CHAPTER 1

DIGITAL ELECTRONICS

1. DIGITAL NUMBER SYSTEM

A number system with base 'b' will have 'b' different digits from 0 to (b – 1).

Number Representation

$$(N)_b = d_{n-1}d_{n-2}...d_1d_0.d_{-1}d_{-2}...d_{-f}...d_{-m}$$

where, $N \rightarrow$ Number

$b \rightarrow$ Base or radix

$d_{n-1}d_{n-2}...d_i...d_1...d_1d_0$ represents integral portion of number $(N)_b$, $d_{-1}d_{-2}.....d_{-f}...d_{-m}$ represent fraction portion of number and between these two there is a radical sign.

Weighted Number System :

Binary, octal hexa decimal, BCD, 2421 etc.

Unweighted Number System :

Gray code, Excess 3-Code :

Number System	Base (b)	Digits
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Octal	8	0, 1, 2, 3, 4, 5, 6, 7
Binary	2	0, 1
Hexa Decimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9 A, B, C, D, E, F

In binary number system, a group of "Four bits" is known as "Nibble" and group of "Eight bits" is known as "Byte".

4bits = 1Nibble

8bits = 1Byte

CODES :

Binary Coded Decimal Code (BCD) :

Each digit of a decimal number is represented by binary equivalent.

In 4-bit Binary Formats :

Total number of possible representation = $2^4 = 16$

Valid BCD codes = 10

Invalid BCD codes = 6

In 8-bit Binary Formats :

Valid BCD Codes = 100

Invalid BCD Codes = 256 – 100 = 156

- ***Excess-Three Code :***

  It is a 4-bit code.

  It can be derived from BCD code by adding "Three" to each coded number.

  It is a "Self-complementing code".

- ***Gray Code :***

  Also called "minimum change codes" in which only one bit in the code group changes when going from one step to the next.

Remember :

In self-complemented code, the sum of weightage = 9.

The largest number that can be represented by using N-bits is $(2^N - 1)_{10}$.

2. LOGIC GATES

EXOR, EXNOR are Other Gates

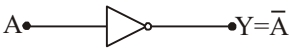
OR, AND, NOR are Basic Gates

NAND, NOR are Universal Gates.

- ***Not Gate :***

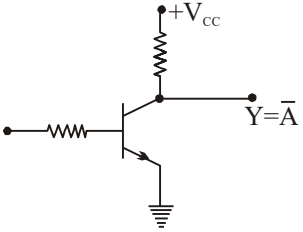
  Also referred to as "Inversion" or "Complementation."

  Symbol and Truth table :



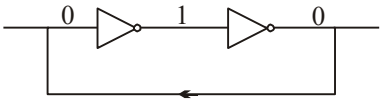
Input	Output Y = $\overline{A}$
0	1
1	0

Transistor Circuit :



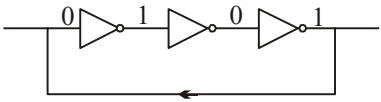
A	T	Y
0	OFF	$+V_{cc} \approx 1$
1	ON	Ground ≈ 0

When 'Even number' of NOT Gates are connected in Feedback, then it acts like as "Bistable Multivibrator".



When "Odd number" of NOT Gates are connected in Feedback, then it acts like as Astable Multivibrator (AMV) or Squarewave generator or Clock generator or Ring oscillator.

All inverters take some time to get the response "Y", this time is called "Propagation delay time" (t_{pd}).



For An Astable Multivibrator (AMV) :

Time Period of AMV :

$$\boxed{T = 2nt_{pd}}$$

Where, $n \rightarrow$ Number of inverters (NOT Gates)

$t_{pd} \rightarrow$ Propagation delay time of each inverter

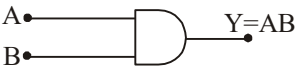
$T \rightarrow$ time period of a square wave generated by AMV or Ring oscillator.

