) } A+BC = (A+B)(A+C)}$

Proof:- $(A+B)(A+C) = A\cdot A + AC + BA + BC$

$= A + AC + AB + BC$

$= A[1+C] + AB + BC \quad \because 1+C=1$

$= A + AB + BC$

$= A[1+B] + BC \quad \because 1+B=1$

$= A+BC$

4) Absorption law :-

a) $A + AB = A$

b) $A(A+B) = A$

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

Proof :- a) $A + AB = A[1+B] = A$

b) $A(A+B) = A \cdot A + AB = A + AB = A$

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

5) consensus law :-

a) $AB + \bar{A}C + BC = AB + \bar{A}C$

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

Proof :- a) $AB + \bar{A}C + BC = AB + \bar{A}C + BC[A+\bar{A}]$

$= \underline{AB} + \bar{A}C + \underline{ABC} + \bar{A}BC$

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

$= AB + \bar{A}C$

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

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

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

$\because A(A+B) = A$

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

Scanned by CamScanner

* Demorgan's law :-

Two theorems that are an important part of Boolean Algebra were proposed by Demorgan.

First theorem :-

The complement of a product is equal to the sum of their complements.

i.e. $\overline{A \cdot B} = \bar{A} + \bar{B}$

Second theorem :-

The complement of a sum is equal to the product of their complements.

i.e. $\overline{A+B} = \bar{A} \cdot \bar{B}$

Proof of Demorgan's law -

A	B	$\bar{A}$	$\bar{B}$	$A+B$	AB	$\overline{A+B}$	$\bar{A} \cdot \bar{B}$	$\overline{AB}$	$\bar{A}+\bar{B}$
0	0	1	1	0	0	1	1	1	1
0	1	1	0	1	0	0	0	1	1
1	0	0	1	1	0	0	0	1	1
1	1	0	0	1	1	0	0	0	0

Q Simplify the following Boolean expression.

1) $AB + BC + \bar{B}C \rightarrow AB + C$

2) $\bar{A}B + AB + \bar{A}\bar{B} \longrightarrow B + \bar{A}$

3) $A + A\bar{B} + \bar{A}B \longrightarrow A + B$

4) $AB + \overline{AC} + A\bar{B}C(AB+C) \longrightarrow 1$

$AB + \bar{A} + \bar{C} + 0 + A\bar{B}C$

$= A[B + \bar{B}C] + \bar{A} + \bar{C}$

$= A[B + C] + \bar{A} + \bar{C}$

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

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

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

$\cancel{= \bar{A}+B+\bar{C}+AC}$ $= \bar{A} + B + A + \bar{C}$

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

$= 1 + B + \bar{C} = 1.$

5.) $(\bar{A}+B)(A+B) \longrightarrow B$

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

$= B.$

Standard representations of logical functions

Logical functions are expressed in terms of logical variables. Any arbitrary logic function can be expressed in the following forms.

a) Sum of Products (SOP) form

b) Product of sums (POS) form

SOP :- A sum of products expression consists of product terms logically added. This canbe realized using AND-OR configuration(two level realization).

first level — AND
2nd level — OR

eg- $Y = AB + A\bar{B} + BC$

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

POS :- A product of sums expression consists of sum term logically multiplied. This canbe realized using OR-AND configuration.

- first level — OR
2nd level — AND

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

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

$Y = A(B+C)$

Standard / cannonical form of SOP & POS :

If each term in SOP and POS forms contains all the variables then these are known as standard or canonical SOP & POS respectively.

Each individual term in canonical SOP form is called minterm & in canonical POS form as maxterm.

eg- $Y = AB + A\bar{B}$
$Y = AB\bar{C} + \bar{A}BC + ABC$ — Canonical SOP

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

Q:- Convert the given expressions into their canonical form.

a) $Y = AB + A\bar{C} + BC$

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

Soln:- a) $Y = AB + A\bar{C} + BC$

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

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

Scanned by CamScanner

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

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

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

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

Minterm & Maxterm designation :-

The concept of minterm & maxterm allows us to introduce a very convenient shorthand notation to express logical functions.

