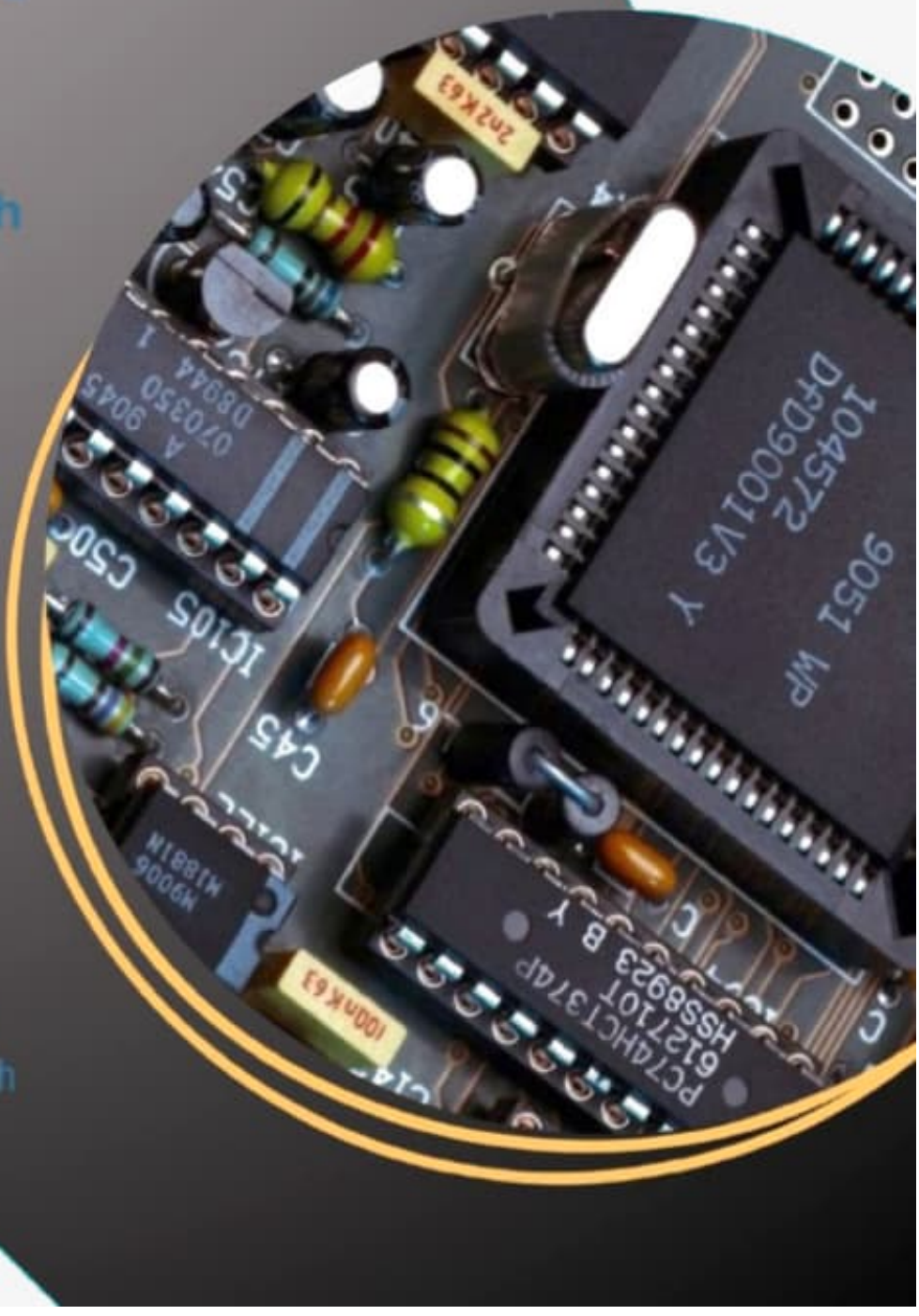
C-21

Digital ELECTRONICS

2024

Prepared by
Abdul Muqsith



Published by
Abdul Muqsith

Scanned with OKEN Scanner

Number System

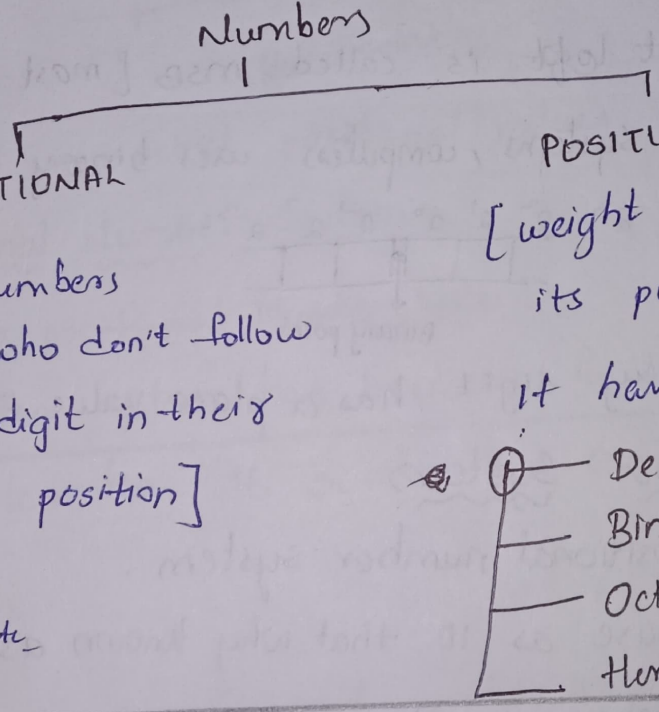
① Computers / digital systems has only 0 & 1 to represent various alpha-numeric characters.

② in digital systems 18 is represented as 10010 in binary

③ as digits increases binary number also increases tremendously to decrease its length new number system arises i.e Octal, Hexa-decimal, Binary coded Decimal, decimal etc.

Number system.

Number = representation of Quantity.



Base

~~The number of symbols can be used~~

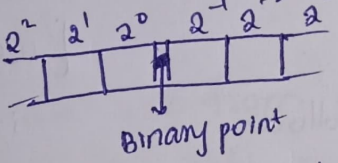
① The no. of symbols that can be used in a number system is known as base or radix.

ex: in binary we use only 0 & 1 so base of binary is 2

ex①. $(01200110)_2$

① Binary Number System

* It is a positioned value system.
* The number system with base 2 [only have 0 & 1 to represent any thing]
* It uses only two symbols i.e 0 and 1.
* The symbol 0 and 1 are called bit instead of digits
* Using 0 & 1 we can express or represent any number / character. ex: 10110
* In 10011, the 1 at right is called LSB [least significant Bit]
* the 1 at left is called MSB [most significant Bit].
* All digital systems, computers uses binary system.



Each binary digit has a place value or weight.

② Decimal number system

* It is a positional number system.
* It has base as 10 that why known as decimal number sy
* Decimal number can have 10 different symbols i.e 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9.
* They can be expressed in units, tens, hundreds, thousands so on
* 5678.9 can be represented as

$$5\times10^3 + 6\times10^2 + 7\times10^1 + 8\times10^0 + 9\times10^{-1}$$

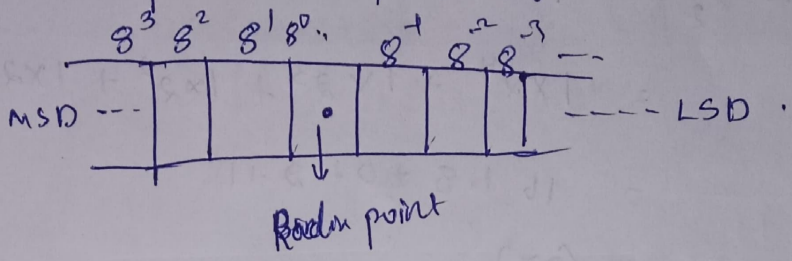
$$5000 + 600 + 70 + 8 + 0.9 = 5678.9$$

* 5 in left has greatest weight so MSB or MSD
* 9 in right has lowest weight so LSB or LSD digit.

