NUMBER SYSTEM – used in computers

- Binary $(\ )_2$
- Octal $(\ )_8$
- Decimal $(\ )_{10}$
- Hexa-Decimal $(\ )_{16}$

→ Number system is used for counting, measurement etc. Computer understands the data in numbers (i.e in binary 0 and 1).

		Base (Radix)	
Binary	(0, 1)	2	
Octal	(0→7) → (0, 1, 2, 3, 4, 5, 6, 7)	8	
Decimal	Range (0→9)	10	(0, 1, 2, 3, 4, 5, 6, 7 8, 9)
Hexadecimal	(0→15)	16	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ~~10~~, ~~11~~, ~~12~~, ~~13~~, ~~14~~, ~~15~~) A, B, C D E F

Base		Range
$(\ )_n$	→	$(0-(n-1))$

E.g: $(\ )_{12}$ → $(0-11)$

Table:- Octal $(\ )_8 \rightarrow 2^3$

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

Scanned with CamScanner

CONVERSION

(1) Decimal to others – $(\ )_{10} \rightarrow (\ )_n$

$\div n$ $\times n$

i) Decimal → Binary

(Ex) $(19.35)_{10} \rightarrow (\ )_2$

$\div 2$ $\times 2$

2	19	
2	9	1 → LSB
2	4	1
2	2	0
	1	0

↑ MSB

$\Rightarrow 10011.0101$

$0.35 \times 2 = 0.70$ 0

$0.70 \times 2 = 1.40$ 1

$0.40 \times 2 = 0.80$ 0

$0.80 \times 2 = 1.60$ 1

2. $(14)_{10} \rightarrow (\ )_2$

2	14	
2	7	0
2	3	1
	1	1

$= (1110)_2$

3. $(30)_{10} \rightarrow (\ )_2$

2	30	
2	15	0
2	7	1
2	3	1
	1	1

MSB

$= (11110)_2$

4. $(0.625)_{10} \rightarrow (\ )_2$

	Real part		fractional part		Real part
$0.625 \times 2 =$	1	.	25	⇒	1
$0.25 \times 2 =$	0	.	50	⇒	0
$0.50 \times 2 =$	1	.	00	⇒	1
$0.00 \times 2 =$	0	.	00	⇒	0

$(0.625)_{10} \rightarrow (0.1010)$

Scanned with CamScanner

Decimal → Octal

1. $(19.35)_{10} \longrightarrow (\ )_8$

$\div 8 \quad \times 2$

Successive division :-

8	19
	2 → 3
	MSB LSB

$0.35 \times 8 = 2.80 \quad 2$

$0.80 \times 8 = 6.40 \quad 6$

$0.40 \times 8 = 3.20 \quad 3$

$\Rightarrow (23.26)_8$

2. $(172.878)_{10} \longrightarrow (\ )_8$

8	172	
8	21	− 4
	2	− 5

$(172)_{10} \rightarrow \cancel{(245)_8}\ (254)_8$

Successive multiplication :-

$0.878 \times 8 = 7.024 \Rightarrow 7$

$0.024 \times 8 = 0.192 \Rightarrow 0$

$0.192 \times 8 = 1.536 \Rightarrow 1$

$0.536 \times 8 = 4.288 \Rightarrow 4$

$(0.878)_{10} = (.7014)_8$

$\Rightarrow (172.878)_{10} \rightarrow (254.7014)_8$

Scanned with CamScanner

iii) Decimal to Hexadecimal

Ex: $(19.35)_{10} \longrightarrow (\#)_{16}$

$$\begin{array}{r|l} 16 & 19 \\ \hline & 1 \end{array} \nearrow 3$$

$0.35 \times 16 = 5.60$ 5

$0.60 \times 16 = 9.60$ 9

$\Rightarrow (13.59)_{16}$

Ex: $(786.983)_{10} = (x)_{16}$. find the value of x.

$$\begin{array}{r|l} 16 & 786 \\ \hline 16 & 49 \quad - 2 \\ \hline & 3 \quad - 1 \end{array}$$

$(786)_{10} = (312)_{16}$

Real part

$0.983 \times 16 = 15.728 \Rightarrow 15\ (F)$

$0.728 \times 16 = 11.648 \Rightarrow 11\ (B)$

$0.648 \times 16 = 10.368 \Rightarrow 10\ (A)$

$0.368 \times 16 = 5.888 \Rightarrow 5$

$(786.983)_{10} = (312.FBA5)_{16}$

Ex: $(82.35)_{10}$ to $(\ )_4$.

$$\begin{array}{r|l} 4 & 82 \\ \hline 4 & 20 \quad - 2 \\ \hline 4 & 5 \quad - 0 \\ \hline & 1 \quad - 1 \end{array}$$

$0.35 \times 4 = 1.4 \Rightarrow 1$

$0.4 \times 4 = 1.6 \Rightarrow 1$

$0.6 \times 4 = 2.4 \Rightarrow 2$

$0.4 \times 4 = 1.6 \Rightarrow 1$

$= (1102.1121)_4$

Scanned with CamScanner

) Others to Decimal — $(\,.\,)_n \rightarrow (\,)_{10}$

n^+ n^-

i) Binary to Decimal -

Positional weight for integer part — Positional weight for fractional part

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

Binary point

Ex: $(1101.01)_2 \rightarrow (\,)_{10}$

2^+ 2^{+-}

$\Rightarrow 1\times2^3+1\times2^2+0\times2^1+1\times2^0 . 0\times2^{-1}+1\times2^{-2}$

$= 8+4+0+0+0+\frac{1}{4} = (13.25)_{10}$

Ex: $(11010.011)_2 \rightarrow (\,)_{10}$

$1\times2^4+1\times2^3\ 0\times2^2+2^1+2^0\times0 . 0\times2^{-1}+1\times2^{-2}+1\times2^{-3}$

$= 16+8+0+2+0+0+0.25+0.125$

$= (26.375)_{10}$

Ex: a) $(1101.011)_2 \rightarrow (\,)_{10}$

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

$= 8+4+0+1+0+0.25+0.125 = (13.375)_{10}$

b) 11101.101

$1\times2^4+1\times2^3+1\times2^2+0\times2^1+1\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$

$= 16+8+4+0+1+0.5+0+0.125$

$= (29.625)_{10}$

Scanned with CamScanner

ii) Octal to Decimal: $(\ )_8 \rightarrow (\ )_{10}$

1. $(17.4)_8 \rightarrow (\ )_{10}$

$(1\times8^1 + 7\times8^0 . 4\times8^{-1}) = (15.5)_{10}$

2. $(763.45)_8 \rightarrow (\ )_{10}$

$7\times8^2 + 6\times8^1 + 3\times8^0 + 4\times8^{-1} + 5\times8^{-2} = (499.578125)_{10}$

3. $(467.342)_8 \rightarrow (\ )_{10}$

$4\times8^2 + 6\times8^1 + 7\times8^0 + 3\times8^{-1} + 4\times8^{-2} + 2\times8^{-3}$

$= (311.4414)_{10}$

4. $(23.734)_8 \rightarrow (\ )_{10}$

$\Rightarrow 2\times8^1 + 3\times8^0 + 7\times8^{-1} + 3\times8^{-2} + 4\times8^{-3}$