Frequency of AMV :

$$\boxed{f = \frac{1}{T} = \frac{1}{2nt_{pd}}}$$

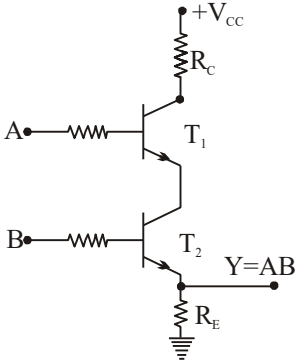
AND GATE :

Symbol and Truth table :



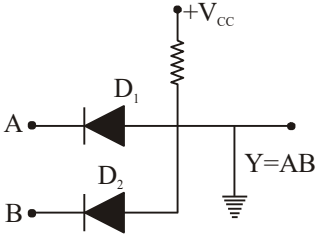
Input		Output
A	B	Y =AB
0	0	0
0	1	0
1	0	0
1	1	1

Transistor Circuit :



A	B	T_1	T_1	Y
0	0	OFF	OFF	0
0	1	OFF	ON	0
1	0	ON	OFF	0
1	1	ON	ON	1

Diode Circuit Diagram :



A	B	D_1	D_2	Y
0	0	ON	ON	0
0	1	ON	OFF	0
1	0	OFF	ON	0
1	1	OFF	OFF	1

Remember :

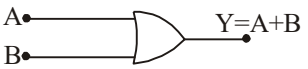
In AND gate operation, any unused inputs (Floating input) may be connected as:

Logic '1' for TTL circuit

Logic '0' for ECL circuit

- ***OR GATE :***

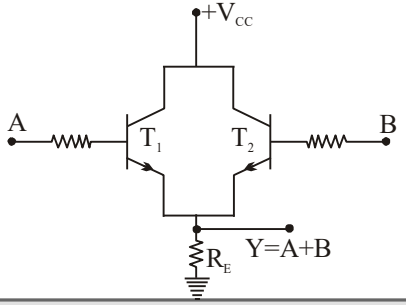
Symbol and Truth Table



Inputs		Output
A	B	Y =A+B
0	0	0
0	1	1
1	0	1
1	1	1

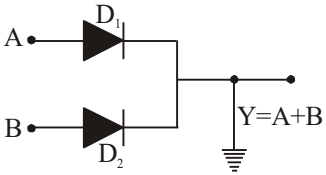
Transistor Circuit :

A	B	T_1	T_1	Y
0	0	OFF	OFF	0
0	1	OFF	ON	1
1	0	ON	OFF	1
1	1	ON	ON	1



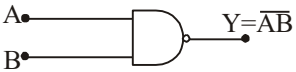
Diode Circuit Diagram :

A	B	D_1	D_2	Y
0	0	ON	ON	0
0	1	ON	OFF	0
1	0	OFF	ON	0
1	1	OFF	OFF	1



NAND GATE :

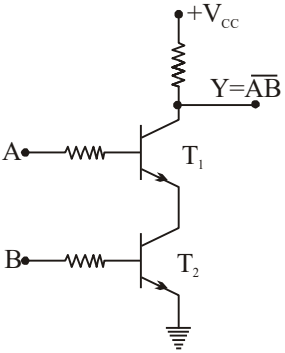
Symbol and Truth Table



Inputs		Output
A	B	$Y = \overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

Transistor Circuit :

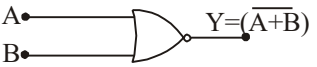
A	B	T_1	T_1	Y
0	0	OFF	OFF	1
0	1	OFF	ON	1
1	0	ON	OFF	1
1	1	ON	ON	0



NOR GATE :

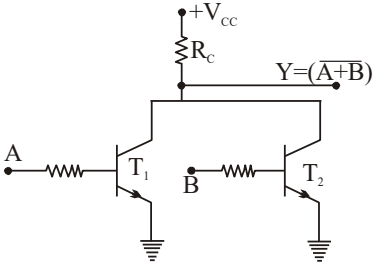
Symbol and Truth Table

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



Transistor Circuit :

A	B	T_1	T_1	Y
0	0	OFF	OFF	1
0	1	OFF	ON	0
1	0	ON	OFF	0
1	1	ON	ON	0



EXOR GATE :

It is also called "stair case switch".

Symbol and Truth Table

Inputs		Output
A	**B**	**$Y=A\oplus B$**
0	0	0
0	1	1
1	0	1
1	1	0

Boolean function of 2-input EXOR operation.

