Digital Logic

1 **computer:-**

computer is an electromechanical device which accesses input data and process it to the user instruction.

OR

Computer is an electromechanical device which process arithmatic and logic operation.

characteristics of computer system:-

(i) Automatic
(ii) Versatility
(iii) speed
(iv) power of remembering
(v) Accuracy
(vi) No I.Q.
(vii) Diligence
(viii) No feelings

Hardware:-

Hardware is the physical component of computer system which are visible and touchable.

Ex:- Keyboard, mouse, monitor, CPU e.t.c.

Software:-

Software is the physical component of computer which isn't touchable.

OR

Software is the set of programms which enable the computer system to do some work.

Software
- System Software
- Application Software

(1) System software :-
System software is the set of programme which enable the hardware to do work.
Ex- OS, utility program, translation system, community S/W

(2) Application software :-
It is also a set of program to perform a specific task.
Ex- Word, Excell, PPt e.t.c.

Operating system (OS) :-
OS is a set of programme which acts as an interface between the users and the computer system.
(Interface — communicate)

Ex- Windows, Android e.t.c.

Types of OS :-

Examples

i) Single user - DOS (Disk Operating System)
ii) Multi user - Linux, Unix, Windows
iii) Multi tasking - MS windows, Mac OS
iv) Multi processing - Windows NT, 2000, XP and Unix
v) Real time - VRTX, PSOS, RT Linux e.t.c.

Language processor :-
There are 3 types of language processor of translator.

(1) Assembler
(2) Interpreter
(3) compiler

(1) Assembler :-
i) A programme which translates an assembly language program into a machine language programme is called an assembler.

ii) computer only understand machine language. The programme written in other language must be translated into machine language.

(2) compiler :-

(i) It is a program which translates a high-level language programme into a machine language programme.

(ii) A compiler is more intelligent than assembler.

(iii) Its programme run time is more and occupies a larger part of the memory. So it has low speed.

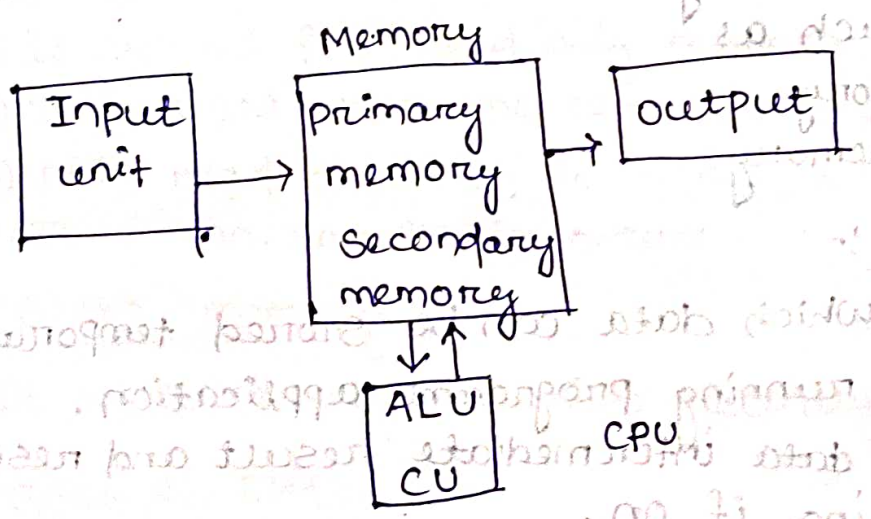
(3) Interpreter :-

(i) It is a programme which translates statements of a programme into machine code.

(ii) It translates only one statement at a time.

(iii) It reads only one statement of programme translates it and execute it.

Component of computer system :-



CPU :-

(i) It is otherwise known as brain of computers.

(ii) It performs a major calculations and comparisons.

(iii) There are 2 basics components are :-

(1) control unit (CU)

(2) Arithmatic Logic Unit (ALU)

(iv) No other single component as a computer determines the overall performance as much as the CPU.

(v) It processed all input data.

(vi) The computational and control unit of a computer is known as CPU.

ALU :-

(i) It stands for Arithmatic logic unit.
(ii) It performs all arithmatic operation or calculation (+, -, *, -/).
(iii) It also performs logical operations or comparision. (>, <, =).
(iv) A computer performs all calculation and comparision operations in the ALU.

CU :-

(i) It acts as the central nerves system of a computer.
(ii) It stands for control unit.
(iii) It controlls the raw data between the peripherals of computer.

Memory :-

(i) It is a place in which data can be store either temporary or permanently.
(ii) It is 2 types such as
  (1) primary memory
  (2) secondary memory

primary memory :-

(i) It is a place in which data can be stored temporarily
(ii) It is used to hold running programme application.
(iii) It is used to hold data intermediate result and result of on going processing if on.
(iv) It is volatile in nature.
(v) It has fast in operation.
(vi) It has small capacity.
(vii) It is expensive.

Secondary memory :-

(i) It is a place in which data can be stored permanently.
(ii) It is used to hold stored programme and instruction.
(iii) It is used to hold data and intermation stored jobs.
(iv) It is slower than primary memory.
(v) It has large capacity.
(vi) It is cheaper than primary storage.

(vii) It is otherwise known as auxilary memory of computer.

Primary memory:-

It is basically 3 types.

(1) RAM
(2) ROM
(3) Cache memory

RAM :-

(i) It stands for random access memory.
(ii) It is the main memory of the computer.
(iii) Read and write operations are permissible.
(iv) It is volatile in nature.
(When the power is off then the data can be lost due to temporary nature.

ROM:-

(i) It stands for read only memory.
(ii) It is used when the computer is shutting up.
(iii) only read operation is permissible.
(iv) It is non-volatile in nature.

Cache memory:-

It is a memory in which recently used data are stored.

Types of ROM :-

(i) PROM (Programmable ROM)
(ii) EPROM (Erasable PROM)
(iii) EEPROM (Electrical ~~EPRO~~ EPROM)

PROM :-

This is other kinds of ROM in programmable ROM. A user can also write his own programme but can't made any further changes.

EPROM:-

(i) This permanent memory Provide both Writing and erasing facility.

(ii) This is known as erasable programable ROM.

(iii) You can erase the already programmed chip by exposing it to ultraviolet light.

EEPROM:-

(i) Electrical Erasable programable ROM is permanent memory which users can write and erasable programme.

(ii) Hence you can erase the chip by exposing it to an electric charge.

RAM:-

It is of 2 types.

(1) Static RAM

(2) Dynamic RAM

Dynamic RAM:-

(i) It can store 4 times ~~store~~ more data than Static RAM.

(ii) It is slower than static RAM.

(iii) It is very cheap than static RAM.

(iv) It is a form of semi conductor ROM in which the data can store integrated circuit (capacitor).

Static RAM:-

(i) It can store 1/4 th data of dynamic RAM.

(ii) It is faster to store the data than dynamic RAM.

(iii) It is more expensive.

(iv) It is a form of semiconductor memory in which we can store the data on a logic circuit known as flip clock which store the information for long time period and it needs more power to run the device.

Number System :-

1. Binary Number System (Base 2, Digits used 0,1)
2. Octal Number System (Base 8, Digits used 0 to 7)
3. Hexa decimal Number System (Base 16, Digits used: 0 to 9, letters used A-F)

· There are many methods or techniques which can be used to convert numbers from one base to another like.

→ Decimal to other base system
→ Other base system to decimal
→ Other base system to non-decimal
→ Shortcut method - binary to octal
→ Shortcut method - octal to binary
→ Shortcut method - binary to Hexadecimal
→ Shortcut method - Hexadecimal to Binary

Conversion :-

(1) Decimal to Binary :-

Example :-

(i)
```
2 | 2
2 | 1 | 0 ↑
    0 | 1
```
$= (10)_2 = 2$

(ii)
```
2 | 3
2 | 1 | 1 ↑
    0 | 1
```
$= 3 = (11)_2$

(iii)
```
2 | 4
2 | 2 | 0
2 | 1 | 0 ↑
    0 | 1
```
$= 4 = (100)_2$

(iv)
```
2 | 90
2 | 45 | 0
2 | 22 | 1
2 | 11 | 0
2 | 5 | 1
2 | 2 | 1
2 | 1 | 0 ↑
    0 | 1
```
$= 9 = (1011010)_2$

Decimal	Binary
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10 (A)	1010
11 (B)	1011
12 (C)	1100
13 (D)	1101
14 (E)	1110
15 (F)	1111

(2) Binary to Decimal :-

(i) $(1100100)_2 = 1\times2^6 + 1\times2^5 + 0\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0$

(654321 0)

$= 64 + 32 + 0 + 0 + 4 + 0 + 0$

$= (100)_{10}$ (Ans)

(ii) $(1010.0101)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 0\times2^{-3} + 1\times2^{-4}$

(3210 1234)

$= 8 + 0 + 2 + 0 + 0 + 0.25 + 0 + 0.0625$

$= (10.3125)_{10}$ (Ans)

(3) Octal to Decimal :-

(i) $(234)_8 = 2\times8^2 + 3\times8^1 + 4\times8^0$

(210)

$= 128 + 24 + 4$

$= (156)_{10}$ (Ans)

(ii) $(245)_8 = 2\times8^2 + 4\times8^1 + 5\times8^0$

(210)

$= 128 + 32 + 5$

$= (165)_{10}$ (Ans)

(4) Decimal to octal :-

(i)
```
8 | 156
8 | 19 | 4
8 | 2 | 3
    0 | 2
```
$= (156)_{10} = (234)_8$ (Ans)

(ii)
```
8 | 165
8 | 20 | 5
8 | 2 | 4
    0 | 2
```
$= (165)_{10} = (245)_8$ (Ans)

(5) Hexadecimal to Decimal :-

(i) $(ABC)_{16} = A\times16^2 + B\times16^1 + C\times16^0$

$= 10\times256 + 11\times16 + 12$

$= (2748)_{10}$ (Ans)

(ii) $(EF)_{16} = E\times16^1 + F\times16^0$

$= 14\times16 + 15$

$= 224 + 15 = (239)_{10}$ (Ans)

(6) Decimal to Hexadecimal :-

i) 16 | 2748
16 | 171 | 12
16 | 10 | 11
0 | 10

$= (10\ 11\ 12)_{16} = (ABC)_{16}$ (Ans)

(7) Octal to Hexadecimal :-

ii) $(357)_8 = (\ )_{16}$

1st convert octal into decimal and then hexadecimal

$(357)_8 = 3\times8^2 + 5\times8^1 + 7\times8^0$

$= 3\times64 + 40 + 7$

$= (239)_{10}$ (Ans)

$(239)_{10} = (\ )_{16}$

16 | 239
16 | 14 | 15
0 | 14

$= (14\ 15)_{16} = (EF)_{16}$ (Ans)

(8) Binary to octal :-

1st convert binary to decimal and then octal.

① $(1010111)_2 = 1\times2^6 + 0\times2^5 + 1\times2^4 + 0\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

$= 64 + 0 + 16 + 0 + 4 + 2 + 1$

$= (87)_{10}$

$(87)_{10} = (\ )_8$

$= 8 | 87$
8 | 10 | 7
8 | 1 | 2
0 | 1

$= (127)_8$ (Ans)

Shortcut method binary to octal :-

① $(1010111)_2 = \overline{001}\ \overline{010}\ \overline{111}$

$(001)_2 = 0\times2^2 + 0\times2^1 + 1\times2^0 = 1$

$(010)_2 = 0\times2^2 + 1\times2^1 + 0\times2^0 = 2$

$(111)_2 = 1\times2^2 + 1\times2^1 + 1\times2^0 = 7$

$(1010111)_2 = (127)_8$ (Ans)

⑨ Binary to Hexadecimal :-

1st binary to decimal and then hexadecimal

(i) $(10101111)_2 = 1\times2^8+0\times2^7+1\times2^6+0\times2^5+1\times2^4+1\times2^3+1\times2^2+1\times2^1+1\times2^0$

$= 256+0+64+0+16+8+4+2+1$

$= (351)_{10}$

$$\begin{array}{r|l} 16 & 351 \\ 16 & 21 \;|\; 15 \\ 16 & 1 \;|\; 5 \\ & 0 \;|\; 1 \end{array}$$

$= (1\ 5\ 15)_{16} = (15F)_{16}$ (Ans)

Shortcut method binary to hexadecimal :-

(i) $(10101111)_2 = \overline{0001}\ \overline{0101}\ \overline{1111}$

$(0001)_2 = 0\times2^3+0\times2^2+0\times2^1+1\times2^0 = 1$

$(0101)_2 = 0\times2^3+1\times2^2+0\times2^1+1\times2^0 = 5$

$(1111)_2 = 1\times2^3+1\times2^2+1\times2^1+1\times2^0 = 15$

$(1\ 5\ 15)_{16} = (15F)_{16}$ (Ans)

⑩ Fractional decimal to binary :-

(i) $(0.25)_{10} = (\ )_2$

$= 0.25\times2 = 0.5$

$= 0.5\times2 = 1.0$

$(0.25)_{10} = (0.01)_2$ (Ans)

(ii) $(0.9)_{10} = (\ )_2$

$= 0.9\times2 = 1.8 \to 1$

$0.8\times2 = 1.6 \to 1$

$0.6\times2 = 1.2 \to 1$

$0.2\times2 = 0.4 \to 0$

$0.4\times2 = 0.8 \to 0$

$0.8\times2 = 1.6$

$0.6\times2 = 1.2$

$0.2\times2 = 0.4$

$0.4\times2 = 0.8$

$(0.9)_{10} = (0.11100)_2$

(iii) $(21.1)_{10} = (\ )_2$

$$\begin{array}{r|l} 2 & 21 \\ 2 & 10\ 1\ 1 \\ 2 & 5\ 1\ 0 \\ 2 & 2\ 1\ 1 \\ 2 & 1\ 1\ 0 \\ & 0\ 1\ 1 \end{array}$$

$= (10101)_2$

$0.1 \times 2 = 0.2 \rightarrow 0$

$0.2 \times 2 = 0.4 \rightarrow 0$

$0.4 \times 2 = 0.8 \rightarrow 0$

$0.8 \times 2 = 1.6 \rightarrow 1$

$0.6 \times 2 = 1.2 \rightarrow 1$

$0.2 \times 2 = \cancel{2.4}\ 0.4 \rightarrow 0.$

$0.4 \times 2 = 0.8 \rightarrow 0$

$(21.1)_{10} = (10101.0001100)_2$

(11) Fractional octal to decimal :-

(i) $(0.325)_8 = (\ )_{10}$

$= 3 \times 8^{-1} + 2 \times 8^{-2} + 5 \times 8^{-3}$

$= 3 \times \frac{1}{8} + 2 \times \frac{1}{8^2} + 5 \times \frac{1}{8^3}$

$= \frac{3}{8} + \frac{2}{64} + \frac{5}{512} = \frac{192+16+5}{512} = \frac{213}{512} = (0.416)_{10}$

(Ans)

(12) Fractional hexadecimal to decimal :-

(i) $(.ABC)_{16} = (\ )_{10}$

$= A \times 16^{-1} + B \times 16^{-2} + C \times 16^{-3}$

$= 10 \times \frac{1}{16} + 11 \times \frac{1}{16^2} + 12 \times \frac{1}{16^3}$

$= \frac{10}{16} + \frac{11}{256} + \frac{12}{4096} = \frac{2560+176+12}{4096} = \frac{2748}{4096}$

$= (0.670)_{16}$ (Ans)

Boolean Algebra :-

(i) It is used to analyze and simplify the digital (logic) circuits.

(ii) It uses only the binary numbers i.e. 0 & 1.

(iii) It is also called as binary algebra or logical algebra.

(iv) Boolean algebra was invented by George boole in 1854.

Rules in Boolean Algebra :-

(i) Variable used can have only 2 values binary 1 for high & binary 0 for low.

(ii) complement of a variable is represented by an overbar (−). Thus complement of variable B is represented as $\bar{B}$. Thus if $B=0$ then $\bar{B}=1$ and $B=1$ then $\bar{B}=0$.

(iii) OR ing of variables is represented by a plus (+) sign between them. For example ORing of A, B, C is represented as $A+B+C$.

Boolean laws :-

There are 6 types of boolean laws.

(i) communicative law :- (1) $A \cdot B = B \cdot A$

(2) $A+B = B+A$

(ii) Associative law :- (1) $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

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

(iii) Distributive law :- (1) $A \cdot (B+C) = A \cdot B + A \cdot C$

(iv) AND law :- (1) $A \cdot 0 = 0$ (3) $A \cdot A = A$

(2) $A \cdot 1 = A$ (4) $A \cdot \bar{A} = 0$

(v) OR law :- (1) $A+0 = A$ (3) $A+1 = A$

(2) $A+A = A$ (4) $A+\bar{A} = 1$

(vi) Inversion law :- $\bar{\bar{A}} = A$

Postulates of Boolean Algebra :-

(1) $X = 1$ or 0

(2) $1 \cdot 1 = 1$

(3) $1 \cdot 0 = 0 \cdot 1 = 0$

(4) $0 \cdot 0 = 0$

(5) $0 + 0 = 0$

(6) $1 + 0 = 0 + 1 = 1$

(7) $1 + 1 = 1$

(8) $\bar{1} = 0$

(9) $\bar{0} = 1$

Logic gates :-

(i) Logic gates are the basic building blocks of any digital system.

(ii) It is an electronic circuit having one or more than one input and only one output.

(iii) The relationship between the input and the output is based on a certain logic.

(iv) In Boolean Algebra 3 boolean operator OR, AND, NOT, XOR.

The AND Operator :-

(i) The AND is used to define for 2 integers X & y, if x & y both are 1 and the result is 1.

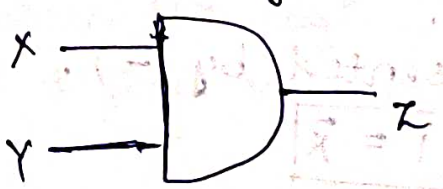
(ii) The Dot (·) symbol is used for AND operator.

$$\boxed{Z = X \cdot Y}$$

Truth table

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Logic diagram



OR operator :-

(i) The OR operator is used to define for 2 integers x & y if x & y both are 0 & the result is 0.

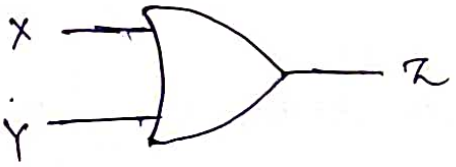
(ii) The plus (+) symbol is used for OR operator.

$$\boxed{Z = X + Y}$$

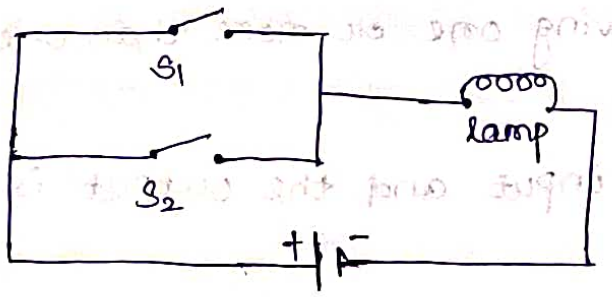
Truth table :-

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Logical diagram :-

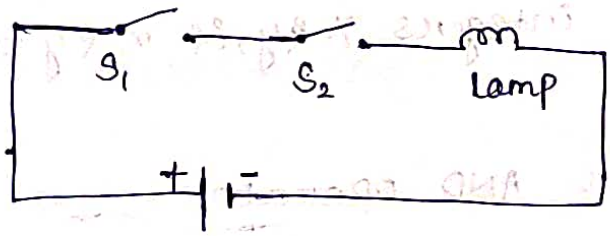


circuit diagram :-



(Parallel)

circuit diagram for AND Gate :-



(series)

NOT :-

(i) NOT Gate is also known as inverter. It has one input x and one output y.

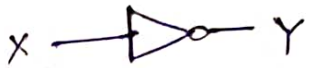
(ii) It is represented by (–).

$$\boxed{Y = \bar{X}}$$

Truth table:-

X	$Y = \bar{X}$
0	1
1	0

Logic diagram



NAND:-

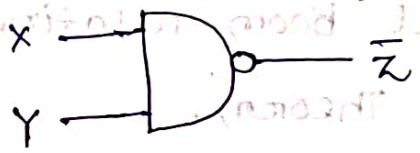
(i) A NOT-AND ~~operatore~~ operation is known as NAND operation.

(ii) It defines for 2 integers x & y, if x & y both are 1 & the result is 0.

Truth table :-

X	Y	$\bar{Z}$
0	0	1
0	1	1
1	0	1
1	1	0

Logic diagram:-



NOR:-

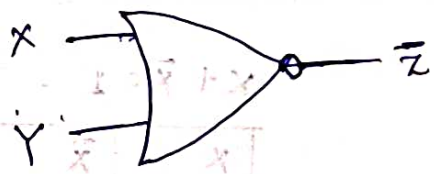
(i) A NOT-OR operation is known as NOR operation.

(ii) It defines for 2 integers x & y, if x & y both are 0 & the result is 1.

Truth table:-

X	Y	$\bar{Z}$
0	0	1
0	1	1
1	0	1
1	1	0

Logic diagram :-



XOR:-

(i) XOR or EX-OR Gate is a special type of gate.

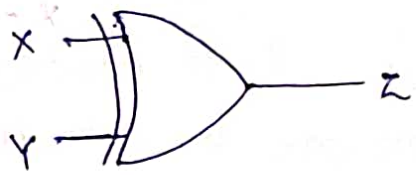
(ii) The EX-OR differentiate between the logical variables which are identical & the logical variable which are different.

(X, Y) $\boxed{Z = X \oplus Y}$

Truth table:-

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Logic diagram:-



It defines for 2 integers x & y if x & y both are 0 & the result is 1 & if x & y both are 1 & the result is 0.

Theorems of Boolean Algebra:-

(1) A fundamental Boom relating to Boolean variable called Boolean Theorem.

(1) complementation law:-

i) The term complementation to invert $A=1$ $\bar{A}=0$ / $A=0$ $\bar{A}=1$.

ii) In Boolean Algebra this means to convert 0's to 1's and 1's to 0's.

Theorem - 1(a) - $x \cdot \bar{x} = 0$

1(b) - $x + \bar{x} = 1$

Truth table:-

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

X	$\bar{X}$	$X \cdot \bar{X}$
0	1	0
1	0	0

$x + \bar{x} = 1$

X	$\bar{X}$	$X + \bar{X}$
0	1	1
1	0	1

(2) commutative law:-

i) This law allows the change in the position of the logical variable in ORing and ANDing i.e.

Theorem - 2(a) - $X + Y = Y + X$

2(b) - $X \cdot Y = Y \cdot X$

Truth table:-

$X+Y = Y+X$

X	Y	X+Y	Y+X
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

$X \cdot Y = Y \cdot X$

X	Y	X·Y	Y·X
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

③ AND law:-

ANDing operation of a variable with itself or its complement is of special significance.

Theorem - 3(a) - $X \cdot 0 = 0$

3(b) - $X \cdot 1 = X$

3(c) - $X \cdot X = X$

3(d) - $X \cdot \bar{X} = 0$

Truth table:-

$X \cdot 0 = 0$

X	Y=0	Z=0
0	0	0
1	0	0

$X \cdot 1 = X$

X	Y=1	Z=X
0	1	0
1	1	1

$X \cdot X = X$

X	Y=X	Z=X
0	0	0
1	1	1

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

X	Y=$\bar{X}$	Z=0
0	1	0
1	1	1

(4) OR Law:-

The ORing operation of a variable with itself or its complement is of special significance.

Theorem :- 4(a) - $X + 0 = X$

4(b) - $X + 1 = 1$

4(c) - $X + X = X$

4(d) - $X + \bar{X} = 1$

Truth table:-

$X + 0 = X$

X	Y=0	Z=X
0	0	0
1	0	1

$X + 1 = 1$

X	Y=1	Z=1
0	1	1
1	1	1

$X + X = X$

X	Y=X	Z=X
0	0	0
1	1	1

$X + \bar{X} = 1$

X	Y=$\bar{X}$	Z=1
0	1	1
1	0	1

(5) Associative law :-

Theorm :- 5(a) - $x+(Y+Z) = (x+Y)+Z$

5(b) - $x\cdot(Y\cdot Z) = (x\cdot Y)\cdot Z$

Truth table :-

$x+(Y+Z) = (x+Y)+Z$

X	Y	Z	X+Y	Y+Z	(X+Y)+Z	(X+Y)+Z
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	0	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Equal

$x\cdot(Y\cdot Z) = (x\cdot Y)\cdot Z$

X	Y	Z	Y·Z	X·Y	X·(Y·Z)	(X·Y)Z
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

Equal

(6) Distributive law :-

Theorem :- 6 (a) - $X \cdot (Y+Z) = (X \cdot Y) + (X \cdot Z)$

6 (b) - $X + (Y \cdot Z) = (X+Y) \cdot (X+Z)$

Truth table :-

$X \cdot (Y+Z) = (X \cdot Y) + (X \cdot Z)$

X	Y	Z	Y+Z	X·Y	X·Z	X·(Y+Z)	(X·Y)+(X·Z)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	0	1	1	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1

Equal

$X + (Y \cdot Z) = (X+Y) \cdot (X+Z)$

~~Truth~~

X	Y	Z	Y·Z	X+Y	X+Z	X+(Y·Z)	(X+Y).(X+Z)
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0
0	1	0	0	1	0	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Equal

Example:-

$x + (\bar{x} \cdot Y) = x + Y$

Truth table:-

x	Y	$\bar{x}$	$\bar{x} \cdot Y$	$x + (\bar{x} \cdot Y)$	$x + Y$
0	0	1	0	0	0
1	0	0	0	1	1
0	1	1	1	1	1
1	1	0	0	1	1

Equal

Proof:-

$x + \bar{x} \cdot Y$

$\Rightarrow x \cdot 1 + \bar{x} \cdot Y$ ~~(or $1+Y=1$)~~

$\Rightarrow x(1+Y) + \bar{x} \cdot Y$ $(1+Y=1)$

$\Rightarrow x + x \cdot Y + \bar{x} \cdot Y$

$\Rightarrow x + Y \cdot x + Y \cdot \bar{x}$ $(x \cdot Y = Y \cdot x)$

$\Rightarrow x + Y(x + \bar{x})$ $(x + \bar{x} = 1)$

$\Rightarrow x + Y \cdot 1$

$\Rightarrow x + Y$ (Proved)

Distributive law prove:-

① $x + (Y \cdot Z) = (x+Y) \cdot (x+Z)$

$\Rightarrow x + Y \cdot Z$

$\Rightarrow x \cdot 1 + Y \cdot Z$ $(x \cdot 1 = x)$

$\Rightarrow x(1+Y) + Y \cdot Z$ $(Y+1=1)$

$\Rightarrow x + x \cdot Y + Y \cdot Z$

$\Rightarrow x \cdot 1 + x \cdot Y + Y \cdot Z$ $(x \cdot 1 = x)$

$\Rightarrow x \cdot (1+Z) + x \cdot Y + Y \cdot Z$ $(1+Z=1)$

$\Rightarrow x + x \cdot Z + x \cdot Y + Y \cdot Z$

$\Rightarrow x \cdot x + x \cdot Z + x \cdot Y + Y \cdot Z$ $(x \cdot x = x)$

$\Rightarrow x(x+Z) + Y(x+Z)$

$\Rightarrow (x+Z)(x+Y)$ (Proved)

(7) Absorption law :-

Theorm 7(a) - $X + (X \cdot Y) = X$

7(b) - $X \cdot (X + Y) = X$

Prove.