In general, for an 'n' variable logical functions there are '2^n' minterms & an equal no. of maxterms.

eg- minterms/maxterms for three variables

Variables A B C	Minterm	Maxterm
0 0 0	$\bar{A}\bar{B}\bar{C} \equiv m_0$	$A+B+C \equiv M_0$
0 0 1	$\bar{A}\bar{B}C \equiv m_1$	$A+B+\bar{C} \equiv M_1$
0 1 0	$\bar{A}B\bar{C} \equiv m_2$	$A+\bar{B}+C \equiv M_2$
0 1 1	$\bar{A}BC \equiv m_3$	$A+\bar{B}+\bar{C} \equiv M_3$
1 0 0	$A\bar{B}\bar{C} \equiv m_4$	$\bar{A}+B+C \equiv M_4$
1 0 1	$A\bar{B}C \equiv m_5$	$\bar{A}+B+\bar{C} \equiv M_5$
1 1 0	$AB\bar{C} \equiv m_6$	$\bar{A}+\bar{B}+C \equiv M_6$
1 1 1	$ABC \equiv m_7$	$\bar{A}+\bar{B}+\bar{C} \equiv M_7$

for minterms

normal (uncomplemented) variables are taken as 1's & the complemented variables are taken as 0's.

for maxterms

uncomplemented variables are taken as 0's & complemented variables are taken as 1's.

⇒ Minterm representation of

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

$$\bar{A}BC \equiv m_3$$
$$A\bar{B}\bar{C} \equiv m_4$$
$$AB\bar{C} \equiv m_6$$
$$ABC \equiv m_7$$

So, $Y = m_3 + m_4 + m_6 + m_7$

or $\underline{Y = \sum m(3,4,6,7)}$

⇒ Maxterm representation of

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

$$A+B+C \equiv M_0$$
$$A+B+\bar{C} \equiv M_1$$
$$A+\bar{B}+C \equiv M_2$$
$$\bar{A}+B+\bar{C} \equiv M_5$$

$$Y = M_0 \cdot M_1 \cdot M_2 \cdot M_5 = \underline{\prod M(0,1,2,5)}$$

NOTE:- SOP and POS forms of Boolean expressions are complementary forms, so the minterms & maxterms notation are also complementary to each other.

eg- for a three variables case

if $Y = \Sigma m(1, 2, 4, 6)$

then $Y = \Pi M(0, 3, 5, 7)$

Karnaugh map (K-map) representation of logical functions

K-map is a graphical method of simplifying Boolean expressions which provides a systematic method for simplifying and manipulating Boolean expressions. In this techniques the information contained in a truth table or available in POS or SOP form is represented on K-map.

In an n- variable K-map, there are 2^n cells. Each cell corresponds to one of the combinations of n variables. i.e, for each min-term & for each maxterm there is one specific cell in the K-map.

Two variable K-map

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

SOP

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

POS

Three variable K-map

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

Four variable K-map

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

Representation of truth table on K-map

Consider a truth table of 3-variable

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

The o/p Y is logic '1' corresponding to row 1, 2, 4 and 7. Hence we can write the canonical SOP form from truth table as.

$$Y = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$
$$= \sum m(1,2,4,7)$$

Similarly, The o/p Y is logic '0' corresponding to the rows 0, 3, 5 and 6. Hence we can write the 'canonical' POS form from truth table as.

$$Y = (A+B+C)(A+\bar{B}+\bar{C})(\bar{A}+B+\bar{C})(\bar{A}+\bar{B}+C)$$
$$= \prod M(0,3,5,6)$$

NOTE:- Representation of standard SOP/POS form on K-map can also be done by applying the knowledge discussed above in reverse order.