Scanned with OKEN Scanner

Octal Number System

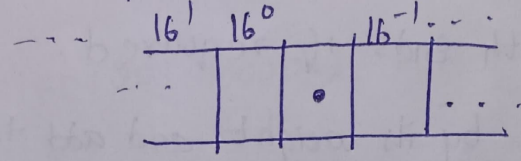
* It is a positional N·S.
* The number system with base 8 is called octal N·S
* This system has eight symbols i.e 0, 1, 2, 3, 4, 5, 6 and 7
  Ex : 1703
* Used in many microcomputers & computers for Entering data as 3-bit binary data.
* weigh is expressed as power of 8.



octal positional values

Hexa-decimal Number System

* It is a positional Number system.
* The number system has base as 16 ; that why called as Hexa-d
* This system has 16 symbols i.e 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and
  A [10], B [11], C [12], D [13],
  E [14], F [15].
* It is a combination of number & alphabets as 2ABD3. etc.
* widely used for representing microcomputer memory location addresses.
* weight is represented as power 16



NUMBER SYSTEMS CONVERSIONS

(1) BINARY TO DECIMAL.

① Write down all bits in a row

② Multiply each bit with its positional weight

③ ADD All ~~bits~~ of them.

(i) Convert binary no' $(11011)_2$ into decimal.

(A) $(11011)_2 = 1^4\ 1^3\ 0^2\ 1^1\ 1^0$

$= 1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0.$

$= 16 + 8 + 0 + 2 + 1$

$= (27)_{10}.$

(ii) $(1101.101)_2$ to decimal.

$N = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

$= 8 + 4 + 0 + 1 + 0.5 + 0.125$

$= (13.625)_{10}.$

(2) BINARY TO OCTAL.

① Arrange the bits in a set of three bit. from right to left & left to right after decimal point

② Add 0's at both ends if required.

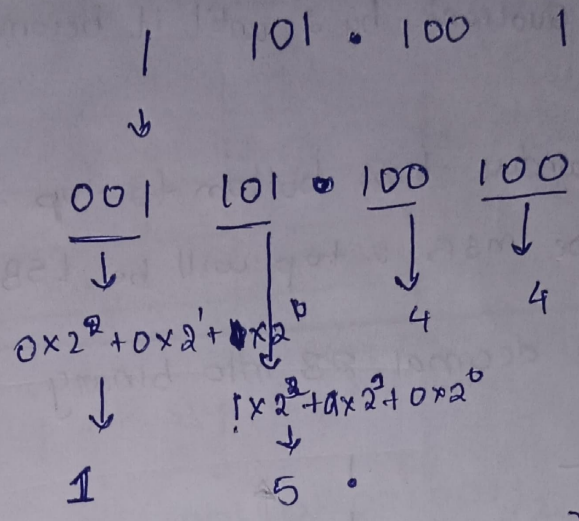
③ multiply each bit by its weight and add them do this for all parts separately.

④ Then write its octal Equivalent

Scanned with OKEN Scanner

Convert 1101.1001 into octal.

Sol^n Step (1)

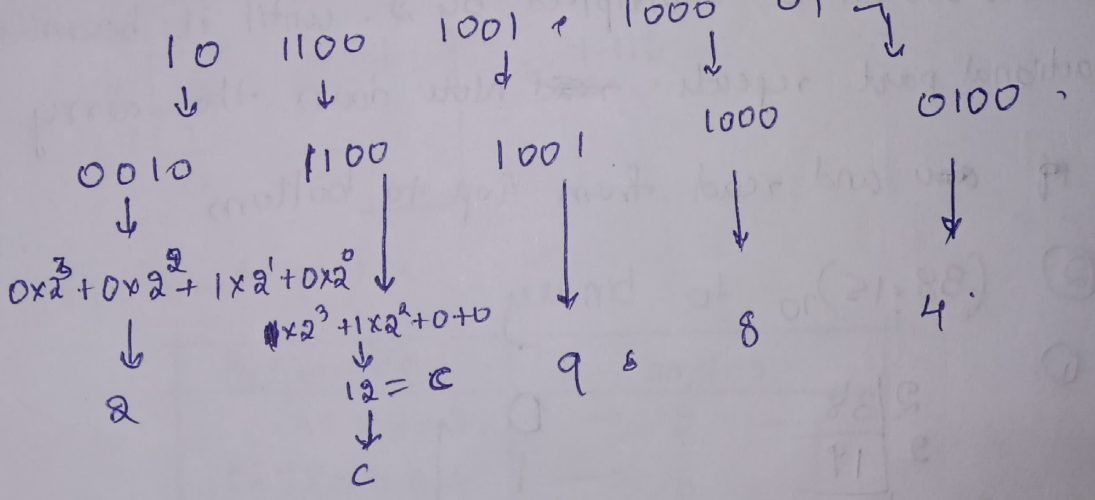


$\therefore 1101.1001 = (15.44)_8$.

Binary to Hexadecimal

* Do same procedure but now create 4 pairs.

convert $(1011001001.100001)_2$ into Hexadecimal.



$\therefore (1011001001.100001)_2 = (2C9.84)_{16}$.

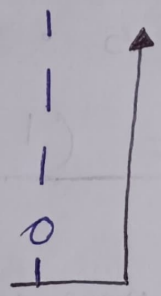
1. DECIMAL TO OTHERS

DECIMAL TO BINARY

(1) Divide by 2
(2) Note down remainder after each division.
(3) Divide the Quotient by 2 until it becomes zero & note the remainder.
(4) write remainders from bottom to top such that bottom will be MSB & top will be LSB.

(1) Convert the decimal 23 into binary.

2	23
2	11
2	5
2	2
2	1
	0



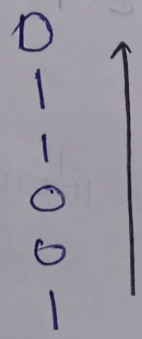
(Down to up) $(23)_{10} = (10111)_2$.

* If the decimal contains fraction then the fractional part should be multiplied by 2. until it becomes 0 or fractional part repeats. Note down the carry part and read from top to bottom

Example (2) $(38.15)_{10}$ to binary

Step (1)

2	38
2	19
2	9
2	4
2	2
2	1
	0



$\Rightarrow 100110$

Step (2) - fractional part

×2	carries
0.15 × 2 = 0.3	→ 0
0.3 × 2 = 0.6	→ 0
0.6 × 2 = 1.2 take 0.2	→ 1
0.2 × 2 = 0.4	→ 0
0.4 × 2 = 0.8	→ 0
0.8 × 2 = 1.6	→ 1
	continues

0 0 1 0 0 1 ↓ = 0.001001

So add step 1 & step 2 results

$(38.15)_{10} = (100110.001001)_2$.

(2) Decimal to Octal

* Same process but divide by 8 & for fractional multiply by 8

convert $(974.35)_{10}$ in octal.

8 | 974 — 6
8 | 121 — 1
8 | 15 — 7
8 | 1 — 1
0

= 1716

fractional part

fractional × 8	carries
0.35 × 8 = 2.80	→ 2
0.8 × 8 = 6.4	→ 6
0.4 × 8 = 3.2	→ 3
0.2 × 8 = 1.6	→ 1
0.6 × 8 = 4.8	→ 4

= 0.26314.

$(974.35)_{10} = (1716.26314)_8$

Scanned with OKEN Scanner

(3) DECIMAL TO HEXADECIMAL CONVERSION

same procedure but divide by 16 and for fractional part multiply by 16.

Convert $(28.6875)_{10}$ into hexadecimal.

16 | 28
16 | 1
0

$-12 = C$
-1

C ↑
1

$= 1C$

$0.6875 \times 16 = 11.0 \rightarrow B$.

$\therefore (28.6875)_{10} = 1C.B$

(1) Octal to BINARY

(1) octal can be converted to binary by writing each bit with its 3-bit equalent binary.