ⓐ $X + X \cdot Y = X \cdot 1 + X \cdot Y$ $\quad (X \cdot 1 = X)$

$= X(1 + Y)$ $\quad (1 + Y = 1)$

$= X \cdot 1$

$= X$ (Proved)

ⓑ $X \cdot (X + Y) = X \cdot X + X \cdot Y$

$= X + X \cdot Y$ $\quad (X \cdot X = X)$

$= X(1 + Y)$ $\quad (1 + Y = 1)$

$= X \cdot 1$

$= X$ (Proved)

Rules :-

(1) $\bar{0} = 1$

(2) $\bar{1} = 0$

(3) $X = 0, \bar{X} = 1$

(4) $X = 1, \bar{X} = 0$

(5) $\bar{\bar{X}} = X$

(6) $X \cdot 0 = 0$

(7) $X \cdot 1 = X$

(8) $X \cdot X = X$

(9) $X \cdot \bar{X} = 0$

(10) $X + 0 = X$

(11) $X + 1 = 1$

(12) $X + X = X$

(13) $X + \bar{X} = 1$

(14) $X + Y = Y + X$

(15) $X \cdot Y = Y \cdot X$

(16) $X + (Y + Z) = (X + Y) + Z$

(17) $X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$

(18) $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$

(19) $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$

(20) $X + (\bar{X} \cdot Y) = X + Y$

(21) $\overline{X + Y} = \bar{X} \cdot \bar{Y}$

(22) $\overline{X \cdot Y} = \bar{X} + \bar{Y}$

$X \cdot 1 = X$

$1 + Y = 1$

$1 + X = 1$

(8) Demorgan's law :-

Theorm 8(a) - $\overline{X + Y} = \bar{X} \cdot \bar{Y}$

8(b) - $\overline{X \cdot Y} = \bar{X} + \bar{Y}$

Truth table:-

(a) $\overline{X+Y} = \overline{X}\cdot\overline{Y}$

X	Y	$\overline{X}$	$\overline{Y}$	$X+Y$	$\overline{X+Y}$	$\overline{X}\cdot\overline{Y}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

Equal

(b) $\overline{X\cdot Y} = \overline{X}+\overline{Y}$

X	Y	$\overline{X}$	$\overline{Y}$	$X\cdot Y$	$\overline{X\cdot Y}$	$\overline{X}+\overline{Y}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

Equal

Ex:-

$a+(\bar{a}\cdot b) = a+b$

$\Rightarrow a\cdot 1 + \bar{a}\cdot b$

$\Rightarrow a(1+b) + \bar{a}\cdot b$ $\quad (1+b=1)$

$\Rightarrow a + a\cdot b + \bar{a}\cdot b$

$\Rightarrow a + b(a+\bar{a})$ $\quad (a+\bar{a}=1)$

$\Rightarrow a + b\cdot 1$

$\Rightarrow a+b$ (Proved)

Truth table:-

$a+(\bar{a}\cdot b) = a+b$

a	b	$\bar{a}$	$\bar{a}\cdot b$	$a+\bar{a}\cdot b$	$a+b$
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

Equal

Duality Principle:-

If Plus (+) is interchanged with dot (·) & 0 is interchanged with 1 ~~and~~ that is called duality principle.

$$\boxed{\begin{array}{l} (1)\ a+(a\cdot b)=a \\ (2)\ a\cdot(a+b)=a \end{array}}$$

(1) $a+a\cdot b=a$

$\Rightarrow a\cdot 1+a\cdot b$

$\Rightarrow a(1+b)$

$\Rightarrow a\cdot 1$

$\Rightarrow a$ (proved)

(2) $a\cdot(a+b)=a$

$\Rightarrow a\cdot a+a\cdot b$

$\Rightarrow a+a\cdot b$ $(a\cdot a=a)$

$\Rightarrow a(1+b)$ $(1+b=1)$

$\Rightarrow a\cdot 1$

$\Rightarrow a$ (Proved)

Truth table:-

(1) $a+(a\cdot b)=a$

a	b	a·b	a+(a·b)
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Equal

(2) $a\cdot(a+b)=a$

a	b	a+b	a·(a+b)
0	0	0	0
0	1	1	0
1	0	1	1
1	1	1	1

Equal

Ex:-

$\bar{\bar{a}}=a$

When $a=0$ $\Rightarrow \bar{a}=1$

$\bar{\bar{a}}=\bar{1}=0$ So when $a=0$

$\bar{\bar{a}}=0=a$ (Proved)

When $a=1$ $\Rightarrow \bar{a}=0$

$\bar{\bar{a}}=\bar{0}=1$ So when $a=1$

$\bar{\bar{a}}=1=a$ (Proved)

→ $\overline{\overline{a\cdot b}}=a\cdot b$

$\overline{a\cdot b}=\bar{a}+\bar{b}$

$\Rightarrow \overline{\bar{a}+\bar{b}}=\bar{\bar{a}}\cdot\bar{\bar{b}}$

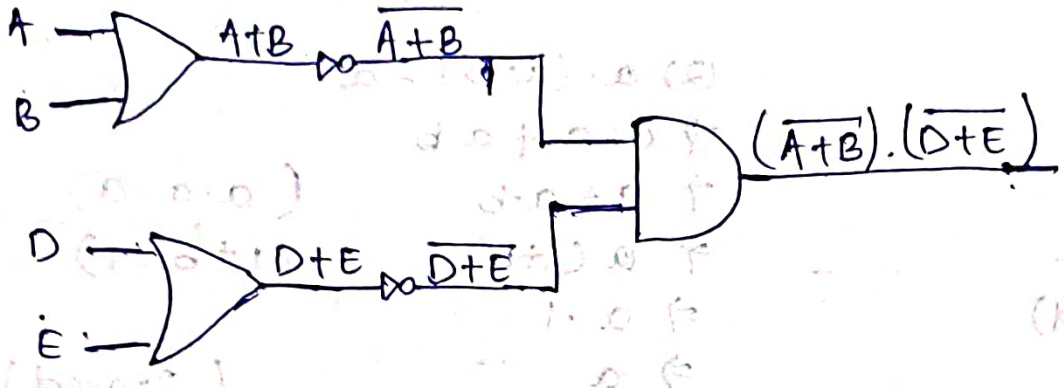
$=ab$ (Proved)

→ $\overline{\overline{a+b}} = a+b$

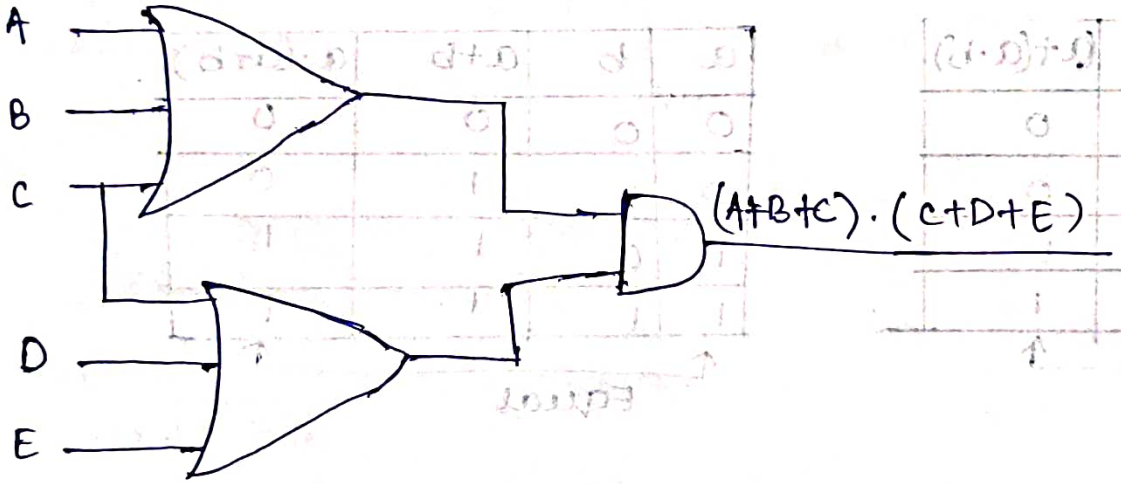
$$\overline{a+b} = \bar{a}\cdot\bar{b}$$

$$= \overline{\bar{a}\cdot\bar{b}} = \bar{\bar{a}}+\bar{\bar{b}} = a+b$$ (Proved)

→ $(\overline{A+B}).(\overline{D+E})$



→ $(A+B+C)\cdot(C+D+E)$



Reducing Boolean Expression :-

When a designer desires to acheive an electronic circuit making use of logic operation he has to reduced every boolean equation to simplest form. so as to reduced cos of the circuit to minimum.

*There are some procedures to reduce Boolean Expression

(1) To remove parenthesis multipy all variables.

(2) collect the identical term & using identity law.

(3) search for the term having a variable and its negation

Ex:- $X \cdot \bar{X} \cdot Y = 0 \cdot Y = 0$

(4) collect the pair of terms which are identical except for 1 variable.

Ex:- $U \cdot X \cdot Y \cdot Z + U \cdot X \cdot Z$

$= U \cdot X \cdot Z (Y+1) = U \cdot X \cdot Z \cdot 1 = U \cdot X \cdot Z$

Ex:- $X \cdot Y \cdot Z + X \cdot Y$

$= X \cdot Y (Z+1) = X \cdot Y$

Presidency of operator:-

(1) Scan the expression from left to right.

(2) First solve the Parenthesis.

(3) Now perform NOT / complement operation

(4) All '.' (AND) operations are performed. after that

(5) In the end perform or operation.

Ex:- $X + X + Y + Y + Y$

$= (X+X) + Y + (Y+Y) = X + Y + Y$

$= X + (Y+Y) = X + Y$

Ex:- $X \cdot Y \cdot Y \cdot Z \cdot Z \cdot Z$

$= X \cdot (Y \cdot Y) \cdot Z \cdot (Z \cdot Z) = X \cdot Y \cdot Z \cdot Z$

$= X \cdot Y \cdot Z$

Ex:- $X \cdot X \cdot Y + X \cdot Y \cdot Y + Y \cdot \bar{Y} \cdot X \cdot X$

$= (X \cdot X) \cdot Y + X \cdot (Y \cdot Y) + (Y \cdot \bar{Y}) X \cdot X$

$= X \cdot Y + X \cdot Y + 0 \cdot X$

$= X \cdot Y$

Ex:- $X \cdot \bar{Y} + \bar{X} \cdot Y + X \cdot Y + \bar{X} \cdot \bar{Y}$

$= X \cdot \bar{Y} + X \cdot Y + \bar{X} \cdot Y + \bar{X} \cdot \bar{Y}$

$= X (\bar{Y} + Y) + \bar{X} (Y + \bar{Y})$

$= X \cdot 1 + \bar{X} \cdot 1 = X + \bar{X} = 1$

Ex:- $X+\bar{X}\cdot Y+\bar{X}\cdot Z$

$= X+\bar{X}\cdot Y+\bar{X}\cdot Z$

$= X+\bar{X}\cdot(Y+Z)$ $\quad (X+(\bar{X}\cdot Y)=X+Y)$

$= X+Y+Z$

Ex:- $(\bar{X}+\bar{Y})\cdot X\cdot\bar{Y}\cdot Z$

$= \bar{X}\cdot X\cdot\bar{Y}\cdot Z+\bar{Y}\cdot X\cdot\bar{Y}\cdot Z$

$= 0+X\cdot\bar{Y}\cdot\bar{Y}\cdot Z$ $\quad$ (If ~~X~~ value 1 or 0)

$= X\cdot\bar{Y}\cdot Z$

Ex:- $X\cdot[Y+Z\cdot\overline{X\cdot Y+X\cdot Z}]$

$= X\cdot[Y+Z(\overline{X\cdot Y}\cdot\overline{X\cdot Z})]$

$= X\cdot Y+X\cdot Z(\overline{X\cdot Y}\cdot\overline{X\cdot Z})$

$= X\cdot Y+X\cdot Z\cdot\overline{X\cdot Z}\cdot\overline{X\cdot Y}$ $\quad (X\cdot\bar{X}=0)$

$= X\cdot Y+0\cdot\overline{X\cdot Y}$

$= X\cdot Y$

Ex:- $X\cdot\bar{Y}+\bar{X}\cdot Y+X\cdot Y+\overline{X\cdot Y}$

$= X\cdot\bar{Y}+X\cdot Y+\bar{X}\cdot Y+\bar{X}+\bar{Y}$ $\quad (\overline{X\cdot Y}=\bar{X}+\bar{Y})$

$= X(\bar{Y}+Y)+\bar{X}(Y+1)+\bar{Y}$

$= X\cdot 1+\bar{X}\cdot 1+\bar{Y} = X+\bar{X}+\bar{Y}$

$= 1+\bar{Y}$

$= 1$

Ex:- $\overline{(YZ+\bar{X}U)\cdot(X\bar{Y}+Z\bar{U})}$

$= \overline{YZ\cdot X\bar{Y}+YZ\cdot Z\bar{U}+\bar{X}U\cdot X\bar{Y}+\bar{X}U\cdot Z\bar{U}}$

$= \overline{0+0+0+0}$

$= \bar{0} = 1$

Ex:- $XY+XYZ+\bar{X}Y+X\bar{Y}Z$

$= XY+\bar{X}Y+XYZ+X\bar{Y}Z$

$= Y(X+\bar{X})+XZ(Y+\bar{Y})$

$= Y\cdot 1+XZ\cdot 1$

$= Y+XZ$

Solve the boolean expression

$$\overline{X\bar{Y}+XYZ+X(Y+X\cdot\bar{Y})}$$

$$= \overline{X(\bar{Y}+YZ)+X(Y+X\cdot\bar{Y})}$$

$$= \overline{X(\bar{Y}+Z)+X(Y+X)} \qquad (X+\bar{X}\cdot Y = X+Y)$$

$$= \overline{X\bar{Y}+XZ+X(Y+X)} \qquad (X+\bar{X}\cdot Y = X+Y)$$

$$= \overline{X\bar{Y}+XZ+XY+X\cdot X}$$

$$= \overline{X\bar{Y}+XZ+X\cdot X+XY}$$

$$= \overline{X(\bar{Y}+Z)+X+XY}$$

$$= \overline{X(\bar{Y}+Z)+X(1+Y)}$$

$$= \overline{\bar{X}+\overline{(\bar{Y}+Z)}+X\cdot 1}$$

$$= \overline{\bar{X}+\bar{\bar{Y}}\cdot\bar{\bar{Z}}+X}$$

$$= \overline{\bar{X}+Y\cdot\bar{Z}+X}$$

$$= \overline{X+\bar{X}+Y\cdot\bar{Z}}$$

$$= \overline{1+Y\cdot\bar{Z}} \qquad (\text{If } Y\cdot\bar{Z} \text{ is constant})$$

$$= \bar{1} = 0$$

Ex:- $XY+\bar{X}Z+X\bar{Y}Z(XY+Z)$

$$= XY+\bar{X}Z+X\bar{Y}Z\,XY+X\bar{Y}Z\cdot Z$$

$$= XY+\bar{X}Z+0+X\bar{Y}Z \qquad (Y\cdot\bar{Y}=0)$$

$$= XY+\bar{X}Z+X\bar{Y}Z$$

$$= XY+\bar{X}+\bar{Z}+X\bar{Y}Z$$

$$= XY+\bar{Z}+\bar{X}+X\bar{Y}Z$$

$$= XY+X\bar{Y}Z+\bar{Z}+\bar{X}$$

$$= X(Y+\bar{Y}Z)+\bar{Z}+\bar{X}$$

$$= X(Y+Z)+\bar{Z}+\bar{X} \qquad (X+(\bar{X}\cdot Y) = X+Y)$$

$$= XY+XZ+\bar{Z}+\bar{X}$$

$$= XY+\bar{X}+Z+\bar{Z}$$

$$= \bar{X}+Y+XZ \qquad (\text{Rule 20}) = X+\bar{X}\cdot Y = X+Y)$$

$$= X+\bar{X}+Y+Z$$

$= 1 + Y + Z$. If $Y + Z$ is constant, then answer is

$= 1$

Q) Prove the following identities using boolean algebra:-

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

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

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

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

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

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

$= \bar{C} \cdot 1$

$= \bar{C} = R.H.S.$ (Proved)

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

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

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

$= B \cdot 1 + AC \cdot 1$

$= B + AC$

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

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

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

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

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

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

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

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

$= \bar{A}(\bar{C} + \bar{C}\bar{B}) + AB$ $\quad (\text{As } X + \bar{X} \cdot Y = X + Y)$

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

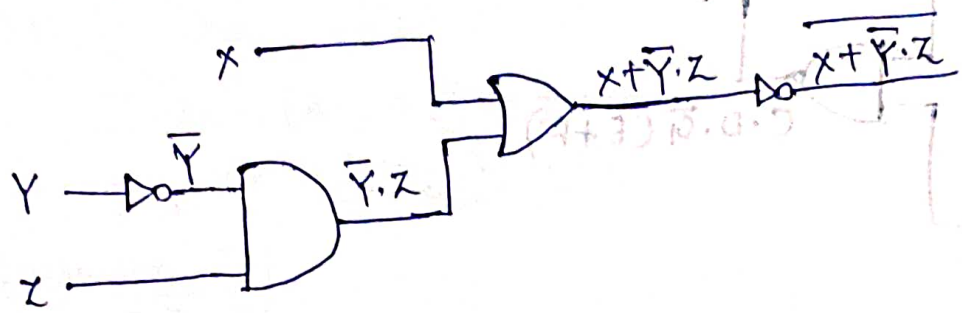
$= AB + \bar{A}\bar{B} + \bar{A}C = R.H.S.$ (Proved)

Logic Diagram and Boolean expression:-

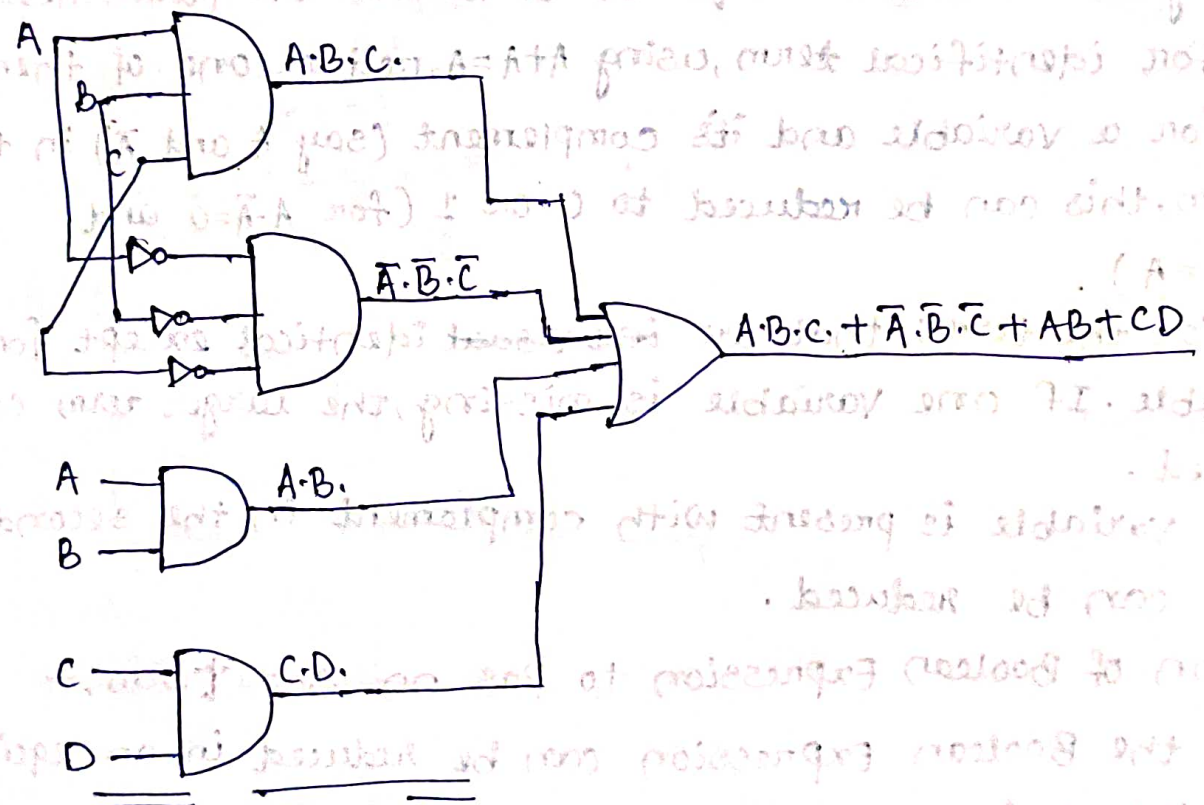
We can implement any Boolean Expression by using the standard logic gates.

For converting an expression into a logic diagram the easiest way to start from output & proceed towards input.

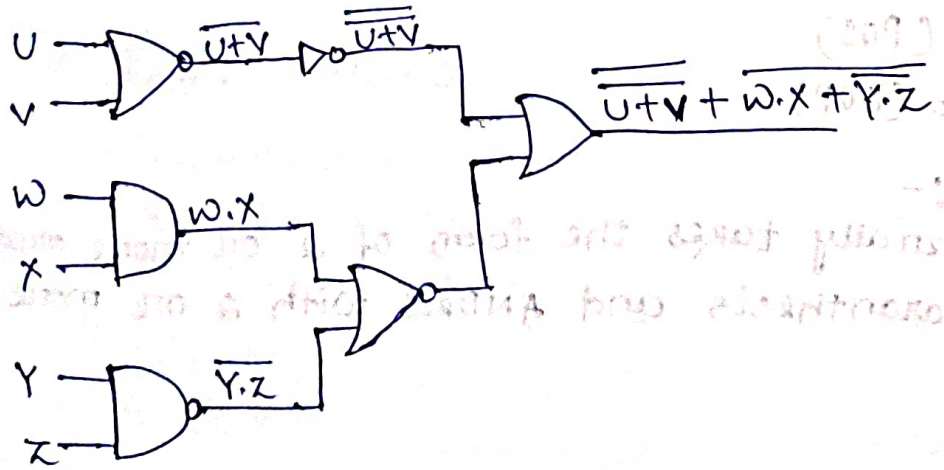
① $\overline{X+\overline{Y}\cdot Z}$



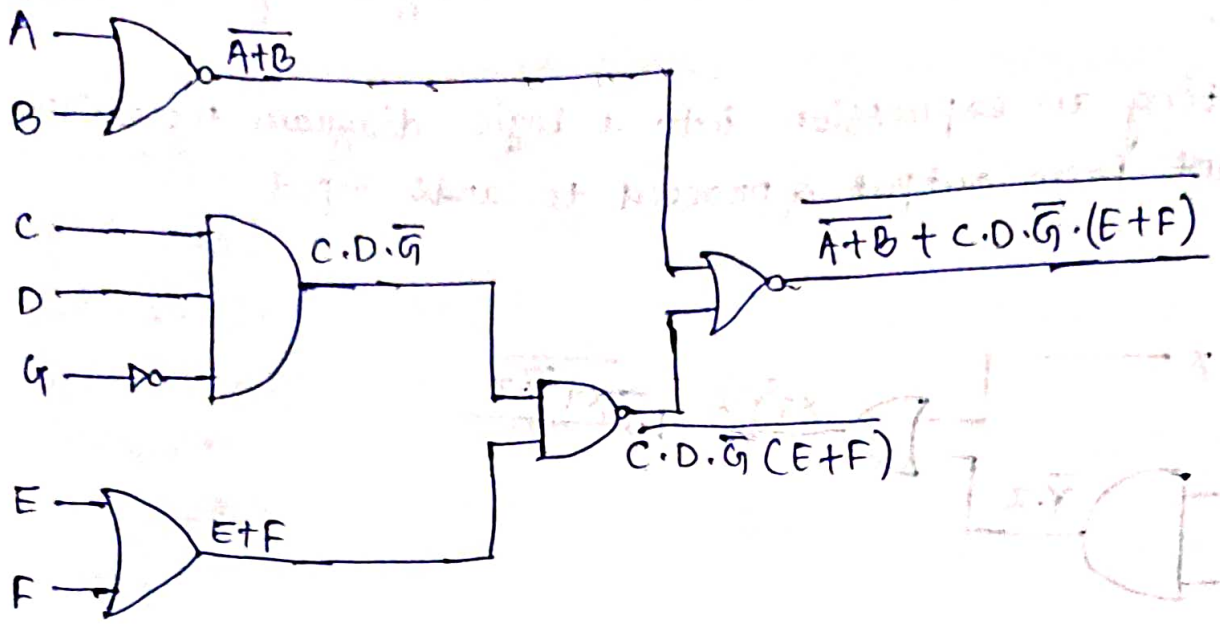
② $X = A\cdot B\cdot C + \overline{A}\cdot\overline{B}\cdot\overline{C} + AB + C\cdot D$



③ $X = \overline{\overline{U+V}} + \overline{W\cdot X + \overline{Y\cdot Z}}$



(4) $\overline{A+B+C.D.\bar{G}.(E+F)}$



General procedure for Reducing Boolean Expression:-

(i) Multiply all variables required to remove the paranthesis.

(ii) Look for identifical term, using $A+A=A$, reduce one of them.

(iii) Look for a variable and it's complement (say A and $\bar{A}$) in the same term, this can be reduced to 0 or 1 (for $A.\bar{A}=0$ and for $A+\bar{A}=A$).

(iv) Look for the terms that are ~~indivisual~~ identical except for one variable. If one variable is missing, the larger term can be dropped.

(v) If one variable is present with complement in the second term. It can be reduced.

Reduction of Boolean Expression to POS or SOP term:-

Most of the Boolean Expression can be reduced in an equ one of the 2 forms.

(a) Product of Sum (POS)

(b) Sum of Product (SOP)

(a) product of Sum:-

POS expression generally takes the form of 2 or more ored variables within paranthesis and ANDed with 2 or more other variables.

Ex:- $a = (x+y)\cdot(y+z)$

$a = (y+\bar{z}+\bar{b})\cdot(y+z+b)$

$a = (x+\bar{z})\cdot(\bar{y}+E)\cdot(z+y)$

(b) Sum of product :-

SOP expression usually takes the form of 2 or more variabl ANDed together ored with 2 or more variables.

Ex:- $y = (A\cdot B\cdot C) + (\bar{C}D) + (\bar{E}F)$

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

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

· Convert SOP :-

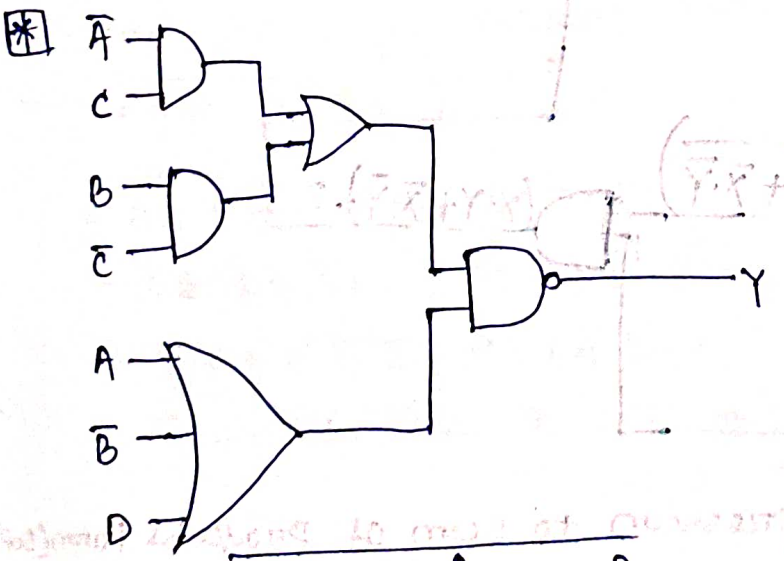
$y = \overline{A\bar{B} + \bar{C}D}$

$= \overline{A\bar{B}}\cdot\overline{\bar{C}D}$ (by Demorgan's law)

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

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

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



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

$Y = \overline{\bar{A}C + B\bar{C}} + \overline{A+\bar{B}+D}$ (by Demorgan's law)

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

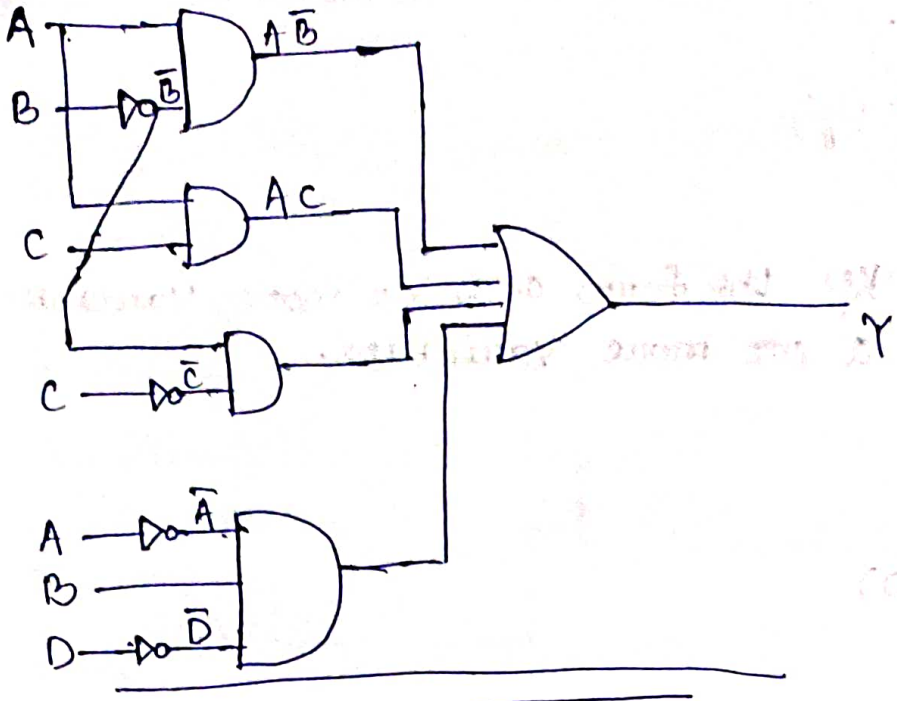
$= (\bar{\bar{A}}+\bar{C})(\bar{B}+\bar{\bar{C}}) + \bar{A}\cdot B\cdot\bar{D}$

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

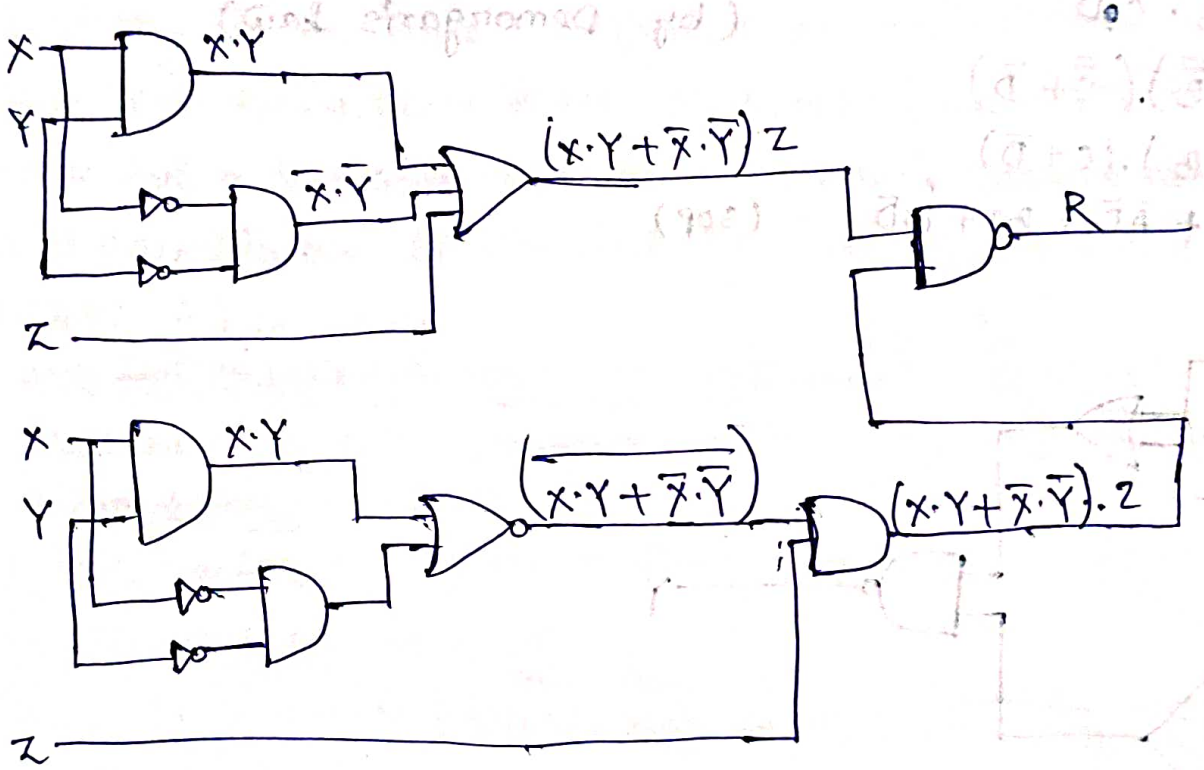
$= A\bar{B} + AC + \bar{B}\bar{C} + \bar{C}\cdot C + \bar{A}\cdot B\cdot\bar{D}$.

