UNIT- III

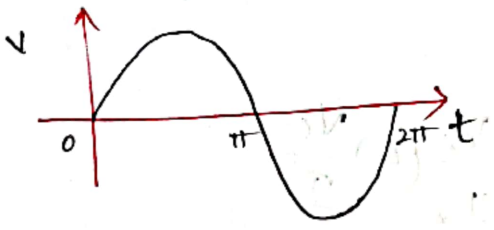
Digital Electronics

* Introduction :-

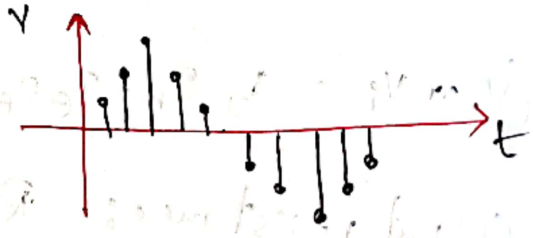
- Now a days digital systems are used in wide variety of industrial and consumer products such as automated industrial machinery, pocket calculators, micro processors, digital computers, digital watches, TV games, signal processing and so on. The areas of applications of digital systems have been increasing every day.

⇒ Analog Vs Digital system :=

- Analog system process information that varies continuously i.e., they process time varying signals that can take on any value across a continuous range of voltage, current or any physical parameter. Ex: voice sig.
- Digital systems use discrete quantities to represent information. Discrete means distinct or separated as opposed to continuous or connected.



(a) continuous signal (Analog)



(b) Discrete sig (Digital)

Advantages of Digital system :-

→ Ease of Design
→ Reproducibility of result
→ Flexibility
→ speed
→ Economy
→ upgrading Technology
→ Audio Recording & video
→ Cameras
→ TV, ~~cells~~, Mobiles

* Number system :-

- In digital electronics the number system is used for representing the information. The number system has different bases and the most common of them are the decimal, binary, octal and hexadecimal.
- the base or radix of the number system is the total number of the digit used in the number system.

Ex Base of system is '10', it represents the digit 0 - 9.

Types of Number systems :-

1. Decimal Number System
2. Binary Number system
3. Octal Number system
4. Hexadecimal Number System.

LSB - least significant bit
MSB - Most " "

Number System	Base/ Radix	All digits / characters
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
Hexa decimal.	16 (or) H	0, 1, 2, 3 4, 5, 6, 7, 8, 9, A, B, C, D, E, F NOTE: A=10, B=11, C=12, D=13, E=14, F=15

1. Decimal Number System :-

- the total number of digits or symbols represented in a decimal number systems are '10'.
- the base or radix of decimal number system is '10'.
- the digits or symbols in decimal number system starts with '0' and end with base -1 i.e, (10-1)is 9)
  i.e, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Ex :- ① $(443)_{10}$ in decimal

$= 4\ 4\ 3$ $\Rightarrow$ $4\times10^2 + 4\times10^1 + 3\times10^0$

$10^2\ 10^1\ 10^0$ $= 400 + 40 + 3 = (443)_{10}$

- The decimal number system is positional weighted system and its weights are expressed as power of 10.
- In floating point representation the left hand side digits have +ve powers and right hand side digits have -ve powers.

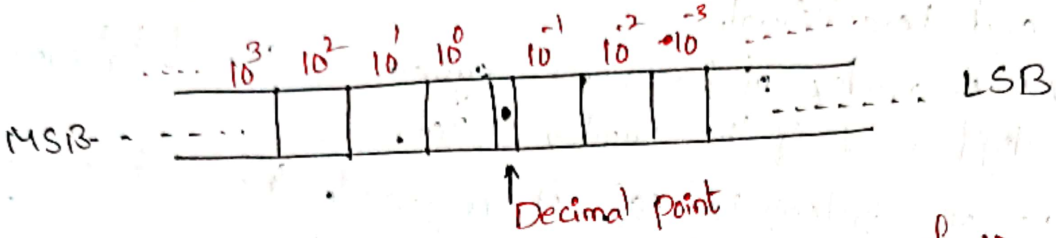


fig:- Decimal position values as powers of 10.

Ex:- 198.72

$= 198.72$
$10^2\ 10^1\ 10^0\ 10^{-1}\ 10^{-2}$

$= (1\times10^2) + (9\times10^1) + (8\times10^0) + 7\times10^{-1} + 2\times10^{-2}$

$= 100 + 90 + 8 + 0.7 + 0.02$

$= (198.72)_{10}$

2. Binary Number System:-

- The total no. of digits or symbols represented in a binary No. system are 2.
- The base or radix of binary no. system is 2
- The digits or symbols in binary No. system starts with '0' and end with base -1 i.e (2-1 = 1) the digits are 0, 1.

Ex:- $2^3\ 2^2\ 2^1\ 2^0$

0 - 0 0 0 0
1 - 0 0 0 1
2 - 0 0 1 0
3 - 0 0 1 1
4 - 0 1 0 0
5 - 0 1 0 1
6 - 0 1 1 0
7 - 0 1 1 1
8 - 1 0 0 0

Ex:- 1000 = 8
$2^3\ 2^2\ 2^1\ 2^0$

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

$= 8 + 0 + 0 + 0$

$= 8$

- In floating point representation the left hand side digit +ve powers and Right hand side digits are -ve powers.

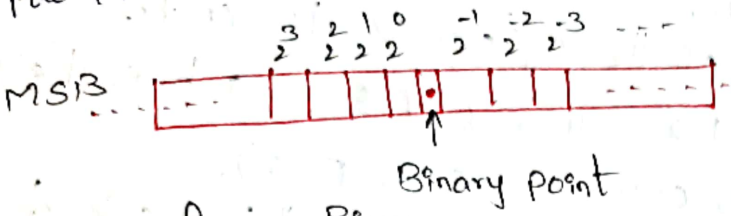


fig: Binary ~~Decimal~~ position values as powers of 2

EX: ① $(1101.101)_2 = (\quad)_{10}$

1 1 0 1 . 1 0 1
2^3 2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3}

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

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

$= (13.625)_{10}$

3. Octal Number System:-

- The total no. of digits or symbols present in the octal no. system is '8'.
- The base or radix of the system is '8'. The digits are

0, 1, 2, 3, 4, 5, 6, 7

- In floating point representation the left hand side digit +ve powers and right hand side digits are -ve powers.

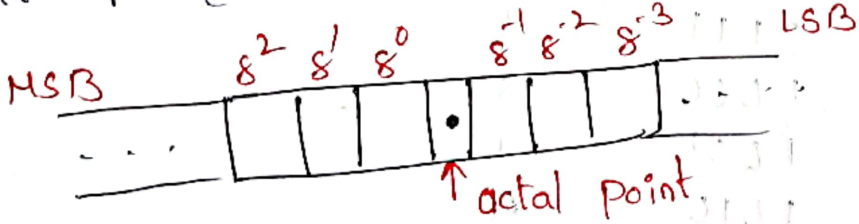


fig: octal position values as powers of '8'.

EX ① $(567)_8 = (\quad)_{10}$

5 6 7
8^2 8^1 8^0

$= (5\times8^2) + (6\times8^1) + (7\times8^0)$

$= (5\times64) + 48 + 7 = (375)_{10}$

$= (\quad)$

② $47.3 = (\quad)_{10}$
8^1 8^0 8^{-1}

$= (4\times8^1) + (7\times8^0) + (3\times8^{-1})$

$= 32 + 7 + 0.37$

$= (39.37)_{10}$

4. Hexadecimal Number System:-

- The total No. system of digits or symbols presents in the hexadecimal is 16.
- The base or radix of the system is '16' or 'H'.
- The digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f (a=10, b=11, c=12, d=13, e=14, f=15).
- In floating point representation the left hand side digits have +ve powers and right hand side digits have -ve powers.

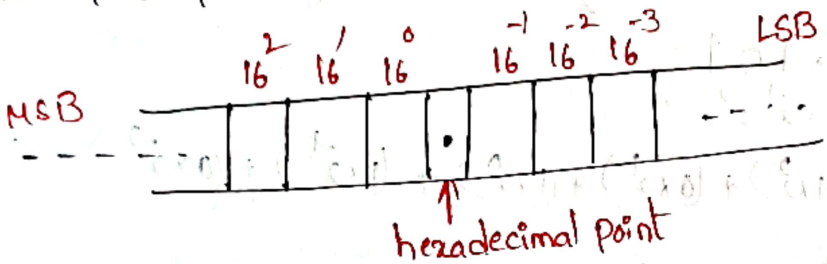


fig:- hexadecimal position values as a power of 16.

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

EX:-

① $(2A1)_{16} = (\quad)_{10}$

$2\ A\ 1$
$16^2\ 16^1\ 16^0$

$= (2\times16^2) + (A\times16^1) + (1\times16^0)$

$= 512 + 160 + 1$

$= (673)_{10}$

② $(1B.2)_H = (\quad)_{10}$

$1\ B\ .\ 2$
$16^1\ 16^0\ \ 16^{-1}$

$= (1\times16^1) + (B\times16^0) + (2\times16^{-1})$

$= 16 + 11 + 0.125$

$= (27.125)_{10}$

* Binary to octal conversion:-

- The base for octal number is 8 and the base for binary number is 2. The base for octal number is the third power of the base for binary numbers.
- Therefore by grouping 3 digits of binary numbers and then converting each group digit to its octal equivalent, we can convert binary to octal equivalent.

Decimal	octal digit 3 bit $2^3=8$ ($2^2\ 2^1\ 2^0$ = 4+2+1)
0	0 0 0
1	0 0 1
2	0 1 0
3	0 1 1
4	1 0 0
5	1 0 1
6	1 1 0
7	1 1 1

EX ① $(101001110)_2 = (\quad)_8$

MSB ... LSB

$101\ |\ 001\ |\ 110$ — Grouping 3 bits

$5\quad 1\quad 6$

$(516)_8$

② $(111\ 01\ 011)_2 = (\quad)_8$

$011\ |\ 101\ |\ 011$

$3\quad 5\quad 3$

$(353)_8$

* Octal to Binary Conversion :-

Conversion from octal to binary is a reversal of the process explained in the binary to octal conversion. Each digit of the octal number is individually converted to its binary equivalent. To get octal to binary conversion of the number.

Ex:- ① Convert $(634)_8$ to binary.