(2nd) convert each digit into binary by using division method

Convert $(23.54)_8$ into binary

Soln

2 3 . 5 4
↓ ↓ ↓ ↓
010 011 101 100

$(23.54)_8 = (10011101100)_2$

Scanned with OKEN Scanner

(2) Octal to decimal

Similar to conversion of binary to decimal but only difference is that the power of 8 is used instead of 2.

Convert $(567)_8$ into decimal

$$5\times8^2+6\times8^1+7\times8^0$$

$$= 320+48+7$$

$$= (375)_{10}.$$

(ii) 22.78

$$= 2\times8^1+2\times8^0 . 7\times8^{-1}+7\times8^{-2}.$$

$$= 16+2+0.875+0.0625$$

$$= (18.9375)_{10}.$$

(3) Octal to Hexa-decimal.

Step - (1) do first octal to binary

Step - (2) Then Binary to hexadecimal

Convert $(615)_8$ to hexadecimal

Step - (1)

6	1	5
↓	↓	↓
110	001	101

Binary number = $(110001101)_2$

Step - (2)

0001	1000	1101
↓	↓	↓
$0\times2^3+0\times2^2+0\times2^1+1\times2^0$		13
↓	↓	↓
1	8	D

∴ Hexadecimal number = $(18D)_{16}$.

Scanned with OKEN Scanner

(1) Hexadecimal to BINARY

Convert $(2BD)_{16}$ to binary.

① convert each bit into 4 bit equivalent binary by division method.

2	B	D
↓	↓	↓
	11	13
↓	↓	↓
0010	1011	1101

$(2BD)_{16} = (1010111101)_2$.

2 | 13
2 | 6 — 1
2 | 3 — 0
2 | 1 — 1
0 — 1

2 | 11
2 | 5 — 1
2 | 2 — 1
2 | 1 — 0
0 — 1

2 | 2
2 | 1 — 0
0 — 1

(2) Hexadecimal to decimal

* same as conversion of binary/octal to decimal.

$(5D.98)_{16}$ D=13

$= 5\times16^1 + 13\times16^0 + 9\times16^{-1} + 8\times16^{-2}$

$= 80 + 13 + 9/16 + 8/256$

$= 80 + 13 + 0.5625 + 0.03125$

$= 93.59375$

$(5D.98)_{16} = (93.59375)_{10}$.

Scanned with OKEN Scanner

(2) Hexadecimal to Octal number System

Same as conversion of octal to Hexadecimal

Convert $(B4)_{16}$ into Hexadecimal.

Step(1) : Convert to Binary.

B 4

↓ ↓

1011 0100.

$\therefore$ Binary $= (10110100)_2$.

2 | 11 — 1
2 | 5 — 1
2 | 2 — 0
2 | 1 — 1
0

2 | 4 — 0
2 | 2 — 0
2 | 1 — 1
0

Step(2) : Binary to ~~Hexa~~ octal :

010 110 100

↓ ↓ ↓

$0\times2^2+1\times2^1+0\times2^0$ ↓ ↓

↓

2 6 4.

$\therefore (BA)_{13} = (264)_8$.

Scanned with OKEN Scanner

12

14

BINARY ARITHMETIC

* Digital systems deals with binary numbers hence the arithmetic operation is also performed logically.

BINARY ADDITION

A+B		
BINARY	Sum	Carries
0+0	0	0
0+1	1	0
1+0	1	0
1+1	0	1

* At last 1+1 gives 2 but in binary it is 10 in which LSB '0' given as sum and MSB '1' is given as carry.

* Add 1010 & 1011 & verify decimal.

Binary Addition.

$$\begin{array}{r} {\scriptstyle 1}\ 1\ 0\ 1\ 0 \\ +\ 1\ 0\ 1\ 1 \\ \hline 1\ 0\ 1\ 0\ 1 \end{array}$$

verify → 1010 → 10
1011 → 11
21

* Add 1011.011 & 110.1

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1} \\ 1011.011 \\ +\ 110.100 \\ \hline 10001.111 \end{array}$$

11.375
6.5
17.875

BINARY SUBSTRACTION

A-B	Difference	Barrow
0-0	0	0
0-1	1	1
1-0	1	0
1-1	0	0

Perform substraction 10101 from 11011

```
  ①
  1 1 0 1 1   =(27)10
- 1 0 1 0 1   =(21)10
-------------
  0 0 1 1 0   =06)10
```

BINARY MULTIPICATION

* Easier than decimal multiplication as has only 0 & 1 but same

A x B = 0

0 x 0 = 0

0 x 1 = 0

1 x 0 = 0

1 x 1 = 1

multiply 1011 by 101

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

Verification

11 x 5

= 55

BINARY complement

Used to represent negative numbers.

They are basically two complements i.e 1's complement & 2's complement.

1's complement

It is the complement of each bit

The complement of 0 is 1 and complement of 1 is 0.

Ex: 1110101 [Find 1's complement]

1's of 1110101 is 0001010

2's complement

If we add '1' to 1's complement - then it is called 1's co

2's complement = 1's complement + 1

Find 2's complement of ~~1010011~~ $(1001)_2$

number = 1001

1s = 0110

2s = 0110
+ 1

0111

I SUBSTRACTION OF smaller number from larger using complement method.

* find 1's complement of given smaller number
* Add the 1's complement to larger number
* remove carry & Add it to the result known as end-around carry.

Ex: 101011 − 111001 using 1's method

```
101011   <- 1's
111001     1's of 101011 is 010100
------
0
1
111 001
010 100
------
1001101
  +1   <- Add end carry
------
 1110  <- result
------
```

11 Substraction of larger number from smaller number using 1's complement method.

* Find 1's of larger number
* Add 1's complement larger to smaller
* Ther result is negative & in 1's form.
* Again do compliment for final answer & put ⊖ sign.

Substract 1100 from 1000 using 1's complement

* 1's complement of larger number 1100 is 0011
*

```
  0011
+ 1000
------
  1011   ← addition of 1's of larger to smaller
```

* True result = 0100 = −0100.

2's complement method.

Substraction of smaller from larger.

1. Find 2's of smaller
2. Add it to larger
3. Discard the carry.

Ex 101.11 from 1100.1 by 2's C.M

smaller = 0101.11 (Add precise zero)

1's of smaller = 1010.00

2's = 1010.01

```
  1100.10
+ 1010.01
---------
 10110.11
```

result = 110.11.

Subtraction of larger from smaller using 2s cm.

* Find 2's of larger number
* Add 2s to smaller number
* result is negative & in 2s complement form.
* To get actual result take 2's complement the result.

Ex: 1101 − 1000

* 1's of 1000 is 0111
  2's of 1000 is 0011

Smaller num is 1000
+ 0011
1011

1011 is in 2s form again write 2s form of 1011

2s of result will be 1011 → 0100+1 = 0101

∴ final answer = 0101

Boolean Algebra: It is defined with set of Elements [variables] (binary) a set of operators & a number of postulates.

Boolean Algebra

* Boolean Algebra invented by George Boolean
* in boolean algebra every variable can have only 2 values "TRUE/FALSE" 1 / 0
* the digital signals are discrete in nature and can only assume 0 or 1.
* The TRUE / FALSE logic of Boolean algebra is the basis of all digital systems like computer / calculator, etc.
* Binary variables can be represented by A, B, C --- & their value can have only 0 or 1.

Postulates in BOOLEAN ALGERA

* Postulates: Anything which is not proved but assumed to be true
* Variables [binary] or [Elements] are actually input signals of the digital circuit.
* variables represented by letters. for analysing & designing of digital systems.
* Three basic types of operators are available i.e AND (.), OR (+). and NOT (-"bar").