$= (19.9296)_{10}$

Scanned with CamScanner

Hexadecimal → Decimal $(\ )_{16} \rightarrow (\ )_{10}$

1. $(A7.C)_{16} \rightarrow (\ )_{10}$

$$(A\times16^1 + 7\times16^0 . C\times16^{-1}) = (10\times16 + 7\times1 . 12\times16^{-1})$$

$$= (167.75)_{10}$$

2. $(ABC.75)_{16} \rightarrow (\ )_{10}$

$$A\times16^2 + B\times16^1 + C\times16^0 + 7\times16^{-1} + 5\times16^{-2}$$

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

$$10\times16^2 + 11\times16^1 + 12\times16^0 + 7\times16^{-1} + 5\times16^{-2}$$

$$= 2560 + 176 + 12 + 0.4375 + 0.0195$$

$$= (2748.457)_{10}$$

3. $(765.ECE)_{16} \rightarrow (\ )_{10}$

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

$$14 \quad 12 \quad 14$$

$$\Rightarrow 7\times16^2 + 6\times16^1 + 5\times16^0 + 14\times16^{-1} + 12\times16^{-2} + 14\times16^{-3}$$

$$= (1893.925)_{10}$$

4. $(73.20)_{16} \rightarrow (\ )_{10}$

$$= 7\times16^1 + 3\times16^0 + 2\times16^{-1} + 0\times16^{-2}$$

$$= (115.125)_{10}$$

Scanned with CamScanner

③ Conversion between Base (Power of 2)

$X = Y$	Binary $()_2$	Octal $()_8$	Hexa decimal $()_{16}$
Power of 2	2^1	2^3	2^4

i) B ⇄ O

ii) O ⇄ H

iii) B ⇄ H

i) * $(11011.110)_2 = (\quad)_8$

Octal need 3 bit from 0 → 7.

→ pair 3 bits.

$$\underbrace{011}\,\underbrace{011}.\underbrace{110}$$

$$(3\ 3\ .6)_8$$

ii) * $\underbrace{001}\,\underbrace{011}\,\underbrace{011}.\underbrace{011}\,\underbrace{011}$

$$(1\ 3\ 3.\ 3\ 3)_8$$

$(3\ 3.6)_8 \longrightarrow (\quad)_2$

$$(011\ 011.110)_2$$

Scanned with CamScanner

ii) $(237.61)_8 = (\quad)_{16}$

$(010\ 011\ 111 \cdot 110\ 001\ 0\ 0)$

$(0\ 9\ \ F \cdot C\ 4)_{16}$

$(09F.C4)_{16} \longrightarrow (\quad)_8$

iii) * Binary → Hexadecimal

$(10\ 1110\ 11 \cdot 1101101)_2$

$(B\ \ B\ .\ D\ \ A)_{16}$

* $(ABC.67B)_{16} \rightarrow (\quad)_2$

Hexadecimal number	A	B	C	.	6	7	B
Binary number	1010	1011	1100		0110	0111	1011

Scanned with CamScanner

conversion b/w base (not power of 2) :-

$$(\ )_n = (\ )_m$$

$$(\ )_n \rightarrow (\ )_{10} \rightarrow (\ )_m$$

Ex $(43.12)_5 = (\ )_7$

Base(7) → range(0 → 6)

$$(4\times5^1 + 3\times5^0 + 1\times5^{-1} + 2\times5^{-2})$$

$$= (23.28)_{10}$$

$$\downarrow$$

$$(\ )_7$$

⇒

7	23.0	
	3	2

$0.28 \times 7 = 1.96$ 1

$0.96 \times 7 = 6.72$ 6

$0.72 \times 7 = 5.04$ 5

$(43.12)_5 = (32.165)_7$

→ When Base increases, number reduces.

Ex: $(786.57)_9 = (\ )_6$

$$(7\times9^2 + 8\times9^1 + 6\times9^0 + 5\times9^{-1} + 7\times9^{-2})$$

$$= \cancel{(645)_{10}} \quad (645.636)_{10}$$

$\cancel{0.57\times9}$

6	645	
6	107	3
6	17	5
	2	5

$0.636 \times 6 = 3.816$ 3

$.816 \times 6 = 4.896$ 4

$.896 \times 6 = 5.376$ 5

⇒ $(2553.345)_6$

Scanned with CamScanner

Compliments

- Compliments are used in digital computers to simplify the subtraction. Complement of a number system is also used to represent negative numbers.
- classified in two types:
  1. r's complement or Radix complement
  2. (r-1)'s complement or Diminished complement

For Decimal number system -
  i) 10's complement
  ii) 9's complement

For Binary number system -
  i) 2's complement
  ii) 1's complement

<u>r's complement</u> - If N is a positive number to base r with integer part of n digits, the r's complement of N is defined as

$$\boxed{r\text{'s comp.} = r^n - N}$$

r → base of number
n → no. of integer digit
N → given +ve no.

$$\boxed{r\text{'s complement} = (r-1)\text{'s complement} + 1}$$

Ex 1. Find 10's complement of the given decimal no.

a) 7634 :-

$n = 4$

Base $r = 10$

10's comp. of 7634 $= 10^4 - 7634$

$= 10000 - 7634 = 2366$

Scanned with CamScanner

b) 76.34

$n = 2$

$r = 10$

10's comp. of 76.34 $= (10^2 - 76.34)$

$= 100 - 76.34$

$= 23.66$

c) 0.456 -

$n = 0$

$r = 10$

10's comp. of 0.456 $= 10^0 - 0.456$

$= 1 - 0.456$

$= 0.544$

Ex 2. Find 2's complement of the given binary no.

a) 11011

$n = 5$

$r = 2$

2's comp. $= 2^5 - 11011$

$= (32)_{10} - 11011$

$= 100000 - 11011 = 00101$

OR

-11011

1's comp. of 11011 $= 00100$

$+1$

2's comp. $= 00101$

OR →

11011

↓

00101

↑ change the 0 to 1 & 1 to 0 after 1st 1 of LSB.

b) 1101.01

$n = 4$

$r = 2$

2's comp. $= 2^4 - 1101.01$

$= (16)_{10} - 1101.01$

$= 10000 - 1101.01$

$= 0010.11$

10000.00

-1101.01

0010.11

Scanned with CamScanner

(r-1)'s complement – If N is positive number in r base with n integer digits and m fraction digits, the (r-1)'s complement is defined as –

$$[(r^n - 1) - N]$$

or $(r^n - r^{-m} - N)$

$$(r-1)\text{'s complement} = [(r^n - 1) - N] \text{ or } [r^n - r^{-m} - N]$$

Ex:- Find 9's complement for the given decimal no.

a) 7543

$n = 4$

$r = 10$

9's comp. $= (10^4 - 1) - 7543$

$= 9999 - 7543 = 2456$

b) 632.456

$n = 3$

$r = 10$

9's comp. $= (10^3 - 1) - 632.456$

$= 999 - 632.456$

$= 366.544$

c) 0.7863

9's comp. $= \cancel{(10^0 - 1)}$ $r^n - r^{-m} - N$

$m = 4$

$\Rightarrow 10^0 - 10^{-4} - 0.7863$

$= 1 - 0.0001 - 0.7863$

$= 0.2136$

Scanned with CamScanner

1's complement – 1's complement can be find by change all the 1's to 0's and all 0's to 1's.