$= A\bar{B} + AC + \bar{B}\bar{C} + \bar{A}B\cdot\bar{D}$

(convert to circuit diagram)



$R = (X\cdot Y + \bar{X}\cdot\bar{Y} + Z)\,\overline{(X\cdot Y + \bar{X}\cdot\bar{Y})\cdot Z}$



convert the following expression to sum of products form(s

(1) $(X+Y)(\bar{Y}+Z)(\bar{X}+Z)$

$= (X+Y)(\bar{Y}\cdot\bar{X} + \bar{Y}Z + Z\bar{X} + ZZ)$

$= (X+Y)(\bar{Y}\bar{X} + \bar{Y}Z + Z\bar{X} + Z)$

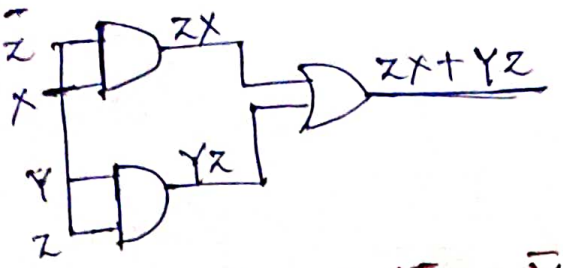
$= X\bar{Y}\bar{X} + X\bar{Y}Z + XZ\bar{X} + XZ + Y\bar{Y}X + X\bar{Y}Z + YZ\bar{X} + YZ$

$= 0 + X\bar{Y}Z + 0 + ZX + 0 + 0 + YZ\bar{X} + YZ$

$= X\bar{Y}Z + ZX + YZ\bar{X} + YZ$

$= ZX(\bar{Y}+1) + YZ(\bar{X}+1)$

$= ZX + YZ$ (As $\bar{Y}+1 = 1$ & $\bar{X}+1 = 1$)



(2) $(X+Z)(X\bar{Y}+XZ)(\bar{X}Z+\bar{Y})$

$= (X+Z)(X\bar{Y}\bar{X}Z + X\bar{Y}\bar{Y} + XZ\bar{X}Z + XZ\bar{Y})$

$= (X+Z)(0 + X\bar{Y} + 0 + XZ\bar{Y})$

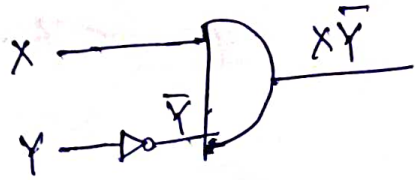
$= (X+Z)(X\bar{Y} + XZ\bar{Y})$

$= XX\bar{Y} + XXZ\bar{Y} + ZX\bar{Y} + ZXZ\bar{Y}$

$= X\bar{Y} + XZ\bar{Y} + XZ\bar{Y} + XZ\bar{Y}$ (As $Z \cdot Z = Z$)

$= X\bar{Y} + XZ\bar{Y}$

$= X\bar{Y}(1+Z) = X\bar{Y}$ $(1+Z=1)$



Q) Draw the logic diagram for following Boolean Expression?

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

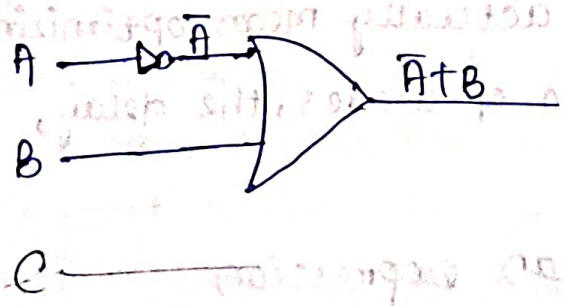
$= \overline{A\bar{B}} + \overline{A+C} + \bar{A}B \cdot \overline{A+\bar{B}\cdot\bar{C}}$ (Demorgan's law)

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

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

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

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



Minimize:-

$$R = \overline{(X\cdot Y + \bar{X}\cdot\bar{Y} + Z)\,(\overline{X\cdot Y + \bar{X}\cdot\bar{Y}})}\cdot Z$$

$$= \overline{(X\cdot Y + \bar{X}\cdot\bar{Y} + Z)} + \overline{\overline{(X\cdot Y + \bar{X}\cdot\bar{Y})}} + \bar{Z}$$

$$= (\overline{X\cdot Y}\cdot\overline{\bar{X}\cdot\bar{Y}}\cdot\bar{Z}) + (X\cdot Y + \bar{X}\cdot\bar{Y}) + \bar{Z}$$

$$= (\bar{X}+\bar{Y})(\bar{\bar{X}}+\bar{\bar{Y}})(\bar{\bar{Z}}) + (XY + \bar{X}\cdot\bar{Y}) + \bar{Z}$$

$$= (\bar{X}+\bar{Y})(X+Y)\cdot\bar{Z} + (XY + \bar{X}\cdot\bar{Y}) + \bar{Z}$$

$$= (\cancel{\bar{X}X}^{0} + \bar{X}Y + \bar{Y}X + \cancel{\bar{Y}Y}^{0}\cdot\bar{Z}) + (XY + \bar{X}\cdot\bar{Y}) + \bar{Z}$$

$$= (\bar{X}Y + \bar{Y}X)\bar{Z} + XY + \bar{X}\cdot\bar{Y} + \bar{Z}$$

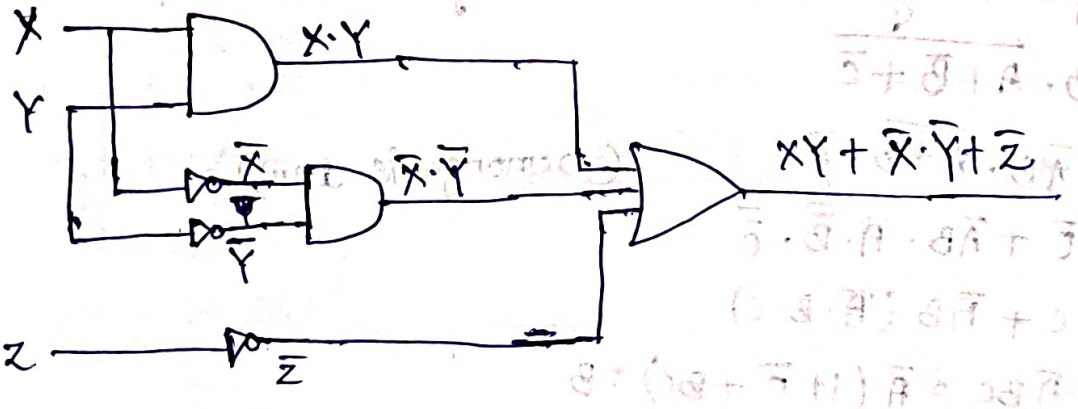
$$= \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XY + \bar{X}\cdot\bar{Y} + \bar{Z}$$

$$= XY + \bar{X}\cdot\bar{Y} + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + \bar{Z}$$

$$= XY + \bar{X}\cdot\bar{Y} + \bar{Z}(\bar{X}Y + X\bar{Y} + 1)$$

$$= XY + \bar{X}\cdot\bar{Y} + \bar{Z}(1)$$

$$= XY + \bar{X}\cdot\bar{Y} + \bar{Z}$$



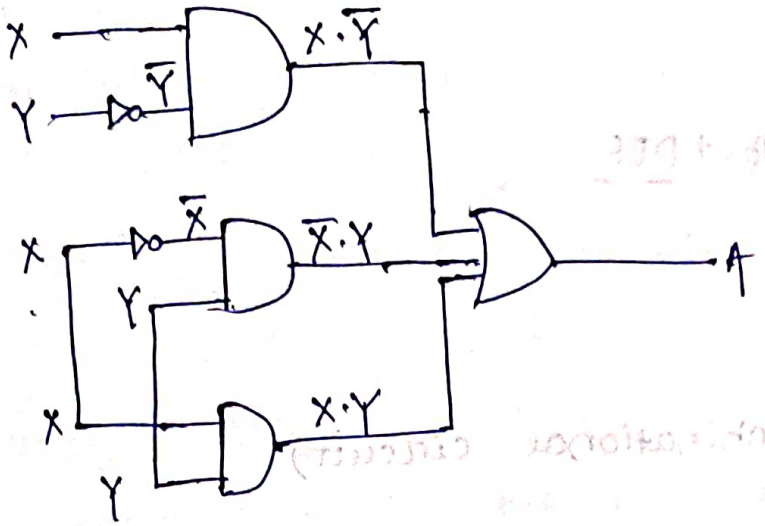
Minimization Techniques :-

By simplyifying or minimizing we actually mean optimizing o minimizing total no. of gates, no of diodes, the delay, maximizing the reliability e.t.c.

Let us consider following Boolean expression

For ex, $X\cdot\bar{Y} + \bar{X}\cdot Y + X\cdot Y = A$

For implementing these circuit we required 6 gates as shown in the figure below.

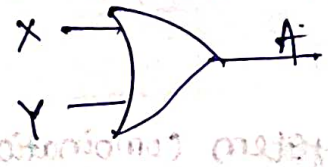


* $X \cdot \bar{Y} + \bar{X} \cdot Y + XY = A$

$A = X \cdot \bar{Y} + Y(X + \bar{X})$

$= X \cdot \bar{Y} + Y \cdot 1$

$= X \cdot \bar{Y} + Y = X + Y$ (Rule no. 20)



Simplification of logic expression is very important as it states the hardware require to design a specific system.

A designer is always try to make optimum use of ICS (Integrated circuits) available in the design of digital system

Digital system can be classified into 2 categories.

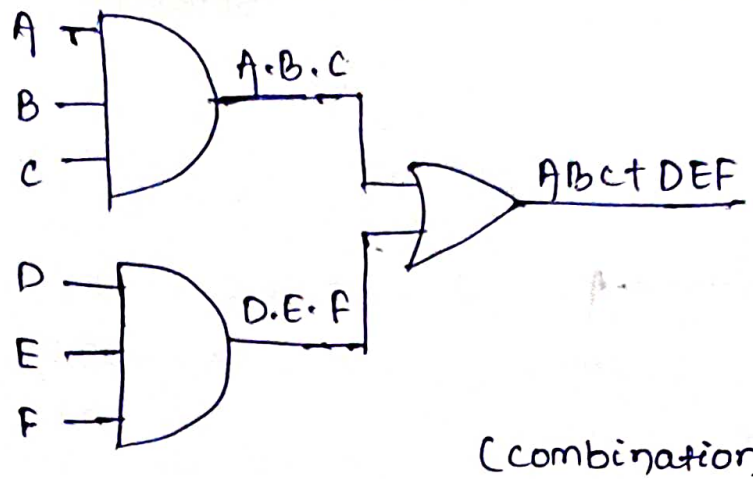
(1) combinational circuit

(2) sequencial circuit

(1) combinational circuit :-

In combinational circuits the output is obtained at every time as a certain logical combination of the input signals i.e. at any instant of time the output depends upon the inputs present at that instant of time. This means there is no effect of the input positions just before the instant, we consider the output i.e., there is no memory element in these circuits.

Ex:-



(combinational circuit)

(2) Sequencial circuit :-

In case of sequencial circuits the outputs at any instant of time depends upon the present inputs as well as past input output i.e. there are elements used to store post information (memory elements). In such circuits we not only consider the input combination but we also consider their accuran in time.

one may have in sequencial logic system combination logic subsystems.

Ex:- FLIP-FLOP

Minterms and maxterms :-

one of the most important theorms of Boolean Algebra states that any Boolean Expression can be expressed as SUM (OR) of Products (AND) of all the variables in the expression.

However the variable can be represented either in direct form or complement form.

Ex:- The term A+B can be expressed as sum of several Products. Each of them containing letters A & B. This product terms are known as minterm.

$$A+B = A\cdot 1 + B\cdot 1$$
$$= A(B+\bar{B}) + B(A+\bar{A})$$
$$= AB + A\bar{B} + BA + B\bar{A}$$
$$= AB + A\bar{B} + \bar{A}B$$

In this case, terms AB, $A\bar{B}$ & $\bar{A}B$ are known minterm because all the term contain both variable A & B.

In general expanding Boolean function into minterm the following systematic procedure should be followed.

(1) Write down all the terms of Boolean function.

(2) By writing x convert each term into min-term.

(3) Generate all possible minterm by using the combination of the x's in each term (Barred letters for 0 & unbarred letters for 1).

(4) Omit all the redundant terms.

Q) Find the minterm for $BC+A$

Ans→ Step-1 :- $BC+A$

Step-2 :- Write x to convert each term into minterm at the place where letters are missing i.e.

$xBC + Axx$

Step-3 :- and in Axx

$\Rightarrow ABC, A\bar{B}\bar{C}, A\bar{B}C, AB\bar{C}$

in $xBC = \bar{A}BC, ABC$ then

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

Step-4 :- Omit all the redundant terms i.e.

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

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

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

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

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

$= B(A+C) + A\bar{B}$ (Rule-20)

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

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

$= A + BC$

Q) Convert $A+B$ to minterms :

Ans → Step-1 :- $A+B$

Step-2 :- $AX+BX$

Step-3 :- $AX \rightarrow AB, A\bar{B}$

$BX \rightarrow BA, B\bar{A}$

Step-4 :- $A+B = AB+A\bar{B}+BA+B\bar{A}$

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

Hence minterm of $A+B = AB+A\bar{B}+\bar{A}B$ (Ans.)

Q) Find the minterm designator for the following minterm

(1) $X\bar{Y}Z\bar{U}$

(2) $XYZ\bar{U}W$

(3) $A\bar{B}\bar{C}\bar{D}$

Ans → (1) $X\bar{Y}Z\bar{U}$

→ Replacing barred letter by 0 & non barred letter by 1 i.e. (1010)

→ Decimal equivalent of this word is

$1010 = 1\times2^3+0\times2^2+1\times2^1+0\times2^0$

$= 8+0+2+0$

$= 10$

→ Expressing as decimal subscript of M

Hence $X\bar{Y}Z\bar{U} = M_{10}$

(2) $XYZ\bar{U}W$

→ Replacing barred letter by 0 & nonbarred letter by 1 i.e. (11001).

→ Decimal equivalent of this world is

$(11001) = 1\times2^4+1\times2^3+0\times2^2+0\times2^1+1\times2^0$

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

$= 25$

→ Expressing as decimal subscript of M,

Hence $XY\bar{Z}\bar{U}W = M_{25}$

(3) $\bar{A}\bar{B}\bar{C}\bar{D}$

→ Replacing barred letter by 0 & non barred letter by 1 i.e. (1000).

→ Decimal equivalent of this word is

$(1000) = 1\times2^3 + 0\times2^2 + 0\times2^1 + 0\times2^0$

$= 8+0+0+0$

$= 8$

→ Expressing as decimal subscript of M,

Hence $\bar{A}\bar{B}\bar{C}\bar{D} = M_8$

Maxterm (POS) :-

① $(X+\bar{Y})(\bar{X}+Y)$

→ Replacing barred num by 1 and unbarred num by 0.

$= \pi(01, 10)$

$= \pi(1,2)$ [decimal equivalent]

$= \pi M(1,2)$

② $(X+Y+\bar{Z}+\bar{U})(X+\bar{Y}+Z+\bar{U})$

→ Replacing barred num by 1 and unbarred num by 0.

$= \pi(0011, 0101)$

$= \pi(3,5)$ [decimal equivalent]

$= \pi M(3,5)$

Karnaugh's map :-

The Karnaugh's map also called as K-map is a method of simplyfying Boolean function in a systematic mathematical way.

It is another way of presenting the information given by the truth table i.e. it shows each output level of a Boolean Expression for each of the possible variables.

It requires that one should reduced the expression to an SOP form, from this one follows a systematic approach, which will always produce the simplest configuration possible for the logic circuit.

In this method, a group of squares (or cells) are created, each output level is placed in a separate square of the K-map.

All the squares are designated in a particular way. Each square of the map represents a minterm.

K-maps can be used to simplify expressions having 2,3,4,5 or 6 different input variables solving 5 or 6 variables K-maps is extremly combersum.

Karnaugh's map techniques :-

* Karnaugh maping is a method of plotting a boolean funct and it used to simplify the expression making use of tru table.

* To draw a Karnaugh map, a boolean function must be writt in a sum of products (SOP) form or in the minterm form.

* It isn't in the form then we should use the truth table to convert.

Ex:- Simplify the following Boolean Expression into SOP form.

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

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

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

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

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

$$= \bar{A}\cdot\bar{B}\cdot\bar{C}+\bar{A}\cdot\bar{D}+0 \qquad (\bar{A}\cdot A=0)$$

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

How to plot a K-map :-

* A K-map is a diagram consisting of squares. Each square of the map represents a minterm i.e. A term in the sum of minterm.
* A K-map for N variables is made up of 2^n squares.
→ Each square designates a product term of a Boolean Expression.
→ For product term, which are present in the expression, for 1's are written in those square which correspond to product terms not present in the expression.

K-map for 2 variables :-

Let us consider a map of 2 variable.

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

The 1st row is for $\bar{B}$ and
the 2nd row is for B similiarly
the 3rd row is for $\bar{A}$ &
the 4th row is for A.

Ex:- ① $Y = \bar{A}B + AB$
② $Y = \bar{A}\bar{B} + \bar{A}B$

① $Y = \bar{A}B + AB$

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

$Y = B$

② $Y = \bar{A}\bar{B} + \bar{A}B$

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

$Y = \bar{A}$

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

A	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

Steps for K-map :-

For using reduction by K-map method following steps should be followed.

① Transform Boolean expression to be reduced into an SOP f

② Determine and draw the desired number of squares (for n variables it is 2^n).

i.e. $2^2=4$, $2^3=8$, $2^4=16$, $2^5=32$

③ Fill in the appropriate squares of the K-map care should taken when moving form one square to an adjacent square or one variables changes.

④ Encircle adjacent squares in groups of 2, 4 or 8 (the mor adjacent squares encircled simpler will be final expression).

⑤ Find each term of the final SOP equation by determining which variable remains constant within each circle.

We show first consider 2 variable K-map and extended this concept to function of more than 2 variables.

Ex:- Draw a K-map for the following

(1) $F(A,B) = 0$

(2) $F(A,B) = 1$

(3) $F(A,B) = A$

(4) $F(A,B) = A+B$

(5) $F(A,B) = (A+B)(A+\bar{B})$

Solution:-

(1) $F(A,B) = 0$

It indicates that the given function is a null function & ~~to represent a null on the~~ to represent a null function on the Karnaugh map all the squares should be assigned 0 values.

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

(2) $F(A,B) = 1$

It indicates that the given function is an identification function and all the squares are assigned a value of 1.

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

(3) $F(A,B) = A$

It indicates that are term representing the function aren't in the form of minterm. Hence before drawing the K-map the expression should be converted into a minterm (SOP) form using Boolean Algebra.

$F(A,B) = A$

$= A(B+\bar{B})$ (As $B+\bar{B}=1$)

$= AB + A\bar{B}$

let it be X

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

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

So $X = A$

(4) $F(A,B) = A+B$

1st we convert this function into minterm i.e. SOP form.

$F(A,B) = A+B$

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

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

$= A\bar{B} + AB + \bar{A}B$ let it be y

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

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

$Y = A$

$Y = B$

By adding $Y = A+B$

(5) $F(A,B) = (A+B)(A+\bar{B})$

1st we convert this term into minterm i.e. SOP form

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

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

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

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

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

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

$= AB + A\bar{B}$ let it be y

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

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

so $Y = A$

K-map for 3 Variables :-

K-map for 3 variables means the number of squares i.e. $2^3 = 8$.

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

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

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

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

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

$= \bar{B}$

$X = \bar{B}$

A \ BC	00	01	11	10
0	M_0	M_1	M_3	M_2
1	M_4	M_5	M_7	M_6

Q) Represent the following function of A, B, C by a K-map?

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

(2) $F = AB + C$

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

(4) $F = A + C$

(5) $F = B$

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

Ans → (1) $F = ABC + A\bar{B}C + A\bar{B}\bar{C}$

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	M_0 0	M_1 0	M_3 0	M_2 0
A	M_4 1	M_5 1	M_7 1	M_6 0

(2) $F = AB + C$

$F = AB + C$

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

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

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

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

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	M_0 0	M_1 1	M_3 1	M_2 0
A	M_4 0	M_5 1	M_7 1	M_6 1

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

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	M_0 0	M_1 0	M_3 1	M_2 0
A	M_4 0	M_5 1	M_7 1	M_6 1

(4) $F = A + C$

$F = A + C$

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

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

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

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

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

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	M_0 0	M_1 1	M_3 1	M_2 0
A	M_4 1	M_5 1	M_7 1	M_6 1

→ LOOP 1

→ LOOP 2

$F = A + C$

LOOP-1

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

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

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

$= \bar{B}C + BC$

$= C$

LOOP-2

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

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

$= A\bar{B} + AB$

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

$= A$

(5) $F = B$

$F = B$

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

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

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

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	M_0 0	M_1 0	M_3 1	M_2 1
A	M_4 0	M_5 0	M_7 1	M_6 1

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

$F = BC + B\bar{C}$

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

$= B$

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

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

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

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

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

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0 M_0	1 M_1	1 M_3	1 M_2
A	0 M_4	1 M_5	1 M_7	0 M_6

→ Loop 1

↳ Loop 2

Loop-1 :- $\bar{A}BC + \bar{A}B\bar{C}$

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

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

Loop-2 :- $\bar{B}C + BC$

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

$F = C \quad (B+\bar{B}=1)$

By adding we get,

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

Rules for K-map in 3 variables :- (2^3) (1, 2, 4, 8, 16, 32)

(1) When we move from column 1 to column 2 the value of C changes from $\bar{C}$ to C.

(2) When we move from column 3 to column 4 the value of C changes to C to $\bar{C}$.

(3) When we move from column 2 to column 3 the value of B changes to $\bar{B}$ to B.

(4) In moving from row 1 to row 2 the value of A changes $\bar{A}$ to A while the value of B remains same.

(5) When we move from 1 square to another adjacent square the value of only 1 variables changes while the value of remaining 2 variables same.

MSB $\bar{A}\ \bar{B}\ \bar{C}$ LSB

Most significal Bit

Least significal Bit

Q) Simplify the function :

(1) $Y = AB\bar{C} + ABC$

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

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

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

Ans→(1) $AB\bar{C} + ABC$

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0 M_0	0 M_1	0 M_3	0 M_2
A	0 M_4	0 M_5	1 M_7	1 M_6

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

$= AB(C + \bar{C})$

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

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

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1 M_0	0 M_1	0 M_3	1 M_2
A	0 M_4	0 M_5	0 M_7	0 M_6

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

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1 M_0	0 M_1	0 M_3	0 M_2
A	1 M_4	0 M_5	0 M_7	0 M_6

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

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

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

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

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

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	M_0 1	M_1 0	M_3 0	M_2 1
A	M_4 0	M_5 0	M_7 0	M_6 1

K-map in 4 Variables :-

(i) A function of 4 variable in a binary can have $2^4 = 16$ possible combination.

(ii) Hence a K-map for 4 variable function will have 16 square and can be drawn in the same way as we did for 2 and 3 variables.

(iii) A 4 variable K-map has 16 squares each representing a minterm or maxterm.

considering ABCD as a binary number the most significant bit (MSB) corresponds to A and the least significant bit (LSB) corresponds to D.

We have to place the MSB (AB) along rows and LSB (CD) along the column.

AB \ CD	$\bar{C}\bar{D}$ (00)	$\bar{C}D$ (01)	CD (11)	$C\bar{D}$ (10)
$\bar{A}\bar{B}$ (00)	$\bar{A}\bar{B}\bar{C}\bar{D}$ (0000) 0	$\bar{A}\bar{B}\bar{C}D$ (0001) 1	$\bar{A}\bar{B}CD$ (0011) 3	$\bar{A}\bar{B}C\bar{D}$ (0010) 2
$\bar{A}B$ (01)	$\bar{A}B\bar{C}\bar{D}$ (0100) 4	$\bar{A}B\bar{C}D$ (0101) 5	$\bar{A}BCD$ (0111) 7	$\bar{A}BC\bar{D}$ (0110) 6
AB (11)	$AB\bar{C}\bar{D}$ (1100) 12	$AB\bar{C}D$ (1101) 13	$ABCD$ (1111) 15	$ABC\bar{D}$ (1110) 14
$A\bar{B}$	$A\bar{B}\bar{C}\bar{D}$ (1000) 8	$A\bar{B}\bar{C}D$ (1001) 9	$A\bar{B}CD$ (1011) 11	$A\bar{B}C\bar{D}$ (1010) 10

Q) Simplify the following expression using k-map ?

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

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

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

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	1
$\bar{A}B$	1	1	0	0
AB	1	1	0	0
$A\bar{B}$	1	0	0	1

loop 1

loop 2

loop 3

LOOP-1:- $\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D}$

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

LOOP-2:- $\bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + AB\bar{C}\bar{D} + AB\bar{C}D$

$= B\bar{C}$

LOOP-3:- $\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D}$

$= \bar{C}\bar{D}$

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

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

Decimal equivalent $= 5 + 13 + 7 + 15 + 0$

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	0	0
$\bar{A}B$	0	1	1	0
AB	0	1	1	0
$A\bar{B}$	0	0	0	0

① Four-1 square being adjacent square having grouped together.

② column wise the variable D is common to all the four squares. Row wise variable B is common to all the four squares.

③ So these 4 squares will give BD after simplification.

④ The square representing $\bar{A}\bar{B}\bar{C}\bar{D}$ is left alone so it is to be considered as a group of a single one squares.

The simplified function is given by $Y = BD + \bar{A}\bar{B}\bar{C}\bar{D}$

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

Loop 1

→ Loop 2

Loop-1:- $Y = \bar{A}\bar{B}\bar{C}D + A\bar{B}\bar{C}D$

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

Loop-2:- $Y = \bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} + \bar{A}BC\bar{D} + ABC\bar{D}$

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

$= B\bar{C}\bar{D} + BC\bar{D}$

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

$= B\bar{D}$

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

~~Unit-1~~

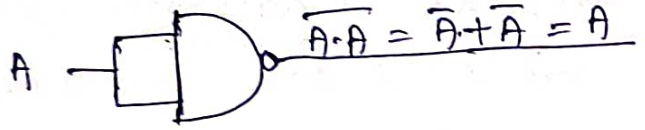
Unit-I (Rest Part):-

Universal NAND Gate:-

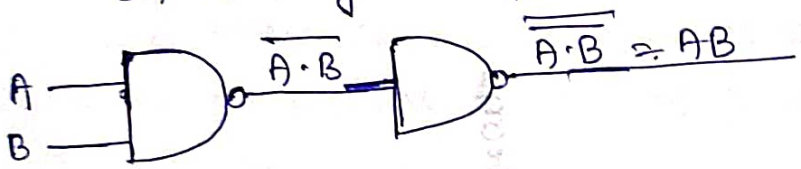
AND, OR & NOT gates are logically complete because only boolean expression can realize using these 3 gates.

The NAND gate is said to be universal gate because it alone sufficient to implement any boolean expression.

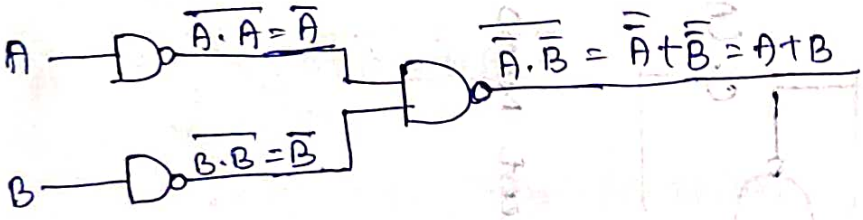
To show that any Boolean Expression can be implemented with NAND Gate only we need to show that logical ~~expression~~ operation AND, OR & NOT can be implemented with NAND Gate only.