Scanned with OKEN Scanner

6 Postulates of Boolean Algebra

1. The result of boolean Algebra is either 0 or 1.
   for example, if A=0 & B=1, then A+B=1, A·B=0.
2. They are two Elements 0 & 1
   0 is added to binary variable A [result = A] ie A+0=A
   1 is ~~added~~ multiplied to binary variable A [result = A] ie A·1=A
3. Binary elements follows commutative law
   $A+B = B+A$
   $A \cdot B = B \cdot A$
4. Binary elements follows distributive law
   $A+(B \cdot C) = (A+B)(A+C)$.
   $A \cdot (B+C) = A \cdot B + A \cdot C$.
5. The complement of A is such that if A=1 then $\bar{A}=0$
   if A=0 then $\bar{A}=1$ i.e $A+\bar{A}=1$ & $A \cdot \bar{A}=0$
6. Their are atleast two elements [Boolean] A & B, where
   $A, B \in C$ (The Elements of) such that $A \neq B$.

Laws of Boolean Algebra.

1. OR laws
   (i) A+0=A
   (ii) A+A=A
   (iii) A+1=1
   (iv) $A+\bar{A}=1$

   let A=0 then 0+0=0 — let A=1 then 1+0=1
   0+0=0 — 1+1=1
   0+1=1 — 1+1=1
   0+1=1 — 1+0=1

2. AND laws
   (i) A·0=0
   (ii) A·1=A
   (iii) A·A=A
   (iv) $A \cdot \bar{A}=0$.

3. NOT laws
   (i) $\bar{0}=1$
   (ii) $\bar{1}=0$
   (iii) If A=0, then $\bar{A}=1$
   (iv) If A=1, then $\bar{A}=0$
   (v) $\bar{\bar{A}}=A$.

4. commutative laws:
   (i) A+B = B+A
   (ii) A·B = B·A.

5. Associative laws
   (i) A+(B+C) = (A+B)+C
   A·(B·C) = (A·B)·C

6. Distributive laws
   (i) A·(B+C) = A·B + A·C
   (ii) A+(B·C) = (A+B)·(A+C)
   (iii) A+ĀB = A+B.

7. A unitary law
   (i) A+ĀB = A+B
   (ii) A+A·B = A.

8. De-morgan's law
   (i) $\overline{A+B} = \bar{A} \cdot \bar{B}$
   (ii) $\overline{A \cdot B} = \bar{A} + \bar{B}$

Scanned with OKEN Scanner

DEMORGAN'S THEOREM

They are two laws of demorgan

I Theorem - 1

1st theorem states that the complement of sum of ~~two~~ Equal the product of the complements.

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

Proof by truth table

LHS

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

RHS

A	B	$\overline{A}$	$\overline{B}$	Result $\overline{A} \cdot \overline{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

As the result of both LHS & RHS is equal i.e 1,0,0,0 so

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

II Theorem - 2

The second theorem says that the complement of product is equals to the sum of complements.

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

Proof

A	B	$\overline{AB}$	(Result) $\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

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

Result of LHS = RHS ∴ $\overline{AB} = \overline{A} + \overline{B}$

Logic gates : logic gate is defined as the basic logic circuit used to perform logical operation such as AND, OR, NOT etc.

* logic gate is electronic circuit
* Operation of logic gates can be examined by truth table
* It takes ~~one~~ one / more inputs & generates one output

Truth table: Shows all the input-output possibilities of a logic circuit

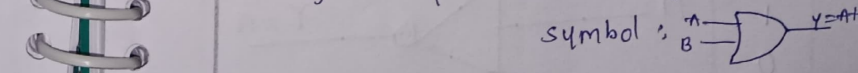
classification of logic gates

- Logic gates
  - Basic logic gate
    - OR gate
    - AND gate
    - NOT gate
  - Universal logic gate
    - NOR gate
    - NAND gate
  - Special logic gate
    - EX-OR Gate
    - EX-NOR Gate

Basic logic gate : They are of three types

(i) OR Gate

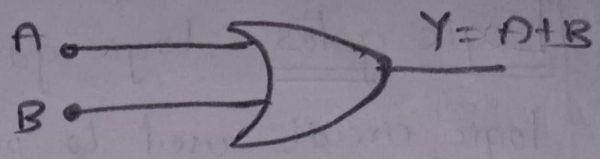
- OR gate performs logical addition.
- It requires two inputs or more (A & B) and gives one output Y
- An OR gate is a gate whose output is ~~high~~ low if any or all of the inputs are ~~high~~ low
- logical expression for OR gate is $Y = A+B$

symbol : 

(26)

Expression $Y = A + B$

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

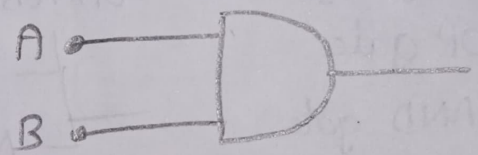


Symbol of logic OR

(2) AND Gate:

- AND Gate performs logical multiplication.
- It is a logic gate which require two/more (A·B) inputs for one output (Y)
- It is output high only if all inputs are high.
- logical Expression for AND Gate is $Y = A \cdot B$.

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

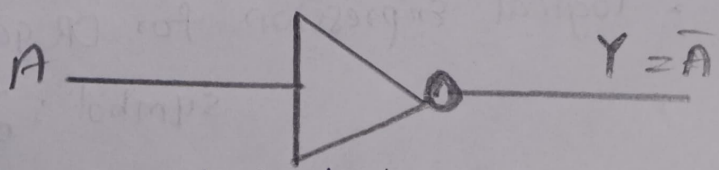


Symbol of logic AND

(3) NOT Gate:

- It requires only one input whose output is always complement of Input also known as inverter
- It shows the output high if input is low & vice versa.
- Expression is $Y = \bar{A}$

Inputs	Outputs
A	$Y = \bar{A}$
0	1
1	0



Symbol of logic NOT

Scanned with OKEN Scanner

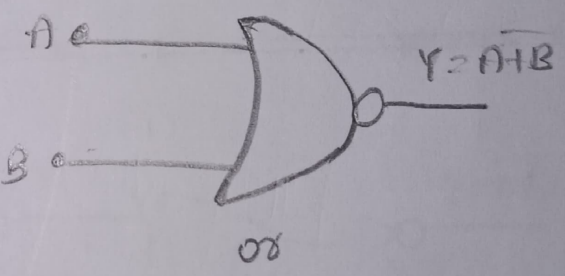
Universal logic Gates

* These logic gates formed by combination of two basic logic gates
* They are of two types

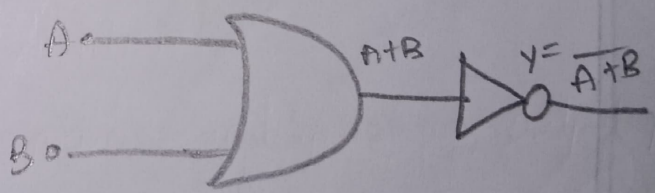
(1) NOR Gate:

- This gate is the combination of ~~two~~ NOT & OR gate.
- This NOR gate requires two / more inputs for one output
- The output of NOR gate is high only when all the inputs are low. If any one or both inputs are high then output will be low.
- logical Expression is $Y = \overline{A+B}$
- output = complement of OR.

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



or

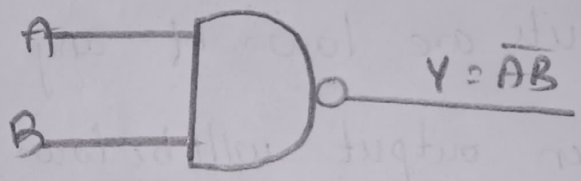


Scanned with OKEN Scanner

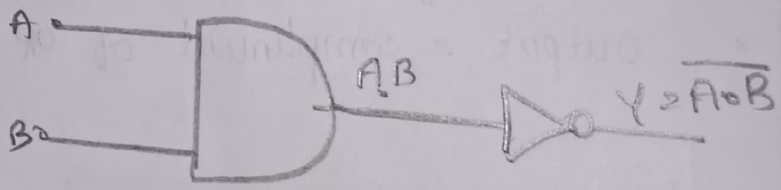
(2) NAND Gate

- This gate is formed by the combination of NOT & AND gate
- This requires two or more inputs but one output
- The output of NAND gate is low when all the inputs are high. If anyone or all inputs are low, then the output of NAND gate will be high.
- logical expression for NAND gate is $Y = \overline{A \cdot B}$

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



or



Ex-NOR (Gate)