6 3 4
| | |
110 011 100

Binary number = $(110011100)_2$

② convert $(725.63)_8$ to binary

7 2 5 . 6 3
↓ ↓ ↓ . ↓ ↘
111 010 101 . 110 011

binary number = $(111010101.11001)_2$

* Binary to Hexadecimal conversion :-

- The base for hexadecimal number is the fourth power of the base for binary numbers. Therefore by grouping 4 digits of binary numbers and then converting each group digit to its hexadecimal equivalent, we can convert binary number to its hexadecimal equivalent.

Ex:- ① convert $(1101100101011001)_2$ to hexadecimal

1101 | 1001 | 0101 | 1001
13 (D) | 9 | 5 | 9

$= (D959)_H$

② Convert 1100 0001 1100 0111

0001 | 1000 | 0011 | 1000 | 1111
1 | 8 | 3 | 8 | F

$(183BF)_H$

* Hexadecimal to Binary conversion :-

Conversion from hexadecimal to binary is a reversal of the process explained in the previ binary to hexa conversion. Each digit of the hexadecimal number is individually converted to its binary equivalent to get hexadecimal to binary conversion of the number.

Ex: ① Convert $(3FD)_H$ to binary

3 F D

0011 1111 1101

binary number = $(001111111101)_2$

② convert $(5A9.B4)_H$ to binary

5 A 9 . B4

0101 1010 1001 . 1011 0100

binary number = $(0101\,1010\,1001.10110100)_2$

* Octal to Hexadecimal conversion :=

The easiest way to convert octal number to hexadecimal number is given below.

1. convert octal number to its binary equivalent.
2. convert binary number to hexadecimal equivalent

Ex :- ① Convert $(615)_8$ to its hexadecimal equivalent

step 1: octal to binary 6 1 5

↓ ↓ ↓

110 001 101

step 2: binary number to hexa decimal

0001 | 1000 | 101

1 8 D

$(18D)_H$

② convert $(443)_8$ to Hexadecimal

step: 1 4 4 3

/ | \

100 100 011

0001 | 0010 | 0011

1 2 3

$(123)_H$

* Hexadecimal to octal conversion :=

the easiest way to convert hexadecimal number to octal number is given below.

1. convert hexadecimal number to its binary equivalent
2. convert binary number to its octal equivalent.

Ex ① convert $(25B)_H$ to octal equivalent

sol: step 1 : Hexadecimal to binary

2 5 B

0010 0101 1011

binary number = 001001011011

step 2 : binary to octal equivalent

001|001|011|011

1 1 3 3

octal number = $(1133)_8$

② convert $(1F3)_H$ to octal

sol: 1 F 3

⇒ 0001 1111 0011

⇒ 000|111|110|011

0 7 6 3

⇒ $(0763)_8$

* Conversion of Decimal to other Radix number :=

① Decimal to binary conversion

② Binary to Decimal " (Reverse process)

③ Decimal to octal "

④ octal to Decimal " (Reverse)

⑤ Decimal to hexa decimal "

⑥ Hexadecimal to decimal " (Reverse)

* Decimal to binary conversion :- (Succesive by division Method)

① Convert decimal Number 37 to its binary equivalent.

$(37)_{10} = (\quad)_2$

	Q	R
2	37	
2	18	1 LSB
2	9	0
2	4	1
2	2	0
	1	0 MSB

2)37(18
-2
17
-16
1 R

2)18(9
-18
0

2)9(4
8
1

2)4(2
4
0

when get small number zero compare and stop the processor.

Binary equivalent = $(100101)_2$ (MSB … LSB)

② $(55)_{10} = (\quad)_2$

	Q	R
2	55	
2	27	1
2	13	1
2	6	1
2	3	0
	1	1 MSB

$(110111)_2$

Recheck: $1\ 1\ 0\ 1\ 1\ 1$ (positions $2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$)

$= 1\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

$= 32 + 16 + 0 + 4 + 2 + 1 = 55$

③ Convert decimal number 24.6 to binary

$(24.6)_{10} = (\quad)_2$

step1: Separate fraction part and integer part

integer part = 24

step2:

	Q	R
2	24	
2	12	0
2	6	0
2	3	0
	1	1

$(11000)_2$

Fraction part = 0.6

Fraction	Radix	Result		Recorded Carries
0.6	× 2	= 1.2	= 1	MSB
0.2	× 2	= 0.4	= 0	
0.4	× 2	= 0.8	= 0	
0.8	× 2	= 1.6	= 1	
0.6	× 2	= 1.2	= 1	LSB

Fraction (MSB … LSB)

∴ The binary equivalent is $(11000.10011)_2$

② $(43.7)_{10} = (\quad)_2$

integer part = 43

2 | 43	
2 | 21	-1
2 | 10	-1
2 | 5	-0
2 | 2	-1
	1 -0 ↑

Fraction = 0.7

0.7 × 2	= 1.4	⇒ 1	MSB
0.4 × 2	= 0.8	⇒ 0	
0.8 × 2	= 1.6	= 1	
0.6 × 2	= 1.2	= 1	
0.2 × 2	= 0.4	= 0	LSB

101011

$\therefore (43.7)_{10} = (101011.10110)_2$

$a^0 = 1$

* Binary to Decimal conversions:-

by using successive multiplication method

① $(1101)_2 = (\quad)_{10}$

MSB LSB

1 1 0 1

2^3 2^2 2^1 2^0 (2 powers)

⇒ $(1\times2^3)+(1\times2^2)+(0\times2^1)+(1\times2^0)$ (Mul, Sum)

⇒ $(1\times8)+(1\times4)+(0\times2)+(1\times1)$

$= 8+4+0+1$

$= (13)_{10}$

$\therefore (1101)_2 = (13)_{10}$

② $(10011)_2 = (\quad)_{10}$

1 0 0 1 1

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

$=(1\times2^4)+(0\times2^3)+(0\times2^2)+(1\times2^1)+(1\times2^0)$

$=(1\times16)+(0\times8)+(0\times4)+(1\times2)+(1\times1)$

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

$= (19)_{10}$

$\therefore (10011)_2 = (19)_{10}$

③ $(110.101)_2 = (\quad)_{10}$

1 1 0 . 1 0 1

2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3}

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

$= 4+2+0+0.5+0.25+0.125$

$= (6.875)_{10}$

0.125

$\frac{1}{8}$

0.125
0.25
0.5
0.875

* Decimal to octal conversion :-

Successive division by 8

digits - (0 to 7)

base - 8

① $(443)_{10} = (\quad)_8$

8 | 443	
8 | 55 – 3	LSB
6 – 7	MSB

$8)443(55$, 40, 43, 40, 3

$8)55(6$, 48, 7

$(673)_8$

Recheck :-

$(673)_8$

$6\ 7\ 3$

$8^2\ 8^1\ 8^0$

$(6\times8^2)+(7\times8^1)+(3\times8^0)$

$= 384+56+3$

$= (443)_{10}$

$6\times6 = 36$, carry 2; $6\times4+2 = 26$, carry 2; $6\times1+2 = 8$

384, 56, 3

443

② $(35.45)_{10} = (\quad)_8$

Integer part = 35

Fraction part = 0.45

8 | 35	
4 – 1	
$(41)_8$	

$8)35(4$, 32, 3

$0.45\times8 = 3.60 - 3$ MSB

$0.6\times8 = 4.8 - 4$

$0.8\times8 = 6.4 - 6$

$0.4\times8 = 3.2 - 3$

$0.2\times8 = 1.6 = 1$ LSB

$\therefore (35.45)_{10} = (41.34631)_8$

* Octal to Decimal conversion :-

Successive Multiplication with '8'.

① $(673)_8 = (\quad)_{10}$

$6\ 7\ 3$

$8^2\ 8^1\ 8^0$

$(6\times8^2)+(7\times8^1)+(3\times8^0)$

$= (6\times64)+(7\times8)+(3\times1)$

$= (443)_{10}$

② $(35.61)_8 = (\quad)_{10}$

35.61

$8^1\ 8^0\ 8^{-1}\ 8^{-2}$

$= (3\times8^1)+(5\times8^0)+(6\times8^{-1})+(1\times8^{-2})$

$= 24+5+0.75+0.01$

$= (29.79)_{10}$

* Decimal to Hexadecimal conversion :- Successive division by '16'

base = 16 or H

0-9 } a=10, b=11
A-F } c=12, d=13, e=14, f=15

① $(1005)_{10} = (\quad)_{16}$

16 | 1005	
16 | 62 – 13 (D)	
3 – 14 (E)	

16) 1005 (62
96
45
32
13

16) 62 (3
48
14

$(3ED)_{16}$

Recheck :

$(3ED)_{16}$

$16^2\ 16^1\ 16^0$

$(3\times16^2) + (E\times16^1) + (13\times16^0)$

$= 768 + 224 + 13$

$= (1005)_{10}$

② $(22.64)_{10} = (\quad)_{H}$

Integer = 22

16 | 22	
1 – 6	

16) 22 (1
16
6

16

fractional = 0.64

$0.64\times16 = 10.24$ – 10 MSB (A)

$0.24\times16 = 3.84$ – 3

$0.84\times16 = 13.44$ – 13 (D)

$0.44\times16 = 7.04$ – 7

$\therefore (22.64)_{10} = (16.A3D7)_{H}$

* Hexadecimal to decimal conversion :-

① $(A1)_{16} = (\quad)_{10}$

A 1

$16^1\ 16^0$

$(A\times16^1) + (1\times16^0)$

$(10\times16) + (1\times1)$

$= 160 + 1$

$= (161)_{10}$

② 2D.6A

2 D . 6 A

$16^1\ 16^0\ 16^{-1}\ 16^{-2}$

$= (2\times16^1) + (D\times16^0) + (6\times16^{-1}) + (A\times16^{-2})$

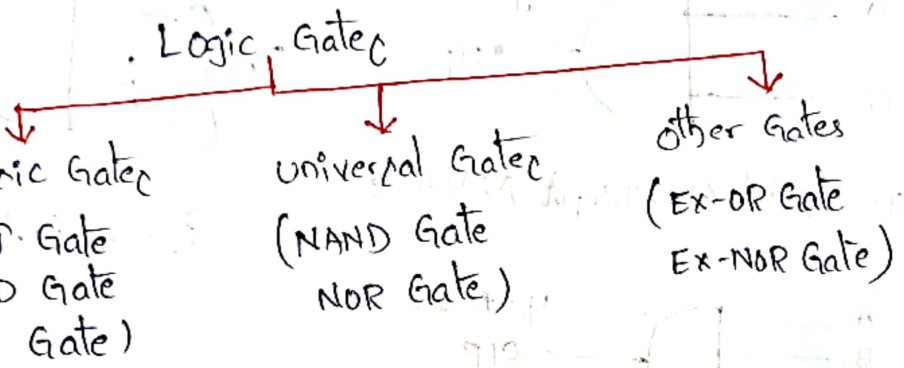
$= 32 + (13\times1) + (6\times16^{-1}) + (10\times16^{-2})$

$= 32 + 13 + 0.37 + 0.03$

$= (45.40)_{10}$

* Logic Gates :-

- Logic gates are the basic elements that make up a digital system. The logic gates allows or delays the logic signals (input signals).
- The gate is a digital circuit with one or more i/ps and single output. The operation of a logic gate can be easily understood with the help of "Truth table".
- The Truth tables indicates the relationship between i/ps and o/ps.