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

Remember :

It acts like an "odd number of 1's detector in the input."

It is mostly used in " parity generation and detection."

- ***In EXOR Operation :***

  For BUFFER CIRCUIT $\Rightarrow$ Logic '0'

  For INVERSION CIRCUIT $\Rightarrow$ Logic '1'

Note : $A \oplus A \oplus A \oplus$ .... *upto n terms = 0, when n = even*

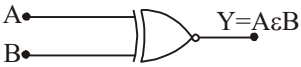
$=$ A, when n = odd

- ***EXNOR GATE :***

  It acts like an "even number of 1's detector."

  Symbol and Truth Table :

Inputs		Output
A	**B**	**Y =AεB**
0	0	1
0	1	0
1	0	0
1	1	1



Boolean function of 2-input EXNOR Operation :

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

Note : $A \odot A \odot A \odot$ .....upto n terms = 1, when n = even

$=$ A, when n = odd

Remember :

- ***In EXNOR Operation :***

  For BUFFER CIRCUIT $\Rightarrow$ Logic '1'.

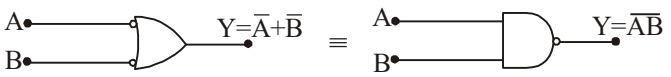
  For INVERSION CIRCUIT $\Rightarrow$ Logic '0'.

  $\overline{A} \oplus B = A \odot B$ and $A \oplus \overline{B} = A \odot B$

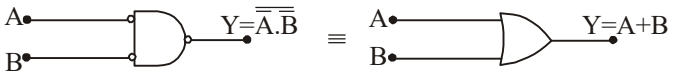
  $\overline{A} \odot B = A \oplus B$ and $A \odot \overline{B} = A \oplus B$

- ***ALTERNATIVE SYMBOLS OF GATES :***

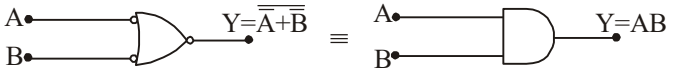
  Bubbled – OR Gate ≡ NAND Gate :



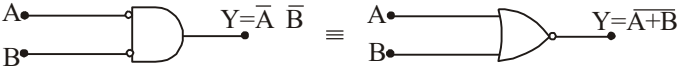
Bubbled – NAND Gate ≡ OR Gate :



Bubbled – NOR Gate ≡ AND Gate :



Bubbled – AND Gate ≡ NOR Gate :



NAND and NOR Gate as Universal :

Logic Gates	No. of NAND Gate Required	No. of NOR Gate Required
NOT	1	1
AND	2	3
OR	3	2
EX OR	4	5
EX NOR	5	4

Note : *AND-OR Logic ≡ NAND-NAND Logic ⇒ SOP*

OR-NAND logic ≡ NOR-NOR logic ⇒ POS

3. BOOLEAN ALGEBRA AND MINIMIZATION TECHNIQUES

- ***Commutative Law :***

$$A + B = B + A \text{ and } A \cdot B = B \cdot A$$

- ***Associative Law :***

$$\boxed{Y = A \odot B = \overline{A \oplus B} = (\overline{A}\overline{B} + A\overline{B}) = AB + \overline{A}\,\overline{B}}$$

and $A.(B.C) = (A.B)C = ABC$

- ***Distributive Law :***

$$A(B + C) = AB + AC$$

$$\boxed{(A + B)(C + D) = AC + AD + BC + BD}$$

BOOLEAN ALGEBRAIC THEOREMS :

- ***AND-Operation Theorem :***

$$\boxed{A.A = A} \quad \boxed{A.0 = 0}$$

$$\boxed{A.1 = A} \quad \boxed{A.\bar{A} = 0}$$

- ***Involution Theorem :***

$$\boxed{(A')' = \bar{\bar{A}} = A}$$

- ***Or-Operation theorem :***

$$\boxed{A + A = A} \quad \boxed{A + 0 = A}$$

$$\boxed{A + 1 = 1} \quad \boxed{A + \bar{A} = 1}$$

- ***DE Morgan's Theorem :***

$$\overline{(A_1.A_2.A_3.\ldots..A_n)} = \bar{A}_1 + \bar{A}_2 + \bar{A}_3 + \ldots + \bar{A}_n$$

$$\boxed{\overline{(A_1 + A_2 + A_3 + \ldots..+ A_n)} = \bar{A}_1.\bar{A}_2.\bar{A}_3.\ldots..\bar{A}_n}$$

- ***Transposition theorem :***

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

- ***Distribution Theorem :***

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

- ***Consensus Theorem :***

Used to eliminate redudant term.