Ex. 10110 ↓↓↓↓↓ 1's. 01001

101.1101 ↓↓↓ ↓↓↓↓ 010.0010

2's complement – Take 1's complement and add 1 to that.

Ex:- a) 101101

2's complement = 1's complement + 1

1's complement of 101101 = 010010

+ 1

2's complement of 101101 = 010011

b) 1011001

1's complement of 1011001 = 0100110

+ 1

2's comp. = 0100111

Scanned with CamScanner

Binary Addition –

$0+0=0$, $0+1=1$, $1+1=10$ i.e 0 with carry 1.
$1+0=1$

Ex-

$$\begin{array}{r} 1101.101 \\ +\ 111.011 \\ \hline 10101.000 \\ \hline \end{array}$$

Binary Subtraction –

$0-0=0$, $1-1=0$, $1-0=1$, $0-1=1$, with a borrow.

Ex.

$$\begin{array}{r} 1010.010 \\ -\ 111.111 \\ \hline 0010.011 \\ \hline \end{array}$$

Subtraction with r's complement –

Steps to perform the subtraction of two positive base 'r' number systems with M (minuend) and S (subtrahend)

1. Equating the no. of digits between the minuend and subtrahend if they are not equal padding with zero is used to make them equal.
2. Find the r's complement of subtrahend.
3. Add minuend with r's complement of S.
   i) If the sum produces a carry, discard it.
   ii) If the sum does not produce an end around carry, take the r's complement to the result of sum and place a negative sign in front.

Scanned with CamScanner

Ex: Using 10's complement,

a) 76532 − 4250

Step 1: No. of digits in M = 5

No. of " " S = 4

Padding 0's to S to make equal digits.

76532 − 04250

Step 2: 10's complement of 04250 $= 10^5 - 4250$

$= 95750$

Step 3:

$$\begin{array}{rl} M & = 76532 \\ \text{10's comp. of } S & = +95750 \\ \hline & \ 1\,72282 \end{array}$$

→ Ans.

discard

b) 4250 − 76532

⇒ 04250 − 76532

10's comp. of S $= 10^5 - 76532 = 23468$

$= 23468$

$$\begin{array}{rl} M & = 04250 \\ & + \\ \hline & \ 27718 \end{array}$$

No carry

10's comp. of 27718 $= 10^5 - 27718$

$= 72282$

→ put '−' sign in front = −72282 → Ans.

Scanned with CamScanner

Subtraction with (r-1)'s complement -

For subtraction of (M-S), both +ve numbers with in base r, the steps are -

1. Equating the no. of digits of M and S.
2. Find (r-1)'s complement of S.
3. Add → M + (r-1)'s comp. of S,

   i) If the addition produce a carry, add this carry to LSB of the sum.

   ii) If the above does not produce a carry, take the (r-1)'s complement to the result and place a negative sign in front.

Ex1. Perform the subtraction by using 9's complement.

a) 76532 − 4250

⇒ M = 76532

S = 04250

9's complement of S = $(10^5 - 1) - 04250$

= 95749

$$\begin{array}{rl} M = & 76532 \\ \text{9's comp. of } S = & 95749 \\ + & \\ \hline & 172281 \end{array}$$

← ~~Ans~~.

add carry ~~discard~~ +1 = 72282 Ans.

b) 4250 − 76532

⇒ M = 04250

S = 76532

9's comp. of S = $(10^5 - 1) - 76532 = 23467$

$$\begin{array}{rl} M = & 04250 \\ & +23467 \\ \hline & 27717 \end{array}$$

9's comp. of 27717 = $(10^5 - 1) - 27717 = 72282$

= $\boxed{-72282}$ Ans.

Scanned with CamScanner

Ex- Perform subtraction using 2's complement

a) 11101 − 1100

M S

⇒ 11101 − 01100

1's comp. of S = 10011

+ 1

2's comp. of S = 10100

M = 11101

2's comp of S = + 10100

11 10001 → Ans.

discard carry

b) 1100 − 11101

⇒ M = 01100

S = 11101

2's comp. of S (11101) = 00010 + 1 = 00011

= 00011

M = + 01100

01111

No carry 1's comp.

2's comp. of 01111 = 10000 + 1

= −10001 Ans.

Scanned with CamScanner

Perform the subtraction by using 1's complement.

11011 − 1010

⇒ 11011 − 01010

1's comp. of S = 10101

M = 11011

+10101

110000 ← ~~Ans~~.

+1

10001 Ans.

b) 1010 − 11011

⇒ M = 01010

1's comp of S = 00100

+ 01110

No carry

1's comp of 01110 = 10001

Add −ve sign = −10001 Ans.

Scanned with CamScanner

SIGNED BINARY NUMBER

- In decimal number system (+) sign – positive no.
  (−) sign – negative no.

- In binary number system –
  Signed bit is placed at MSB.
  0 → positive number
  1 → negative number

- The signed binary number is represented in three ways :-
  1. Sign magnitude representation
  2. 1's complement representation
  3. 2's complement representation

① Sign Magnitude Representation –

- In an n-bit number, the right most (n−1) bit holds the magnitude of the given no. and n^{th} bit is assigned for sign bit.

e.g:

$$+18 \Rightarrow \underbrace{0}_{\text{sign bit}}\ \underbrace{0\,0\,1\,0\,0\,1\,0}_{\text{magnitude bits}}$$

$$-18 \Rightarrow 1\,0\,0\,1\,0\,0\,1\,0$$

→ When arithmatic operations are implemented in a computer, it is more convinient to use signed complement system for representing negative numbers. In this negative no. is indicated by its complement.

→ 2's (1's) complement system for representing signed numbers :-

1. If the number is +ve, the magnitude is represented in its true binary form and a sign bit 0 is placed infront of MSB.

Scanned with CamScanner

If the no. is -ve, the magnitude is represented in its 2's (or 1's) complement form and a sign bit '1' is placed in front of the MSB.

+51 = | 0 | 1 1 0 0 1 1 |
Sign bit Magnitude

-51 = | 1 | 1 1 0 0 1 1 | – In Sign Magnitude form

-51 = | 1 | 0 0 1 1 0 1 | – In 2's complement form

-51 = | 1 | 0 0 1 1 0 0 | – In 1's complement form
Sign bit Magnitude

Ex – a) 01101 b) 010111 c) 10111. Determine the decimal value in i) Sign magnitude ii) 2's comp. iii) 1's comp.

Given Number	Signed Magnitude	2's Comp. form	1's Comp. form
01101	+13	+13	+13
010111	+23	+23	+23
10111	-7	-7	-7

Special case in 2's complement representation:–

– Whenever a signed number has a 1 in the sign bit and all 0s for the magnitude bits, the decimal equivalent is -2^n, where n is the no. of bits in the magnitude.

eg: → 1000 = -8 and -10000 = -16.

Characteristics of 2's complement numbers:– Properties of 2's comp.