* NOT Gate :- (Inverter)

- The Inverter or NOT Gate performs a basic logic function called 'inversion' or 'complementation'. NOT Gate consists of single i/p and single o/p. The output of NOT Gate is complement of input signal.

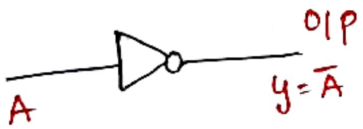


fig:- NOT Gate Symbol

Truth table :-

i/p A	o/p y = Ā
0	1
1	0

Inverter operation :-

- if the input of NOT Gate is '0' (Low) then o/p of NOT Gate is '1' (High).
- if the input of NOT Gate is '1' (High) the output of NOT Gate is '0' (Low).

* AND Gate :-

- The AND Gate performs logical multiplication of inputs signals. The AND Gate consists of two or more than two inputs and single output.
- The output of AND Gate is product of input signals. (IC 7408)

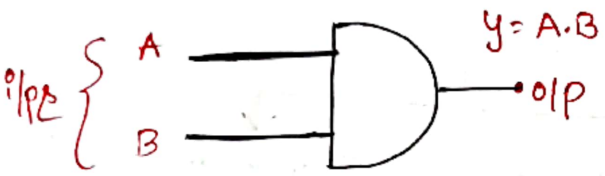


fig:- Symbol of 2-input AND Gate

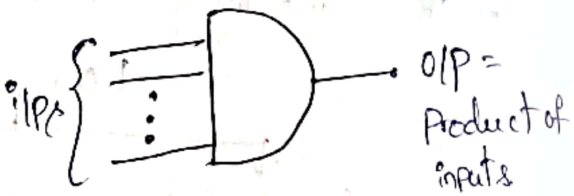


fig:- multiple i/p AND Gate

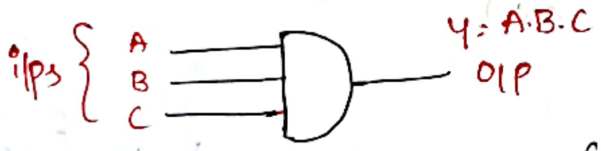


fig:- Symbol of 3-i/p AND Gate

Truth table :- $Y = A.B$

inputs		output
A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate operation :-

- if both or two inputs of AND Gate is '1' (high) then output of AND Gate is 1 (high).
- → if any one of the input of AND Gate '0' (zero) (Low) then output of AND Gate is '0' (Low)

* OR Gate :=

- The OR Gate performs logical addition of input signals. The OR Gate consists of two or more than two i/ps and single output.
- The output of OR Gate is addition of inputs.

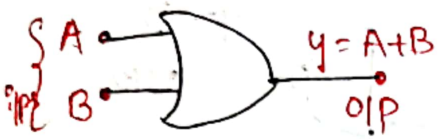


fig: symbol of 2-i/p OR Gate

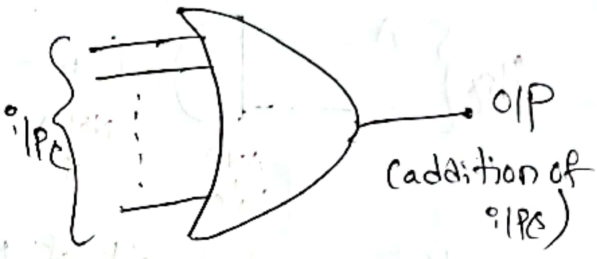


fig:- symbol of multiple i/p OR Gate

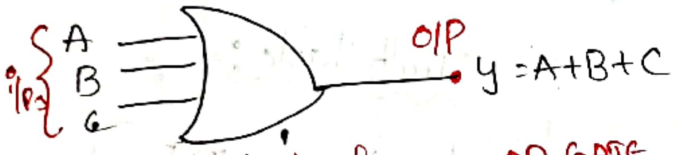


fig: symbol of 3-i/p OR GATE

OR operation :-

- If both inputs of OR Gate is '0' (Low), then output of OR Gate is '0' (Low).
- If any one of the input is 1 (High) then o/p of OR Gate is 1 (High).

Truth table :

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

* UNIVERSAL GATES :-

→ The universal gates are two NAND Gate and NOR gate. By using universal gates we can design all other gates so that NAND and NOR gates are universal gates.

→ universal gates consists of two or more than two inputs and single output.

1) NAND Gate - (AND Gate + NOT Gate)

2) NOR Gate - (OR Gate + NOT Gate)

* NAND Gate :=

- The NAND Gate is combination of AND Gate and NOT Gate. The NAND gate output is AND Gate function with a inverted (complemented) output.

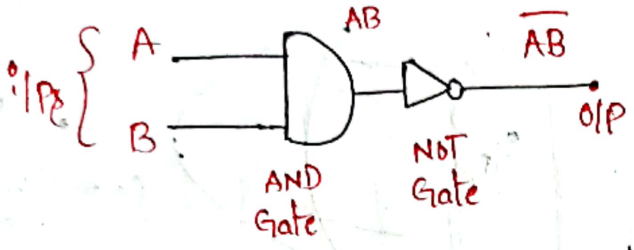


fig: Equivalent ckt of NAND Gate

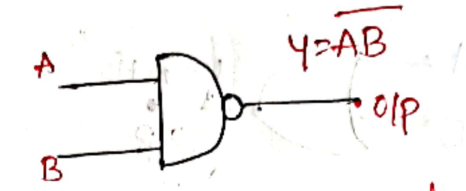


fig: -symbol of NAND Gate

Truth table:

inputs		AND O/P	NOT O/P
A	B	A.B	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

(or)

Truth table :=

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

NAND Gate operation :-

- If both inputs of NAND Gate is '1' (High) then output of NAND gate is '0' (Low).
- If any one of the input of NAND Gate is '0' (Low) the output of NAND Gate is '1' (High).

* NOR Gate :=

- The NOR Gate is combination of OR Gate and NOT Gate. The NOR gate output is OR Gate function with a inverted (complemented) output.

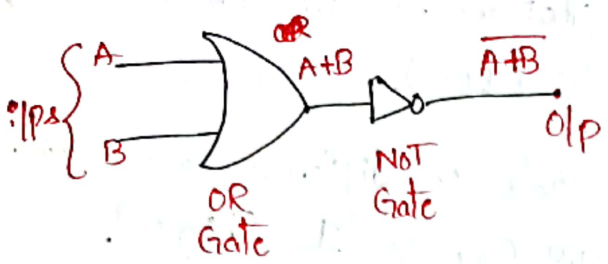


fig: OR + NOT Gate

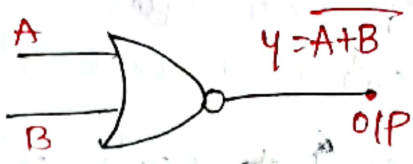


fig: symbol of NOR Gate

Truth Table :-

Inputs		OR A+B	NOT A+B
A	B		
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

(or)

Inputs		NOR O/P A+B
A	B	
0	0	1
0	1	0
1	0	0
1	1	0

NOR Gate operation :-

- If both inputs of NOR Gate is '0' (LOW) then output of NOR Gate is 1 (High).
- If any of the input of NOR Gate is 1 the output of NOR gate is '0' (LOW).

* Ex-OR Gate (or) X-OR Gate :=

- The Ex-OR gate is an abbreviation for Exclusive-OR gate. An Ex-OR gate has two or more than two inputs and one output.

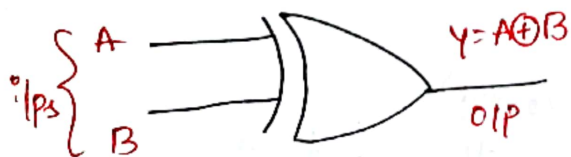


fig: Symbol of 2-I/P Ex-OR Gate

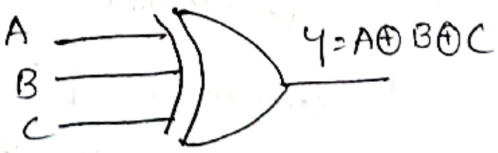


fig: symbol of 3-I/P Ex-OR Gate

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

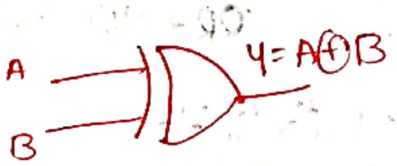
- if both inputs are Same that means 0,0 or 1,1 then output of Ex-OR gate is '0' (Low).
- if both inputs of Ex-OR Gate is different (or) odd no. of ones then output of Ex-OR Gate is '1' (high).

Truth table :

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

Inputs A	Inputs B	$\bar{A}$	$\bar{B}$	$A\bar{B}$	$\bar{A}B$	$A\bar{B} + \bar{A}B$
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

(or)



Inputs A	Inputs B	output Y = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

* Ex-NOR Gate :-

- The Ex-NOR Gate combination of Ex-OR Gate and NOT Gate. The Ex-NOR Gate consists of two or more than two inputs and single output.
- Ex-NOR Gate output is compliment of Ex-OR Gate.

fig: Symbol of 2-i/p Ex-NOR Gate

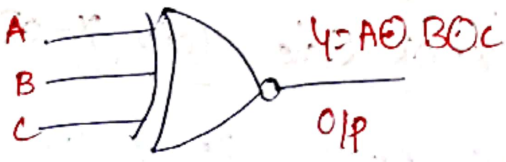


fig: Symbol of 3-i/p Ex-NOR Gate

Expression: $Y = A \odot B$ (or) $\overline{A \oplus B}$

$Y = \bar{A}\bar{B} + AB$ (0,0)(1,1).

- If both inputs are Same, then output of Ex-NOR Gate is 1 (High).