(a) NOT gate implementation



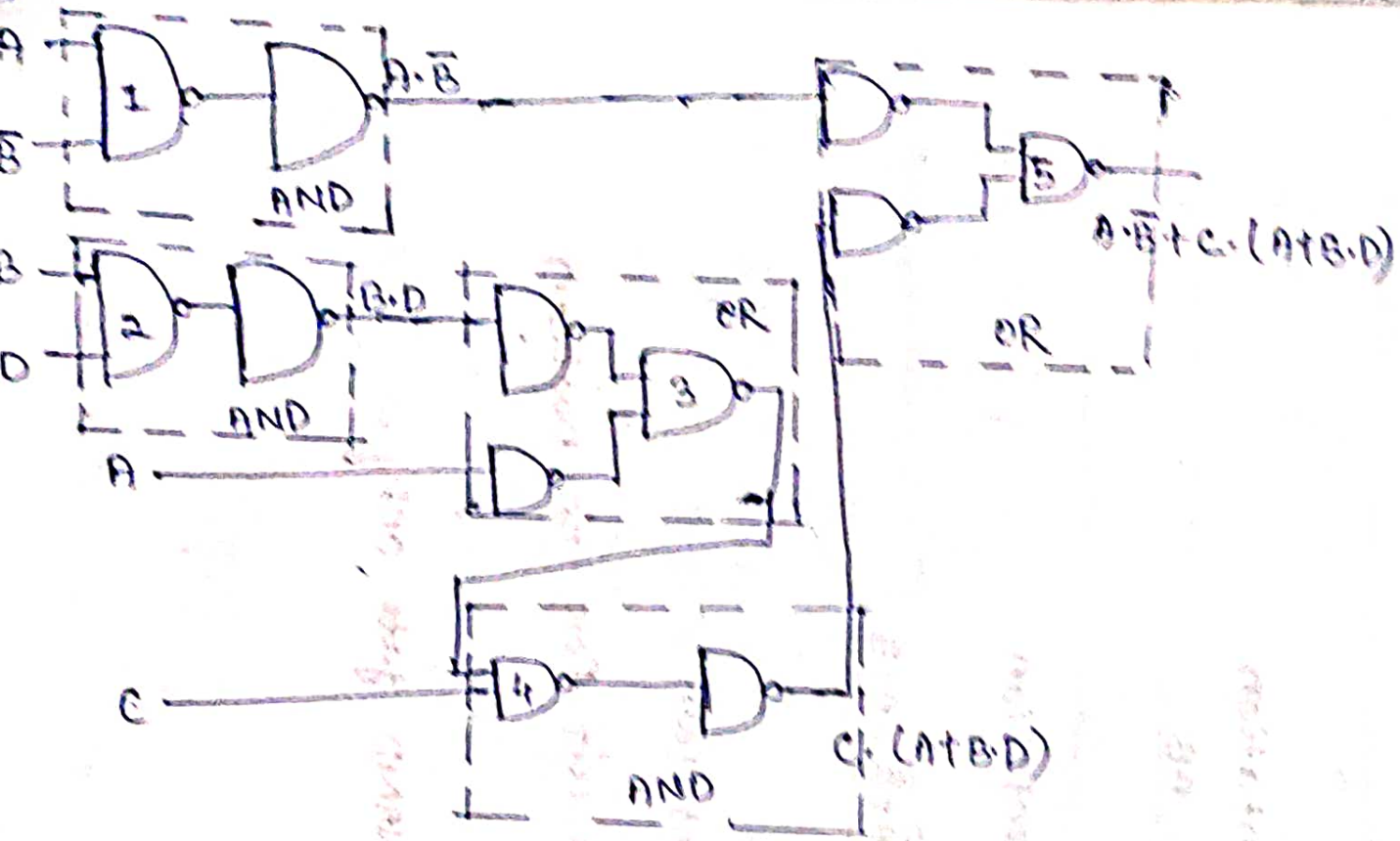
(b) AND gate implementation



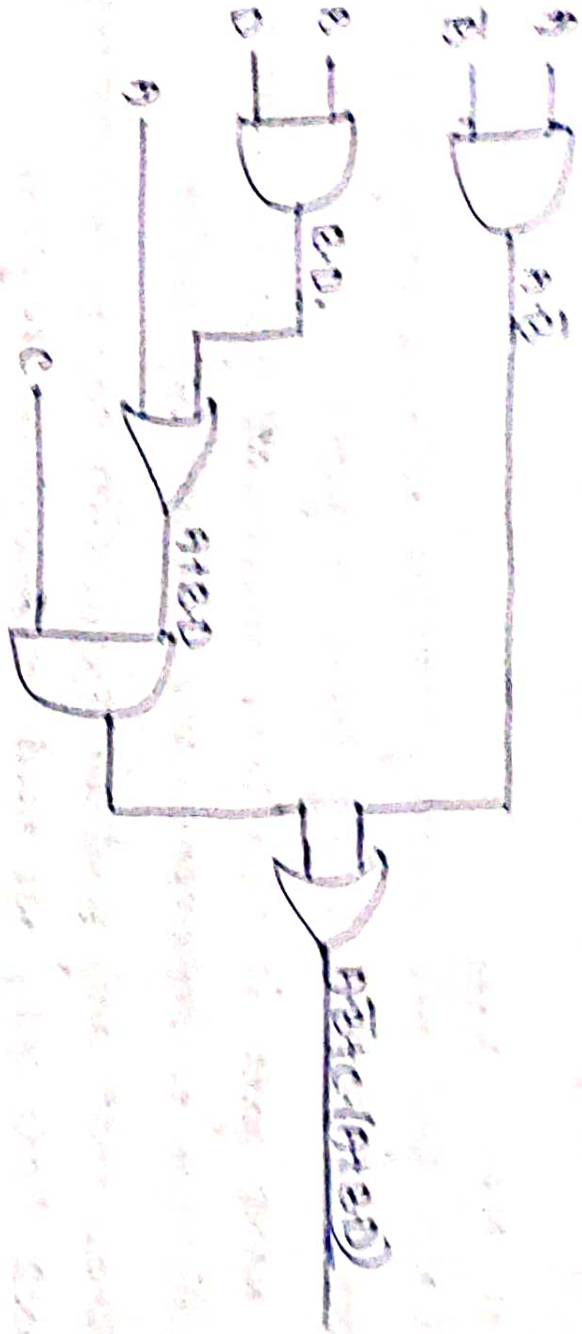
(c) OR gate implementation

Q) construct a logic circuit for the boolean expression?

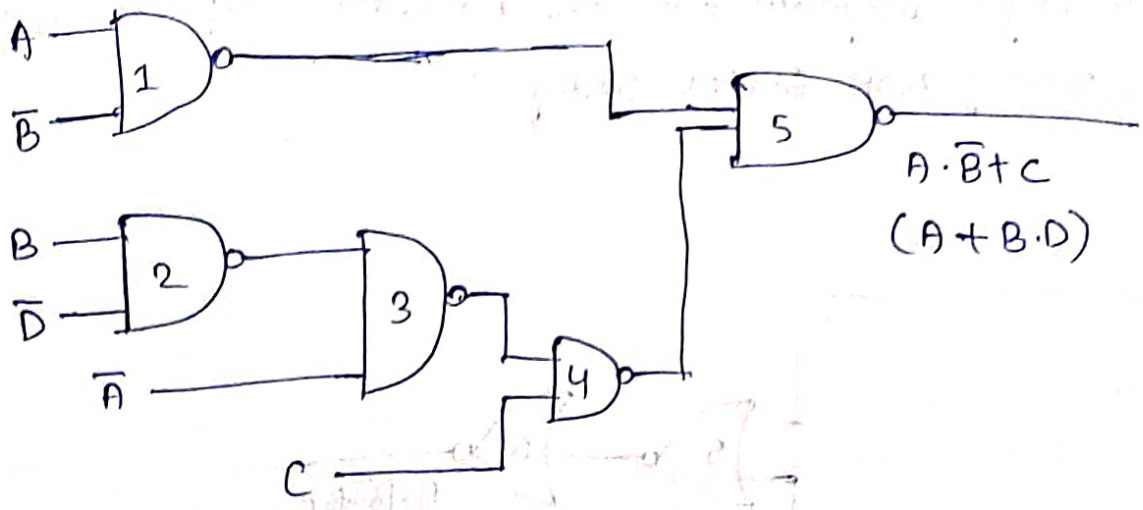
$\overline{A} \cdot \overline{B} + C \cdot (A + B \cdot D)$ using NAND gate only.



(b) substituting equivalent NAND functions



(c) NAND/NAND implementation

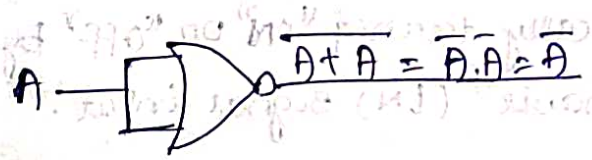


(c) NAND Implementation

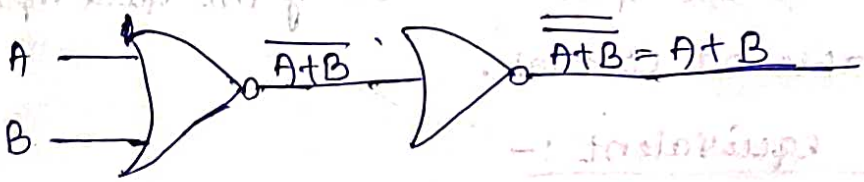
Universal NOR Gate :-

NOR function is the dual of NAND function. Hence all procedures and rules for NOR logic from a dual of the corresponding procedures and rules for NAND logic.

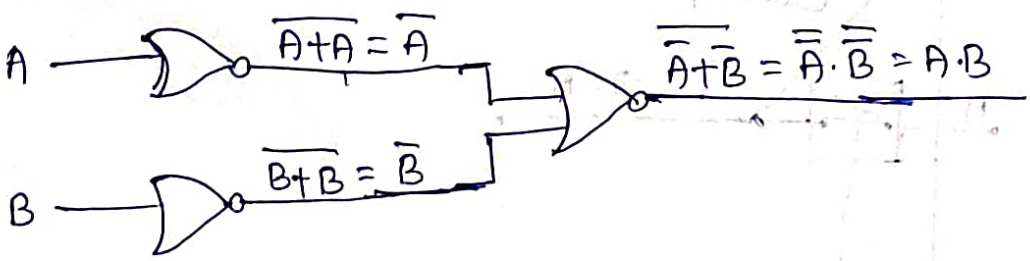
Like NAND Gate, NOR gate is also universal because it is alone sufficient to implement any boolean expression. To show this we need to show that we can implement the logical operation AND, OR and NOT with NOR gates only.



(a) NOT gate implementation

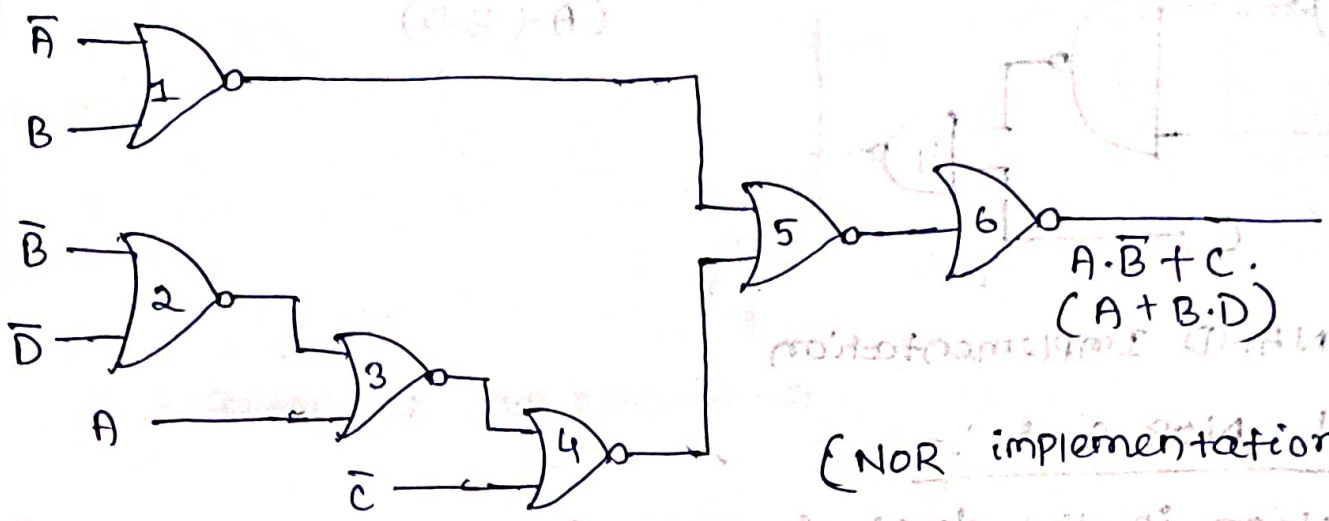


(b) OR gate implementation



Q) construct a logic diagram for the boolean expression $A\bar{B}+C(A+B\cdot D)$ using NOR Gates only ?

Ans→



(NOR implementation)

Tristate buffer :-

It is another type of digital buffer circuit whose output can be "electronically" disconnected from its output circuitary when required.

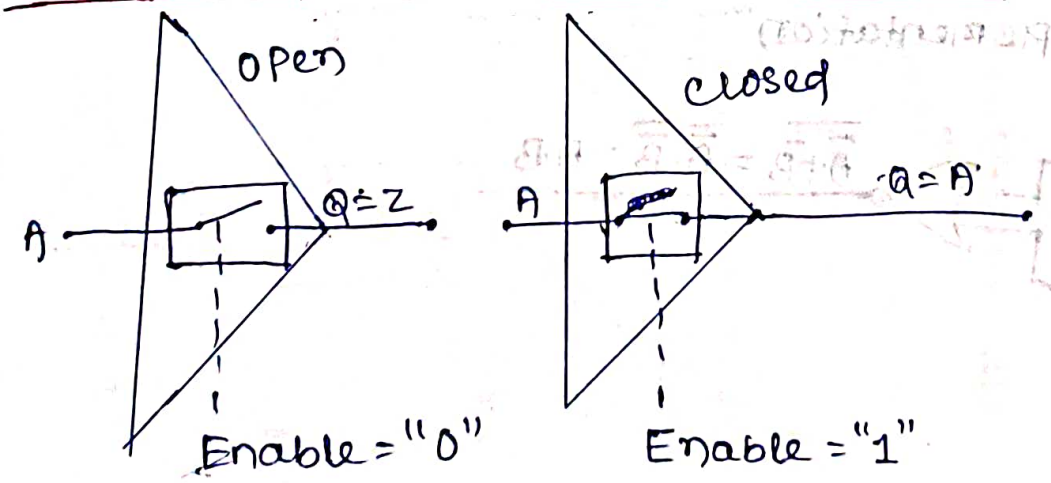
This type of buffer is known as a 3 state buffer or more common a tri-state buffer.

A tristate buffer can be thought of as an input controlled switch with an output that can be electronically turned "on" or "off" by means of an external "control" or "Enable" (EN) signal input.

The logic can be 0 or 1.

Then a tristate buffer requires 2 inputs. One being the data input & the other being the enable or control.

Tristate Buffer switch equivalent :-



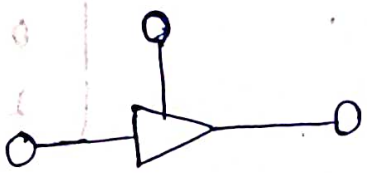
In other way,

A tristate buffer is similiar to a Buffer but it adds an additional "enable" input that contras wheather the primary input is passed to its output or not.

If the "enable" inputs signal is true, the tristate Buffer behaves like a normal buffer. If the "enable" input signal is false, the tristate buffer passes a high impedance (or Hi-Z) signal, which effectively disconnects its output from the circuit. Tristate Buffer are often connected to a bus which allows multiple signals to travel along the same connection.

Symbols :-

The symbol below represents a tri-state buffer.



Truth-table :-

Enable Input	Input A	Output
False	False	Hi-Z
False	true	Hi-Z
True	False	False
True	true	true

(Here 'Hi' stands for high impedence)

Unit-2

Binary addition:-

Rule
0+0=0
0+1=1
1+0=1
1+1=10

① 1011
+ 1111
11010

② 1101
+ 1110
11011

Binary Substraction:-

Rule
0-0=0
1-0=1
0-1=1
1-1=0

For 0011:-

1's complement is the reciprocal of given number & 2's complement = the 1's complement +1.

0011 = 1's complement = 1100

2's complement = 1100 + 1 = 1101

Ex:- +4
-6
-2

* Binary value of 4 = 100
It convert into 4 digit i.e. 0100.

* Binary value of 6 = 0110
1's complement = 1001
2's complement = 1001 + 1 = 1010

0100
+ 1010
(1110)

(for -ve num we have to find 2's complement)

Ex:- +7
-3
(+4)

* Binary value of 7 = 0111
* Binary value of 3 = 0011
1's complement = 1100
2's complement = 1100 + 1 = 1101

0111
+ 1101
$\boxed{1}$0100

(0100) is the binary value of 4.

Ex. -7
-3
(-10)

Binary value of 7 - 111
1's - 000
2's - 000 + 1 = 001

for sign (-7) = 1001

Binary value of 3 - 011
1's - 100
2's - 101

for sign (-3) = 1101

$$\begin{array}{r} 1001 \\ +\ 1101 \\ \hline \boxed{1}0\ 110 \end{array}$$

Binary value of 10 = 1010
1's = 0101
2's = 0110
(-10)

Ex. -5
-3
-8

Binary value of 5 = 101
1's = 010
2's = 011

for sign (-5) = 1011

for 3 binary :- 011
1's = 100
2's = 101

for sign = 1101

$$\begin{array}{r} 1011 \\ 1101 \\ \hline \boxed{1}1000 \end{array}$$

Binary value of 8 = 1000
1's = 0111
2's = 1000

Ex. $+5$
-3
$(+2)$

Binary value of 5 = 101 (0101)
Binary value of 3 = 011
1's = 100
2's = 101

for sign = 1101

$$\begin{array}{r} 0101 \\ 1101 \\ \hline \boxed{1}0010 \end{array}$$

Binary value of 2 = 10

Ex:- -9
-3
-12

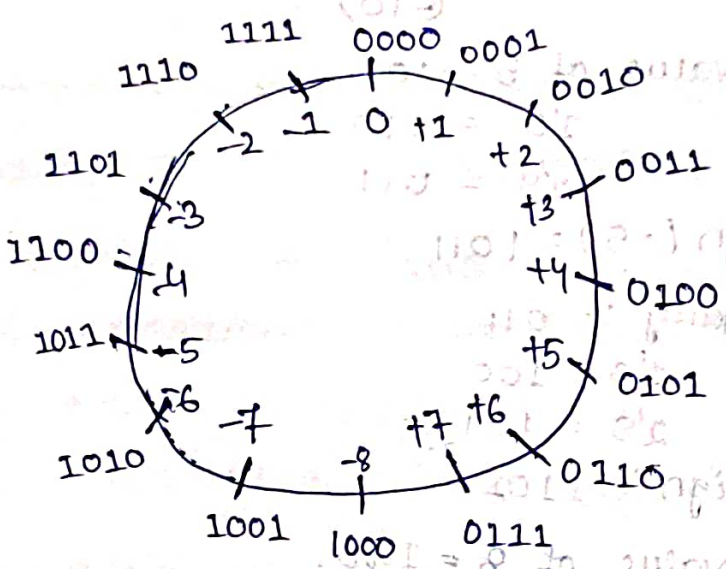
Binary value of 9 = 1001
1's = 0110
2's = 0111

for sign - 10111

for 3 - 0011
1's - 1100
2's - 1101
for sign - 11101

$$\begin{array}{r} 10111 \\ +\,11101 \\ \hline \boxed{1}\,10100 \end{array}$$

12 = 1100
1's = 0011
2's = 10100 (-12)



(mod 16 system for 2's complement numbers)

Design of half adder :-

A half adder needs two binary inputs and 2 binary outputs. Input variables designate augend and addend bits where as output variables produce sum and carry bits.

Let A & B be the two inputs and S (for sum) and C (for carry) be the two outputs.

Truth table:-

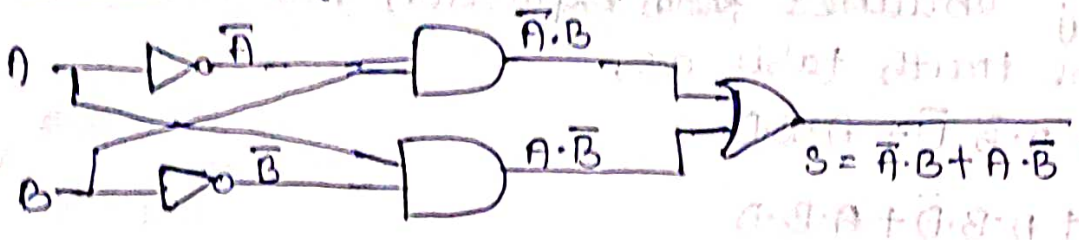
Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

* Boolean Expressions for the two outputs (obtained from the truth table) are

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

$C = A.B$

Circuit diagram:-



Design of full adder:-

A full adder forms the arithmatic sum of 3 inputs bits. Hence, it consists of 3 inputs and 2 outputs.

Two of the input variables (A & B) represent augend and addend bits and the 3rd input variable (D) represents carry from the previous lower significant position.

Truth table:-

Inputs			Outputs	
A	B	D	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

* We can directly obtained ~~from~~ SOP expression for the 2 outputs from the truth table as,

$S = \bar{A}\cdot\bar{B}\cdot D + \bar{A}\cdot B\cdot\bar{D} + A\cdot\bar{B}\cdot\bar{D} + A\cdot B\cdot D$

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

Although it is not possible to simplify the expression for S, we can simplify the expression for C as follows.

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

$= \bar{A}BD + A\bar{B}D + AB\bar{D} + ABD + ABD + ABD \quad (X + X = 1)$

$= (\bar{A}BD + ABD) + (A\bar{B}D + ABD)\ (AB\bar{D} + ABD)$

$= (A + \bar{A})\ BD + (B + \bar{B})\ AD + (D + \bar{D}) \cdot AB$

$= BD + AD + AB \quad (\text{since } X + \bar{X} = 1)$

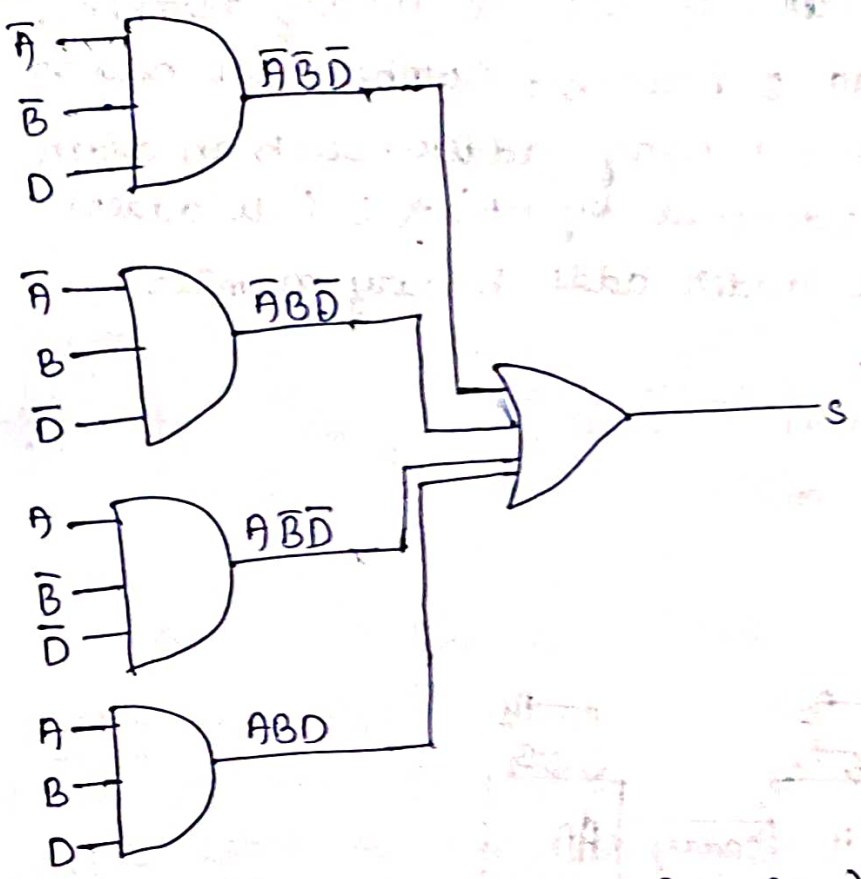
$= AB + AD + BD$

Hence we obtain the following expression for the 2 outputs.

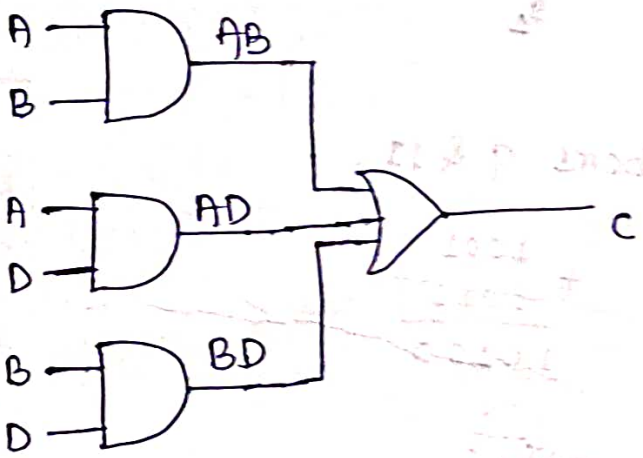
$S = \bar{A}\bar{B}D + \bar{A}B\bar{D} + A\bar{B}\bar{D} + ABD$

$C = AB + AD + BD$

circuit diagram :-

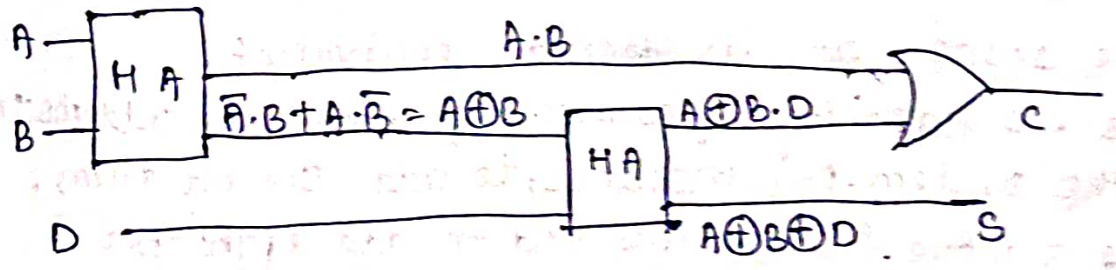


(circuit diagram for sum)



(circuit diagram for carry)

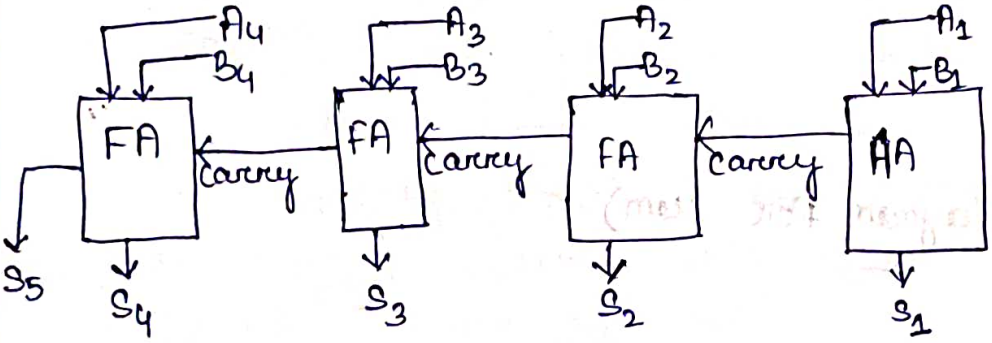
We can also implement a full adder with 2 half adders & one OR Gate.



A Parallel binary adder :-

We use parallel binary adders to add two binary numbers. For ex, If we want to add 2 four-bit numbers we need to construct a parallel 4-bit binary adder. Such an adder requires one half adder (denoted by HA) & 3 full adders (denoted by FA), Here the adder adds binary number A_4, A_3, A_2, A_1 & $B_4 B_3 B_2 B_1$.

$$\begin{array}{r} A_4\ A_3\ A_2\ A_1 \\ +\ B_4\ B_3\ B_2\ B_1 \\ \hline S_5\ S_4\ S_3\ S_2\ S_1 \end{array}$$

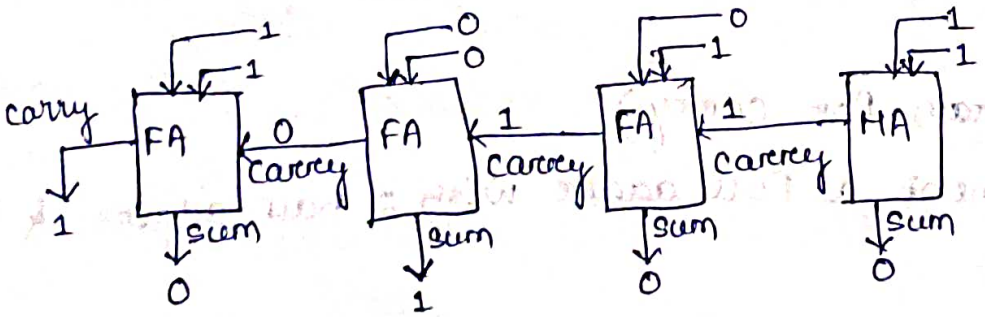


Example:-

Let us see how it will add 2 numbers 9 & 11.

Binary value of 9 is - 1001
Binary value of 11 is - 1011

$$\begin{array}{r} 1001 \\ +\ 1011 \\ \hline 10100 \end{array}$$



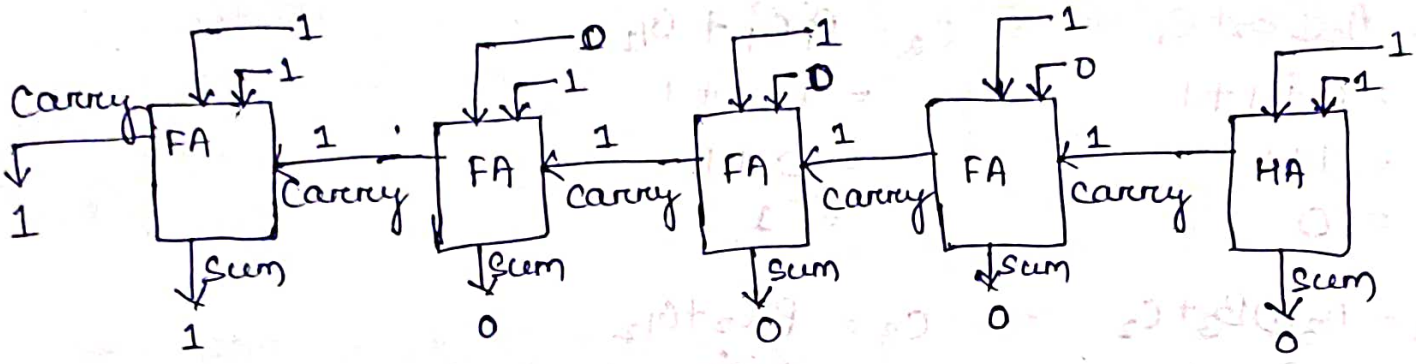
The sum will be 10100 = 20 in decimal equivalent.
We can increases its capacity by connecting more full adders to the left end of the system. For instance, to add six-bit num. We must connect 2 more full adders and to add eight-bit numbers. We must connect 4-more full adders to the left most full adder.