1. There is one unique zero.
2. The 2's complement of 0 is 0.
3. The left most bit cannot be used to express a quantity.

Scanned with CamScanner

46	2327EC1185	SAJAL GUPTA	ASHOK KUMAR	ECE	X	P	P	P	P	P	P	4
47	2327EC1195	SAKSHI	[illegible] KUMAR	ECE	[illegible]	A	P	P	P	P	P	[illegible]

It is a sign bit. If it is '1', the number is -ve, if it is '0', the number is +ve.

4. For an n-bit word which includes the sign bit, there are $(2^{n-1}-1)$ positive integers, 2^{n-1} negative integers and one '0', for a total of 2^n unique states.

5. Significant information is contained in the 1s of the +ve numbers and 0s of the -ve numbers.

6. A -ve number may be converted into a +ve number by finding its 2's complement.

BCD — Weighted Binary codes. (Binary Coded Decimal).

Decimal Digit	8421
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Six illegal combinations: 1010, 1011, 1100, 1101, 1110 and 1111. These are not present in 8421 (BCD) code.

Non weighted codes — i) Excess-3 code
ii) Gray code

Scanned with CamScanner

BCD Arithmetics

* The rules of binary addition and subtraction do not apply to the entire 8421 number but only to the individual 4-bit groups.
* The BCD addition is performed by individually adding the corresponding digits to the decimal numbers expressed in 4-bit binary groups starting from LSB. If there is a carry out of one group to the next group, or if the result is an illegal code, the 6_{10} (0110) is added to the sum of that group and resulting carry is added to the next group.

Ex. a)

$$\begin{array}{r} 25 \\ +\ 13 \\ \hline 38 \end{array} \Rightarrow \begin{array}{r} 0010\ \ 0101 \\ +\ 0001\ \ 0011 \\ \hline 0011\ \ 1000 \end{array}$$

no carry. Ans.

b)

$$\begin{array}{r} 679.6 \\ +\ 536.8 \\ \hline 1216.4 \end{array} \Rightarrow \begin{array}{r} 0110\ \ 0111\ \ 1001\ .\ 0110 \\ +\ 0101\ \ 0011\ \ 0110\ .\ 1000 \\ \hline 1011\ \ 1010\ \ 1111\ .\ 1110 \end{array}$$

← illegal codes

$$\begin{array}{r} +\ 0110 \\ \hline 10001 \end{array} \quad \begin{array}{r} +\ 0110 \\ \hline 10000 \end{array} \quad \begin{array}{r} +\ 0110 \\ \hline 10101 \end{array} \quad \begin{array}{r} +\ 0110 \\ \hline 10100 \end{array}$$

$$\begin{array}{r} +1 \\ \hline 0001 \end{array} \quad \begin{array}{r} +1 \\ \hline 0010 \end{array} \quad \begin{array}{r} +1 \\ \hline 0001 \end{array} \quad 0110$$

Result = 0001 0010 0001 0110 . 0100

Scanned with CamScanner

~~Boolean Algebra and Logic Gates~~

Ex: Add the BCD numbers

a) 1001 + 0100 $\qquad$ 9 + 4 = 13

$$\Rightarrow \begin{array}{r} 1001 \\ +\,0100 \\ \hline 1101 \\ +\,0110 \\ \hline 0001\ 0011 \end{array}$$

1101 Invalid BCD no. (>9)

+ 0110 Add 6

0001 0011 Valid BCD

1 3

b)

$$\begin{array}{rr} 0110 & 0111 \\ +\,0101 & 0011 \\ \hline 1011 & 1010 \\ +\,0110 & +\,0110 \\ \hline 1\,0001 & 1\,0000 \\ & +\,1 \\ \hline 0001 & 0010\ \ 0000 \end{array}$$

(>9) i.e invalid BCD

Add 6

(1 2 0)

Scanned with CamScanner

BCD Subtraction

- The BCD subtraction is performed by subtracting the digits of each 4-bit group of the subtrahend from the corresponding 4-bit group of the minuend in binary starting from the LSB. If there is a borrow from the next group, then 6_{10} (0110) is subtracted from the difference term of this group. Subtraction is performed by the complement method using 9's or 10's complement.

Ex a) 38 BCD 0011 1000

$$\begin{array}{r} -15 \Rightarrow \\ \hline 23 \text{ form} \end{array} \quad \begin{array}{r} -0001\ 0101 \\ \hline 0010\ 0011 \end{array}$$

b)

$$\begin{array}{r} 206.7 \\ -147.8 \\ \hline \end{array} \Rightarrow \begin{array}{r} 0010\ 0000\ 0110\ .\ 0111 \\ -0001\ 0100\ 0111\ .\ 1000 \\ \hline 0000\ 1011\ 1110\ .\ 1111 \\ -0110\ \ -0110\ \ -0110 \\ \hline \end{array}$$

Ans. (0000 0101 1000 . 1001)

Ex: Perform BCD subtraction by 9's complement.

a)

$$\begin{array}{r} 85 \\ -24 \\ \hline 61 \end{array} \Rightarrow$$

9's complement of 24 = $\begin{array}{r} 99 \\ -24 \\ \hline 75 \end{array}$

$$\begin{array}{r} 85 \\ +75 \\ \hline (1)60 \\ \rightarrow +1 \\ \hline 61 \end{array} \Rightarrow \begin{array}{r} 1000\ 0101 \\ 0111\ 0101 \\ \hline 1111\ 1010 \\ +0110\ +0110 \\ \hline 1001\ 1010\ 0000 \end{array}$$

← invalid BCD

⇒ (0110 0001) Ans.

Scanned with CamScanner

Q b)

$$\begin{array}{r} 305.5 \\ -\ 168.8 \end{array} \rightarrow \text{9's comp} \rightarrow \begin{array}{r} 305.5 \\ 831.1 \\ \hline 1\ 136.6 \\ +1 \\ \hline 136.7 \end{array}$$

$$\Rightarrow \begin{array}{r} 0011\ 0000\ 0101.\ 0101 \\ +\ 1000\ 0011\ 0001.\ 0001 \\ \hline 1011\ 0011\ 0110.\ 0110 \\ +\ 0110 \\ \hline 1\ 0001 \\ +1 \\ \hline 0001\ 0011\ 0110.\ 0111 \end{array}$$

Ans.

c)