- If both inputs of Ex-NOR gate is different then output of Ex-NOR gate is '0' (Low).

Truth Table:-

$Y = A \odot B$ (or) $Y = \overline{A \oplus B}$

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

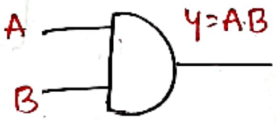
Input		$\bar{A}$	$\bar{B}$	$\bar{A}\bar{B}$	AB	$\bar{A}\bar{B} + AB$
A	B					
0	0	1	1	1	0	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	0	1	1

(or)

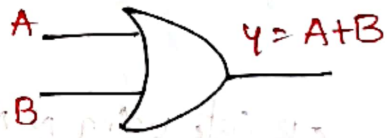
INPUTS		O/P $Y = A \odot B$
A	B	
0	0	1
0	1	0
1	0	0
1	1	1

NOTE:- The logic expression for all the logic gates.

NOT Gate

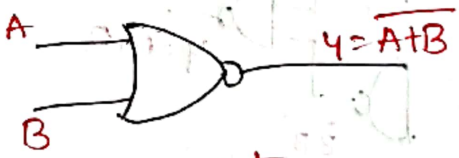


AND Gate

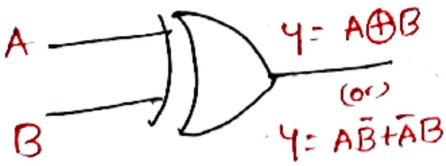


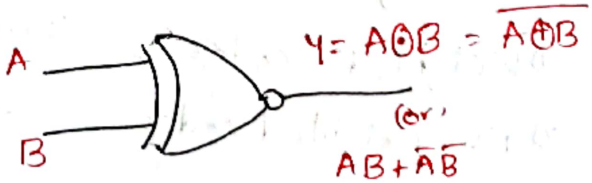
OR Gate

NAND Gate



NOR Gate





→ NOT Gate consists single i/p and single O/P.

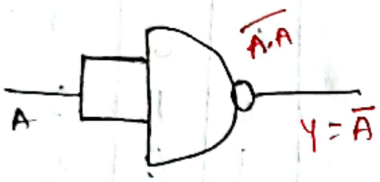
→ All other Gates consists of two or more than two i/ps and single O/P.

→ Universal gates are NAND and NOR Gate.

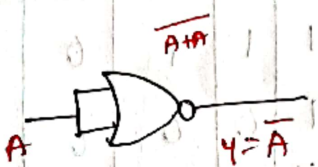
* NOT, AND, OR Gates using universal Gates:-

- The NAND and NOR gates are known as universal gates.
- Any logic gate or logic function can be implemented using NAND or NOR Gates, so that NAND and NOR gates are called as universal gates.

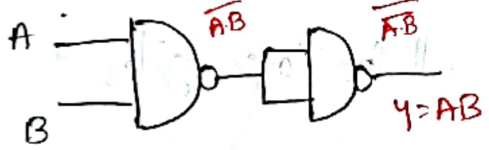
NOT Gate Using NAND Gate:



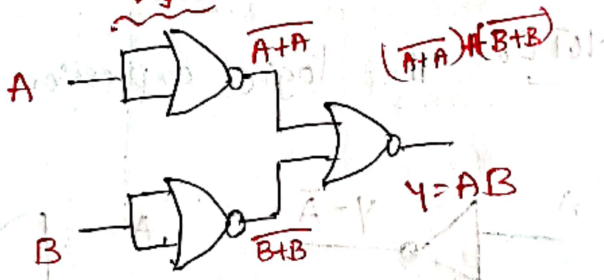
NOT Gate using NOR Gate



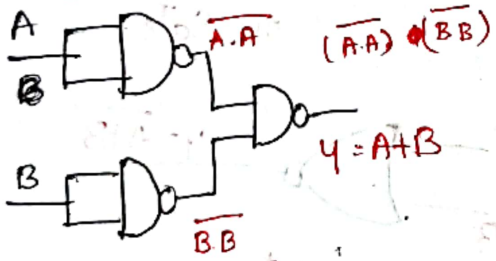
AND Gate using NAND Gate



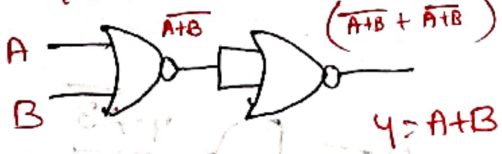
AND Gate using NOR Gate



OR Gate using NAND Gate



OR Gate using NOR Gate



* Binary codes:-

1) Weighted codes — 8421, 2421, 5211
2) Non-weighted codes — Excess-3, Hamming code
3) Reflective codes — (0-9) (0-8) (1,9) (2,6) (3,5)
4) sequential codes — 8421, excess-3
5) Alphanumeric codes — ASCII, EBCDIC
6) Error detecting and correcting codes

American standard code for information interchange

* BCD code :- [Binary coded Decimal 8-4-2-1]

- BCD is an abbreviation for binary-coded-decimal. BCD is a numeric code in which each digit of a decimal number is represented by a separate group of bits. The most common BCD code is 8-4-2-1 BCD, in which each decimal digit is represented by a 4-bit binary number.
- It is called 8-4-2-1 BCD because the weights associated with 4-bits are 8-4-2-1 from left to right.

Decimal digit	BCD code			
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Table: 8-4-2-1 BCD code

NOTE :-

Addition

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Ex :-

```
  1001  → 9
  0011  → 3
+ 11
------
  1100  → 12
```

subtraction

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Ex sub $(0101)_2$ from $(1011)_2$

```
  1011  → 11
(-)0101  → 5
---------
  0110  → 6
```

BCD Addition :-

BCD addition procedure as follows.

1. Add two BCD numbers using ordinary binary addition
2. If four-bit sum is equal to or less than 9, no correction is needed. The sum is in proper BCD form.
3. If the four-bit sum is greater than 9 or if a carry is generated from the four bit sum, the sum is invalid.
4. To correct the invalid sum, add $(0110)_2$ to the four-bit sum. If a carry results from this addition, add it to the next higher-order BCD digit.

Ex:- ① Addition of 3 and 6 in BCD

$$\begin{array}{rl} 6 & = 0110 \\ +3 & = 0011 \\ \hline 9 & \quad 1001 \end{array}$$

② Addition of 6 and 8 in BCD

$$\begin{array}{rl} 6 & = 0110 \\ +8 & = 1000 \\ \hline 14 & \quad 1110 \leftarrow \text{invalid so add '6'} \\ & \quad 0110 \\ & \quad 10100 \end{array}$$

$$0001\ 0100$$
$$1 \quad 4$$

③ $8 \quad 1000$

$+9 \quad 1001$

$+17 \quad 0001\ 0001$ — invalid add six

$0000\ 0110$

$0001\ 0111$ — BCD for 17

17.

④ $24 = 0010\ 0100$

$+18 = 0001\ 1000$

$42 \quad 0011\ 1100$ — > 9 so add '6'

$0000\ 0110$

$0100\ 0010$ — BCD for 42

* BCD subtraction :-

Addition of signed BCD numbers can be performed by using 9's or 10's complement methods, The 9's complement of a decimal number is found by subtracting each digit in the number from 9. The 9's complement of each the decimal digit is as follows

Digit	9's complement
0	9
1	8
2	7
3	6
4	5
5	4
6	3
7	2
8	1
9	0

Ex :-

① 8 — 8
−2 — 7 — 9's complement of 2
6 — (+) 15
+ 1
6

② 28 — 28
−13 — 86 — 9's complement of 13
15 — (+) 114
+ 1
15

③ 79 — 0111 1001
−26 — (+) 0111 0011 — 9's complement of 26 is 73
53 — 1110 1100 — Invalid 'so' add '6'

(+) 0110 0110
1 0101 0010
+1
0101 0011 — BCD 53

* Excess-3 code :- [self complementing code]

Excess-3 code is a modified form of a BCD Number. The Excess-3 code can be derived from the natural BCD code by adding '3' to each coded number. Excess-3 codes to represent single decimal digit.

Decimal digit	Excess-3 code
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

0000
0011
0011

0001
0011
0100

Ex 9 → 1001
0011 – add '3' (0011)
1100

∴ 9 – 1100 – Excess-3 code

* Gray Code :-

Gray code is a special case of unit-distance code. In unit-distance code, bit patterns for two consecutive numbers differ in only one bit position. These codes are also called cyclic codes (or) reflector code.

* Binary to Gray conversion :-

Let us assume binary number as
$B_1, B_2, B_3, B_4 \ldots\ldots B_n$ and its equivalent Gray code as $G_1, G_2, G_3 \ldots\ldots, G_n$

with this representation gray code bits are obtained from the binary bits as follows

$G_1 = B_1$

$G_2 = B_1 \oplus B_2$

$G_3 = B_2 \oplus B_3$

$G_4 = B_3 \oplus B_4$

$\vdots$

$G_n = B_{n-1} + B_n$

EX-OR

A	B	o/p
0	0	0
0	1	1
1	0	1
1	1	0

Ex:- ① convert 1011101 in binary into its gray code equivalent

sol:

1 0 1 1 1 0 1

1 1 1 0 0 1 1 - Gray code

② convert 111001 in binary into its Gray code

sol:

binary → 1 1 1 0 0 1

Gray code → 1 0 0 1 0 1

Decimal code	Binary code	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

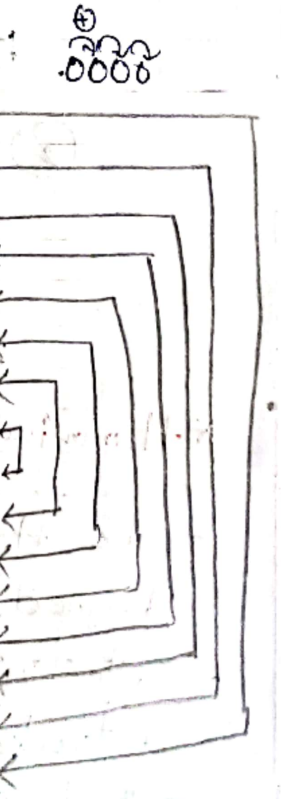


Table:- Gray code

* Gray to binary conversion :-

The gray to binary code conversion can be achieved using following steps:

1. The most significant bit (MSB) of the binary number is the same as the most significant bit of the gray code number, so write it down
2. To obtain the next binary digit perform an exclusive-OR operation between the bit just written down and the next gray code bit. write down the result.
3. Repeat step-2 until all gray code bits have been exclusive-ORed with binary digits. The sequence of bits that have been written down is the binary equivalent of the gray code number.