Ex:- Let us consider 23 & 25.

Binary value of 23 - 10111

Binary value of 25 - 11001

$$\begin{array}{r} 10111 \\ +\ 11001 \\ \hline 110000 \end{array}$$



The sum will be 10000 = 48 is decimal equivalent.

Carry Look Ahead adder :- (CLA)

A CLA or fast adder is a type of electronic adder used in digital logic. A carry look ahead adder improves speed by reducing the amount of time required to determine carry bits.

$Sum = A_i \oplus B_i \oplus C_{in}$

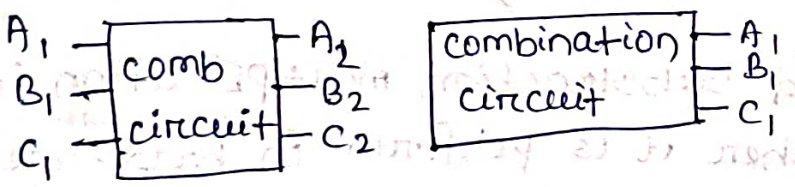
$C_{i+1} = G_i + (P_i \cdot C_i)$

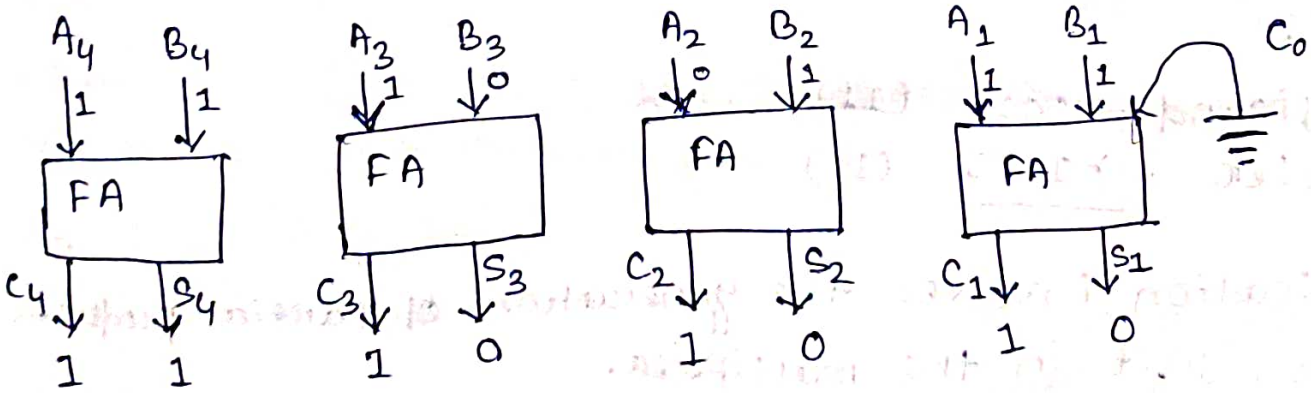
$P_i = A_i \oplus B_i$

$G_i = A_i \cdot B_i$

* Suppose $A = 1101$ (MSB, LSB) — 11

$B = 1011$ — 13





$\Rightarrow S_1 = A_1 \oplus B_1 \oplus C_{in}$

$= 1 \oplus 1 + 0$

$= 0 + 0$

$= 0$

$C_1 = G_1 + (P_i \cdot C_{in})$

$= A_1 \cdot B_1 + (A_1 \oplus B_1 \cdot C_{in})$

$= 1 \cdot 1 + (1 \oplus 1 \cdot 0)$

$= 1 + 0 = 1$

$\Rightarrow S_2 = A_2 \oplus B_2 + C_1$

$= 0 \oplus 1 + 1$

$= 1 + 1$

$= 0$

$C_2 = P_1 C_1 + G_1$

$= 0 \cdot 1 + 1$

$= 0 + 1$

$= 1$

$\Rightarrow S_3 = A_3 \oplus B_3 + C_2$

$= 1 \oplus 0 + 1$

$= 1 + 1$

$= 0$

$C_3 = P_2 C_2 + G_2$

$= P_2 (P_1 C_1 + G_1) + G_2$

$= P_2 P_1 C_1 + P_2 G_1 + G_2$

$= 0 + 1 \cdot 1 + 1 = 1$

$\Rightarrow S_4 = A_4 \oplus B_4 + C_3$

$= 1 \oplus 1 + 1$

$= 0 + 1$

$= 1$

$C_4 = P_3 C_3 + G_3$

$= P_3 (P_2 C_2 + G_2) + G_3$

$= P_3 P_2 C_2 + P_3 G_2 + G_3$

$= P_3 P_2 (P_1 C_1 + G_1) + P_3 G_2 + G_3$

$= P_3 P_2 P_1 C_1 + P_3 P_2 G_1 + P_3 G_2 + G_3$

$= 1 \cdot 1 \cdot 0 \cdot 1 + 1 \cdot 1 \cdot 1 + 1 \cdot 0 + 0$

$= 1$

so final ans. is 11000 which is equal to 24 in decimal equivalent.

Multiplication:-

compared with addition and substraction multiplication is a complex operation. wheather it is perform in hardware or software.

multiplicand → 1011 (11)

multiplier → 1101 (13)

① Multiplication involves the generation of partial products 1 for each digit in the multiplier.

These Partial products are then summed to procedure the final Product.

② The partial product is 0, then the multiplicant is 1.

③ The total product is procedure when they are added.

④ The multiplication of two n bit binary integer results in a product of upto 2n bits in length.

$$\begin{array}{r} 1011 \\ +\ 1101 \\ \hline 1011 \end{array}$$

Add and shift operation (1)

only shift operation (0)

⑤ The multiplier & multiplicand are loaded in 2 registers. i.e. your Q & M.

⑥ A 3rd resister, the A register is also need & is initially set to 0.

⑦ There is also a 1 bit C register initialized to 0; which holds a potential carry bit resulting from addition.

The Operation of the multiplier :-

Control Logic :-

It reads the bits of the multiplier one at a time.

$Q = 101 \vdash a_0$

→ If Q_0 is 1, then the multiplicant is added to the A register, and the result is stored in the A register with the C bit used for overflow.

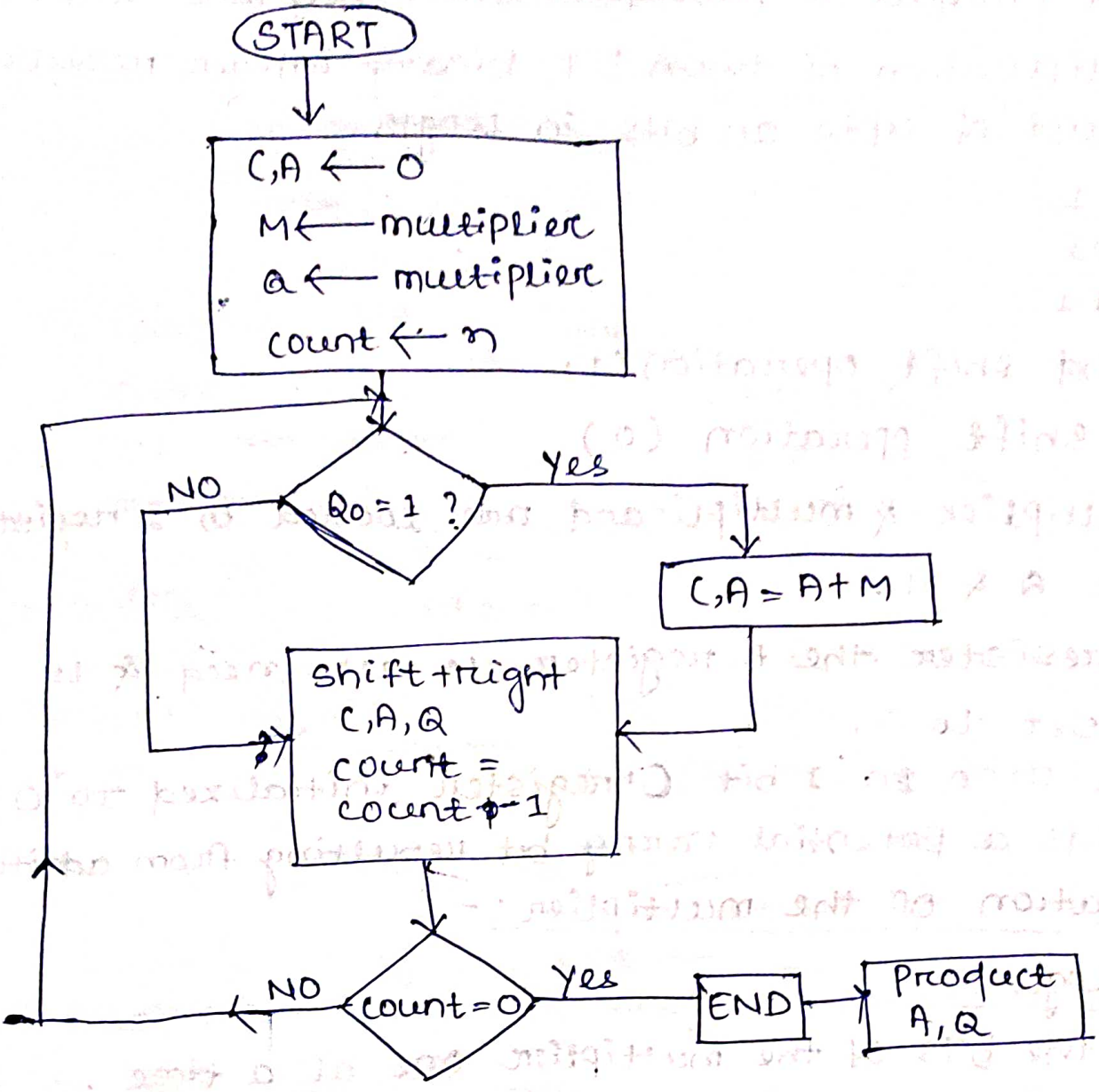
→ Then all of the bits of C, A & Q resister are shifted to the right 1 bit so that C bit goes into A_{n-1}, A_0 goes into Q_{n-1} & Q_0 is lost.

→ If Q_0 is 0 then no addition is performed just the shift.

→ This process is repeated for each bit of the original multiplier

→ The resulting 2n bit product is contained in the n & a register.

Flowchart :-



C	A	Q	M		
0	0000	1101	1011	initial value	
	A+M				
0	1011	1101	1011	Add	(1st cycle)
0	0101	1110	1011	shift	
0	0010	1111	1011	shift	(2nd cycle)
	A+M				
0	1101	1111	1011	Add	(3rd cycle)
0	0110	1111	1011	shift	
1	0001	1111	1011	Add	(4th cycle)
0	1000	1111		shift	

10001111

$= 1\times2^7 + 0\times2^6 + 0\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$

$= 128 + 8 + 4 + 2 + 1$

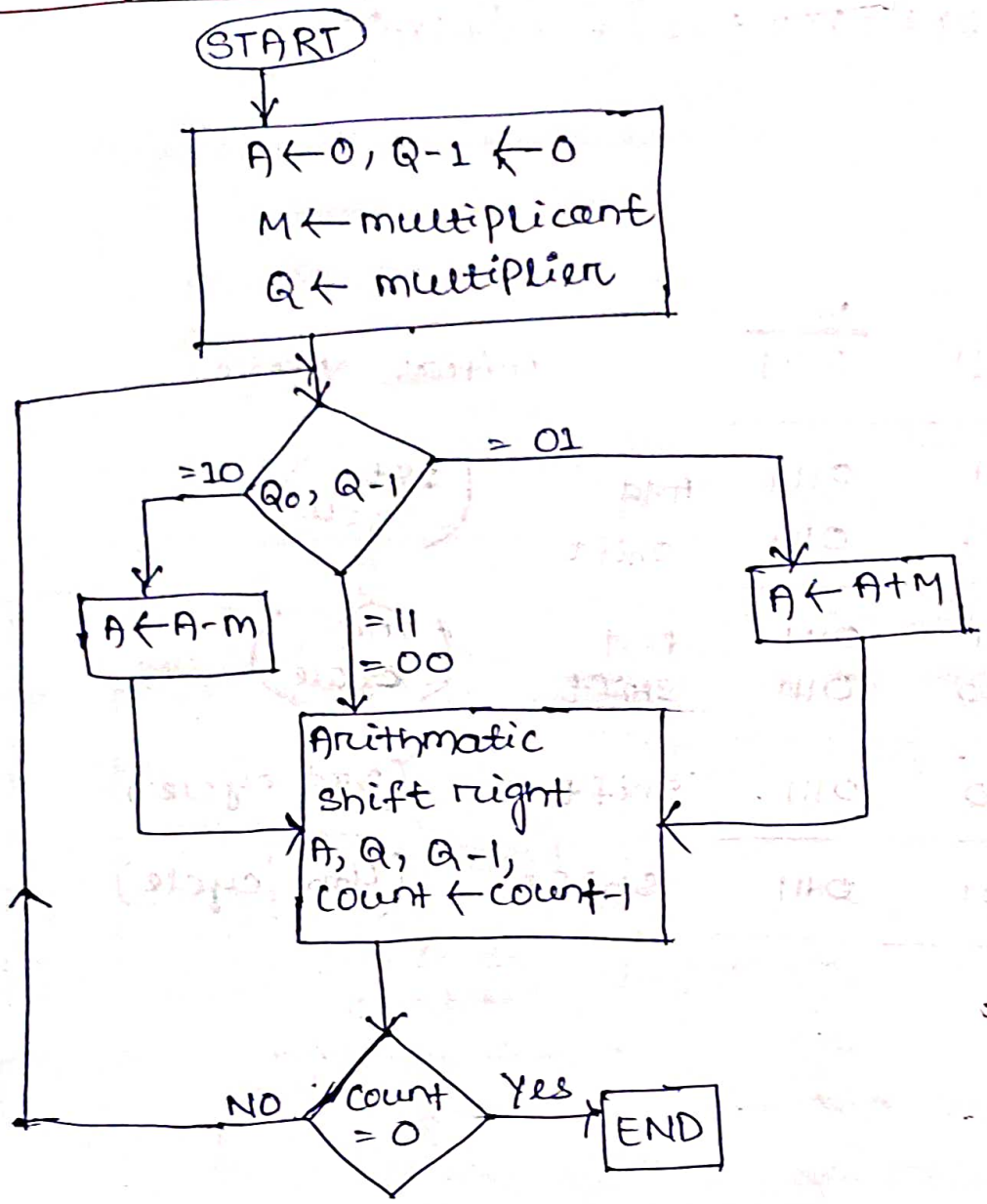
$= 143 \quad (11\times13)$

* $7\times3 = 21$

C	A	Q	M		
0	0000	0011	0111		initial value
0	0111	0011	0111	Add	(1st cycle)
0	0011	1001	0111	shift	
0	1010	1001	0111	Add	(2nd cycle)
0	0101	0100	0111	shift	
0	0010	1010	0111	shift	(3rd cycle)
0	0001	0101	0111	shift	(4th cycle)

00010101 - 21

Booth Algorithm for 2s' complement multiplication :-



A	Q	Q-1	M	
0000	0011 (10)	0	0111	→ Initial Value
1001	0011	0	0111	— A ← A-M
1100	1001 (11)	1	0111	— shift (1st cycle)
1110	0100 (01)	1	0111	shift (2nd cycle)
0101	0100	1	0111	— A ← A+M (3rd cycle)
0010	1010 (00)	0	0111	— shift
0001	0101	0	0111	— shift (4th cycle)

$\therefore 00010101 = 21$

```
 1110
 0111
 ----
 0101
```

Rough

A - M

A = 0000

M - 0111

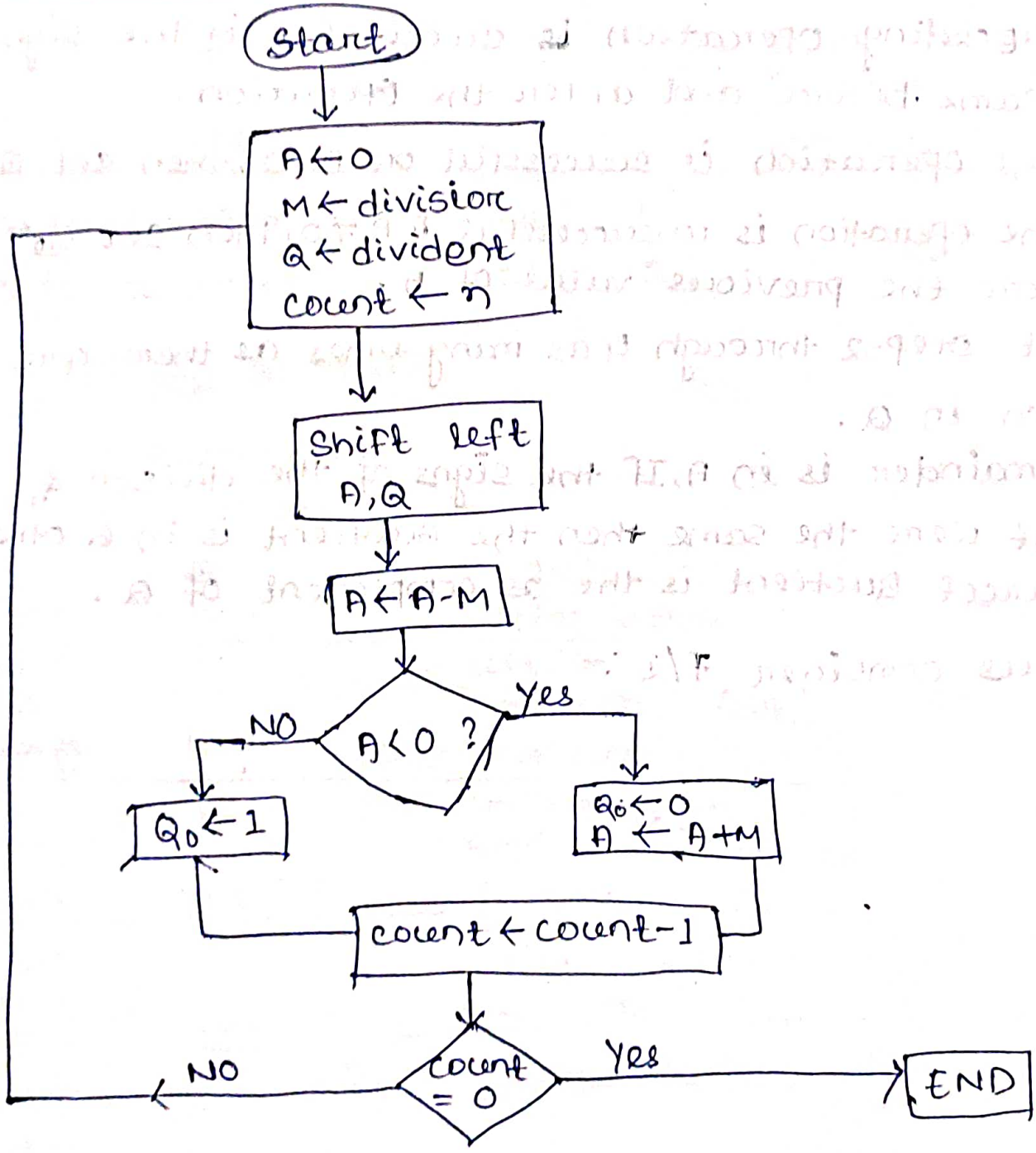
12 - 1000

21 - 1001

```
32  16  8  4  2  1
            1  1  1  — 7
               1  1  — 3
    1   0  1  0  1  — 21
```

Binary Division :-

Flow chart for unsigned Binary Division :-



Algorithm for 2's complement division :-

1). Load the division into the M resister and the divident into the A,Q resister.

The divident must be expressed as a 2n bit 2's complement number.

Thus for ex, the 4 bit 0111 becomes 00000111 & 1001 becomes 11111001.

② Shift A,Q left 1 bit position.

③ If M & A have the same sign then perform $A \leftarrow A-M$ otherwise $A \leftarrow A+M$.

④ The preceding operation is successful if the sign of A is same before and after the operation.

a) If the operation is successful or $A=0$, then set $Q_0 \leftarrow 1$.

b) If the operation is unsuccessful & $A \neq 0$, then set $Q_0 \leftarrow 0$ & restore the previous value of A.

⑤ Repeat step-2 through 4 as many times as there are bit position in Q.

⑥ The remainder is in A, If the signs of the divisor & divident were the same then the quotient is in Q otherwise the correct quotient is the 2s complement of Q.

Let us consider 7/3 :-

7/3 :-

Ex:-

A	Q	M	
0000	0111	0011	initial value
0000	1110		Left Shift
1101	1110		Substract (1st)
0000	1110		Restore
0001	1110		Shift
1110	1100		Substract (2nd)
0001	1100		Restore
0011	1000		Shift
0000			Substract (3rd)
0000	1001		Set $Q_0 \leftarrow 1$
0001	0010		Shift
1110			Substract (4th)
0001	0010		Restore

A is remainder i.e. 0001 = 1

Q is quiotent i.e. 0010 = 2

$$3 \overline{)7} \quad (2) \text{ Quotient}$$
$$\quad\ \ 6$$
$$(1) \text{ Remainder}$$

* Let us consider 7/-3 :-

Ex:-

A	Q	M = 1101	
0000	0111		initial value
0000	1110		Left shift
1101			Addition (1st)
0000	1110		Restore
0000	1100		Shift
1110			Add (2nd)
0001	1100		Restore
0011	1000		Shift
0000			Add (3rd)
0000	1001		set $Q_0 = 1$
0001	0010		Shift
1110			Add (4th)
0001	0010		Restore

A is remainder i.e. 0001 = 1

Q is quiotent i.e. 0010 = 2

In both case the answer is same.

Ex:- -7/3

	A	Q	M = 0011
	1111	1001	initial value
	1111	0010	Shift left
1st	0010		Add
	1111	0010	Restore
	1110	0100	Shift left
2nd	0001		Add
	1110	0100	Restore
	1100	1000	Shift left
3rd	1111		Add
	1111	1001	$Q_0 \leftarrow 1$
	1111	0010	Shift left
4th	0010		Add
	1111	0010	Restore

1's	0000	0111
2's	1111	1000
		1
	1111	1001

$$3 \overline{\big|\, -7 \,\big|} -2$$
$$\quad -6$$
$$\quad (-1)$$

A is Remainder i.e. 1111 which is the 2s complement of 1.

⇒ Remainder = (-1)

Q is Quotient i.e. 0010 Here the sign of division & divident is different so the correct Quotient is the 2s complement of Q i.e. 1110 which is equal to (-2).

Ex:- -7/-3

	A	Q	M	
	1111	1001	1101	initial value
	1111	0010		Left shift
1st	0010			Substract
	1111	0010		Restore
	1110	0100		Left shift
2nd	0001			Substract
	1110	0100		Restore
	1110	1000		Left shift
3rd	1111			Substract
	1111	1001		~~Restore~~ Set $Q_0 \leftarrow 1$
	1111	0010		Left shift
4th	0010			Substract
	1111	0010		Restore

$$-3 \overline{\left)\begin{array}{r} -7 \\ -6 \\ \hline -1 \end{array}\right.} 2$$

A is Remainder i.e. which is the 2s complement of 1

→ Remainder = (-1)

Q is Quotent i.e. 0010 which is equal to 2.

Floating Point Representation:-

976,000,000,000 is Represented by 9.76×10^{14}

$\boxed{\pm S \times B^{\pm E}}$

S - Mantisa / significant bit

B - Base

① sign : Plus or minus

② significant

③ Exponent

④ Base

The Base B is implicity & not be stored because it is same for all the numbers.

Sign (1 bit)	(8 bit) Biased exponent	(23 bit) significant

Ex:- 1.1010001×2

significant must be 23 bit so 10100010000000000000000

The principle used in representing binary floating point numbers is shown in the above example.

32- bit floating point format

The left most bit stores the sign of the number i.e. 0 for (+)ve, 1 for (-)ve.

The exponent value is stored in the next 8 bits of the representation used is known as biased representation.

A fixed value, called the bias is, substracted from the filled to get the true exponent value.

Typically the bias = $\boxed{2^{k-1} - 1}$

where, k is the number of bits in the binary exponent.

$2^{8-1} - 1 = 2^{7} - 1$

$= 128 - 1 = 127$

In this case the 8 bit field the numbers 0 through 255 with a bias of 127.

The true exponent values are in the range -127 to +127

In this case, the base is assume to be 2.

127 - 1111111

$$\begin{array}{r} 1111111 \\ +\ 10100 \\ \hline 10010011 \end{array}$$

32 bit

sign (1 bit)	Biased exponent (8 bit)	significant (23 bit)
0	10010011	10100010000 00000000 0000

+ve - 0

-ve - 1

Booth Multiplication:-

The major advantages of the booth technique as propose by Adrew D Booth.

* It handles both (+)ve &(-)ve numbers.
* It may also have an addend advantage to reducing the num of operation depending upon the multiplier.

consider a (+)ve multiplier consisting a block of is surrounded by zeros.

For ex - 00111110

$M \times 00111110$

$= M \times (1\times2^5 + 1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1)$

$= M \times (32+16+8+4+2)$

$= M \times 62$

where M is the multiplicant

The no of operation can be reduced to 2 by rewritting the same as

$M \times (2^6 - 2^1)$

$= M \times (64-2) = M \times 62$

$= M \times (01000010)$

Impact it can be shown that any sequence of 1s in a binary number can be broken into the difference of 2 binary numbers.

$(---01----10)_2 = (\overbrace{---10-01-}^{n\,bit})_2 - (----00-10-)_2$

Hence we can actually replace the multiplication by the string of 1s in the original Number by simpler operation. Adding the multiplier shifting the partial product thus formed by appropriate place & then finally substracting the multiplier.

It is making use of the fact that we don't have to do anything but shift while we are dealing with zeros in a binary multiplier & is similar to using the mathematical property that

99 = (100 - 1) while multiplying by 99. The scheme can be extended to any number of block of 1s in a multiplier.

$M \times 00111010 = M \times (2^6 + 2^4 + 2^3 + 2^1)$
$= M \times (32 + 16 + 8 + 2)$
$= M \times 58$

Booth's Algorithm follows the scheme by performing an addition when encounters the 1st digit of block of one (01) & a subtracting when it encounters the end of the block of (10)

$M \times 00111010$
$= M \times 0100 - 11 - 10$

00 - 0×M	Booth multiplier Recoding table
01 - +1×M	
10 - -1×M	
11 - 0×M	

Bit Pair recording :-

Rules:-

(a) If BPR = q_i = +1 the multiplicant is entered in the appropriate position to be added to the partial product

(b) If BPR = q_i = 0 → the 0s are entered in the appropriate position to be added to the partial product.

(c) If BPR q_i = -1 the 2s complement of m entered in the appropriate position to be added to the partial product

(d) If BPR = +2 Perform (M×10) in the appropriate position.

(e) If BPR = -2 perform 2s complement (M×10) = partial product in the appropriate position.

Ex:- +13
- 6
-78

8	4	2	1	
1	1	0	1	-13
	1	1	0	-6

2s complement of 6 → 0110
1s - 1001
2s - 1
1010

0 1101 (+13)
1 1010 (-6)
0 -1 1 -1 0

```
          01101
          0-1+1-10
        ----------
      00000000000
      1111100111
      00001101
      1110011
      000000
      ----------
1111 | 0110010 (-78)
```

sign

Binary value of 78 = 1001110
1s = 0110001
2s = 1
0110010 (-78)

So Ans is 0110010 (-78)

Table:-

Multiplier Bitpair		Multiplier on the right	Multiplicant selected at
i+1	i	(i-1)	(i)
0	0	0	0 X M
0	0	1	+1 X M
0	1	0	+1 X M
0	1	1	+2 X M
1	0	0	-2 X M
1	0	1	-1 X M
1	1	0	1 X M
1	1	1	0 X M

Ex:- -12
$\underline{\quad 6}$
-72

8	4	2	1	
1	1	0	0	-12
	1	1	0	-6

2s complement of 12 (1100)

```
   1100
1s- 0011
2s-    1
   ----
   0100
```

sign

```
1 0 1 0 0 (-12)
0 0 1 1 0 (6)
```

6 = 001100
= 0+10-10

```
*          10100
         x 010-10
         --------
     00000000000
     00000 1100
     00000 0000
      111 0100
      000 000
     --------
111. | 0111000
```

```
      10100
      0 2 -2
     ---------
000000011.00
111 0 2 0000
000000
     ---------
111 | 0111000      (-72)

1s- 0001000111
                 1
     ---------
000 | 001000      (72)
```

so ans. is (-72)

Multiplier	LSB	MSB
+ve	0	0
-ve	0	1

Floating Point Numbers : & Arithmatic Operators

① Floating Point numbers

$X = X_s \times B^{X_E}$

$Y = Y_s \times B^{Y_E}$

② Arithmatic operations

$$X+Y = (X_S \times B^{X_E - Y_E} + Y_S) \times B^{Y_E}$$

$$X-Y = (X_S \times B^{X_E - Y_E} - Y_S) \times B^{Y_E}$$

$$X \times Y = (X_S \times Y_S) \times B^{X_E + Y_E}$$

$$X/Y = \left(\frac{X_S}{Y_S}\right) \times B^{X_E - Y_E}$$

Let us take an example

$x = 0.3 \times 10^2$

$y = 0.2 \times 10^3$

$$X+Y = (0.3 \times 10^{2-3} + 0.2) \times 10^3$$

$$= (0.3 \times 10^{-1} + 0.2) \times 10^3$$

$$= (0.03 + 0.2) \times 10^3$$

$$= 0.23 \times 10^3$$

$$= 230$$

$$X-Y = (0.3 \times 10^{2-3} - 0.2) \times 10^3$$

$$= 0.03 - 0.2 \times 10^3$$

$$= -0.17 \times 10^3$$

$$= -170$$

$$X \times Y = (0.3 \times 0.2) \times 10^{2+3}$$

$$= 0.06 \times 10^5$$

$$= 6000$$

$$\frac{X}{Y} = \frac{0.3}{0.2} \times 10^{2-3}$$

$$= 1.5 \times 10^{-1}$$

$$= 0.15$$

sign	Biased exponent	significant
1 bit	11 bit	52 bit

→ A) Floating point operation may produce one of these condition.