* An Ex-NOR gate has two $\frac{1}{2}$more inputs but only one output (Y)
* The output of Ex-NOR (Gate) is HIGH if even number of inputs are 1.
* The output of EX-NOR gate is LOW if odd no. of inputs are 1.
* logical Expression is $Y = \overline{A \oplus B} = AB + \bar{A}\bar{B}$

Derived as $A \oplus B = \bar{A}B + A\bar{B}$

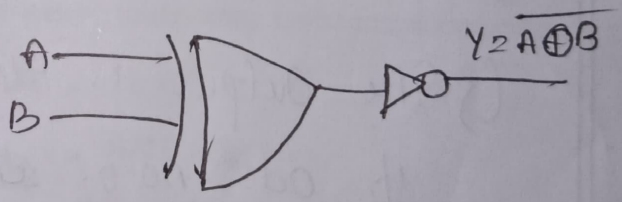
$= \bar{A}B + A\bar{B}$

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

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

Truth Table

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

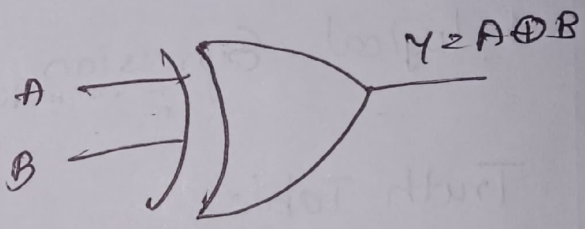


→ Used as bit comparator as if inputs both inputs are 0 or 1 then only the result is HIGH.

EX-OR gate

- Ex-OR can be called as Exclusive OR gate
- Ex-OR gate has two/more inputs but one output
- The output of Ex OR gate is high if two inputs are different
- The output of Ex OR gate is low if all the inputs are either low or high.

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

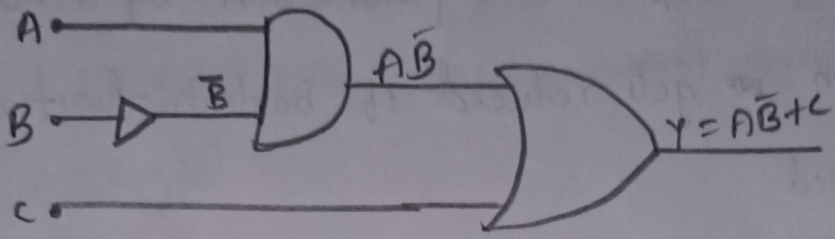


Input		Outputs
A	B	$Y = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

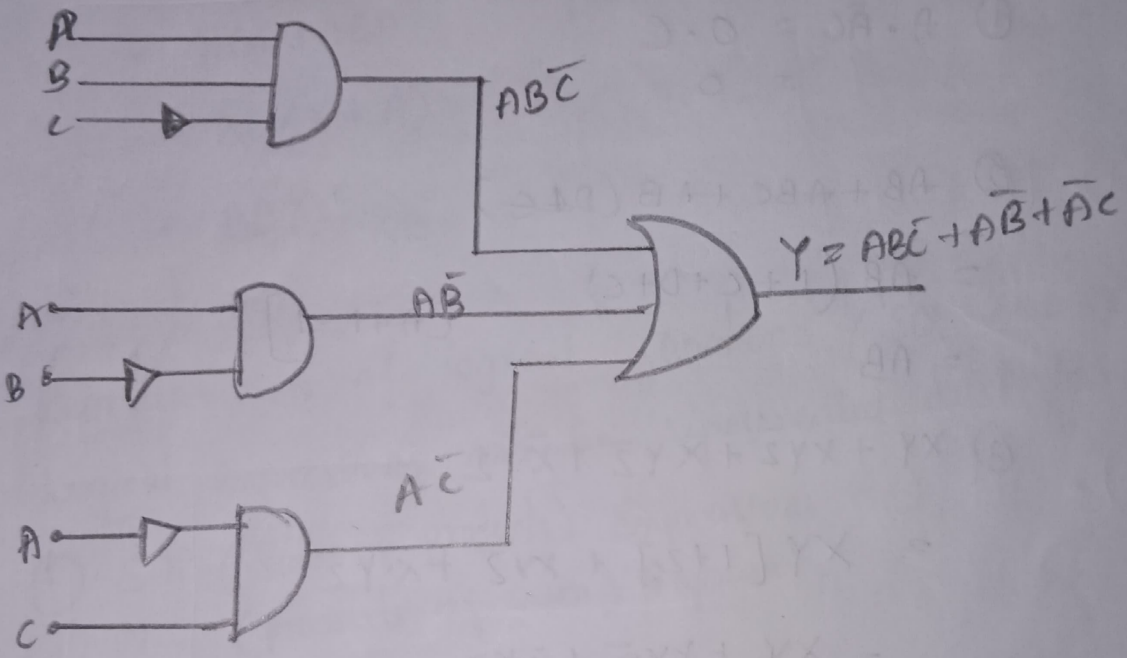
(i) The Output is High if odd no. of input are 1

Realisation using NAND Gate

$Y = A\bar{B} + C$ A, $\bar{A}$ are literals



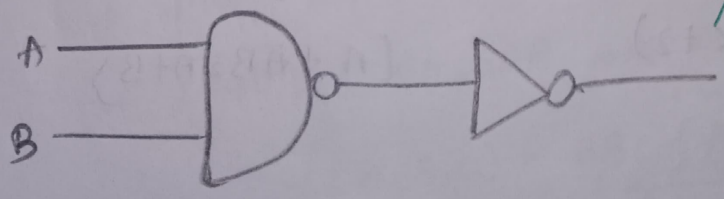
$Y = AB\bar{C} + A\bar{B} + \bar{A}C$



AND GATE USING NAND GATE

Complement of NAND GATE is AND GATE.

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



Simplification of Boolean Expression

* The no. of gates required & the number of input terminals for the gate for the implementation of a boolean function gets reduced if boolean function is simplified
* ∴ Reduce complexity of hardware.

Examples

① $A \cdot \bar{A}C = 0 \cdot C$
$= 0.$

② $AB + ABC + AB(D\bar{E})$
$= AB(1 + C + D + \bar{E})$ $[A+1=1]$
$= AB$

③ $XY + XYZ + XY\bar{Z} + \bar{X}YZ$
$= XY[1+Z] + XY\bar{Z} + \bar{X}YZ$
$= XY + XY\bar{Z} + \bar{X}YZ$ $[A+1=1]$
$= XY(1+\bar{Z}) + \bar{X}YZ$
$= XY + \bar{X}YZ$ $[A+1=1]$
$= Y(X + \bar{X}Z)$
$= Y(X+Z)$. $[A + \bar{A}B = A+B]$

④ $\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC$

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

$\bar{A}\bar{C} + \bar{A}BC \quad (A+\bar{A}=1)$

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

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

⑤ $\bar{A}B\bar{C}D + \bar{A}BCD + ABD.$

$= \bar{A}BD(\bar{C}+C) + ABD$

$= \bar{A}BD + ABD$

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

$= BD(1)$

$= BD.$

Representation of logical functions (SOP AND POS)

Logical Expressions can be represented in 2 forms

(i) SOP (sum of products) Expressions

(ii) POS (products of sum) Expressions.

SOP Expressions

- SOP Expressions contains two or more AND functions and ORed functions together

② • These products are called as minterms

③ • Examples of SOP are as follows

(i) $f(A,B,C) = ABC + \bar{A}\bar{B}C$ (sum; product terms)

① * definition : SOP expressions is a set of AND (product) terms that are summed (ORed) together

POS Expression

① [POS Expressions are the set of OR (sum) terms which are ANDed (product) together]

X. The product of sums expression contains two or more OR functions & AND together.

② These products are called maxterms.

③ The product of sum is a dual of sum of product

$f(A,B,C)$

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

(Sum terms; product)

Standard form of boolean Expression

① If each product term in SOP form contains all the variables then the SOP form is known as Standard or canonical SOP form.