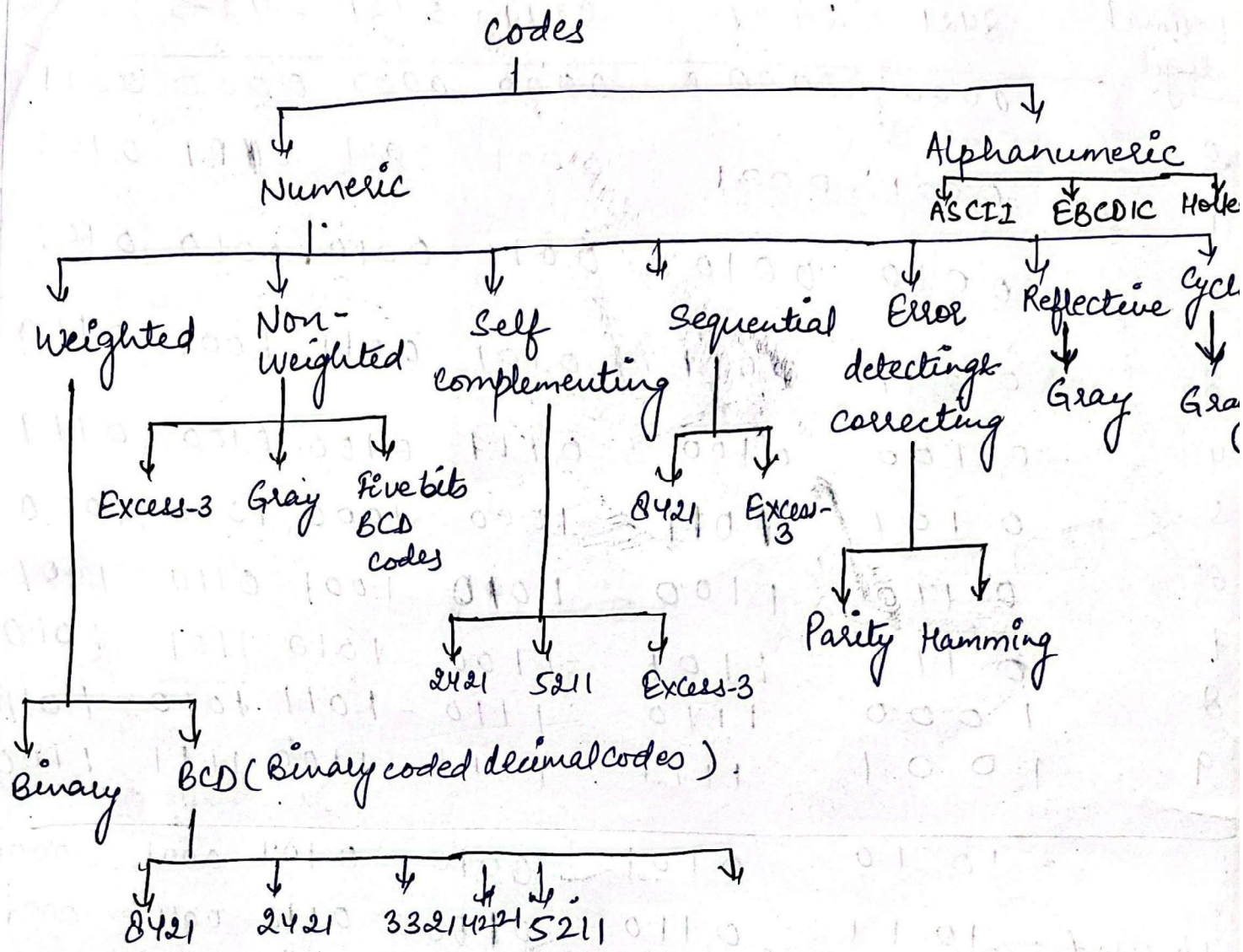
$$\begin{array}{r} 679.6 \\ -\ 885.9 \end{array} \Rightarrow \text{9's comp.} \begin{array}{r} 679.6 \\ 114.0 \\ \hline 793.6 \end{array}$$

no carry. so result is -ve
Take 9's comp. of sum and place a -ve sign infront.

$$\Rightarrow \begin{array}{r} 0110\ 0111\ 1001.\ 0110 \\ +\ 0001\ 0001\ 0100.\ 0000 \end{array}$$

Scanned with CamScanner

BINARY CODES



→ The codes 8421, 2421, 5211, 3321 and 4311 are positively weighted codes.

→ 642-3, 631-1, 84-2-1 and 74-2-1 are negatively weighted codes.

Scanned with CamScanner

Excess-3 code:- It is a non-weighted BCD code. This code derives its name from the fact that each binary code word is the corresponding 8421 code plus 0011 (3). It is a sequential code and can be used for arithmetic operations. The XS-3 code has six invalid states 0000, 0001, 0010, 1101, 1110 and 1111.

→ It is self-complementing code.

→ ~~Binary~~ XS-3 of binary code not possible.

Ex:-

$$\begin{array}{r} 99 \\ +(33) \\ \hline 12\ 12 \end{array}$$

← add ~~bit~~ 3 digit-wise.

(1100 1100) XS-3

Scanned with CamScanner

GRAY CODE (REFLECTIVE-CODE) / CYCLIC CODE

It is a non-weighted code and is not suitable for arithmetic operations.

* It is a cyclic code b'cox successive code words in this code differ in one bit position only i.e it is a unit distance code.

- It is also a reflective code.
- The n LSBs for 2^n through $2^{n+1}-1$ are the mirror images of those for 0 through 2^n-1.
- An N-bit Gray code can be obtained by reflecting an N-1 bit code about an axis at the end of the code, and putting the MSB of '0' above the axis and the MSB of '1' below the axis.

* Binary number is converted to Gray code to reduce switching operations.

- These are used in digital systems for error corrections. Also called as minimum error code.

→ It is self complementing.

Scanned with CamScanner

Reflection of Gray codes

Gray Code				Decimal	4-bit binary
1-bit	2-bit	3-bit	4-bit		
0	00	000	0000	0	0000
1	01	001	0001	1	0001
	11	011	0011	2	0010
	10	010	0010	3	0011
		110	0110	4	0100
		111	0111	5	0101
		101	0101	6	0110
		100	0100	7	0111
			1100	8	1000
			1101	9	1001
			1111	10	1010
			1110	11	1011
			1010	12	1100
			1011	13	1101
			1001	14	1110
			1000	15	1111

Self-reflective

Scanned with CamScanner

Binary-to-Gray Conversion

n-bit binary number $\Rightarrow B_n B_{n-1} \cdots B_1$

Its Gray code equivalent $\Rightarrow G_n G_{n-1} \cdots G_1$

$\uparrow$ MSB

Gray Code =	$G_n = B_n$	$G_{n-1} = B_n \oplus B_{n-1}$	$G_{n-2} = B_{n-1} \oplus B_{n-2}$	$\cdots$

$\uparrow$ X-OR

$\cdots G_1 = B_2 \oplus B_1$

Ex:- Convert the binary 1001 to the Gray code.

Binary 1 $\oplus \rightarrow$ 0 $\oplus \rightarrow$ 0 $\oplus \rightarrow$ 1

Gray 1 1 0 1

X-OR	
00	0
01	1
10	1
11	0

Ex:- Binary 1 1 0 1 ($\oplus$ $\oplus$ $\oplus$)

Gray 1 0 1 1

Ex:- Binary 1 0 1 0

Gray 1 1 1 1

Ex: $(3A7)_{16} \rightarrow$ Gray number?

$(3A7)_{16} \Rightarrow 0011\,1010\,0111 \Rightarrow 1001\,1110\,100$

skip

$(527)_8 \Rightarrow 101,010,111 \Rightarrow 111111.00$ Gray

Scanned with CamScanner

Gray to - Binary -

Gray ⇒ $G_n \; G_{n-1} \; \ldots\ldots \; G_1$

Binary ⇒ $B_n \; B_{n-1} \; \ldots\ldots \; B_1$

Binary code	$B_n = G_n$	$B_{n-1} = B_n \oplus G_{n-1}$	$B_{n-2} = B_{n-1} \oplus G_{n-2}$ --	$B_1 = B_2 \oplus G_1$

Ex:- Gray 1 1 0 1

Binary 1 → 0 → 0 → 1 = 1001

Ex: Gray 0 1 0 1

Binary 0 1 1 0

Ex. 10110010

Binary $(11011100)_2$

$= (DC)_{16}$

$= (334)_8$

$= (220)_{10}$

Scanned with CamScanner

Fixed Vs. Floating Numbers -

Fixed representation has fixed number of bits for integer part and for fractional part.