① Exponent overflow:- A (+ve) exponent exceeds the maximum possible exponent value. In some systems this may be designated as $+\infty$ or $-\infty$.

② Exponent underflow:- A (-)ve exponent is less than the maximum possible exponent value (e.g: -200 is less than -127). This means that the number is too small to be represented & it may be reported as 0.

③ Significant underflow:- In the process of aligning significand digits may flow off the right end of the significand. As we shall discuss, some form of rounding is required.

④ Significand overflow:- The addition of 2 significands of the same sign may result in a carry out of the most significant bit. This can be fixed by realignment, as we shall explain.

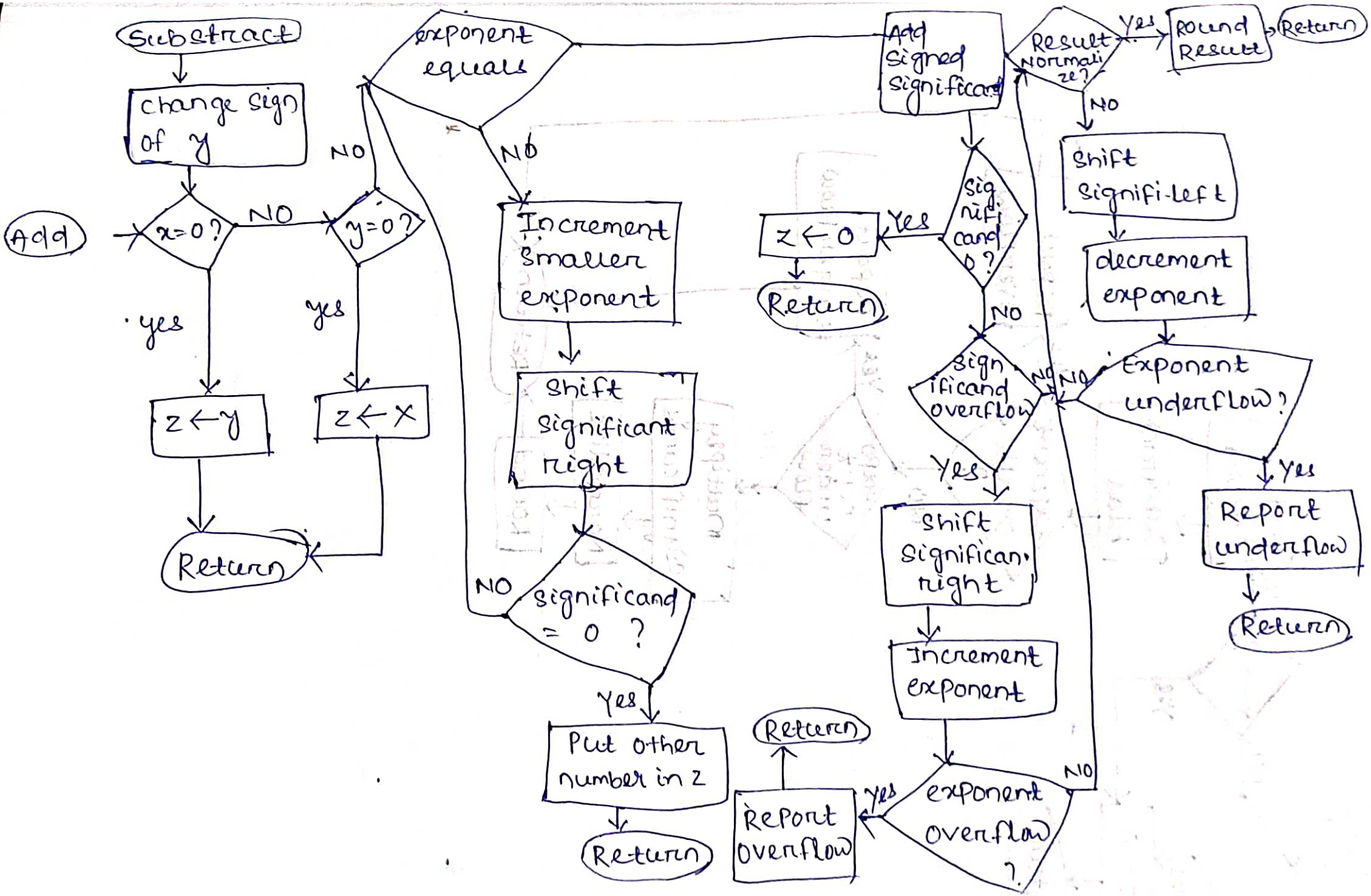
Addition and Substraction:-

Floating point arithmatic, addition and substraction are more complex than multiplication and division. This is because for the need of alignment.

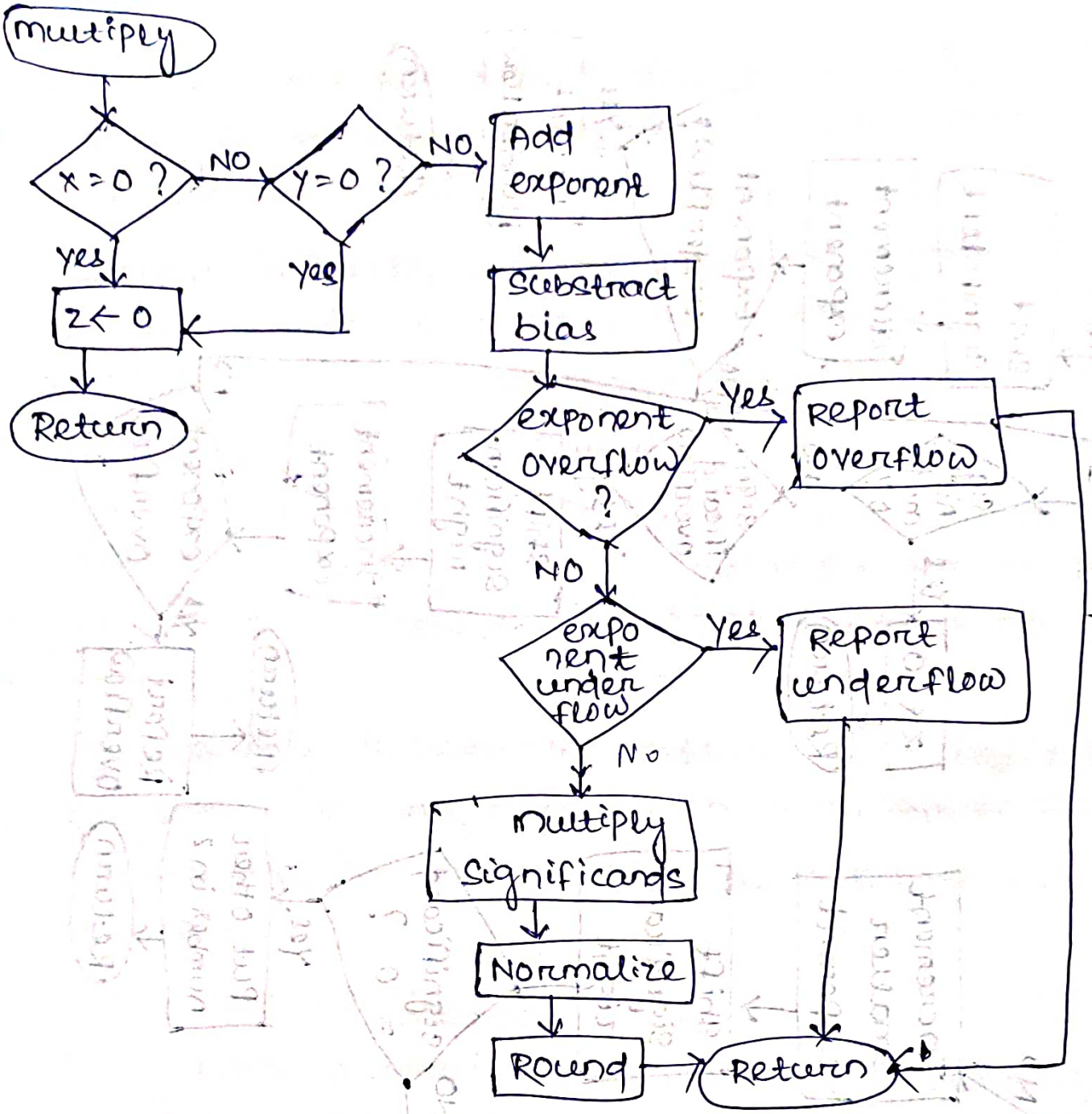
There are 4 basic cases for the algorithm for addition & substraction.

(1) Check for zeroes
(2) Align the significands
(3) Add or substract the significands
(4) Normalize the result.

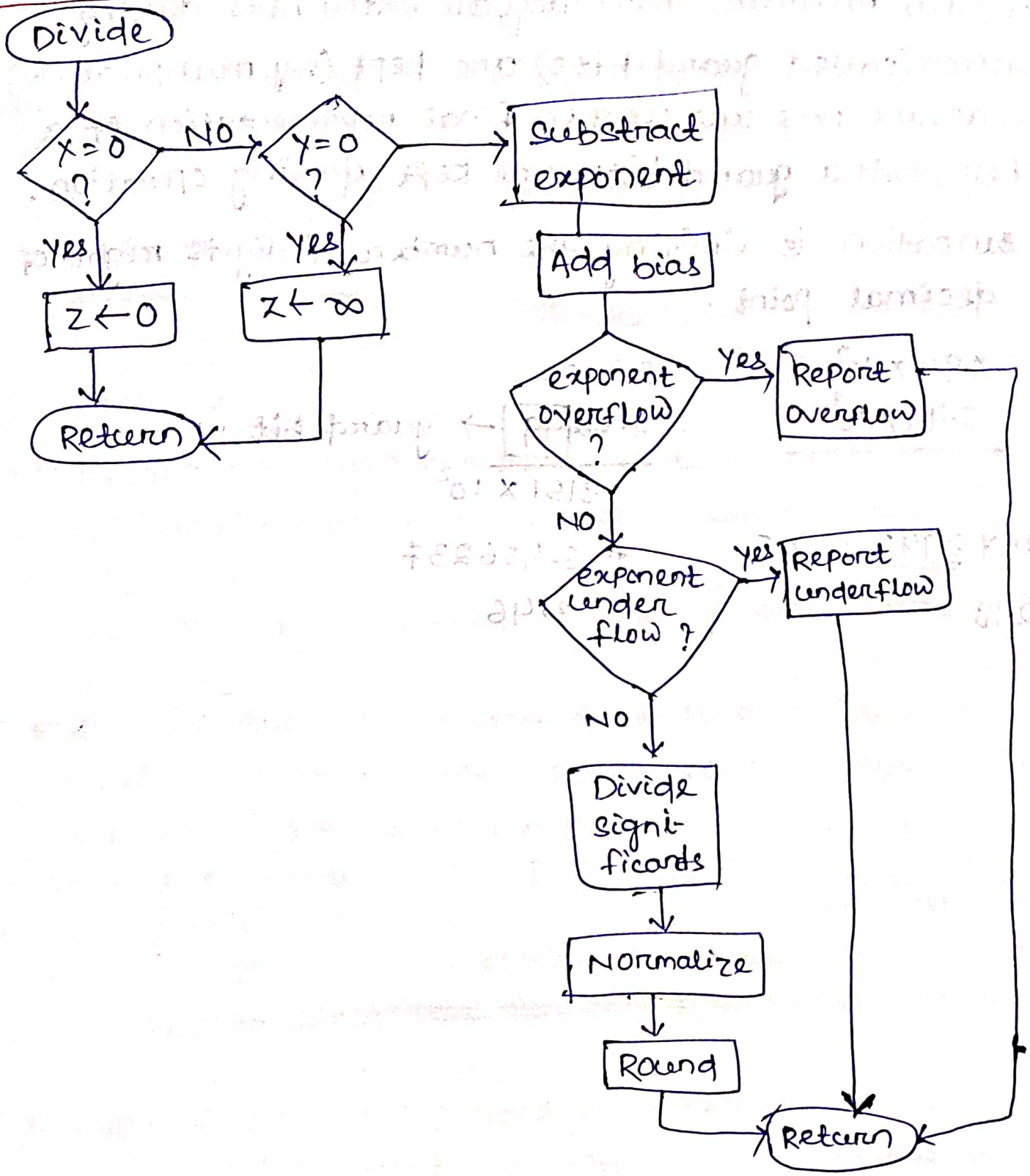
Flowchart:-


Add
x=0?
yes
NO
y=0?
yes
NO
z←y
z←x
Return
Substract
change sign of y
exponent equals
NO
NO
Increment smaller exponent
shift significant right
significand = 0 ?
NO
Yes
Put other number in z
Return
Add signed significand
Sig nificand 0 ?
Yes
z←0
Return
NO
Sign ificand overflow
Yes
shift significan right
Increment exponent
exponent overflow ?
Yes
Report overflow
Return
NO
Result Normalize?
Yes
Round Result
Return
NO
shift signifi-left
decrement exponent
Exponent underflow?
NO
Yes
Report underflow
Return

Multiplication flowchart :-



Division flow chart :-



Guard bit & Truncation:-

To retain maximum accuracy, all extra bits during operation (called guard bits) are kept (e.g. multiplication). If we assure bits are used in final representation of a number, extra guard bits are kept during operation.

Truncation is limiting the number of digits right of the decimal point.

Ex:- 2.84×10^2 2.8400

2.49×10^2 0.02$\boxed{49}$ → guard bit

$.8151 \times 10^2$

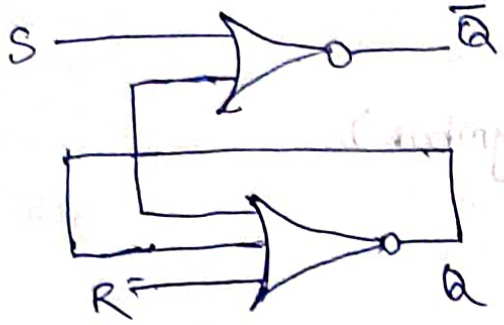
* 2.178942   > 5   * 3.456237

2.18   3.46

Unit-3 Flipflops

Introduction:-

Flipflops are the basic memory element used in sequencial circuit.



SR- flip-flop

→ flipflops are also known as bistable multivibrator, latches, multibinary & toggle are used in various device.

→ flip-flops can be wired from logic gates such as NAND & NOR gates.

→ Flip-flops are interconnected to form sequencial logic circuit such as storage circuit, counter, register & in other computer application.

→ It has 2 stable states logic 0 & logic 1.

→ A flip-flop is a bistable electronic circuit that has 2 stable states i.e. the electronic circuit is either in ~~state~~ 0 or state ~~i.e. the electronic circuit is either in state~~ 1.

→ A flip flop can be regarded as a memory device.

→ When a flip-flop has its output set as 0 it can be regarded as storing a logic 0 & when its output is set as storing 1.

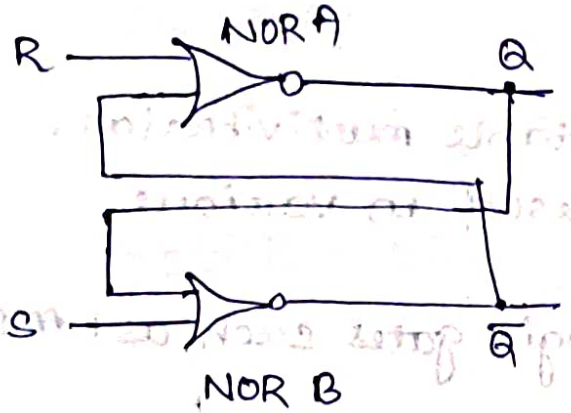
→ It is the smallest storage unity of a computer system.

R-S flip-flop :-

We can construct a R-S flip-flop using NOR gate. Here in case of R-S flip-flop the outputs are define as Q & Q'.

There are 2 inputs to the flip flop define as R & S flip-flop.

(Logic symbol)



Truth table :-

A	B	Z
0	0	last state
0	1	1
1	0	0
1	1	forbidden

→ In the above table R=0 & S=0 since a zero at the input A NOR gate has no effect on its output.

The flip-flop simply remains in its present state that is, Q remains unchanged.

→ The 2nd input condition R=0, S=1 forces the output of NOR gate B to be low. Both inputs to NOR gate A are now low and the NOR gate output must be high. Thus a 1 at the S input is said to set the flip-flop and it switches to the stable state where Q=1.

→ The 3rd condition is R=1 & S=0 this condition forces the output of NOR gate A low & switches because the both inputs to NOR gate B are now low & the output be high.

Thus a 1 at the R input is said to be reset the flip-flop and it switches to the stable state where Q = 0.

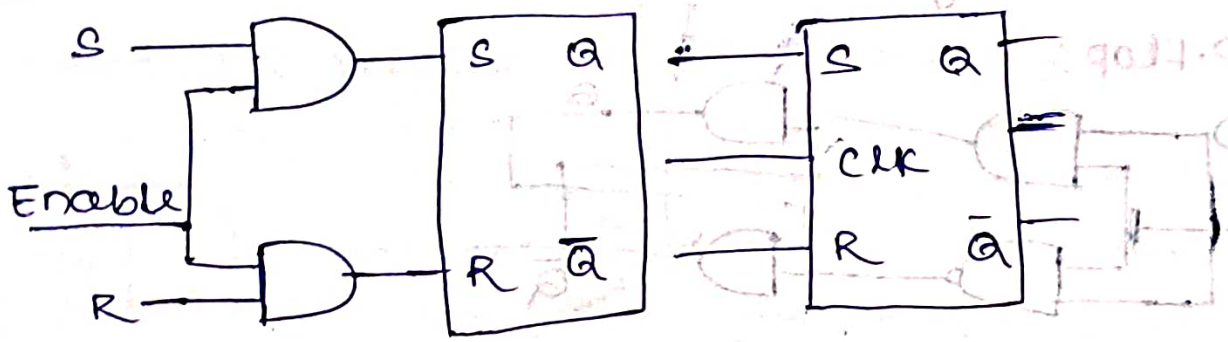
→ The last condition that is R = 1, S = 1, is forbidden & one can't predict what would be the value of Q & Q'.

Latch :-

An R-S flip-flop also called a latch or "Bistable Multivibrator".

Any change in the input information at R or S is transmitted immediately to Q or Q'.

Clocked R-S flip-flop :-



Truth-table :-

S	R	Q
0	0	No change
0	1	0
1	0	1
1	1	Forbidden

→ In the above figure when the enable input is high, information at R & S would be transmitted directly to the output of the AND gates.

→ The output of the R-S flip-flop will change in response to input changes as long as the enable is high.

→ When the enable input goes low, the output of the R-S flip-flop will retain the information.

Thus it is possible to stroke or clock the flip-flop in order to store the information (set it or Reset it) at any

time & then hold the information for any desired period of time.

such a system is called as R-S flip-flop.

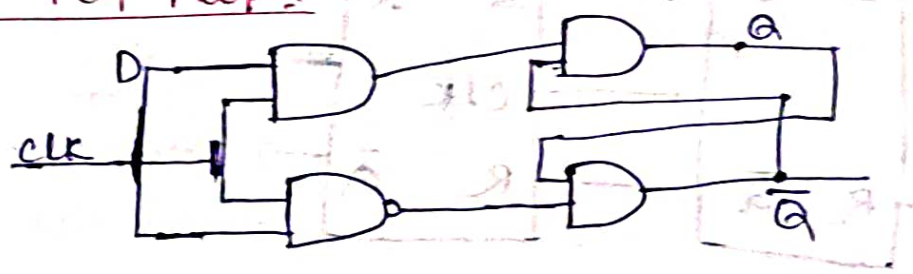
Disadvantages of R-S flip-flop:-

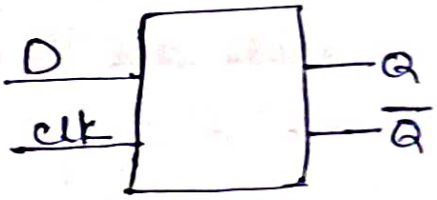
→ R-S flip-flop needs 2 data inputs namely a high S to store 1 bit similiarly to store 0 bit we need a high R.

Generation of 2 signals to drive a flip-flop in many application is difficult.

→ The R-S flip-flop put a forbidden condition that both R & S can't be high at the same time.

D-flip-flop:-





→ It gives a simply way to build a delay or D-flip-flop. This types of flip-flop prevents the value of D from reaching the Q output untill a clk pulse is received.

When the clk pulse is low both AND gates to which clock is applied are disabled. Therefore D can change value without affecting the value of Q. When clock is high both AND gates are enabled.

In this case Q is forced to equal the value of D. When the clock again goes low Q retains or stores the last value of D.

Thus D flip-flop is a bistable circuit whose D-inputs is transferred to the output Q after clk pulse is received.

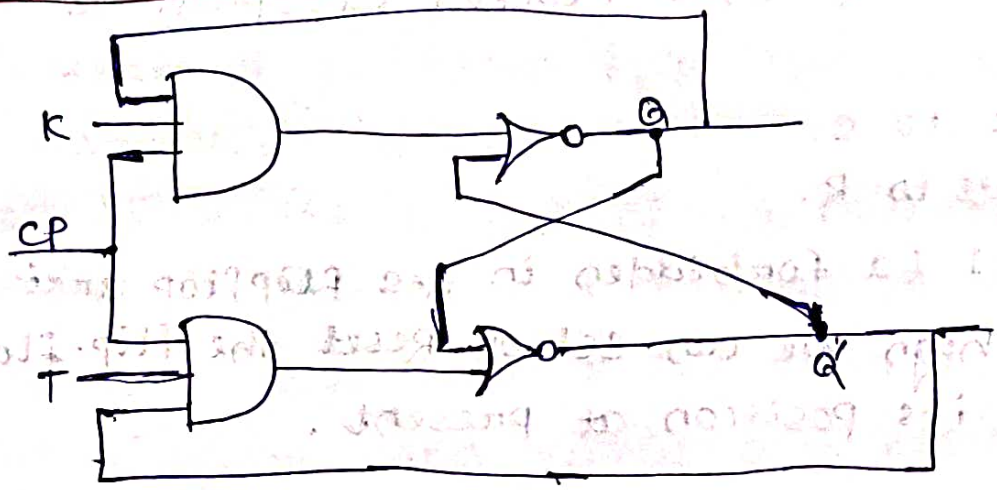
J-K Flip-flop :-

In a R-S flip-flop, the state of output is not predictable when R=1 & S=1.

→ The J-K flip-flop allows inputs J=1 & K=1. In this situation, the state of output is changed. Inputs J & K behave like input S & R to set & Reset. The flip-flop in J-K the letter J is for set & K is for reset.

→ When inputs are applied to both J & K simultaneously as 1 the flip-flop switches to its complement state i.e. if Q=1, it switches Q=0; & vice versa.

A clocked J-K flipflop :-



The above figure shows the diagram of clocked J-K flip-flop.

Working on J-K flip-flop :-

Output Q is ANDed with K & CP (clock pulse) so that the flipflop is cleared during a clock pulse only if Q was previously 1.

Similiarly output, Q' is ANDed with J & CD inputs so that the flip-flop is set with a clock pulse only if Q' was previously one.

Truth table:-

Q	J	K	Q(t+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

Function of flipflop determines by these signals. One is J and another is K when clocked ~~wise~~ pulse is arrive.

J is corresponds to S.

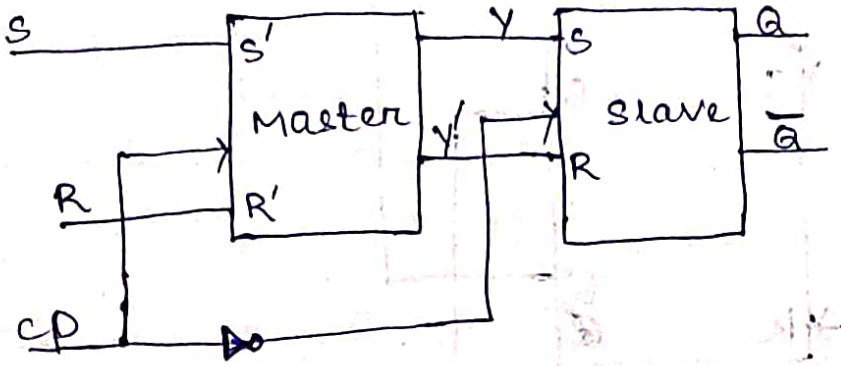
K is corresponds to R.

⇒ When J=1 & K=1 (a forbidden in R-S flipflop arrises J & K both are high we can set or Reset the flip-flop depending upon its position at present.

Truth table:-

Input		output
J	K	Q
0	0	last state
0	1	0
1	0	1
1	1	complement of last state

Master slave flip-flop :-



A master slave flip-flop is made up of a separate ~~sequence~~ flipflop circuit.

First part of the circuit serves as a master SR flip-flop & the 2nd part of the circuit serves as the slave flip-flop.

It consist of a master flip-flop a slave flip-flop & an inverter.

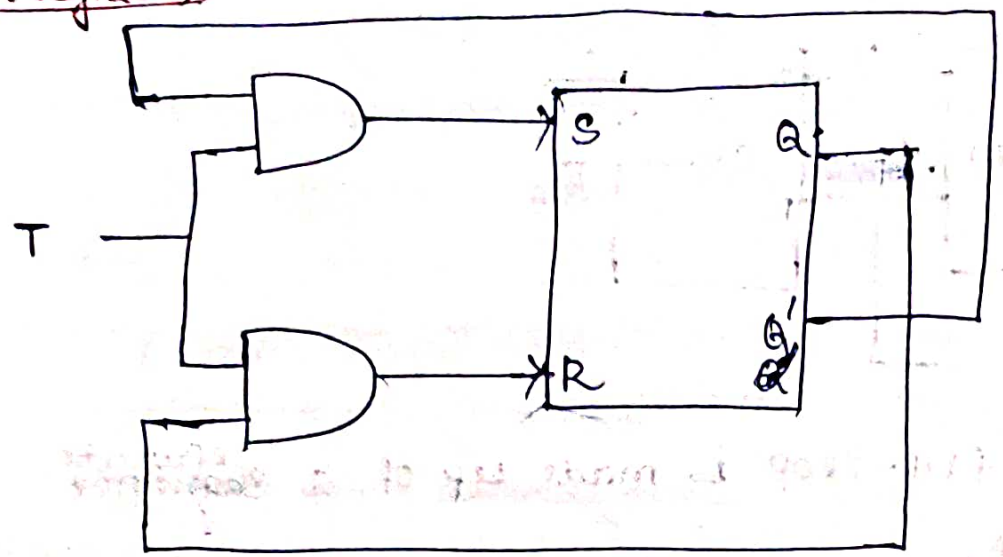
When CP = 0 the output of the inverter is 1. This is applied to the slave flip-flop SR flip-flop.

Since the clock input of the slave is now 1. The flip-flop is enable and output Q is equal to Y. While Q' is equal to Y'. The master flip flop will remain disabl during this time as the clock pulse is 0.

When the clock pulse again goes to 1 the information at the external R&S input is transmitted to the master flip-flop & thus Y and Y' get the values corresponding to the input of R-S flip-flop.

The slave flip-flop however is isolated as long as the clock pulse is at its 1 level because the output of the inverter through which the clock pulse must as to apply to the slave flip-flop is 0.

Diagram:-



Truth table:-

T	Q(t+1)
0	Q
1	$\overline{Q}$

Decoder:-

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n output.

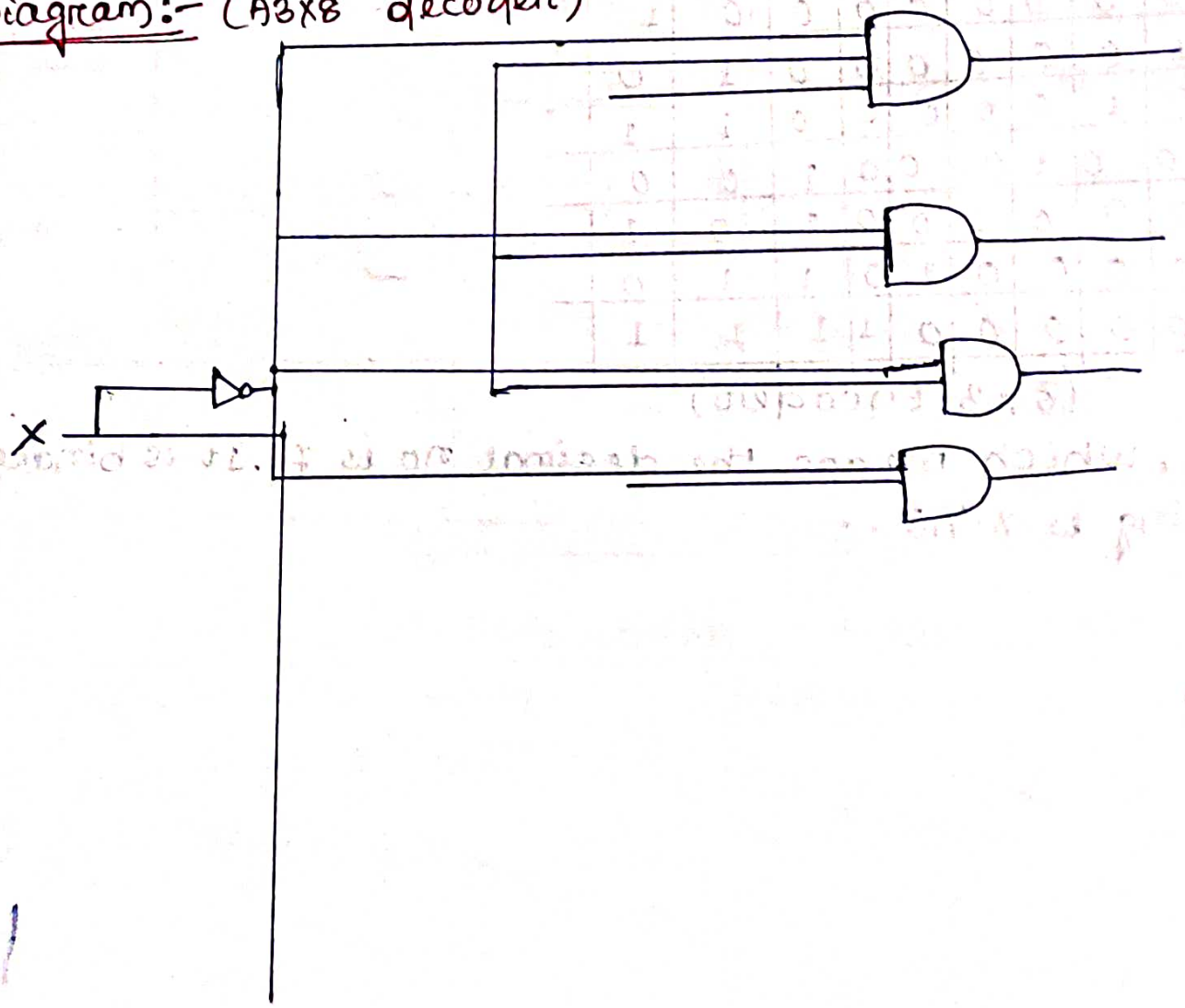
A decoder is said to be n×m decoder if it accepts n input and gives m output where $m <= 2^n$.