Ex :- (1) Convert gray code 101011 into its binary equivalent

Gray code : 1 0 1 0 1 1

Binary code : 1 1 0 0 1 0

Ex-OR	
0 0	0
0 1	1
1 0	1
1 1	0

(2) convert gray code 11001 into its binary

Gray code : 1 1 0 0 1

Binary code : 1 0 0 0 1

* Hamming code :-

- The Hamming code method is a network technique designed by R.W. Hamming, for damage and error detection during data transmission between multiple network channels.
- The hamming code method is one of the most effective ways to detect single-data bit errors in the original data at the receiver end.

- It is not only used for error detection but is also for correcting errors in the data bit

1. Redundant Bits :- These are the extra binary bits added externally into the original data bit to prevent damage to the transmitted data and are also needed to recover the original data. The expression applied to deduce the redundant value is

$$2^r = d + r + 1$$

Here d - data bits.
r = redundant bits $r = (1, 2, 3 \dots n)$

Ex :- Assuming the number of data bits is 7, find the no. of redundant bits

$$2^r = r + 7 + 1$$

$$2^r \geq r + 8$$

$$\boxed{r = 4}$$

2. Parity bits :- The parity bit is the method to append binary bits to ensure that the total count of 1's in the original data is even bit or odd. It is also applied to detect errors on the receiver side and correct them.

Types of Parity bits
(a) odd parity bits
(b) Even parity bits

(a) Odd parity Bits :- In this parity type, the total no. of 1's in the data bit should be odd in count, then the parity value is '0' and the value is 1.

(b) Even parity bits :- In this parity type, the total number of 1's in the data bit should be even in count, then the parity value is '0', and the value is 1.

→ In even parity the added parity bit will make the total no. of 1's an even count.

→ In odd parity the added parity bit will make the total no. of 1's an odd count.

→ The table shows the 3-bit message with odd parity and even parity.

3-bit Message			Message with odd Parity		Message with even Parity	
A	B	C	Message	Parity	Message	Parity
0	0	0	000	1	000	0
0	0	1	001	0	001	1
0	1	0	010	0	010	1
0	1	1	011	1	011	0
1	0	0	100	0	100	1
1	0	1	101	1	101	0
1	1	0	110	1	110	0
1	1	1	111	0	111	1

* **1's complement:-** In the 1's complement representation, '0' are replaced by 1 (or) 1's are replaced by '0'.

Ex:- ① Find 1's complement of $(1001)_2$

1001

1's compl → 0110

② 100110

011001 — 1's complement

* **2's complement:-** 1's complement + 1

Ex ① find 2's complement of $(10010)_2$

1's complement → 01101

+1 1

2's complement → 01110

* Boolean Algebra :-

- In 1854, George Boole introduced a systematic treatment of logic and developed for the purpose, an algebraic system, now it called as Boolean Algebra.
- In 1938 C.E. shannon introduced a two-valued boolean algebra called switching algebra. Boolean algebra is a system of mathematical logic. It differs from both ordinary algebra and the binary number system.

* Rules in Boolean Algebra :-

1. The symbol which represent an arbitrary elements of an boolean algebra is known as Variable. Any single variable or a function of several variables can have either a 1 or 0 value.

   Ex :- $Y = A + BC$

   here A, B, C are variables it values 0 or 1. value
   Y is function it value 0 or 1.

2. A complement of a variable is represented by a "bar" over the letter.

   Ex :- Variable is : $A \rightarrow \overline{A}$

   $0 \rightarrow 1$

   $1 \rightarrow 0$

3. The logical AND operator of two variables is represented either by writing a dot (·) between two variables

   Ex : A·B , A·B·C or AB, ABC

4. The logical OR operator of two variables is represented by writing a '+' sign between the two variables.

   Ex : A+B, A+B+C,

* Rules :-

i) $A+0 = 0+A = A$

Ex

A	0	A+0
0	0	0+0=0
1	0	1+0=1

ii) $A+1 = 1+A = 1$

iii) $A+A = A$

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

v) $A.0 = 0.A = 0$

vi) $A.1 = 1.A = A$

vii) $A.A = A$

viii) $A.\bar{A} = 0$

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

Additional Rules :

$A+AB = A$

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

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

* Laws of Boolean Algebra :-

i) Commutative Law : a) $A+B = B+A$

b) $A.B = B.A$

a) $A+B = B+A$: This states that the order in which the variables are ORed makes no difference in the o/p.

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

=

B	A	B+A
0	0	0
0	1	1
1	0	1
1	1	1

b) $A.B = B.A$

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

=

B	A	BA
0	0	0
0	1	0
1	0	0
1	1	1

ii) Associate law :- a) $A+(B+C) = (A+B)+C$

b) $(AB)C = A(BC)$

iii) Distributive law: a) $A(B+C) = AB+AC$

$A(B+C)$

A	B	C	B+C	A(B+C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

=

A	B	C	AB	AC	AB+AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

* DeMorgan's Theorems :=

Theorem 1 :- $\overline{AB} = \bar{A}+\bar{B}$

The complement of a product is equal to the sum of the complements. This illustrated by truth table

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

A	B	AB	$\overline{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

=

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

Theorem 2 :- The complement of a sum is equal to the product of the complements.

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

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

=

A	B	$\bar{A}$	$\bar{B}$	$\bar{A}.\bar{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

NOTE :- Not in syllabus

Boolean Function :- Boolean expressions are constructed by connecting the boolean constants and variables with the boolean operations. These boolean expressions are also known as **boolean formulas**. We use boolean expressions to describe **boolean functions**.

- A variable in either a complemented or uncomplemented form is called a **literal**.

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

① **Sum of Product Forms** = (SOP) (Minterms)

A sum of products (SOP) is a group of product terms ORed together.

Ex: $f(A,B,C) = A\bar{B}\bar{B} + ABC + A\bar{B}\bar{C}$

(sum (OR) over the whole expression; product (AND) over each term)

② **Product of Sum Form** :- (POS) (Maxterms)

A product of sums is any groups of an sum terms ANDed together.

Ex :- $f(A,B,C) = (A+B+C).(A+\bar{B}+C).(\bar{A}+B+\bar{C})$

(Product over the whole expression; sum terms over each factor)

Problems :-

① Simplify the following expression $y = (A+B)(\bar{A}+C)$

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

$$= A\bar{A} + AC + B\bar{A} + BC \qquad (A \cdot \bar{A} = 0)$$

$$= 0 + AC + \bar{A}B + BC$$

$$\therefore y = AC + \bar{A}B + BC$$

② $(x+\bar{y}+xy)(x+\bar{y})\,\bar{x}\,y = 0$

$= (x+\bar{y}+xy)(x+\bar{y})(\bar{x}y)$

$= (x(1+y)+\bar{y})(x+\bar{y})(\bar{x}y)$ $\quad 1+A=1$

$= (x+\bar{y})(x+\bar{y})(\bar{x}y)$ $\quad A\cdot A = A$

$= (x+\bar{y})(\bar{x}y)$ $\quad A\cdot\bar{A}=0$

$= x\cdot\bar{x}\cdot y + \bar{y}\cdot\bar{x}\cdot y$ $\quad x\cdot\bar{x}=0,\ y\cdot\bar{y}=0$

$= 0$

③ $\overline{A\cdot(A+C)}$

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

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

$= \bar{A} + \bar{A}\cdot\bar{C}$ $\quad 1+\bar{A}=1$

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

$= \bar{A}(1)$

$= \bar{A}$

* Combinational circuit :-

- When logic gates are connected together to produce a specified o/p for certain specified combinations of i/p variables, with no storage involved, the resulting circuit is called combinational logic.
- A combinational circuit consists of i/p variables, logic gates and o/p variables. The logic gates accept sigs from the i/p variables and generate output sigs.

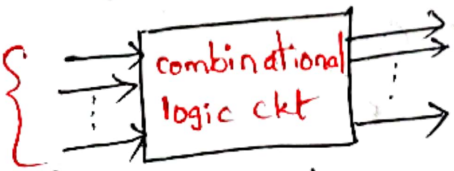
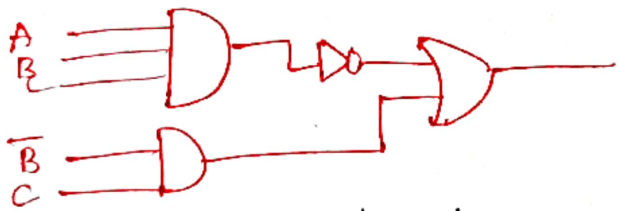


fig:- Block diagram of a combinational circuit

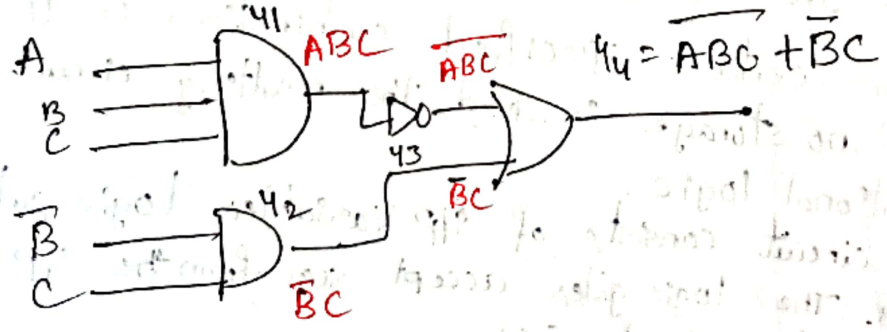
* Analysis Procedure :-

1. First make sure that the given circuit is combinational circuit and not be sequential circuit. The combinational circuit has logic gates with no feedback path or memory elements.
2. Label all gate o/ps that are a function of i/p variables with arbitary symbols, and determine the boolean functions for each gate o/p.
3. Label the gates that are a function of input variables and previously labeled gates and determine the boolean function for them.
4. Repeat the step 3 until the boolean function for o/ps of the circuit are obtained.
5. Finally, substituting previously defined boolean functions, obtain the o/p boolean functions in terms of i/p variables.

EX:- ① obtain the boolean functions for o/ps of the given ckt.