- Floating point refers to the fact that a number's radix point (decimal point or binary point) can "float" i.e it can be placed anywhere relative to the significant digits of the number.

Ex:- 10.0

.15 ← fixed points

Shifting decimal pt.

$2^1 \leftarrow \cdot \rightarrow 2^{-1}$

101.1 → 5.5

↑ if this pt. shifted to right

1011 → 11

⇒ 1011×2^{-1} ← binary base

→ Floating pt. helps to represent the number in any range (not fixed).

Scanned with CamScanner

Floating Point Numbers

- In decimal system, very large and very small numbers are expressed in scientific notation by stating a number (mantissa) and an exponent of 10.

- 16-bit word consists of two parts

S	Exponent	Mantissa

10-bit Mantissa

↑

Mantissa is in 2's complement form.

6-bit Exponent

↓

$(0 \rightarrow 63)$

→ MSB is a sign bit.

→ To express negative exponents, the number 32_{10} $(100000)_2$ has been added to the desired exponent. The actual exponent of the number is equal to the exponent given by the 6 bits minus 32.

Ex:- $(5.625)_{10} \longrightarrow (101.101)_2$

$(5.625)_{10} \longrightarrow 0.\underbrace{5625}_{\text{Mantissa}} \times 10^{(1)}$ Exponent

$(101.101)_2 \longrightarrow 0.\underbrace{101101}_{\text{Mantissa}} \times 2^{(3)}$ Exponent.

move radix pt. to left of MSB (moved 3 bits)

or $(10\underset{\curvearrowleft}{1.}101)_2 \longrightarrow 10.1101 \times 2^{1}$

or $(101.\underset{\curvearrowright}{101})_2 \longrightarrow 101101 \times 2^{-3}$

Scanned with CamScanner

the Radix point can be moved by different ways, so to we need Normalization.

Need of Normalization -

$(101.101)_2 \rightarrow 0.101101 \times 2^3$

$(101.101)_2 \rightarrow 1.01101 \times 2^2$

$(101.101)_2 \rightarrow 101101 \times 2^{-3}$

→ B'coz of so many ways of representation, it leads to confusion. So, we need normalization.

Normalization -

Two types :

1. Explicit Normalization : Move the radix point to the LHS of most significant '1' in the bit sequence.

$(101.101)_2 \rightarrow 0.101101 \times 2^3$

2. Implicit Normalization : Move the radix pt. to the RHS of the most significant '1' in the bit sequence.

$(101.101)_2 \rightarrow 1.01101 \times 2^2$

→ Implicit normalization is the better one in the case of precision.

Scanned with CamScanner

Biasing - This technique is used b'coz the comparator cannot handle +ve and -ve numbers together.

S	Exponent	Mantissa

↑ some bits for exponent

↑ some dedicated to mantissa

→ S → sign bit

→ Exponent can also be +ve or -ve. e.g 2^3 or 2^{-3}.

→ If we use 2's complement representation for exponent then it will not work b'coz in 2's complement sign representation ~~num~~ range starts from 0 → 7 and -8 → -1. Which ~~is~~ do not represent the values sequentially. Due to this we go for Biasing. We don't use 2's comp. representation for exponent.

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

Scanned with CamScanner

range -8 to 7, cannot be identified by the comparators. As they only work with unsigned numbers (cannot identify both).

- We need to represent this range into unsigned numbers. For that, identify the max^m number in the range and add it to all the numbers.

-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$	$(+8)$
$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

range is now converted into an unsigned form.
Here the exponent is called excess-8.

Ex:- $(0.0101)_2 \rightarrow 0.101 \times 2^{-1}$ (Explicit norm.)

S	Exponent				Mantissa				
0	0	1	1	1	1	0	1	0	0
1 bit	4 bits				5 bits				

in 10-bits.

$-1 \rightarrow 7 = 0111$

→ This is how (0.101×2^{-1}) is stored in memory using explicit normalization method.

$(\cancel{0.0101}\ 101.101)_2 \Rightarrow \cancel{001.01}\ 1.01101 \times 2^{+2}$ (Implicit norm. form)

→ +ve number

S									
0	1	0	1	0	0	1	1	0	1

← Implicit method)

Scanned with CamScanner

Formula to retrieve the decimal number –

1. Explicit Normalization :

$$= \boxed{(-1)^S \times 0.M \times 2^{E-Bias}}$$

S → sign bit
M → Mantissa
E → Exponent saved in computer memory.

2. Implicit Normalization :

$$\boxed{= (-1)^S \times 1.M \times 2^{E-Bias}}$$

– Using these two formula, we can get back the data saved in computer memory into human readable form.

Scanned with CamScanner

Explicit vs Implicit Normalization of floating pt numbers:-

Ex:- $(5.625)_{10} \rightarrow (101.101)_2$

If suppose we have to represent a no. into 10-bits then the format will be:-

S	E (4 bits	M (5 bits)

← 10 bits →

* $(101.101)_2 \rightarrow 0.\underbrace{101101}_{M = 6 \text{ bits}} \times 2^3$ (Explicit Norm.)

$S = 0$

$E = 3 + 8 = 11$ (adding Excess-8 biasing)

$= (1011)_2$

S	E	M
0	1011	10110

fixed memory storage.

→ M = 6 bits but the format is having 5 bits for Mantissa, so the machine will omit the last bit i.e 1 from 101101.

* $(101.101)_2 \rightarrow 1.01101 \times 2^2$ (Implicit Norm.)

$S = 0$

$E = 2 + 8 = 10 = (1010)_2$

$M = 01101$ (5 bits)

S	E	M
0	1010	01101

→ Here the all bits are saved in 10 bit format without loosing any bit.

Scanned with CamScanner

0	1011	10110

Explicit form

$\Downarrow$

converting to human readable form by formula.

$= (-1)^S \times 0.M \times 2^{E-Bias}$

$= (-1)^0 \times 0.10110 \times 2^{11-8}$

$= (101.1)_2 \rightarrow (5.5)_{10}$

= incorrect

0	1010	0110

Implicit form

$\Downarrow$

$= (-1)^S \times 1.M \times 2^{E-Bias}$

$= (-1)^0 \times 1.01101 \times 2^{10-8}$

$= 1 \times 1.01101 \times 2^2$

$= (101.101)_2$

$= (5.625)_{10}$ correct

This is the reason, Implicit Normalization is more precise than Explicit Normalization.

Scanned with CamScanner

IEEE Standard for Floating-Point Arithmetic (IEEE 754)

IEEE → Institute of Electrical and Electronics Engineers.

IEEE 754 → It is a standard for representing floating point numbers.

- It is a technical standard for floating point arithmetic established in 1985 by IEEE.