Ex:- consider a 3 to 8 decoder

The decoder has 8 outputs. Each of the outputs represent one of 8 combination of the 3 variables.

Inputs			Outputs							
X	Y	Z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Diagram:- (A3X8 decoder)



Encoders:- (Decimal to Binary encoder):-

Encoder just the reverse of the decoder. It take 2^n inputs and generates n output.

If we have 8 inputs i.e. 2^3 than the coded output will be in 3 bit shown in the figure.

Truth table:-

Inputs								outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

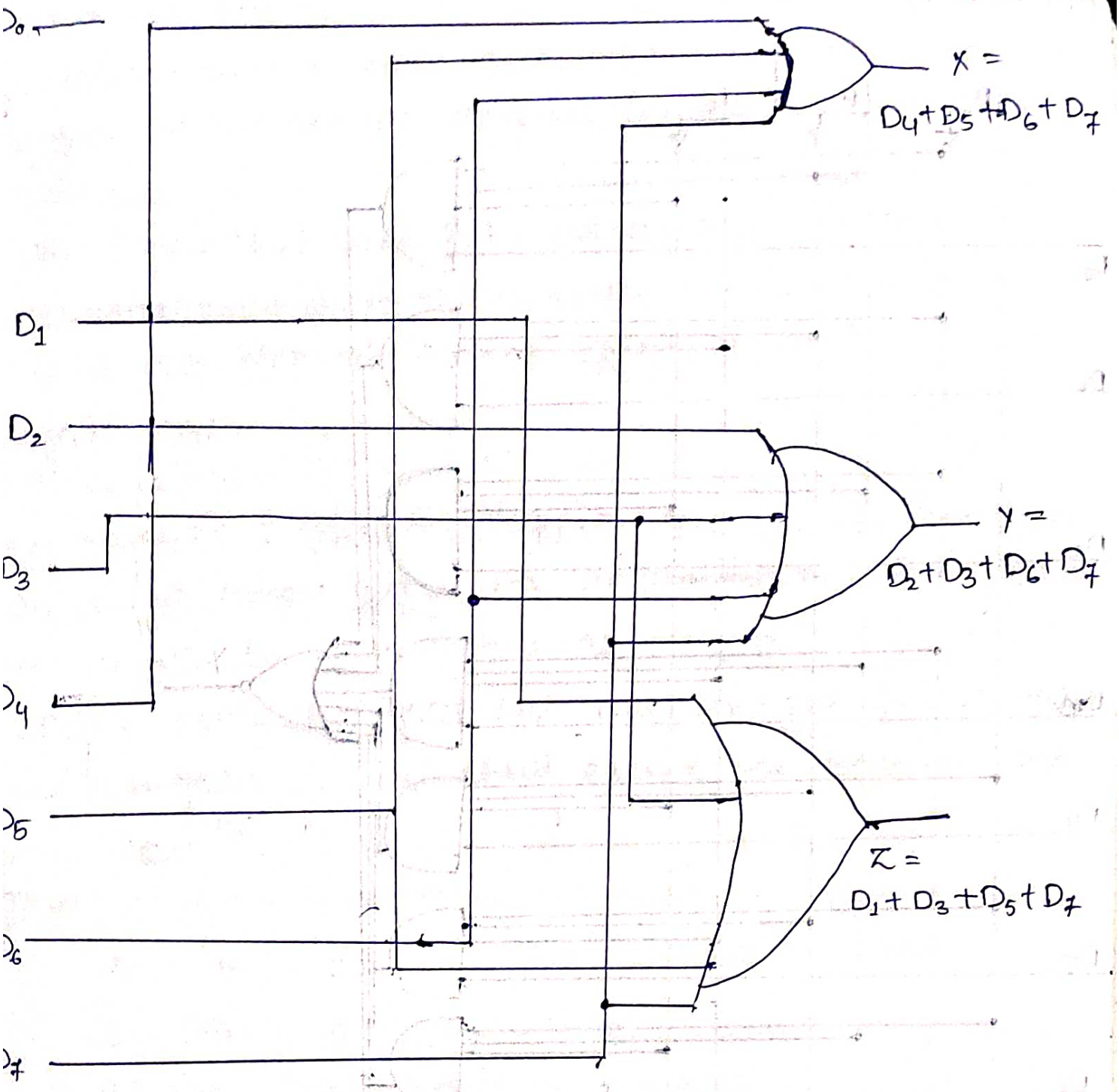
(8 x 3 Encoder)

$D_7 = 1$, which means the decimal no is 7. It is binary encoding is X, Y, Z.

$X = 1$

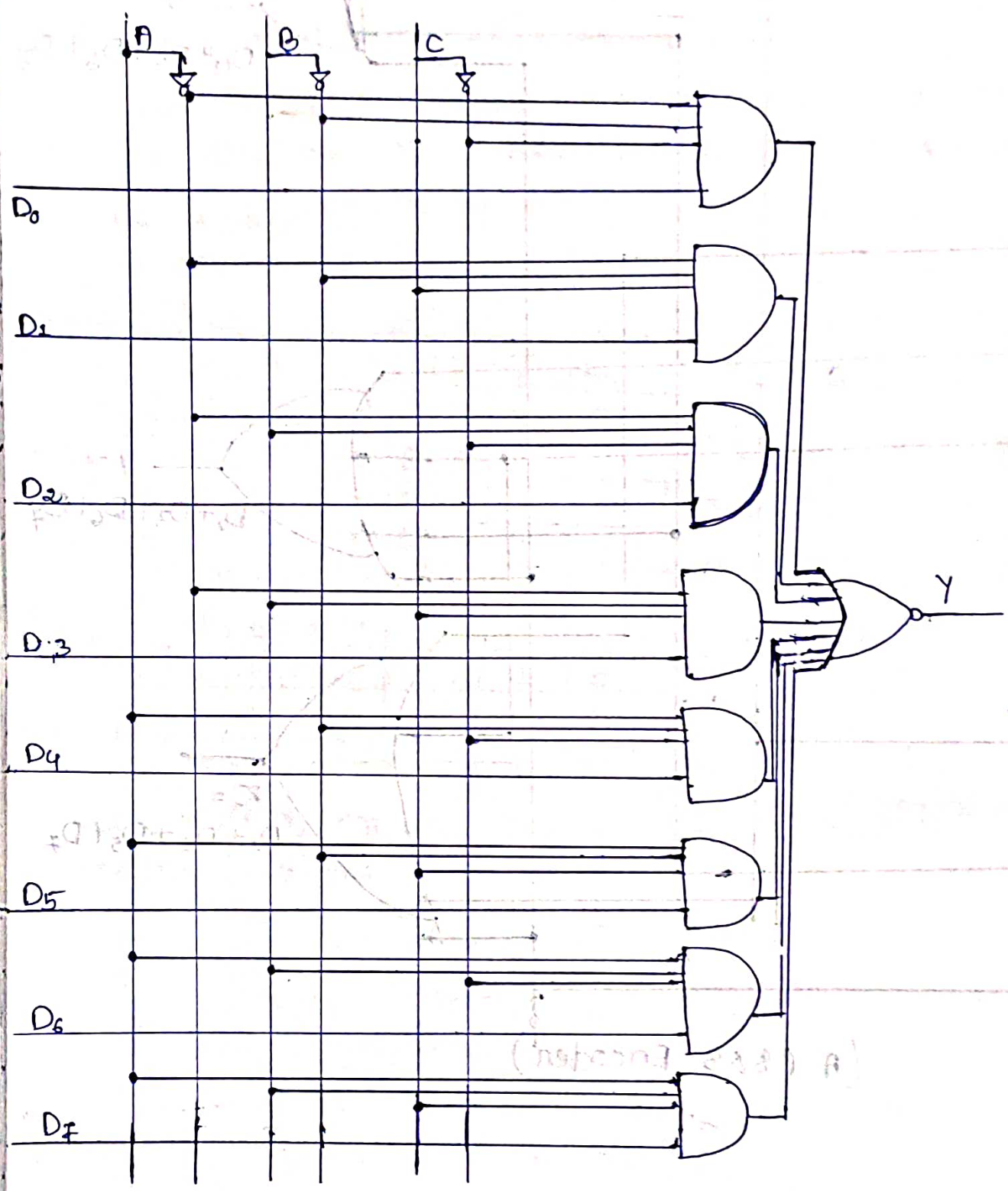
$Y = 1$

$Z = 1$



(A 8x3 Encoder)

Multiplexer :-



(Eight to one multiplexere)

This above figure shows a 8×1 (8 to 1) multiplexer. This is also called a data selector because the data bit coming in the output depends on the selected input data bit.

The input bits are labelled as D_0 to D_7 only of these bits transmitted to the output which bit is transmitted out depend on the value of control input signal ABC.

For ex:-

If ABC=000, then the upper AND gate is enabled while all other gates are displayed. Therefore data bit D_0 will be transmitted as output Y.

If the control signed ABC are changed to 111 then the all gates are disabled except the bottom most AND Gate.

Table:-

A	B	C	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

Resister:-

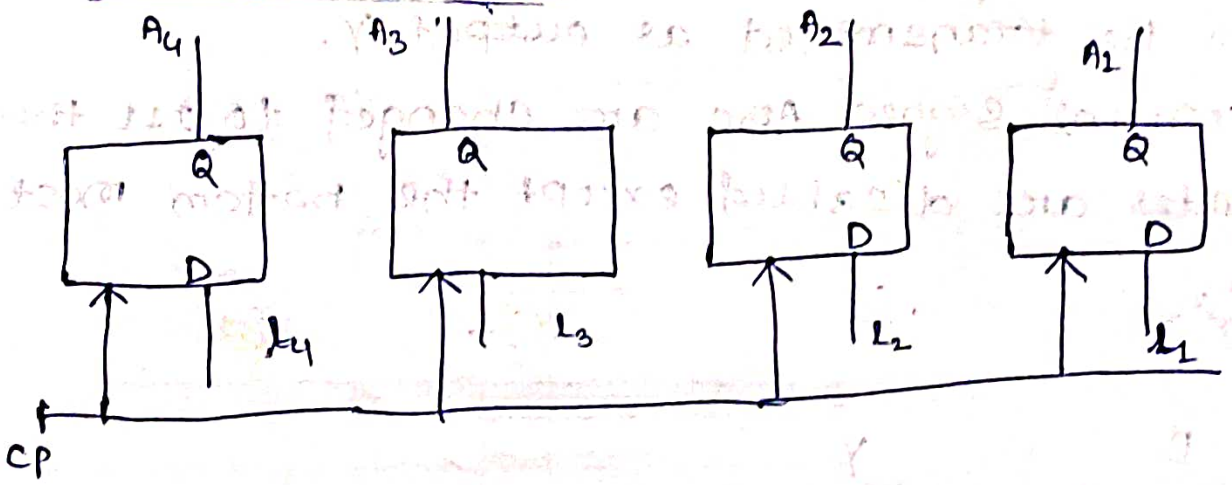
→ Resister are the temporary storage units of a computer that keep both data & instructions in a binary form.

→ A register stores a binary word. It also contains flipflops. The flipflops are connected in parallel.

→ All the n bits of an n-bit register are transferred in parallel.

→ The binary word in a register remain store untill a new binary word is applied to the register.

Ex:- 4 bit register containing D-flipflops



This is a resister constructed with 4 bit D type flipflop and a common clk pulse input, CP, enables all flipflop show that the information presently available at the 4 inputs can be transfered into the 4 bit register.

A clk pulse (CP) allowed on the above figure will load the values of I_1, I_2, I_3, I_4, into a resister and the values at A_1 to A_4 will be corresponding to the values of input I_1, I_2, I_3, I_4.

What is register loading ?

The transfer of new information into a resister is known as loading the resister.

→ When all the bits of the register are loaded simultaneously with a single clock pulse then we say that loading is done in parallel.

* In the right side, the figure shows the circuit ~~Program~~ diagram of a resister with parallel loading. The resister contain D-flipflop. The clock and clear inputs as applied is shown in the figure.

When the load input is 1, All the input such as I_1, I_2, I_3 & I_4 are transferred into the resister.

When the load input is 0, The D-flipflop is reloaded with their present value.

4 bit resister with clear input to remove the earlier values stored in the D-flipflop.

Shift-register :-

Shift Register are also resister and they are used for temporary storage of data.

→ but They are used where bit shifting is required for manipulating or processing the data.

→ Shift register can be constructed by taking J-K flip-flop, D Flipflop or R-S Flipflop.

→ In the shift register with parallel loading information is available from all bits of register and all bits can be transferred simultaneously using single clock pulse.

Some shift resister provide the neccessary input and output terminals for parallel transfer. They have the following capabilities.

(a) A clear control to clear the register to 0.

(b) A clock pulse input CP for clock pulses to synchronize at operation.

(c) A parallel load control to enable a parallel transfer and the N-input lines associated with the parallel transfer.

(d) N-parallel output lines

(e) A control state that leaves the information in the register unchanged even through CLK pulse are applied.

There are 2 types of register :-

(1) unidirectional shift register

(2) Bidirectional shift register

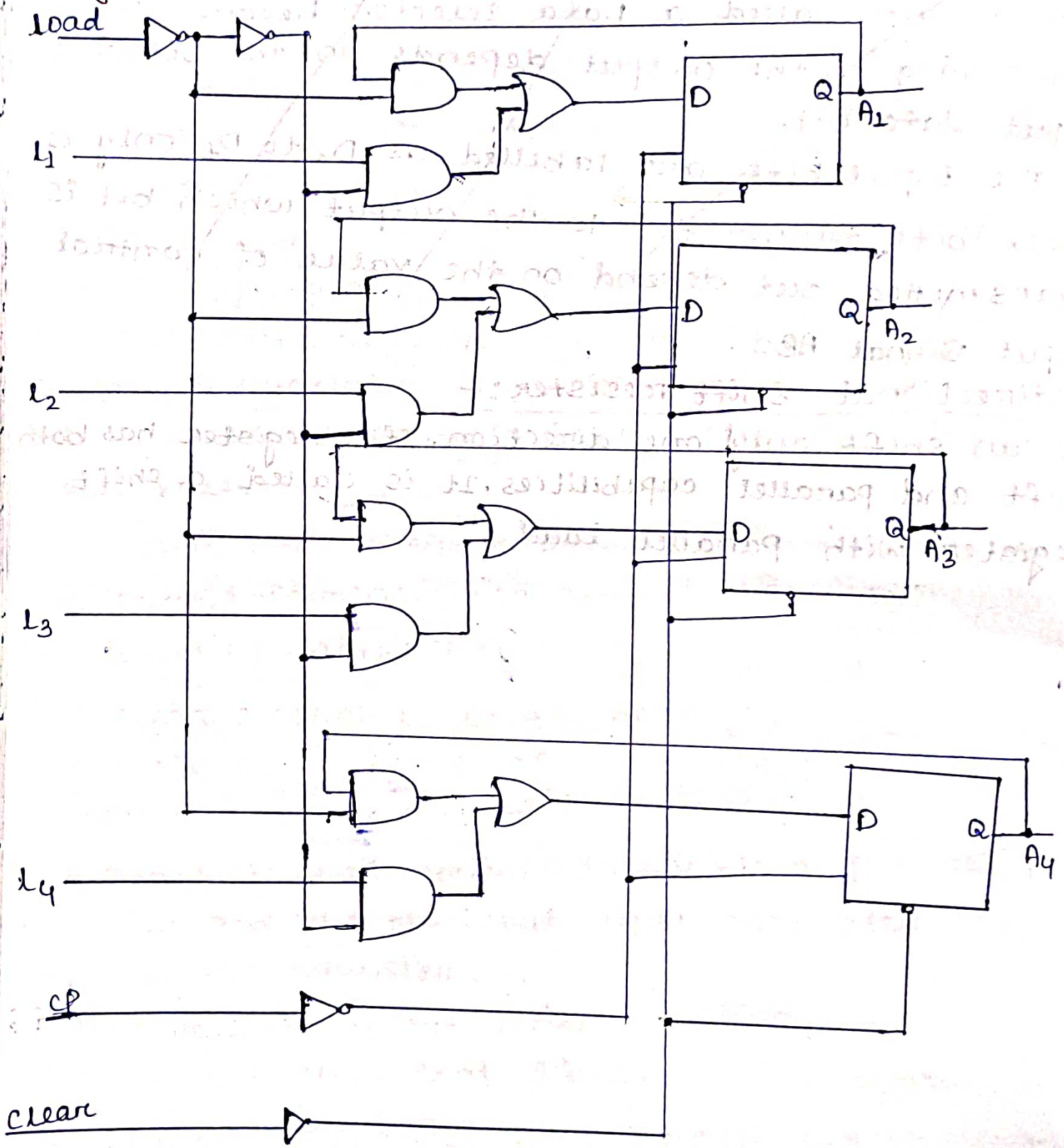
This above figure shows a 8 X 1 (8 to 1) multiplexer. This is also called a Data selector because the data bit coming in the output depends on the selected input data bit.

The input bits are labelled as D_0 to D_7 only of these bits transmitted to the output which bit is transmitted out depend on the value of control input signal ABC.

Undirectional Shift resister:-

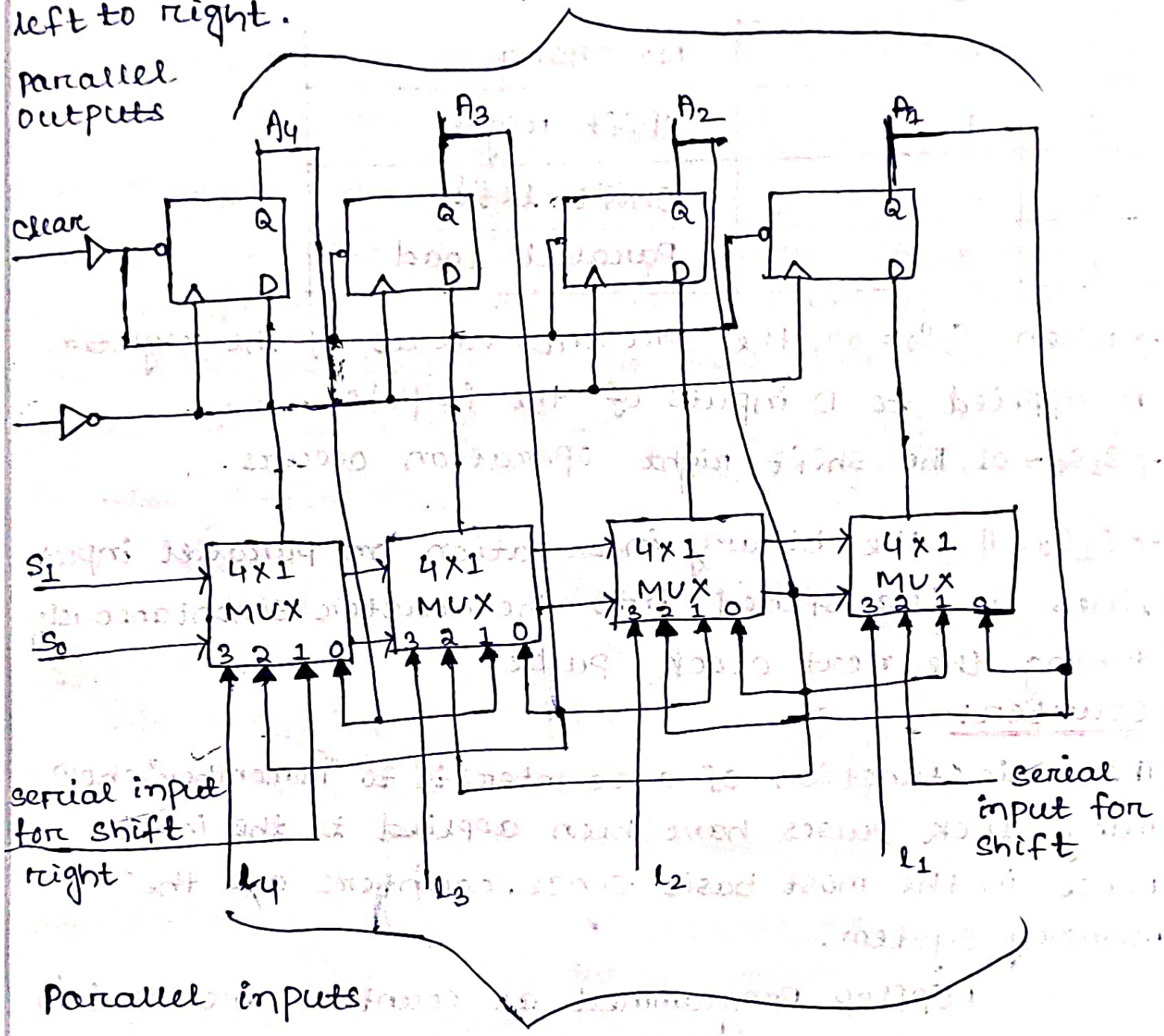
It can shift only one direction. If a register has both shift and parallel capabilities. It is called a shift register with parallel load.

Diagram (resister loading) :-



Bidirectional shift register:-

It is capable of shifting bits both right to left and left to right.



The above figure shows a bidirectional shift resister with Parallel loads. The 4 multiplexer are part of the register.

These multiplexer has a common selective variable S_1 & S_0.

The S_1 & S_0 inputs controll the made of operation of register as per the function entries in the given table.

Mode of control		Register operation
S_1	S_0	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

→ When $S_1S_0 = 00$, the present values of the register is applied to D inputs of the flipflop.

→ $S_1S_0 = 01$, The shift right operation occurs

→ $S_1S_0 = 11$ – the binary information on parallel input lines is transferred into the resister simontaneously during the next clock pulse.

Counter:-

The basic function of a counter is to "remember" how many clock pulses have been applied to the input. Hence in the most basic sense, counters are the memory system.

Flipflop programmed as counters are used in wide variety of counting application.

(1) In scientific instrument
(2) Industrial controls
(3) computer and communication equipments

counters are classified into 2 types.

(1) The ripple or Asynchronus counter
(2) Synchronus counter

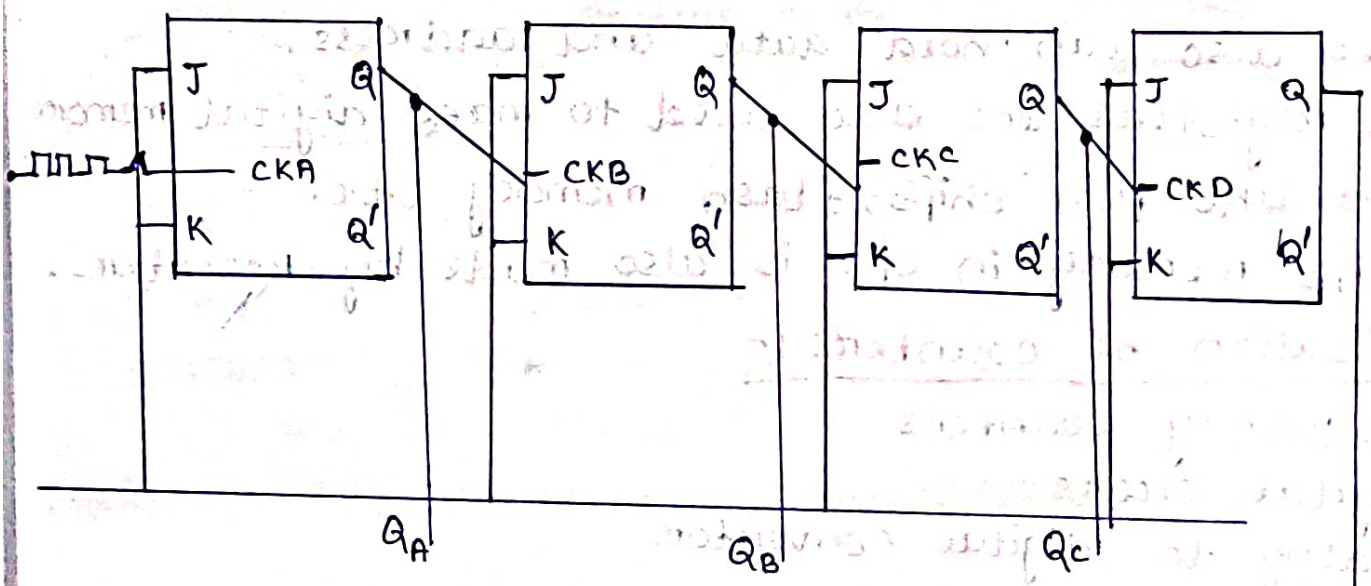
1) Asynchronous counter:-

→ ripple counter is an Asynchronous sequencial circuit, hence they named as Asynchronous counter and here same clock pulse is not applied.

2) Synchronous counter:-

Synchronous counter is a synchronous sequencial circuit. All the flipflops in the synchronous counter are under the control of the same clock pulse. All the counter are made up of flipflops arranged in series.

Diagram:-



(4 bit Ripple counter)

Asynchronous counter:-

A simple 4 bit binary ripple counter that uses 4 J-K flipflop as shown in the above figure.

→ Initially all flipflops are in the logic 0 state. ($Q_A = Q_B = Q_C = Q_D = 0$)

→ A clock pulse is applied to the clock input of a flipflop 'A' causing Q_A to change from logic 0 to 1. At this moment flipflop B, C, D don't state (i.e. remain in the logic 0 state.

→ Since there is no (-ve) going age of the clock pulse present at their clock input if Q_A & Q_D denote the least significant bit (LSB) and most significant bit (MSB) respectively after the 1st clock pulse is applied to the clock input to the flipflop Q_A. At that time counter will read.

MSB Q_D	Q_C	Q_B	Q_A (LSB)
0	0	0	1

Application of Register :-

→ The main application of register is storing data in digital form.

→ They also can hold data and address.

→ The registers are also used to make digital memory chips like ROM chips, flash memory e.t.c.

→ cache memory in CPU is also made by Resister.

Application of counters :-

→ Frequency counters

→ Digital clocks

→ Analog to digital convertor

→ With some changes in their design, counters can be used as frequency divider circuits.

→ In time measurement

→ We can design digital triangular wave genarotor by using counters.

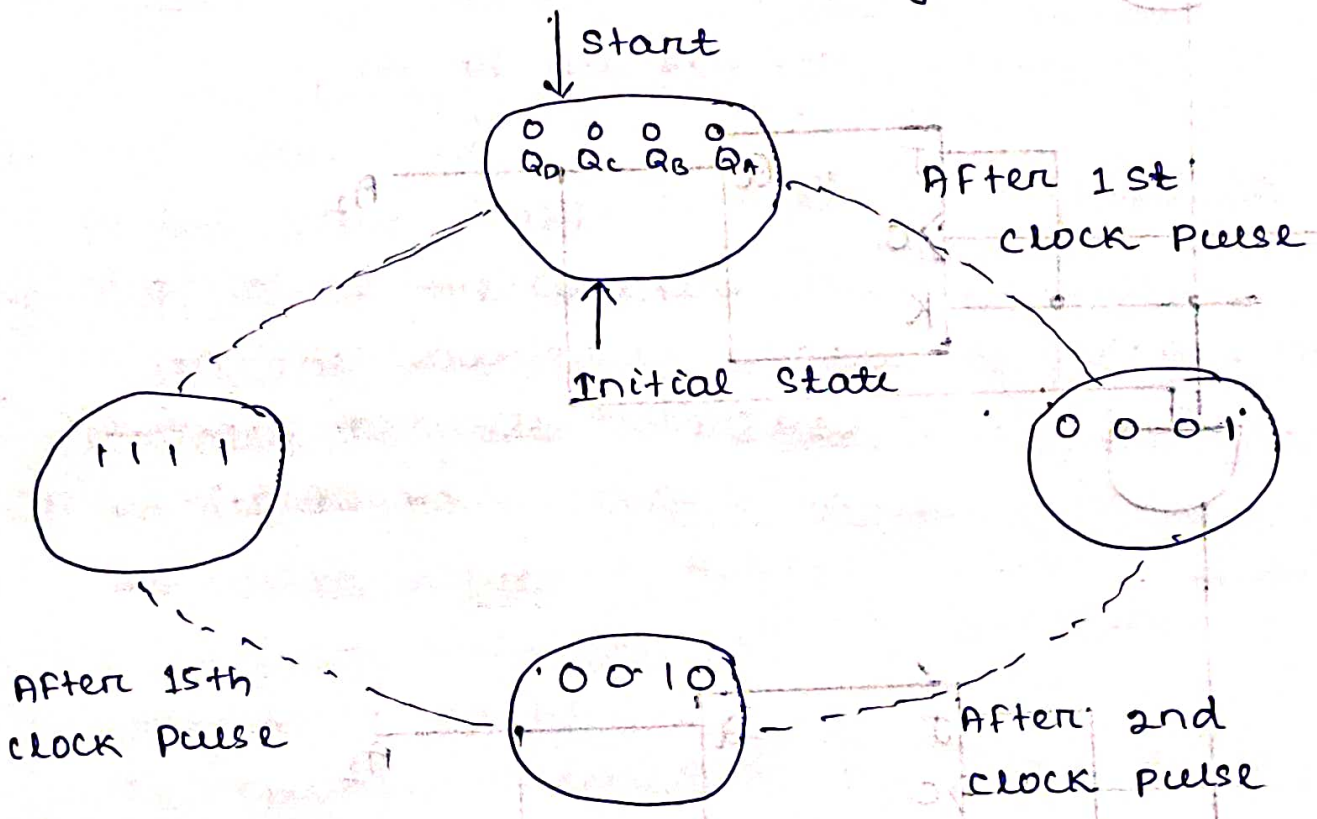
With the arrival of the 2nd clock pulse TO FLIPFLOP A Q_A goes from 1 to 0, This change of state creates the (-)ve going pulse edge needed, to triger flipflop B and thus Q_B goes from 0 to 1. At this moment flipflops C & D still remain in the logic 0

state. Thus counter will read.