It is applicable only when if a boolean function,

(i) Contains 3-variables

(ii) Each variable is used 2-times

(iii) Only one variable is in complemented or uncomplemented form.

BOOLEAN ALGEBRAIC THEOREMS :

Theorem No.	Theorem
1.	$(A + B).(A + \overline{B}) = A$
2.	$AB + \overline{A}C = (A + C)(\overline{A} + B)$
3.	$(A + B)(\overline{A} + C) = AC + \overline{A}B$
4.	$AB + \overline{A}C + BC = AB + \overline{A}C$
5.	$(A + B)(\overline{A} + C)(B + C) = (A + B)(\overline{A} + C)$
6.	$A.B.C = \overline{A} + \overline{B} + \overline{C} + ...$
7.	$A + B + C + ... = \overline{A}.\overline{B}.\overline{C}. ...$

DUALITY THEOREM :

"Dual expression" is equivalent to write a negative logic of the given boolean relation. For this we,

(a) Changes each OR sign by an NAND sign and vice-versa.

(b) Complement any '0' or '1' appearing in expression.

(c) Keep literals as it is.

Note : *For 1st-times Dual, it is called "Self Dual Expression."*

With N-variables, maximum possible distinct logic functions = 2^{2^n}.

For N-variables, maximum possible Self-Dual Function

$$\boxed{= (2)^{2^{n-1}} = 2^{(2^n/2)}}$$

COMPLEMENTARY THEOREM :

For obtaining complement expression we :

(i) Complement any '0' or '1' appearing in expression.

(ii) Changes each OR sign by AND sign and vice-versa.

(iii) Complement the individual literals.

CANONICAL FORM :

All the terms contains each literal.

$$\boxed{F(A,B,C) = \overline{A}BC + ABC + \overline{A}B\overline{C}}$$

STANDARD FORM :

When each term do not have each literals.

$$\boxed{F(A,B,C) = \bar{A} + BC + A\bar{B}C}$$

Note : *A binary-variable is called "LITERALS."*

MINTERMS AND MAXTERMS :

n-binary variables have 2^n possible combinations and each of these possible combination is called "Minterm or Standard Product".

"Maxterm" is the complement of corresponding "Minterm" i.e., $M = \bar{m}$

In an n-variable Karnaugh-map there are 2^n cells.

COMPLETE SIMPLIFICATION RULES :

Construct the K-map and place 1's in those cells corresponding to the 1's in the truth table. Place 0's in the other cells.

Next, look for those 1's which are adjacent to only one other 1. Loop any pair containing such a 1.

Examine the map for adjacent 1's and loop those 1's which are not adjacent to any other 1's. These are called isolated 1's.

Form the OR sum of all the terms generated by each loop.

Loop any octet even if it contains some 1's that have already been looped.

Loop any pairs necessary to include any 1's that have not yet been looped, making sure to use the minimum number of loops.

Loop any quad that contains one or more 1's which have not already been looped, making sure to use the minimum number of loops.

4. DIGITAL NUMBER REPRESENTATION

In unsigned magnitude representation with 'n' bits, the possible integer values are $[0 - (2^n - 1)]$.

Extra bit $\rightarrow$ Sign bit $\rightarrow$ MSB

(i) If MSB = 0 $\rightarrow$ + ve number

(ii) If MSB = 1 $\rightarrow$ –ve number

In a sign magnitude representation the range of number

$$\boxed{-(2^{n-1} - 1) \text{ to } +(2^{n-1} - 1)}$$

In 1's complement representation, the +ve number are represented similar to +ve number in sign magnitude, but for representing (–)ve number, first write +ve number and then take 1's complement of that.