② If each sum term in POS form contains all the variables then the POS form is known as Standard / canonical POS form.

Converting SOP to canonical form.

① Identify missing varible

② AND (multiply) each product term having missing variables with the ORing (adding) the missing variable with its complement.

③ Expand equation by applying distributive law

Example:

$f = A + AB + ABC$.

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

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

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

$= ABC + A$

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

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

$= m_7 + m_6 + m_5 + m_4 = \Sigma m(4,5,6,7)$

Steps to convert POS to canonical form

(1) Identify the missing variable in each sum term if any

(2) OR each sum term having missing variables with term form by ANDing the variable and its complement.

(3) Expand the terms by applying distributive law & reorder the variable in the sum terms.

(4) reduce the expression by omitting repeated Sum terms in any, because $A \cdot A = A$.

Scanned with OKEN Scanner

Example - 1

Write the boolean Expression of minterms (SOP's) from the table

Inputs			Outputs
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	1	1
1	1	0	0
1	1	1	1

Soln

Step ①

Note down input combinations for output 1

There are such cases 001, 011, 101, and 111

Step ②

write the product (AND) terms for each case.

A	B	C	Product
0	0	1	$\bar{A}\bar{B}C$
0	1	1	$\bar{A}BC$
1	0	1	$A\bar{B}C$
1	1	1	ABC

Step 3

$\therefore$ SOP = $\bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + ABC$.

$= m_1 + m_2 + m_3 + m_7$

The Expression can be further Expressed as, SOP =

Scanned with OKEN Scanner

Use karnaugh map to simplify boolean Expression (upto 4 variable) in SOP-form.

Procedure to simplify SOP

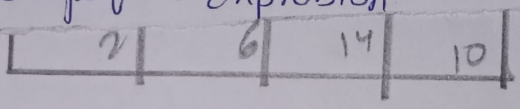
1) Plot the k map and place 1's in those cells corresponding to 1's in the truth table. & others with 0

2) Circle the 1's who are not adjacent to any other.

3) make 8 grouping, 4-grouping, 2-grouping if possible.

4) Combine any pairs necessary to include any 1's that have not been grouped.

5) form the simplified Expression



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

∴ Expression : $Y = B\bar{C} + A\bar{C}D + \bar{A}\bar{B}C\bar{D}$

Use of k-map for POS form.

Same as SOP form but instead of considering 1's we consider 0's output.

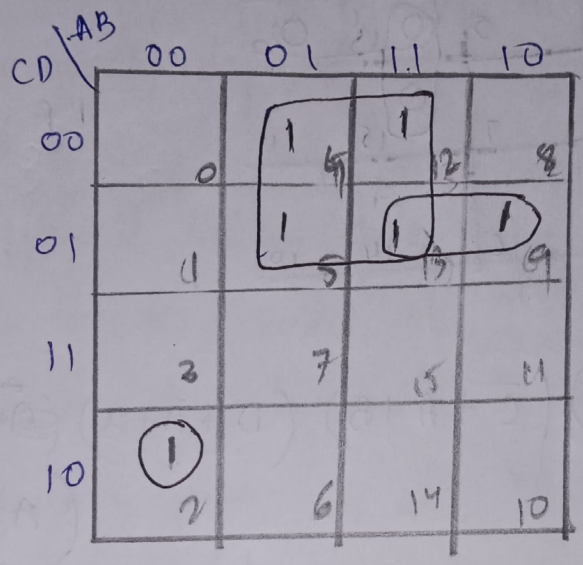
Scanned with OKEN Scanner

Minimize the Expression

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

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

0100	0101	1100	1101	1001	0010
8421	8421	8421	8421	8421	8421
m_4	m_5	m_{12}	m_{13}	m_9	m_2



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

∴ Expression : $Y = B\bar{C} + A\bar{C}D + \bar{A}\bar{B}C\bar{D}$

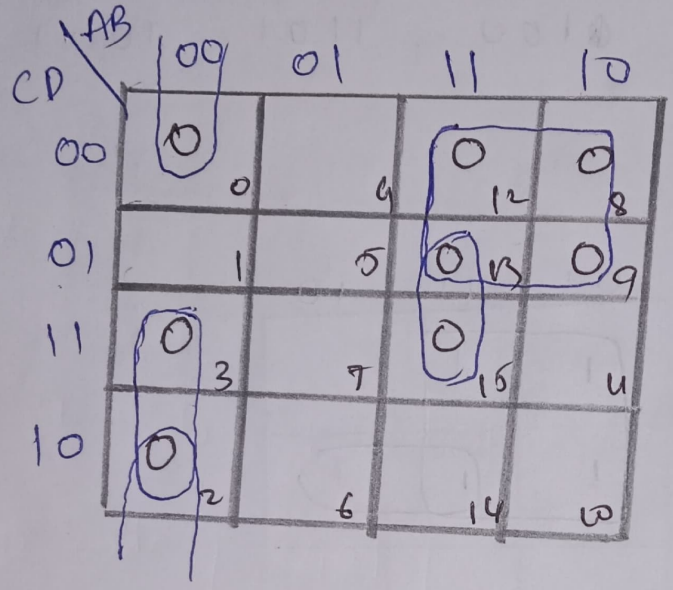
Use of k-map for POS form.

Same as SOP form but instead of ~~with~~ considering 1's we consider 0's output.

Example - 7

Reduce the following Equation using k-maps.

$f(A,B,C,D) = \pi M(0,2,3,8,9,12,13,15)$.



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

$(A+B+D)$

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

Scanned with OKEN Scanner

Binary Codes

The process of converting numerical, alphabets & special characters into binary format is called "coding"

The combination of binary digits that represents numbers, alphabets & symbols called as "binary codes"

Codes are broadly classified into 5 groups.

(i) weighted binary codes
(ii) Non-weighted binary codes
(iii) Error-detecting codes
(iv) Error-correcting codes
(v) Alphanumeric codes.

Weighted binary codes

In weighted codes, each digits position of the number represents a specific weight.

Ex: number 567 in decimal then 5 weights 100, 6 weights 10's & 7 weights 1's. Ex: 8421 code. (BC

Un-weighted binary codes

Unweighted codes are not assigned with any weights to each digit position, each digit position within a number is not assigned fixed value.

Ex: Excess 3 codes, Gray codes.

Scanned with OKEN Scanner

Use of weighted codes

① Used for I/O operations in digital Circuits

② Data manipulation during Arithmetic operation.

③ To represent decimal digits in digital systems

Un-weighted codes

① Used in error detection & correction in communication systems

② Used for generating karnaugh maps.

BCD code [8421]

① BCD - Binary coded decimal

② It is a numerical code in which each digit of a decimal represented by a seperate group of bits.

③ The most common BCD code is 8421 in which each decimal digit is represented by a 4-bit binary number.

④ It is a weighted code.

⑤ To Express any decimal number in 8421 codes simply replace each decimal digit by 4 bit codes.

Scanned with OKEN Scanner

Parity Bit : parity bit is used to detect errors during transmission of binary information from one location to another

Parity Bit is an extra bit added with a binary mess to make number of 1's either even or odd.

The circuit which generates the parity bit in transmi is parity generator & the circuit which ~~receives~~ & checks parity in the receiver is called parity ch

Scanned with OKEN Scanner

Chapter-2 Digital IC logic families

(1) Integrated circuit: Digital IC's are made using several different circuits & production technologies, on a single chip

(2) A Digital logic family: A digital logic family is a collection of different integrated circuits chips that are compatible devices and performs different logic gate functions such as AND, NOT, NOR etc

(3) Compatible :- the output of a devices is the input for another devices

(4) ICs are smaller in size needs less energy consumption
In early decades IC's was on small scale integration (SSI) in which small & less circuits are integrated on a chip but Now from SSI, through MSI (Medium scale integrated) to large scale integrated (LSI) to very large scale integration (VLSI) ~~is achieved~~ to Ultra large scale integration (ULSI) is achieved & The technology is Entering to (GSI) Giant scale integration in which millions of circuits are integrated on a single chip.