Sol:- Labelling the gate o/p



$$\therefore Y_4 = \overline{ABC} + \overline{B}C$$

$Y_4 = \overline{A}BC + \overline{B}C$

A	B	C	ABC	$\overline{ABC}$	$\overline{B}$	$\overline{B}C$	$\overline{ABC} + \overline{B}C$
0	0	0	0	1	1	0	1
0	0	1	0	1	1	1	1
0	1	0	0	1	0	0	1
0	1	1	0	1	0	0	1
1	0	0	0	1	1	0	1
1	0	1	0	1	1	1	1
1	1	0	0	1	0	0	1
1	1	1	1	0	0	0	0

Truth Table for given circuit.

* Design Procedure :=

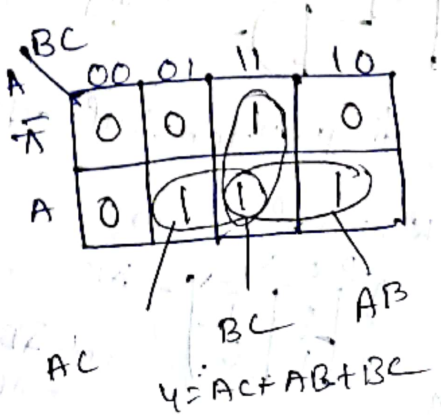
1. The Problem definition.
2. The determination of no.of available i/p variables and required o/p variables.
3. Assigning letter symbols to i/p and o/p variables.
4. The derivation of truth table indicating the relationships between input and output variables.
5. obtain simplified boolean expression for each o/p.
6. obtain the logic diagram.

Ex ① Design a combination logic ckt with three i/p variables the produce a logic 1 o/p when more than one i/p variables are logic 1.

Sol: Three i/p assume A, B, C

when morethan one i/p is 1 then output is 1

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



$Y = AB + AC + BC$

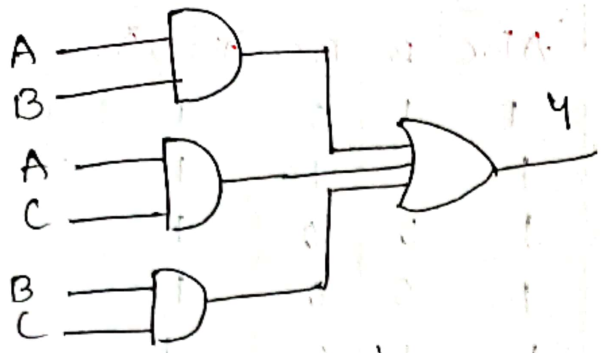


fig: logic diagram

* Half adder :=

- Half adder is a type of digital circuit to calculate the arithmetic binary addition of two single-bit numbers. It is a circuit with two inputs and two o/ps.
- For two single-bit binary numbers A and B, half adder produces two single-bit binary outputs Sum (S) and Carry (C).

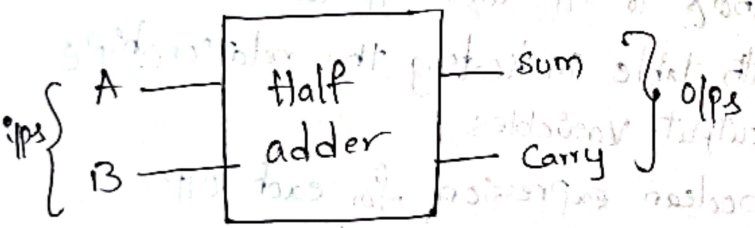


fig: Block schematic of half-adder.

Truth table :-

inputs		outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

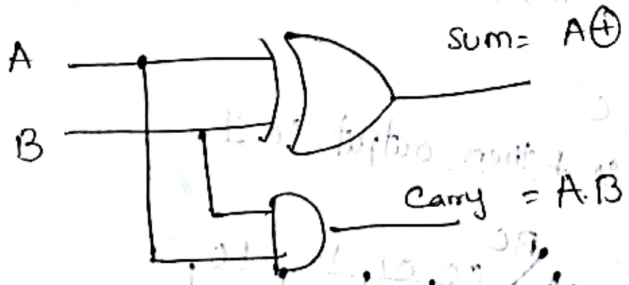


fig: Logic diagram for Half-adder

K-Map for Sum

A \ B	0	1
0	0	1
1	1	0

$\bar{A}B$, $A\bar{B}$

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

for carry

A \ B	0	1
0	0	0
1	0	1

AB

$Carry = AB$

* Full Adder :-

- Full adder is a digital circuit that adds three single digit binary numbers. This is a three - I/P and two O/P digital circuit..
- For three single-bit binary numbers A, B and C_{in}, the full adder circuit generates two-single bit binary O/Ps sum (S) and Carry (Cout).

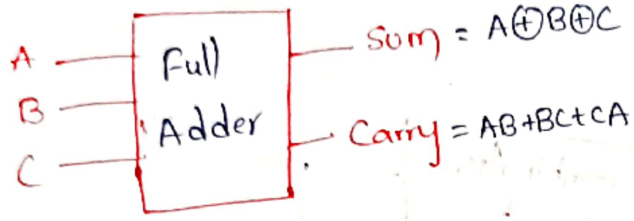
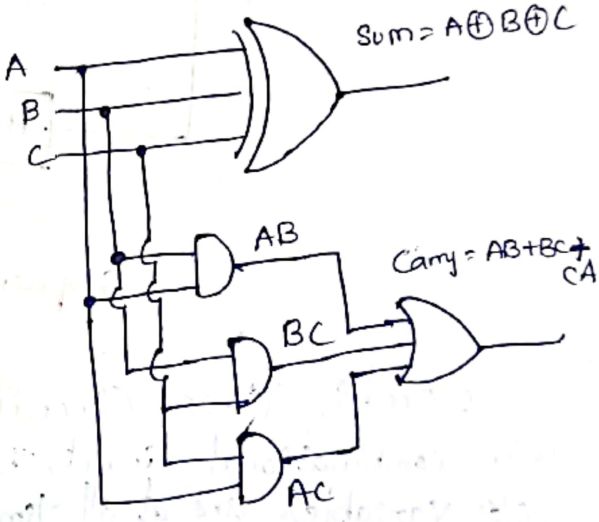


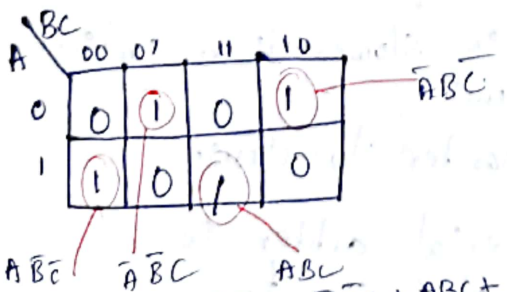
fig: Schematic of Full-Adder



Truth table :-

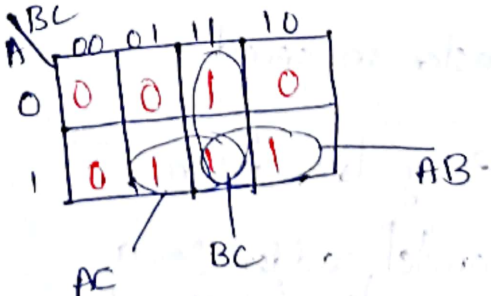
Inputs			Outputs	
A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

K-Map for Sum



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

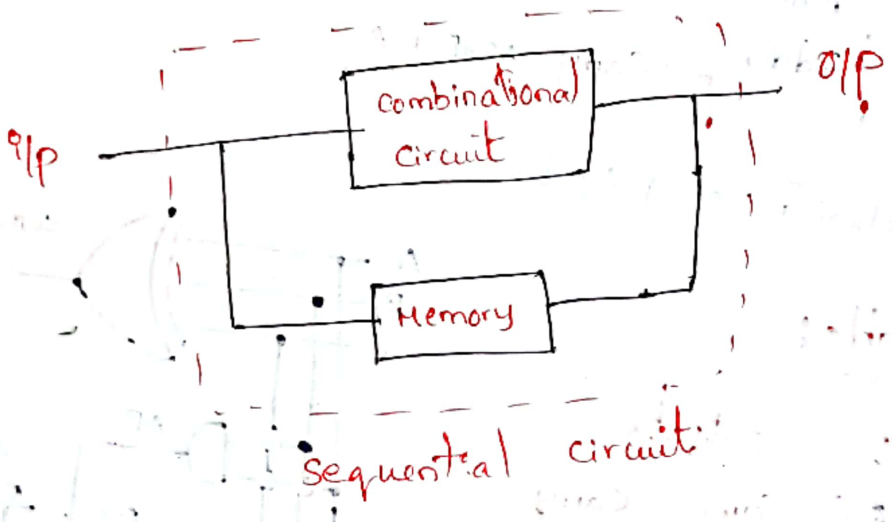
K-Map for Carry



$$Carry = AB + BC + AC$$

* Sequential Circuit :-

- Sequential circuit refers to a special type of circuit. It consists of a series of various inputs and outputs, here the o/p's depends on a combination of both the present inputs as well as the previous o/p's.
- This previous output gets treated in the form of the present state.
- The sequential circuit consists of the combinational circuit along with its memory storage elements.



Combinational Circuits	Sequential Circuits
1. In combinational circuit, the o/p variables are at all times dependent on the combination of i/p variables.	1. In sequential circuits, the o/p variables dependent not only on the present i/p variables but they also depend upon the past history of these i/p variables.
2. Memory unit is not required in combinational ckts.	2. Memory unit is require to store the past history of i/p variables in the sequential ckt.
3. Faster in speed	3. It is slower than Combinational Circuit
4. Easy to design	4. harder to design
5. parallel adder is a combinational circuit	5. Serial adder is a sequential circuit.

Basically two types of sequential circuits

① Asynchronous sequential circuit

② Synchronous sequential circuit

① The asynchronous sequential circuits don't make use of the clock signals. This type of circuit is operated through various pulses.

② The Synchronous sequential circuit :- The clock sig perform the synchronization of the state of memory elements in the case of synchronous sequential circuits.

Synchronous sequential circuit	Asynchronous Sequential Circuits
1. In synchronous ckt, memory elements are clocked flip-flops	1. In asynchronous ckts, memory elements are either unclocked flip-flops or time delay elements
2. The change in i/p signals can affect memory element upon activation of clock sig.	2. change in i/p sigs can affect memory element at any instant of time.
3. The maximum operating speed of clock depends on time delays involved.	3. Because of absence of clock, asynchronous ckts can operate faster than synchronous ckt.
4. Easy to design	4. More difficult to design.

* Flip-Flops :=

-A flip flop in digital electronics is a circuit with two stable states that can be used to store binary data

The stored data can be changed by applying varying inputs.