Simplification of logical functions using K-map

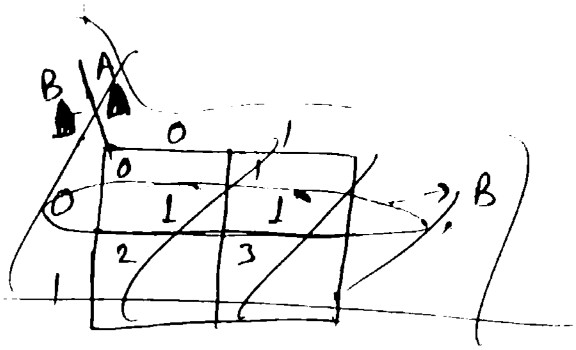
Q: Simplify - a) $Y = \bar{A}\bar{B} + A\bar{B}$

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

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

~~0 0 0 1~~

~~$\Rightarrow Y = \sum m(0,1)$~~

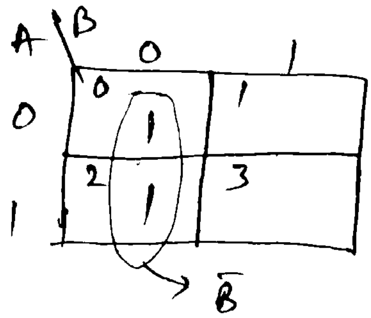


a) $Y = \bar{A}\bar{B} + A\bar{B}$

00 10

$\Rightarrow Y = \sum m(0,2)$

$Y = \bar{B}$

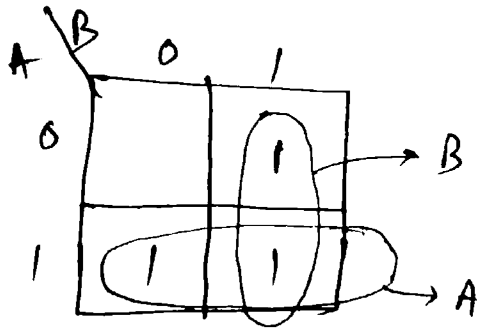


b) $Y = \bar{A}B + AB + A\bar{B}$

01 11 10

$Y = \sum m(1,2,3)$

$Y = A + B$



3 variable K-map

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

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

$= \bar{A}BC + AB\bar{C} + \bar{A}B\bar{C} + \cancel{AB\bar{C}} + A\bar{B}C$

$\quad 011 \qquad 110 \qquad 010 \qquad 110 \qquad 101$

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

$\bar{A}B$

$B\bar{C}$

$A\bar{B}C$

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

Q:- $Y = \Sigma m(0, 2, 4, 6)$

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

$Y = \bar{C}$

Q:- $Y = \Sigma m(0, 2, 3, 4, 6)$

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

4- variable - K map

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

$\quad\quad$ 0101 $\quad$ 1101 $\quad$ 0111 $\quad$ 1111 $\quad$ 0000

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

$\bar{A}\bar{B}\bar{C}\bar{D}$ $\quad$ BD

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

Q:- $Y = \Sigma m(1, 4, 6, 9, 12, 14)$

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

$B\bar{D}$ $\quad$ $\bar{B}\bar{C}D$

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

Minimization of POS expression:-

$$Y = \prod M(0,1,2,3,4,5,7,11,15)$$

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

$(A+B)$

$(\bar{C}+\bar{D})$

$(A+C)$

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

Scanned by CamScanner

* Don't care condition :-

In K-map, we make the entries in the map for either 1's or 0's. The cell which donot contain 1 are assumed to contain 0 & vice versa. This is not always true since there are cases in which certain combinations of input variables donot occur. Also, for some functions the o/P corresponding to certain combinations of i/P variables donot matter. In such situations the designer has a flexibility & it is left to him whether to assume a 0 or a 1 as o/P of these combinations. This condition is known as don't care condⁿ, & denoted by (X). The x mark in a cell maybe used or maynot be used depending upon which one leads to a simpler expression.

eg- $y = \sum m(0,1,2,4) + \underbrace{d(5,7)}_{\text{don't care cond}^n}$.

Q:- Simplify -

$$Y = \Sigma m(1,3,7,11,15) + d(0,2,5)$$

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

$\bar{A}D$

CD

$$Y = \bar{A}D + CD.$$