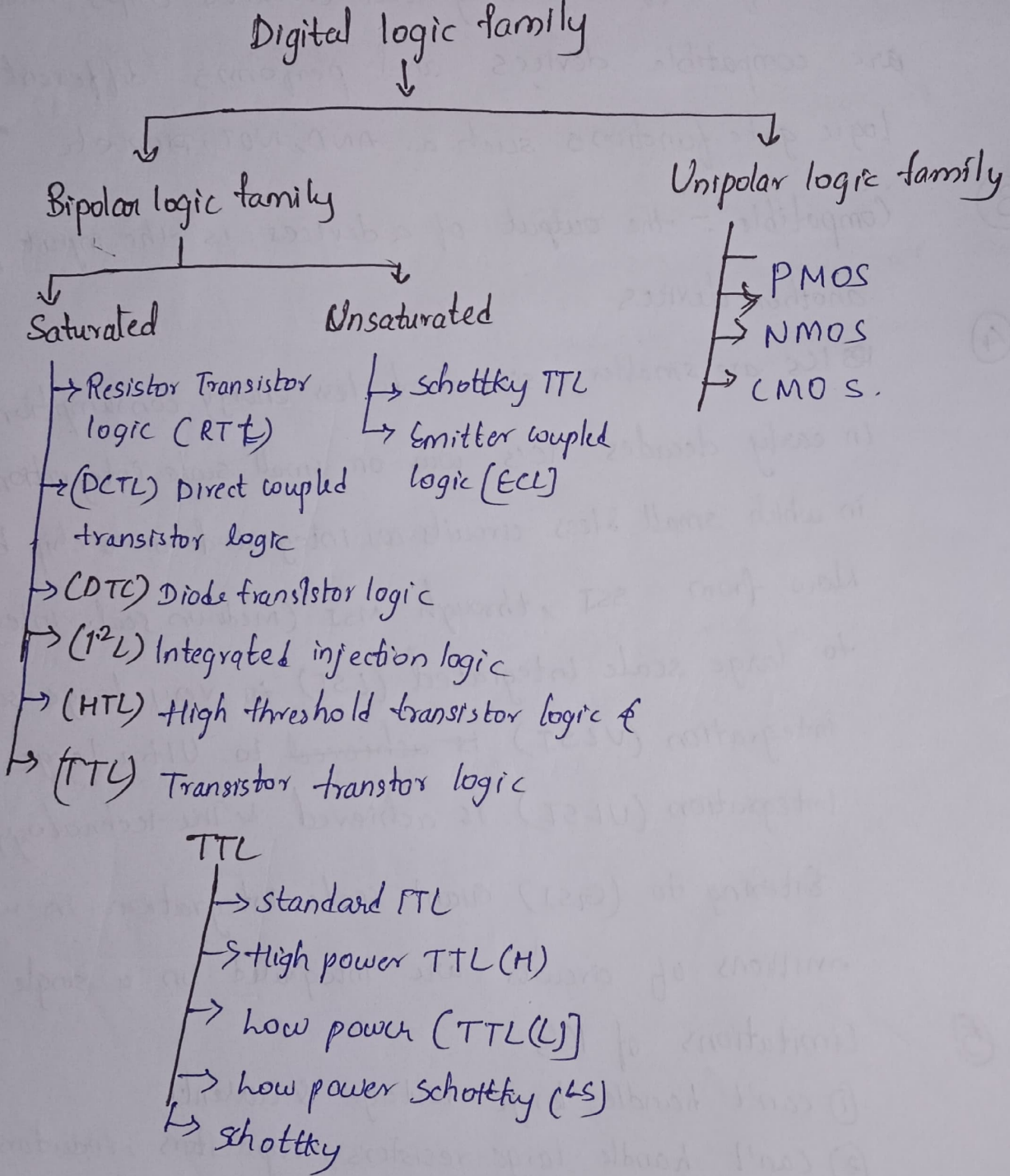
(5) Limitations of IC's
(1) can't handle large voltage & currents
(2) can't handle large resistors, capacitors, inductors & transfo

Scanned with OKEN Scanner

Classification of logic families.

Digital logic family

ICs are designed using bipolar devices or metal oxide semiconductor devices or combination of both



Fabrication: process of manufacture

Characteristics of Digital IC's:

1. Input & Output logic levels
2. Propagation delay
3. Noise margin
4. fan in
5. fan out
6. Power dissipation
7. figure of merit
8. Power supply requirements

(1) Logic level of TTL

The values of voltages or current that are used to represent a LOW (0) state & HIGH (1) state is called logic level.

- In binary logic, 0 & 1 is used to represent this levels

Positive logic system: If higher voltage represents 1 & lower as 0 then it is positive logic system

Negative logic System: If lower voltage represent 1 & Higher as zero

Assume if +5V & 0V are logic level voltages then

Positive logic: High - 1 (5V) low 0 (0V)

Negative logic: High - 0 [5V] low - 1 - (0V)

(2) Voltage requirements of TTL & CMOS

Operating voltage: voltage at which device operates.

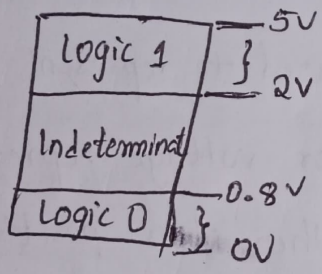
→ Different logic families require different operating voltage

TTL requires +5V as operating voltage.

CMOS requires +3V & +18V

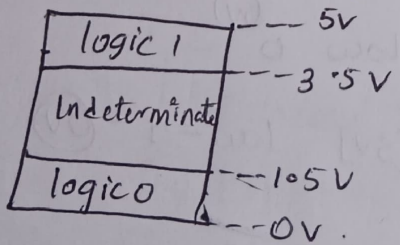
(i) For standard TTL devices a logic 0 ranges from 0 to 0.8V & logic 1 ranges from 2-5V.

(ii) voltage between 0.8V to 2V are said to be indeterminate & should not be used as inputs to any TTL device



(iii) for CMOS voltage between 0 - 1.5V is logic 0 & voltage from 3.5 to 5V are defined for logic 1

(iv) → Indeterminate range 1.50V to 3.5V.



Scanned with OKEN Scanner

11/07/24

(3) Propagation Delay

* The maximum time taken by output to change its state in responding to change in input.
* The speed of a digital circuit is specified in terms of propagate delay
* Shorter the propagation delay, higher the speed of the circuit
* Propagation delay (t_p) is measured using $t_p = \frac{t_{PHL} + t_{PLH}}{2}$

t_{PHL} → time taken to switch logic HIGH from LOW
t_{PLH} → time taken to switch logic LOW from HIGH.

(4) NOISE ~~MARGIN~~

The unwanted voltages called noise may induce into circuit either internally or Externally
This may cause the voltage logic levels to shift undesirable & results gets undesirable operation

NOISE immunity: The ability of a circuit to tolerate noise signals is called noise immunity

NOISE margin: The maximum noise voltage that can be added to an input signal so that it cant cause undesira change in the circuit output is called noise margin.

* In other words, measure of noise immunity is called noise ma

Scanned with OKEN Scanner

(5) Power dissipation

* Every IC requires certain amount of electrical power to operate. This power must not exceed limits.
* Hence, Power dissipation is the amount of power that can be taken for operation without destruction of IC → measured by →

$$P_D = V_{cc} \times I_c$$

P_D = Power dissipation of the gate
V_{cc} = DC voltage
I_c = Average supply current

when power taken is beyond the limits then heat generated & destroys IC.

(6) Figure of merit.

Figure of merit of a digital IC is the product of speed & power.

Figure of merit = Propagation delay (nS) (nano seconds) × Power (mW)

measured in (PJ) - Pico joules.

Normally minimum value is desired.