Name	Common Name	Significand bits	Exponent bits	Exponent Bias	Organization of Bits
binary 16	Half precision	11	5	15	S E(5 bits) M(10 bits)
binary 32	Single precision	24	8	127	S E(8) M(23 bits)
binary 64	Double precision	53	11	1023	S E(11 bits) M(52 bits)
binary 128	Quadruple precision	113	15	16383	S E(15) bits M(112 bits)
binary 256	Octuple precision	237	19	262143	S E(19 bits) M(236 bits)

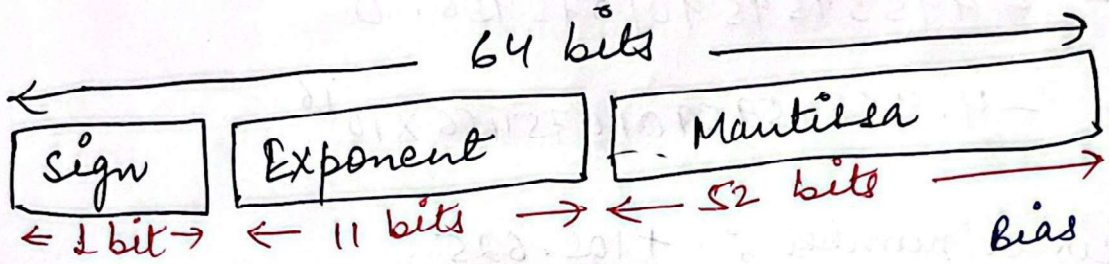
→ Single and Double precision are commonly used.

Scanned with CamScanner

EEE-754 has 3 basic components:

1. Sign of Mantissa
2. The Biased Exponent
3. The Normalised Mantissa

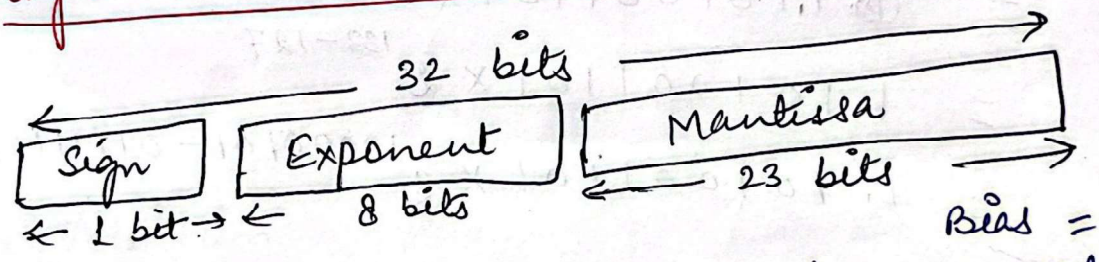
Double Precision IEEE-754 –



Bias = 1023

Single Precision IEEE-754 –

E = 8 bit
$2^8 = 256$



Bias = 127

Bias :-
2^{1-127} to $2^{254-127} = 2^{127}$
$\Rightarrow 2^{-126}$ to 2^{127}

Special Values:- IEEE has reserved some values that can ambiguity.

Exponent	Mantissa	Value
0	0	exact 0
255	0	infinity
0	not 0	denormalised
255	not 0	Not a number (NAN)

Scanned with CamScanner

Ex1- Consider the following 32-bit pattern

1 10110110 011 0000 0000 0000 0000 0000

The value is → $(-1)^S \times 1.M \times 2^{E-127}$

$(-1)^1 \times 2^{10110110 - 01111111} \times 1.011$

$= -1.375 \times 2^{55}$

$= -49539595901075456.0$

$= -4.95395959010754 56 \times 10^{16}$

Ex: Decimal number : +105.625

binary equivalent $= (1101001.101)_2$

$= 1.101001101 \times 2^6$

$= 1.101001101 \times 2^{133-127}$

$= 1.101001101 \times 2^{10000101 - 01111111}$

In IEEE 754 format :

0	10000101	10100110100 0000 00000000

Ex:-

$(-16.25)_{10} \rightarrow (10000.001)_2$

$= 1.0000001 \times 2^4$

$S = 1$

$E = 4+8 = 12 = 1100$

1	1100	0000000	1

$(-1)^S \times (1.M)^{E-Bias}$ $\quad E = 11 - 8 = 3$

$= 1.0000001 \times 2^3$

$= -(1.0000.001)$

$= -16.25$

Ex:- Addition of floating pt numbers:-

$F1 = 270.75$

$F2 = 2.375$

$F1 = (270.75)_{10} = (100001110.11)_2 = 1.000011101 1 \times 2^8$

$F2 = (2.375)_{10} = (10.011)_2 = 1.0011 \times 2^1$

Now, shift the mantissa of F2 right by 8-1 = 7 positions and add.

```
  1.0000 1110110 0000 0000 0000
+        10011 0000 0000 0000
-----------------------------
  1000 10001001 00000000 0000
```

Result $= 1.00010001001 \times 2^8$

Scanned with CamScanner

Ex:- <u>Subtraction</u> -

$F1 = 270.75$

$F2 = 224$

$F1 = (270.75)_{10} = (100001110.11)_2 = 1.000011101 1 \times 2^8$

$F2 = (224)_{10} = (11100000)_2 = 1.110 \times 2^7$

shift mantissa of F2 right by 8-7 = 1 position.

```
  1 000 0111 0110 0000 0000 0000
  - 111 0000 0000 0000 0000 0000
  ------------------------------
  0 001.0111 0110 0000 0000 0000
```

$= 1.01110110 \times 2^5$

Scanned with CamScanner

BOOLEAN ALGEBRA – Boolean algebra is a system of mathematical logic. Any complex logic statement can be expressed by a boolean function.

– Digital systems are designed based on the two digits 0 and 1.

→ No fractions, negative or cube numbers, whereas in binary all values possible.

→ Boolean Addition:

$0+0=0$
$0+1=1$
$1+0=1$
$1+1=1$

Binary Addition:

$0+0=0$
$0+1=1$
$1+0=1$
$1+1=10$

Logic operations –

- AND
- OR
- NOT
- NAND
- NOR

AXIOMS –

$0.0=0$ $\quad 0+0=0$
$0.1=0$ $\quad 0+1=1$
$1.0=0$ $\quad 1+0=1$
$1.1=1$ $\quad 1+1=1$

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

If $A=0$, Then $\bar{A}=1$

If $A=1$, then $\bar{A}=0$

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

Scanned with CamScanner

AND laws

$A.0 = 0$

$A.1 = A$

$A.A = A$

$A.\bar{A} = 0$

OR Laws

$A+0 = A$

$A+1 = 1$

$A+A = A$

$A+\bar{A} = 1$

NOT Laws

If $A = 0, \bar{A} = 1$

If $A = 1, \bar{A} = 0$

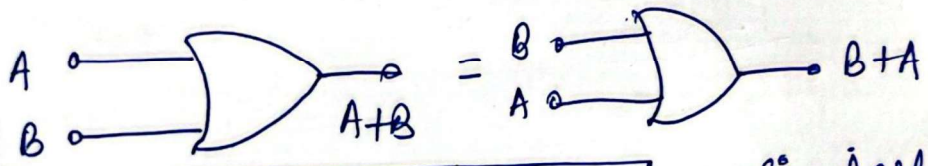
$\bar{0} = 1, \bar{1} = 0$

laws of Boolean Algebra :-

1.) Commutative laws :-

i) **law 1 (for Addition) :-** This law states that A OR B is the same as B OR A i.e order in which the variables are ORed (Addition) makes no difference in the output.