- Flip Flops and latches are fundamental building blocks of digital electronics systems used in computers, communications and many other types of systems.
- Both are used as data storage elements.
- Flip Flops are edge triggered and a latch is level triggered.
- There are basically 4 types of flip-flops in digital electronics
  1. SR Flip-Flop
  2. JK Flip-Flop
  3. D-Flip Flop
  4. T-Flip Flop.

1. SR Flip Flop :=

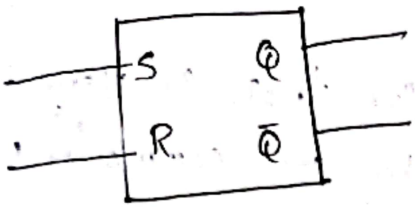
This is the most common flipflop among all. This simple flip flop circuit has a set input (S) and a reset input (R). In this system, when you set "S" as active, the output "Q" would be high, and "$\bar{Q}$" would be low. Once the o/ps are established, the wiring of the ckt is maintained until "S" or "R" go high, or power is turned off.

R	S	Q_{n+1}
0	0	Q_n
0	1	1
1	0	0
1	1	∞

(a) RS Truth Table

Q_n	Q_{n+1}	R	S
0	0	X	0
0	1	0	1
1	0	1	0
1	1	0	X

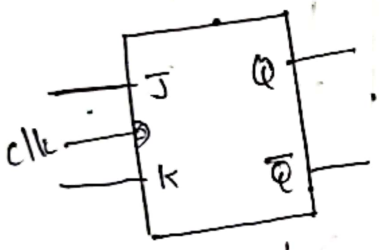
(b) RS excitation Table



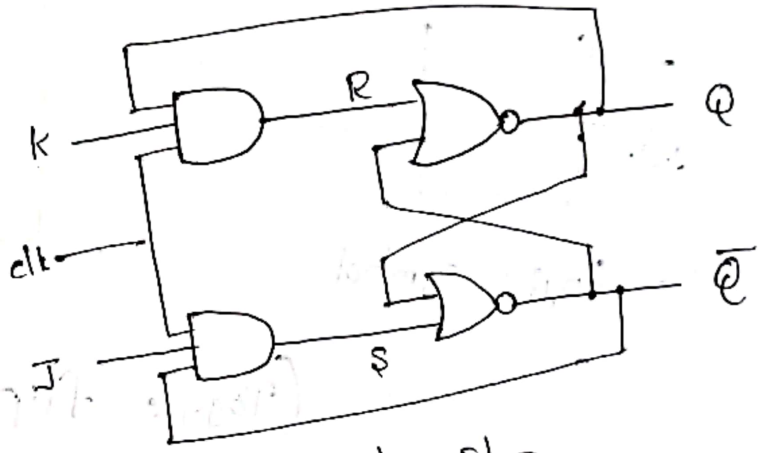
(a) symbol

2. J.K flip-flop

Due to the undefined state in the SR flip flops, another flip flop is required in electronics. The JK flip-flop is an improvement on the SR flip flop ~~where SR=1 is not~~



(a) logic symbol



JK- Flip-Flop

- The input condition of J=K=1 gives an output inverting the o/p state.

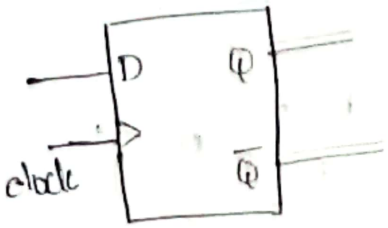
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

=

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

* D-Flip Flop :-

- D Flip-Flop is a better alternative that is very popular with digital electronics. They are commonly used for counters and shift registers and s/p.
- In the D-flip flop the o/p can only be changed at the clock edge, and if the i/p changes at other times, the o/p will be unaffected.

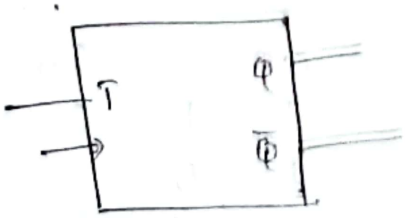


(a). Logic symbol

CP	D	Q_{n+1}
↑	0	0
↑	1	1
0	X	Q_n

* T-Flip Flop :- (Toggle flipflop)

A T-flip flop is like a Jk flipflop. These are basically single i/p version of Jk flipflops. This modified form of the Jk is obtained by connecting i/ps J and K together. It has only one i/p along with the clock i/p.



(a) logic symbol

T	Q_{n+1}
0	Q_n
1	$\overline{Q}_n$

* comparisions between Latch and Flipflops :-

Latch	Flip-Flop
1. Latches do not required clock signal	1. Flipflop have clock signal
2. Latches are building blocks of sequential circuits and those can be built from logic gates.	2. Flipflops are also building blocks of sequential circuits but those can be built from the latches
3. Latch is a level triggered	3. Flipflop is edge triggered
4. O/p response to i/p only at sequential +ve or -ve edge clock pulse.	4. O/p response to the i/p until active level is maintained
5. Latches are level sensitive 	5. Flipflops are edge sensitive 

* Registers :-

- A Flip-Flop is 1-bit memory cell. to increase the storage capacity.
- A register is a group of flip-flops. A flipflop can store 1-bit information. So an n-bit register has a group of n-flipflops and is capable of storing any binary information/number containing n-bits.

Shift Register :-

- In a digital circuit a shift register is a cascade of Flip Flops.
- the binary information (data) in a register can be moved from stage to stage within the register or into or out of the register upon application of clock pulses.

- This type of bit movement or shifting is essential for certain arithmetic and logic operations used in microprocessors.
- This gives rise to a group of registers called "shift registers".
- There are 4 types of shift registers
  1. Serial In serial out shift register [SISO]
  2. Serial In parallel out shift register [SIPO]
  3. Parallel In serial out shift register [PISO]
  4. Parallel In paralle out shift register [PIPO]

1. Serial In Serial Out shift Register [SISO] :-

- Here the data is shifted in serial mode i.e., bit by bit. Same o/p is also shifted in serial mode i.e., bit by bit

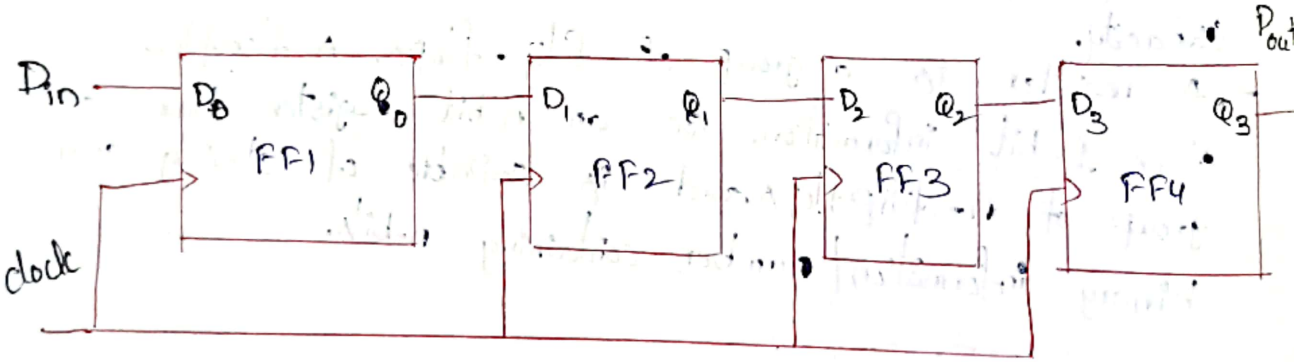
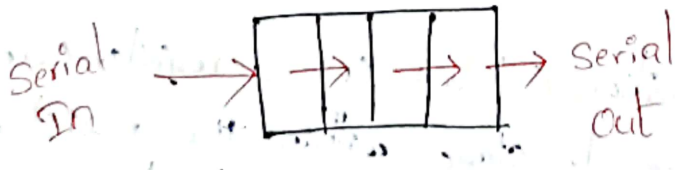


fig:- Serial in serial out shift register



2. Serial In Parallel Out shift Register (SIPO) :=

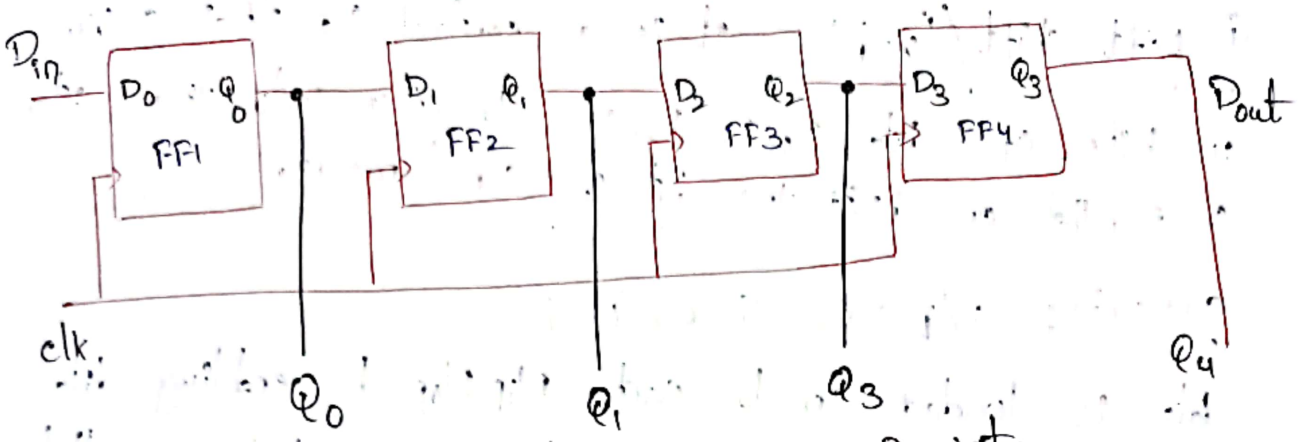
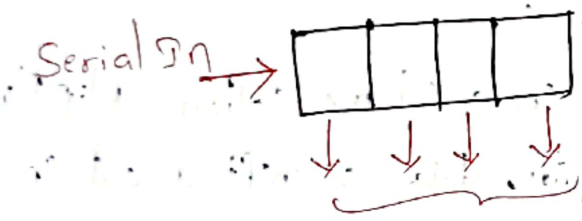


fig: Serial In Parallel out shift Register



- Here the data is shifted In in serial mode i.e., bit by bit position. In order to shift data in parallel it is necessary to have all data bits at its O/p of all FlipFlops.
- Once data is store O/p is collected at a time at Q_0, Q_1, Q_2, Q_3.