(7) FAN IN

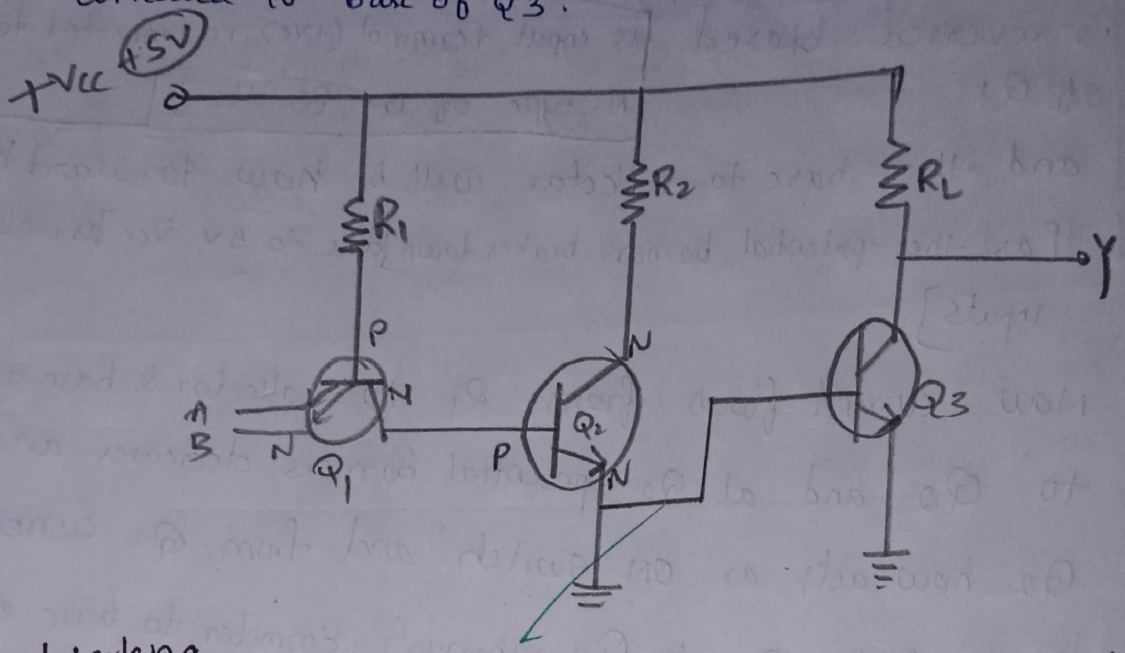
The maximum number of inputs that can be connected to logic circuits is called FAN-IN of the logic circuit

Ex: Inverter has fan-in of 1 (—▷o—)

(8) FAN OUT [loading factor]

The maximum number of outputs than can be driven by a logic circuit is called FAN-OUT

- A two input TTL NAND gate is as follow
- Q_1 is multiple-emitter transistor for two inputs
- Q_1 collector is connected to base of Q_2 & Emitter of Q_2 is connected to base of Q_3.



Working

(1) If A and B inputs are low [assume to be grounded] then the current due to Vcc flows through R_1 and then crosses Base to Emitter junction of Q_1 as A/B are grounded [low] Hence Base Emitter is FB [as +Ve terminal of Vcc is connected to P-type [Base]

(2) The current cannot flows through Base to collector of Q_1 as Collector of Q_1 is connected to Base of Q_2 i.e reversed biased. [N to P] & +Ve terminal of Vcc is connected to N-type collector of Q_2 it is also reversed biased Hence connection of Q_2 is like OFF switch

(3) same with Q_3 as Emitter of Q_2 is connected to base of Q_3 Q_3 is also OFF switch

(4) Hence current from Vcc directly flows from R_L (Pull up resistor to output Y as no chance of grounding.

(5) Even if one of the input may be High it can't pass through Q_2 as due to potential Barrier

Case-2

If A & B are HIGH

① If A & B are HIGH then Emmiter to base junction is reversed biased of Q1 [as input terminal (+ve) is connected to N-type of Q1 [Emitter]]

② and the base to collector will be now forward biased [as the potential barrier brakes down] due to 5V Vcc as well as inputs]

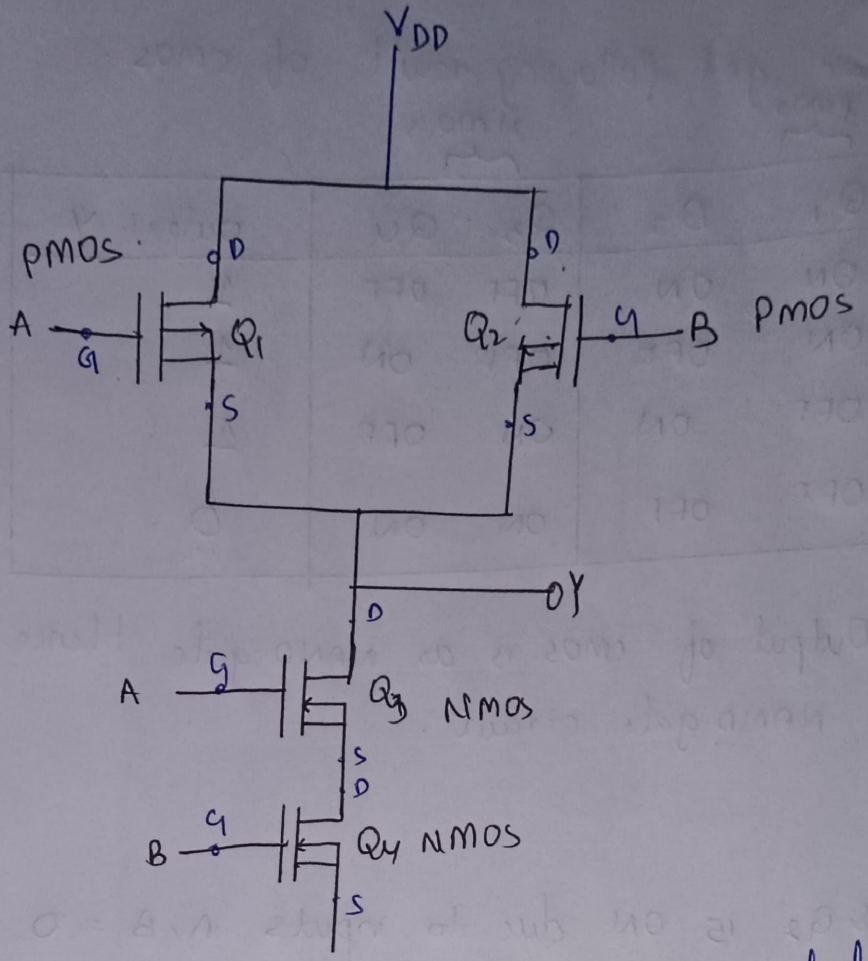
③ Now current flows from R_1 to collector & base of Q_1 to Q_2 and at Q_2 potential barriers decreases and Q_2 now acts as ON switch and from Q_2 current flows from Q_2 to Q_3 through Emmiter to base connection and at Q_3 it is grounded. & ON switch

④ The current from Vcc flows through R_L and then is grounded due to Q_3 & Output is zero.

CMOS inverter

Complementary MOS families

24/7/20



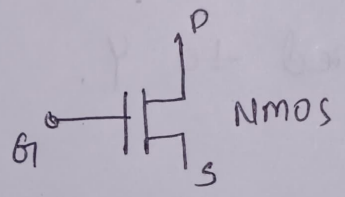
① Q_1, Q_2 are PMOS and are connected in parallel.

② Q_3, Q_4 are NMOS and are connected in series.

③ Transistors used here acts as switches.

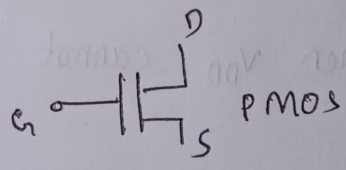
④ At logic 0. Q_1, Q_2 (PMOS) Acts as ON and Q_3, Q_4 as OFF (NMOS)

⑤ At logic 1, Q_1, Q_2 (PMOS) Acts as OFF & Q_3, Q_4 (NMOS) Acts as ON.



G = 1 → ON

G = 0 → OFF



G = 1 → OFF

G = 0 → ON.

⑥ O/p is taken b/w PMOS & NMOS.

Scanned with OKEN Scanner