(MSB) Q_D	Q_C	Q_B	Q_A (LSB)
0	0	1	0

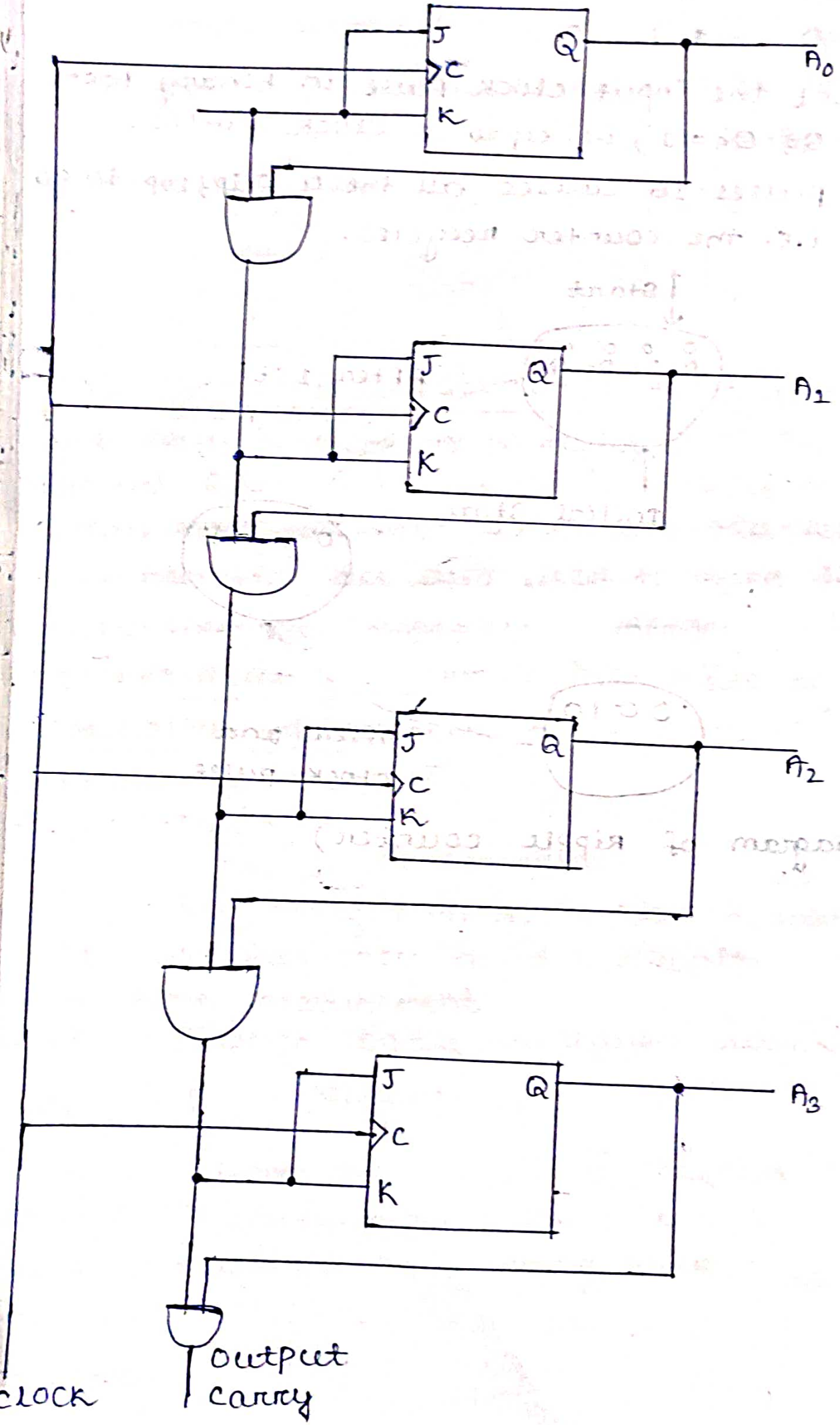
It will count the input clock pulse in binary form upto ($Q_D = Q_C = Q_B = Q_A = 1$) i.e. upto 15 clock pulses.

Clock pulses 16 causes all the 4 flipflop to go to state 0 i.e. the counter recycles.



(State diagram of ripple counter)

Synchronous counter:- (4 bit synchronous counter) :-



A 4 bit synchronous counter using 4 J-K flipflop as shown in the given figure.

A J-K flip flop complements its value when both J & K inputs are 1. The output of J-K flipflop doesn't change when the inputs J=K=0.

In synchronous counter all the 4 flip-flop are controlled by the same clock pulse.

The c inputs of all flipflops receive the common clock at the same time.

If the count enable is 0 all J & K inputs are 0 and the output of the counter doesn't change.

When the counter is enable by giving 1 then A_0 complemented because both J & K of 1st flip flop get 1 and so its toggle.

The other values A_1 A_2 & A_3 will also complemented when they are previoces flipflop are equal to 1 and the counter is enabled.

The counter is based on the principle that in the sequence 0000, 0001, 0010, 0111, 0100, 0101 upto 15th.

The low order bit is complemented after every count and every other bits is complemented if the low order bits are equal to 1.

PLDs:-

Programmable logic devices are the integrated circuits. They contain an array of AND gates and array of OR gates.

There are 3 types of PLDs based on the type of array.

(I) Programmable Read only memory
(II) Programmable array logic
(III) Programmable logic array

Programming:-

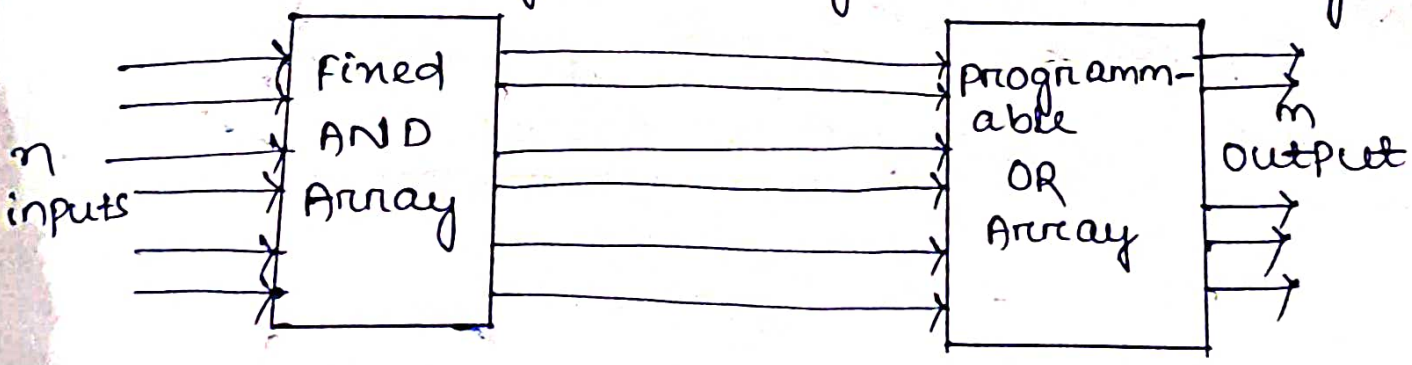
The Process or the method of entering the information or data into these devices is known as programming.

Basically users can program these devices or integrated circuits (ICs) electrically in order to implement the Boolean function based on the requirement.

(I) PROM:-

ROM is a memory device, which stores binary information permanently.

→ We can't change stored information
→ If the ROM has Programmable features, then it is called as Programmable ROM (PROM).
→ The user has flexibility Program the binary information electrically by using PROM Programmer.
→ PROM is a Programmable logic device that has fixed AND array and Programmable OR Array.



Here the inputs of AND gates are not programmable type. So we have to generate 2^n product terms by using 2^n AND gates having n inputs each. So the Decoder generates n minterm.

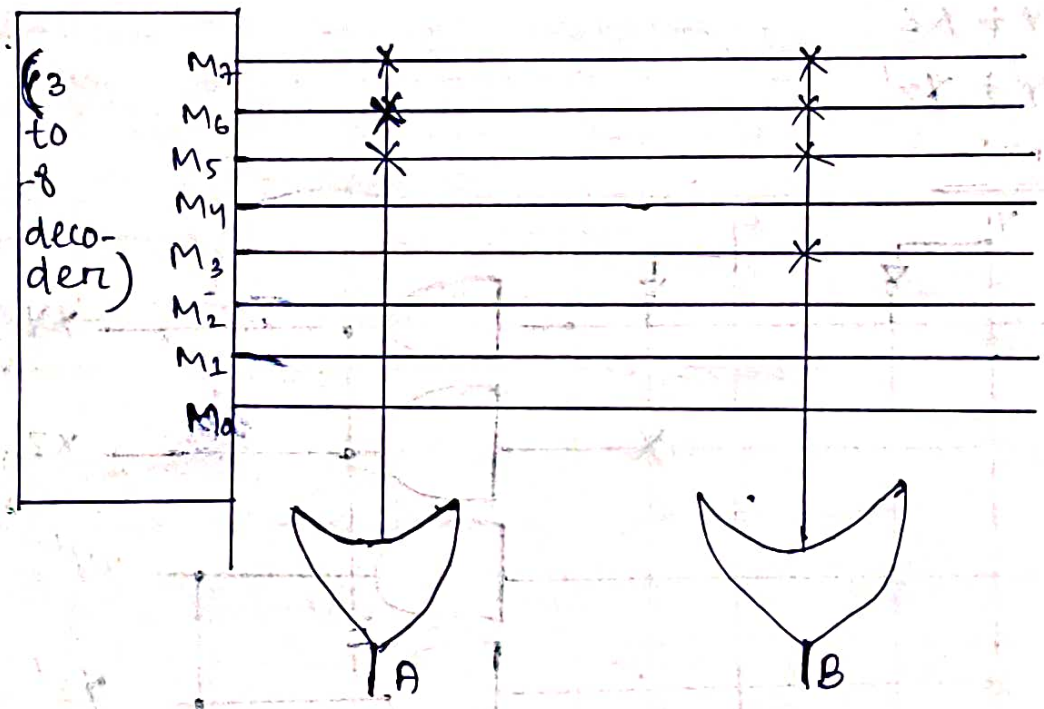
Ex Let us implement the following Boolean function using PROM.

$A(x,y,z) = \Sigma M(5,6,7)$

$B(x,y,z) = \Sigma m(3,5,6,7)$

The given 2 function are in sum of minterm forms & each function is having 3 variables x, y, z.

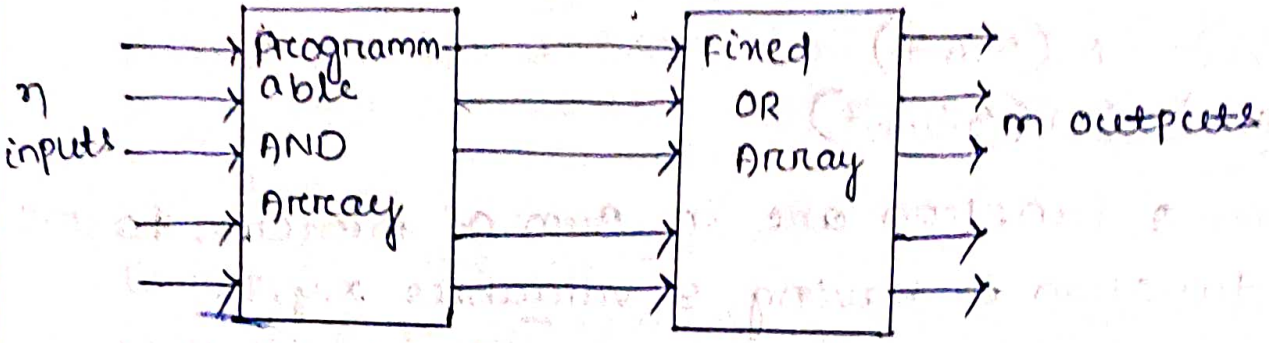
So here we required a 3 to 8 decoder and 2 programmable OR gates for producing 2 functions.



Here 3 to 8 decoder generates 8 minterm the two programmable OR gates have the access of all types of minterms. But only required the minterm are programmed in order to produce the respective boolean function by each OR gate. The symbol 'x' is used for programmable connection.

Programmable Array Logic (PAL):-

PAL is a programmable Logic device and that has programm by taking no. of AND array and OR array are fixed. The advantage of PAL is that he can generate only the required Product terms of Boolean function instead of generating all the minterms by using programmable AND gates.

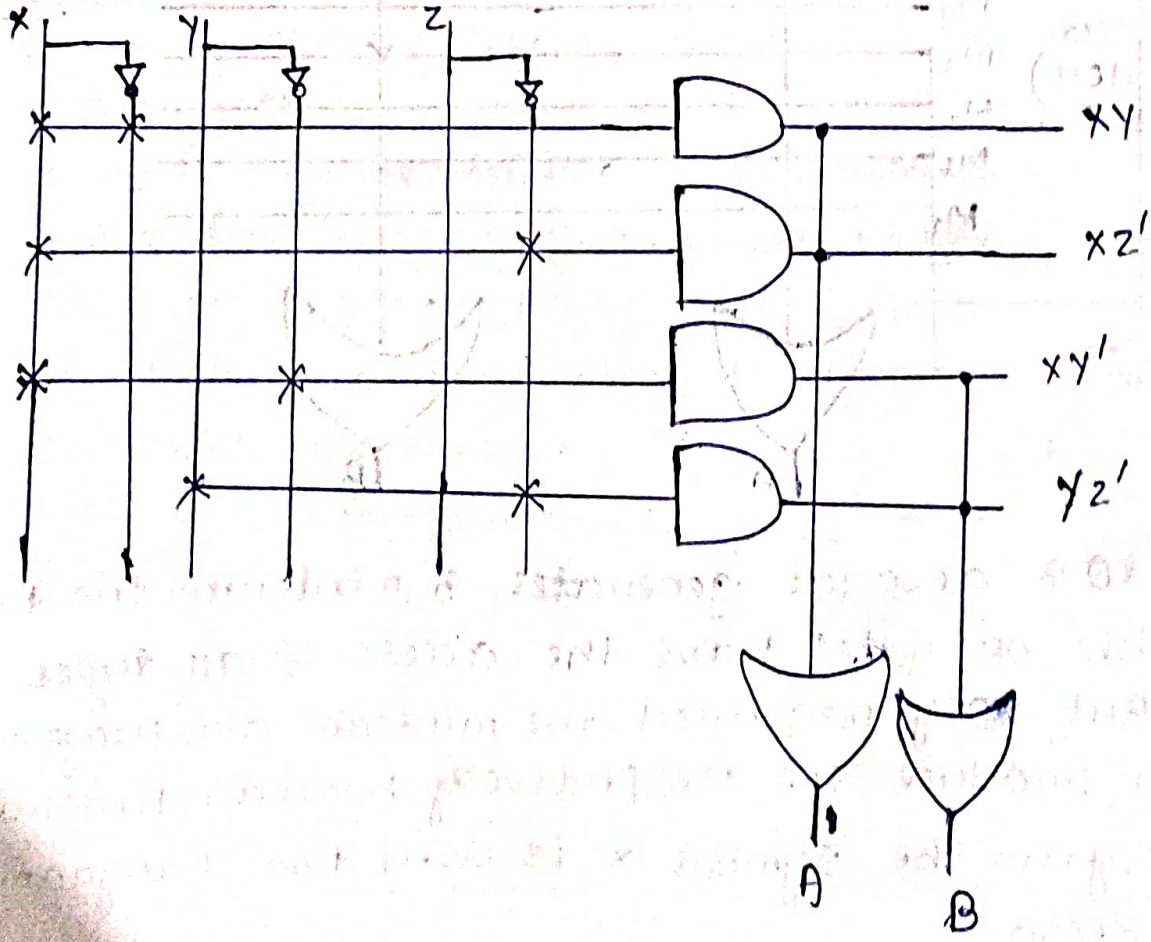


Let us implements the following Boolean function, using (PAL).

$A = XY + XZ'$

$B = XY' + YZ'$

Diagram:-



The Programmable AND gates have the access of both normal and complemented inputs of variable.

The above diagram the inputs are XX', YY', Z & Z'.

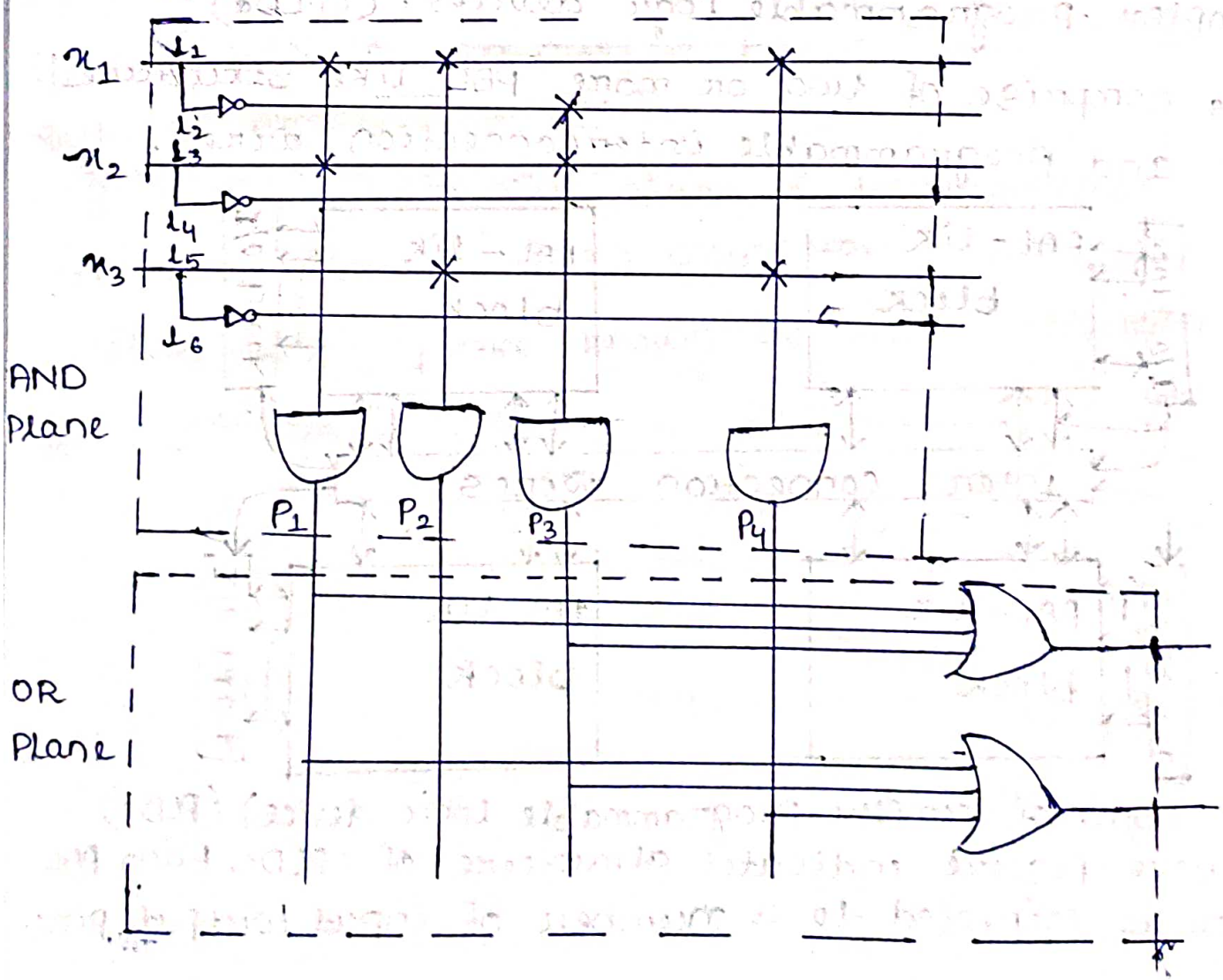
The no. of AND gates connected to each or gate in a PAL determines the maximum number of product terms that can be realised in a sum of product representation of a given function. The AND gates are permanently connected to specific orgates which mean that a particular product term can't be paired among output functions.

Programmable Logic Array (PLA):-

A circuit in which connection to both the AND the OR Array can be programmed is called a programmable Logic Array.

Ex:- $F_1 = x_1x_2 + x_1\bar{x}_3 + \bar{x}_1\bar{x}_2x_3$

$F_2 = x_1x_2 + \bar{x}_1\bar{x}_2x_3 + x_1x_3$



The above figure shows the diagram of Programmable Logic array (PLA).

The programmable connection must be such that if no connection is made to a given input of an AND gate the inputs behaves as if a logic value of 1 is driving it.

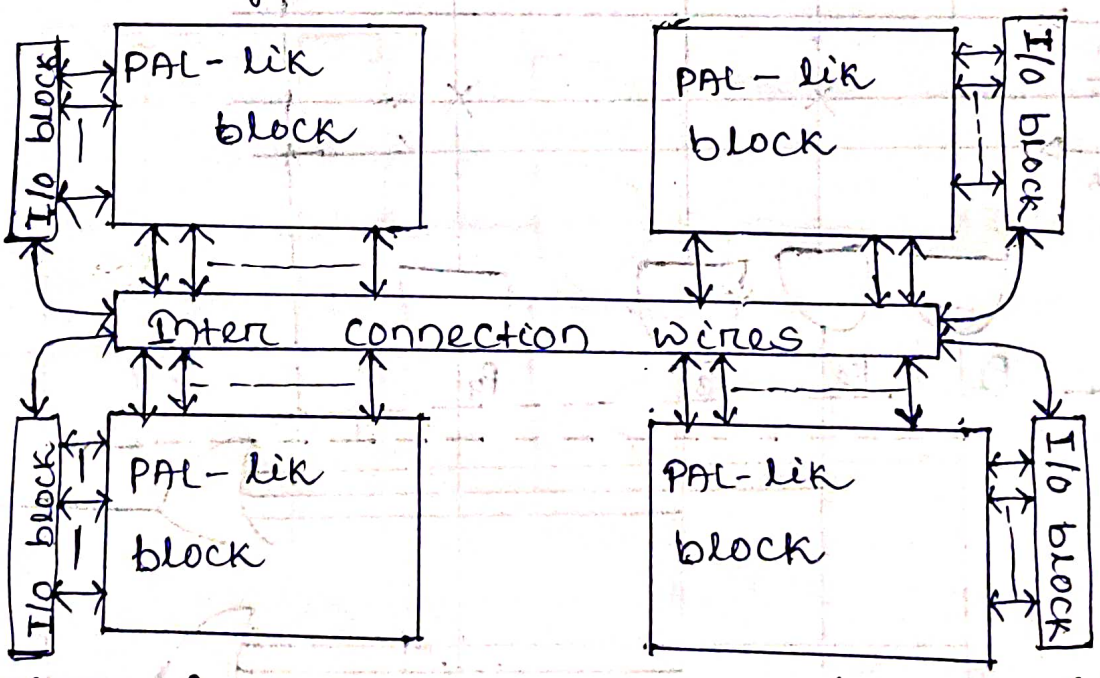
x(Cross) is placed on this line to represent each programmed connection.

CPLDs:-

* PALs are useful devices but that relatively small size means that many such chips may be needed to implement a typical digital system.

Larger devices of a similiar type have been developed to deal with this issue. They are known as complex programmable logic devices. (CPLDs)

→ CPLDs comprise of two or more PAL like structure blocks and programmable interconnection wires.



(Str. of complex programmable logic device) (PLDs)

The above figure indicates structure of CPLDs. Each PAL lik block is connected to a number of input/output pins.

→ connection between PAL lik blocks ~~is connected~~ are establish by programming the switches associated with the interconnected wires.

→ The interconnection resource consists of Horizontal and vertical wires.

Each horizontal wires can be connected to sum of vertical wires by programming the corresponding switches.

CPLD comes in different size ranging from 2 to more than 100 PAL lik blocks.

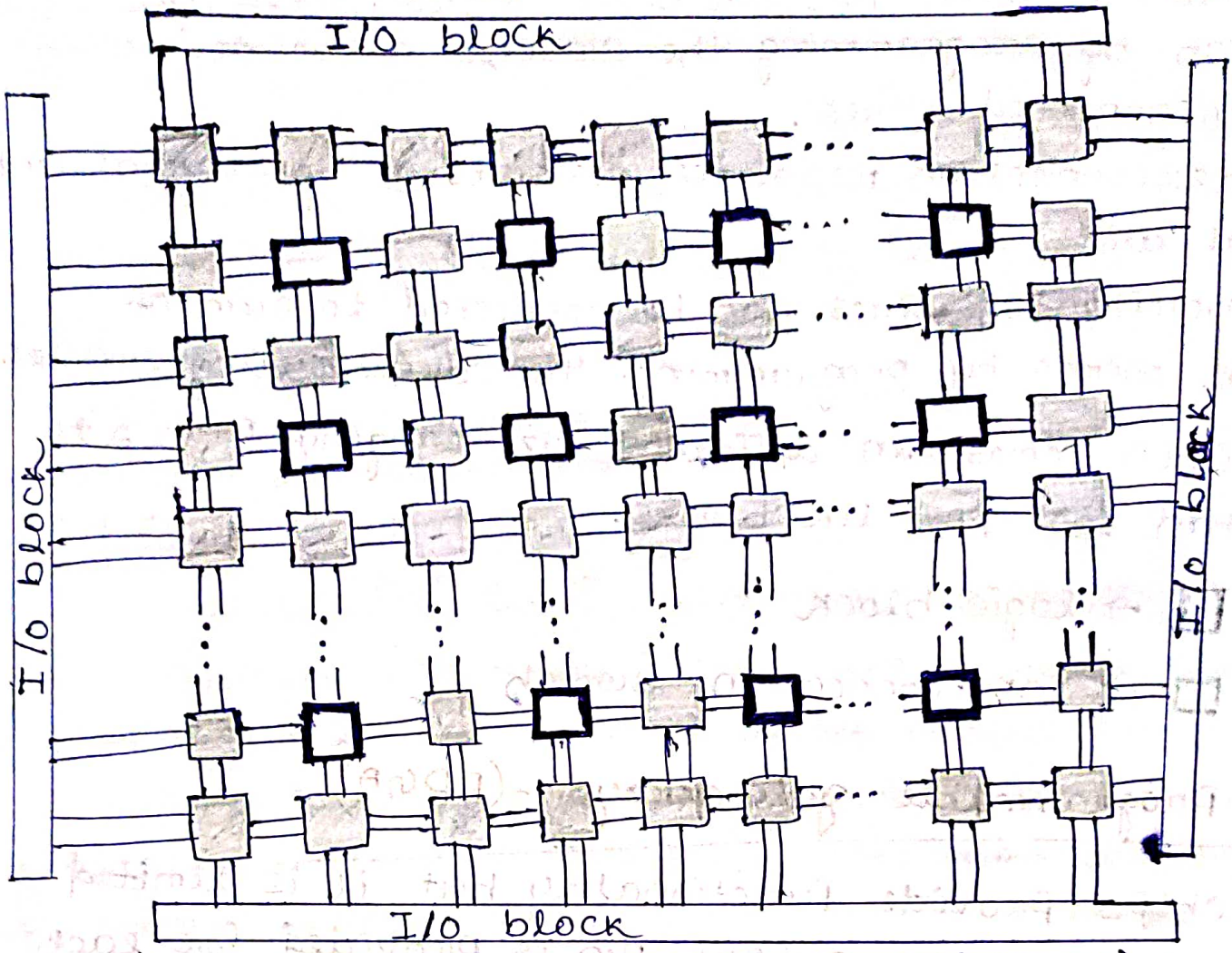
□ → logic block

■ → Interconnection switch

Field Programmable gate arrays :- (FPGA)

→ PAL chips provide functionality but it is limited in size because an output pin is provided for each sum of product circuits.

→ So a more powerful class of programmable devices has been developed to overcome these size limitations. They are known as field programmable gate arrays (FPGAs).



(A conceptual block diagram of FPGA)

The above fig. shows a conceptual block diagram of an FPGA. It consists of an array of logic blocks (indicated as black boxes) that can be connected by general interconnection resources.

The interconnect shown in blue consists of segments of wire and programmable switches.

→ The switches are use to connect the logic blocks to the wire segments and to establish connection between different wire segments.

→ This allows a large degree of rooting flexibility on the chip.

→ Input and output buffers are provided for access to the pins of the chip.

Different between FPGA & CPLDs :-

CPCD	FPGA
① An integrated circuit that helps to implement digital system.	① An integrated circuit designed to be configured by a customer or a designer after manufacturing.
② Stands for complex programmable Logic device.	② Stands for field programmable gate array.
③ provide maximum logic resources.	③ provide a massive amount of logic resources and storage elements to create complex system.
④ cost effective.	④ Expensive than CPCDs.
⑤ made up of larger blocks.	⑤ made up of tiny logic blocks.
⑥ uses EEPROM (non volatile).	⑥ uses RAM (volatile)
⑦ Easier to predict the delays.	⑦ Not easy to predict the delays
⑧ consumes low power.	⑧ consumes more power.
⑨ more secure	⑨ Less secure
⑩ suitable for small to medium scale application.	⑩ suitable for complex applications.