3. Parallel In Serial out shift Registers :=

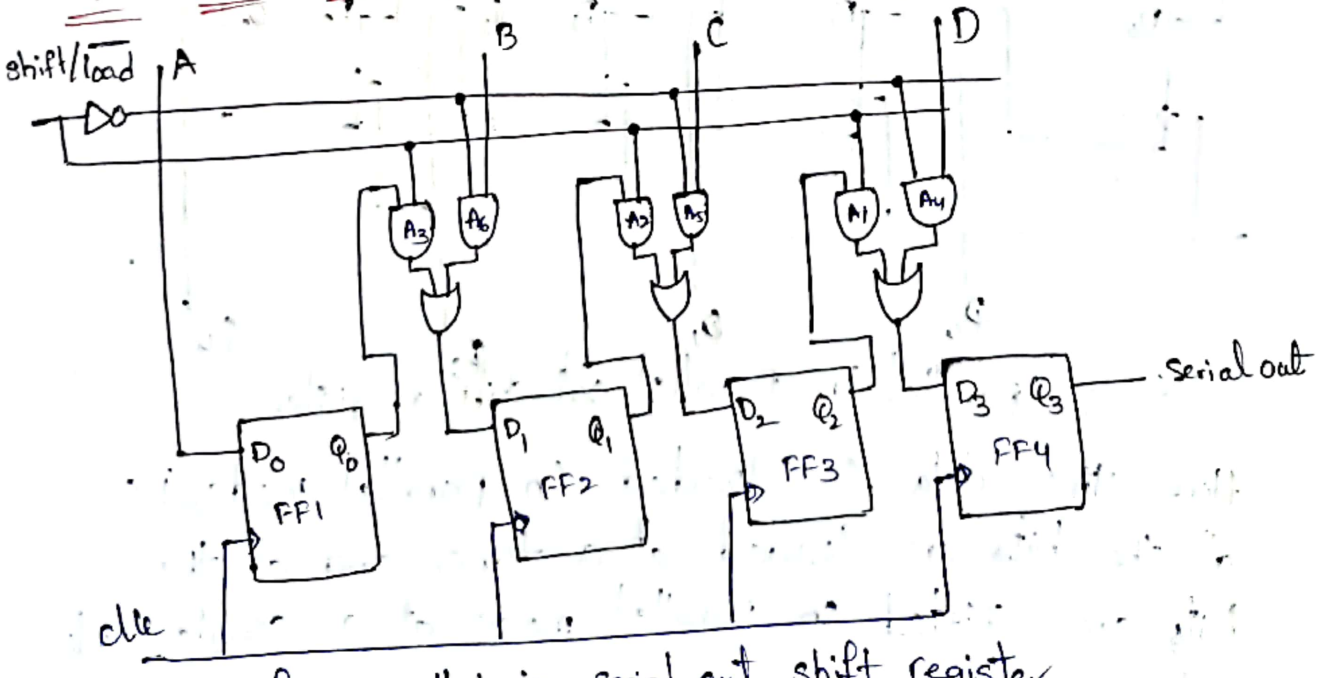


fig: Parallel in serial out shift register

- From flip flop o/p enabling A_1 A_2 A_3 gates data is shifted out in serial mode.

① Left shift / $\overline{load}$:- it is a control i/p that allows 4- bits of data D_0, D_1, D_3, D_3 to enter into the register in parallel or shift the data in serial mode.

case(i) : $S/\bar{L} = 0$, $S = 0$, $\bar{L} = 1$

data is loaded in to each flip flop by enabling data the A_4, A_5, A_6 gates. Hence data is loaded in parallel mode.

case(ii) : $S/\bar{L} = 1$, $S = 1$, $\bar{L} = 0$ . . .

The loaded data is shifted in serial mode from flip flop to o/p enabling A_1, A_2, A_3 gates, data is shifted out in serial mode.

4. Parallel In Parallel out shift Register [PIPO] :-

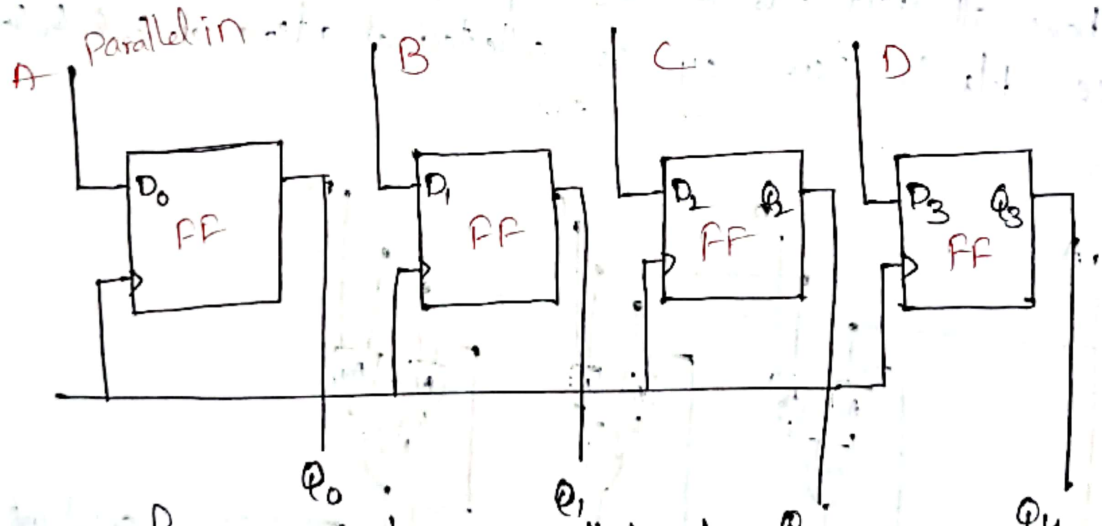


fig:- parallel in parallel out shift register

Here the data is loaded in parallel mode i.e, all the bits are loaded at a time, and all the bits are shifted in parallel mode at a single clock pulse.

* Counter :=

- A counter is a register capable of counting the number of clock pulses arriving at its clock input. count represents the number of clock pulsed pulses arrived.
- There are two types of counters, Synchronous and asynchronous.

* In synchronous counter, the common clock input is connected to all of the flip-flops and thus they are clocked simultaneously.
* In Asynchronous counter, commonly called Ripple counters, the first flipflop is clocked by the external clock pulse and then each successive flipflop is clocked by the Q or $\bar{Q}$ o/p of the previous flipflop.

*Asynchronous counters/ Ripple counter :=

The 2-bit asynchronous counter using J.k flipflops is as shown in diagram. The clock signal is connected to the clock i/p of only first stage flipflop.
- The clock input of the second stage flipflop is trigged by the $\bar{Q}_A$ output of the first stage.
- Because of the inherent propagation delay time through a flipflop, a transition of the i/p clock pulse and a transition of the $\bar{Q}_A$ o/p of first stage can never occur at exactly the same time.
- Therefore, the two flipflops are never simultaneously triggered, which results in asynchronous counter operation.

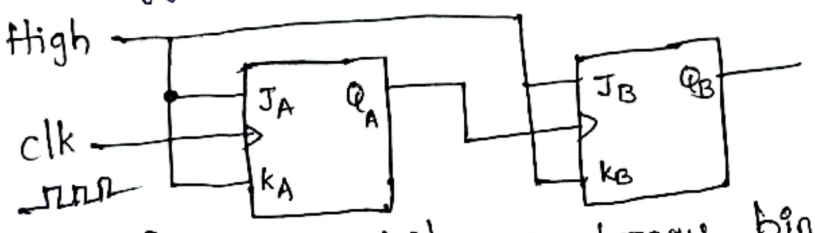


fig:- A two bit asynchronous binary counter

* Synchronous Counter / Binary up or down Counter :-

To form a parallel up/down counter the control input (up/down) is used to control whether the normal flipflop outputs or the inverted flip flop outputs are fed to the J and K inputs of the following flip flops.

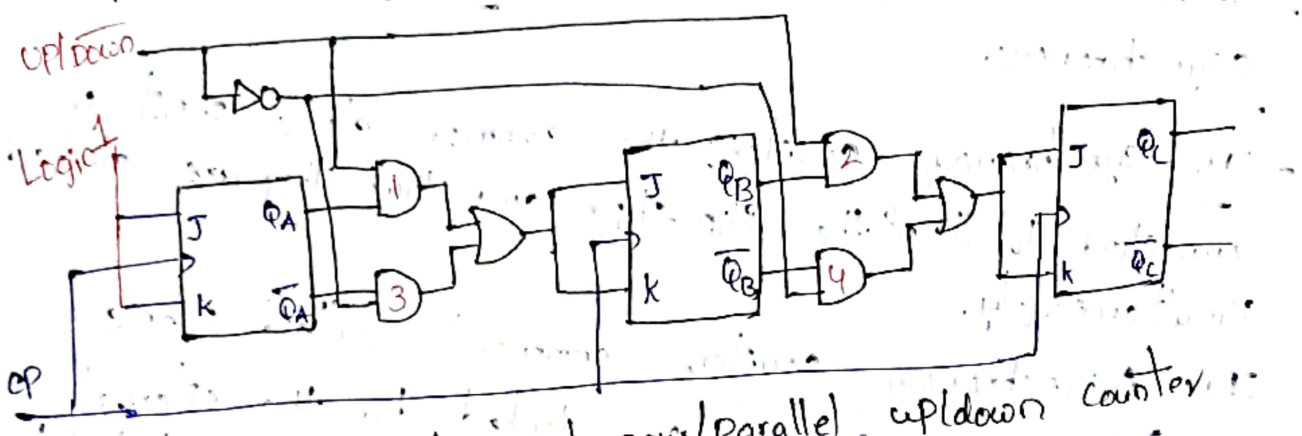


Fig: 3-bit synchronous/parallel up/down counter

* Synchronous Vs Asynchronous counters.

Asynchronous counters	Synchronous Counters
1. In this type of counter Flipflops are connected in such a way that o/p of first flipflop drives the clock for the next FF.	1. In this type there is no connection between o/p of first FF and clock i/p of the next FF.
2. All the ffs are not clocked simultaneously	2. All the FFs are clocked simultaneously
3. Logic circuit is very simple even for more number of states	3. design involves complex logic ckt as no. of states increases
4. low speed	4. High speed.

— X —