- ***Range of 1's complement number :***

$$\boxed{-(2^{n-1}-1) \text{ to } +(2^{n-1}-1)}$$

- ***Range of 2's complement representation number :***

$$\boxed{-2^{n-1} \text{ to } +(2^{n-1}-1)}$$

- ***COMBINATIONAL CIRCUITS :***

Output does not depend on previous value of input.

No feedback is required.

It consists of input variables, logic gate and output variables.

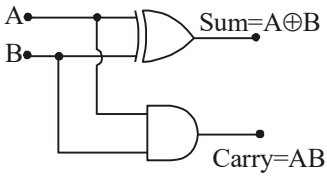
No memory is required.

- ***HALF ADDER (H.A.)***

A logic circuit for the addition of two one-bit numbers is referred to as an "Half Adder (H.A)."

Symbol and Truth Table :

Inputs		Output	
A	**B**	**Sum(S)**	**Carry(C)**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Logical Expression

Sum, $S = \bar{A}B + A\bar{B} = A \oplus B$

Carry, $\boxed{C = AB}$

Remember :

Total number of NAND-gates required to implement half adder = 5.

Total number of NOR-gates required to implement half adder = 5.

FULL ADDER (F.A.) :

It performs the arithmetic sum of the three input bits i.e., addend bit, augend bit and carry bit.

Logical Expression :

Sum, $S = A \oplus B \oplus C$

Carry, $\boxed{C = AB + BC + CA = AB + C(A \oplus B)}$

HALF SUBTRACTOR (H.S)

Logical Expression :

Difference, $\boxed{D = \bar{A}B + A\bar{B} = A \oplus B}$

Borrow, $\boxed{B = \bar{A}B}$

FULL SUBTRACTOR (F.S)

It is a circuit which performs a subtraction between two bits taking into account that a '1' may have been borrowed by a lower significant stage.

Logical Expression :

Difference, $\boxed{D = A \oplus B \oplus C}$

Borrow, $\boxed{B = \bar{A}B}$

Remember :

A F.A. can be implemented with two H.A. and one OR-gate.

Number of NAND/NOR gates required to implement a F.A is equal to '9'.

Total number of NAND/NOR gates required to implement the H.S is equals to '5'.

A F.S. can be implemented with two H.S. and one OR gate.

Number of NAND/NOR gates required to implement the F.S is equals to '9'.

In parallel adder n F.A. or {(n –1) F.A. and 1 H.A.} or {(2n – 1) H.A. and (n – 1) Or-gate} are required to add two n bit numbers.

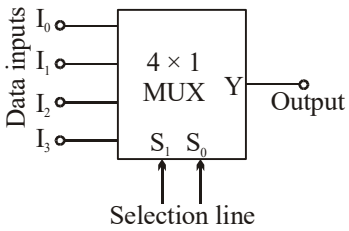
5. MULTIPLEXERS (MUX)

It selects binary information from one of many input lines and directs it to a single output line.

The selection of a particulars input line is controlled by a set of selection lines.

There are 2^n input lines where 'n' is the select line.

4 : 1 MUX :

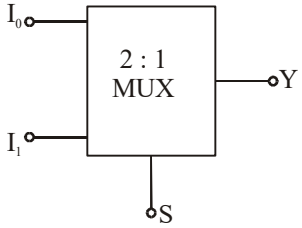


S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$Y = OUTPUT = \bar{S}_1\bar{S}_0I_0 + \bar{S}_1S_0I_1 + S_1\bar{S}_0I_2 + S_1S_0I_3$$

Note : *MUX contains AND gate followed by an OR-gate.*

2 : 1 MUX :



$$Y = \bar{S}I_0 + SI_1$$

HIGHER ORDER MUX USING LOWER ORDER MUX :

Given MUX	To be Implemented MUX	Required No. of MUX
2 : 1	4 : 1	3
4 : 1	16 : 1	4 + 1 = 5
4 : 1	64 : 1	16 + 4 + 1 = 21
8 : 1	64 : 1	8 + 1 = 9
8 : 1	256 : 1	32 + 4 + 1 = 37