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



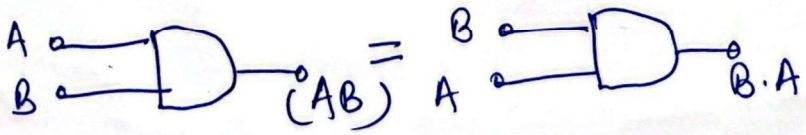
A	B	A+B	B+A
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Similarly,

$A+B+C = B+C+A$

ii) **law 2 (for multiplication) :-**

$$\boxed{A.B = B.A}$$



A	B	A.B	B.A
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

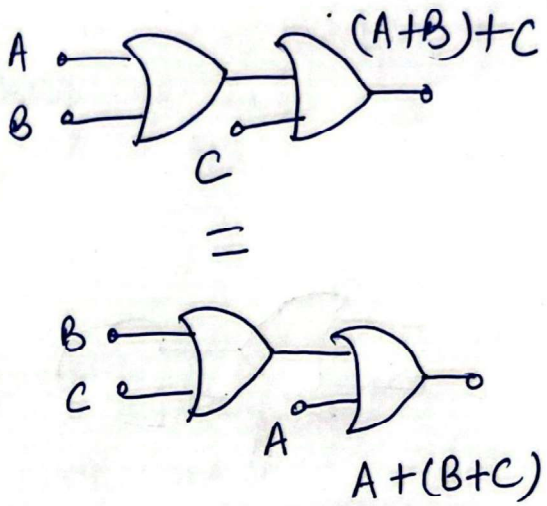
Scanned with CamScanner

Associative Laws :-

i) Law 1 : For addition :-

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

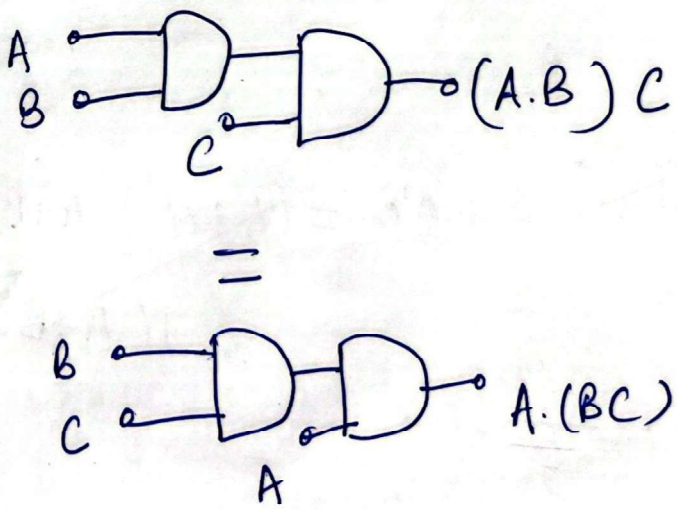
A	B	C	A+(B+C)	(A+B)+C
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1



ii) Law 2 : For Multiplication :-

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

A	B	C	A.(BC)	(AB)C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

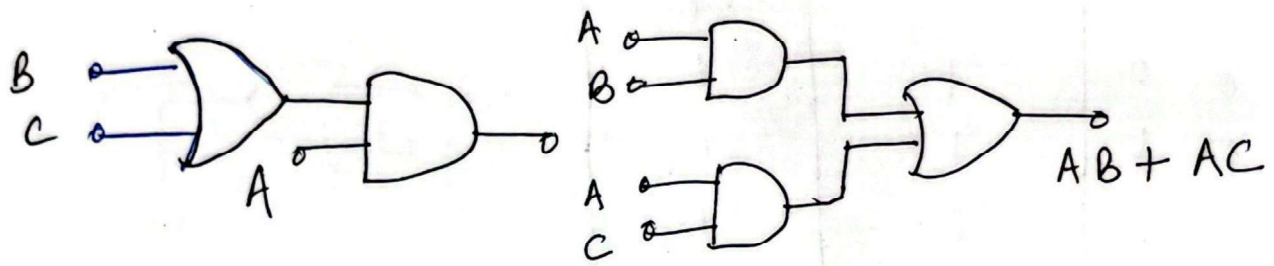


Scanned with CamScanner

(3.) Distributive laws :-

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

This law states that ORing of several variables and ANDing the result with a single variable is equivalent to ANDing that single variable with each of several variables and then ORing the product.



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

Proof: $RHS = (A+B)(A+C)$

$= AA + AC + AB + BC$

$= A + AC + AB + BC$

$= A(1 + C + B) + BC$

$= A(1) + BC = A + BC.$

iii) ~~$A + A'B = (A+A')(A+B)$~~

~~$= 1(A+B) = (A+B)$~~

~~Proof~~ -

Scanned with CamScanner

iii) a) $A + AB = A$ (Absorption law)

Proof:- $A(1+B) = A$

$A(1) = A$

$A = A$.

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

Proof: $A \cdot A + A \cdot B$

$A + AB = A$.

iv) a) $A + \bar{A}B = A+B$ (Redundant Literal Rule) RLR.

Proof:- as $A+AB = A \therefore A = A+AB$

L.H.S:-

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

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

$= A+B(1)$

$= A+B$. R.H.S.

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

Proof: $A\bar{A} + AB$

$1 + AB$

$= AB$.

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

~~$= AA + AC + A$~~

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

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

Duality Theorem:- According to duality theorem, a boolean relation can be written to form another boolean relation by -

1. Changing each OR sign to an AND sign.
2. Changing each AND sign to an OR sign.
3. Complementing any 0 or 1 appearing in the expression.

e.g.: Distributive law:- $A.(B+C) = A.B + B.C$

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

Scanned with CamScanner

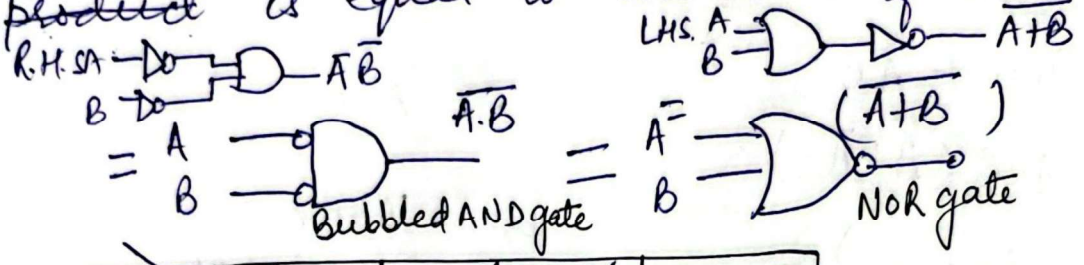
Imp.

DeMorgan's Theorem :- Demorgan's theorem represents two of the most powerful laws in Boolean algebra.

Law 1 (Theorem 1) :-

$$\cancel{\overline{A \cdot B} = \bar{A} + \bar{B}} \quad \boxed{\overline{A+B} = \bar{A} \cdot \bar{B}}$$

This law states that the complement of a ~~sum~~ sum ~~product~~ is equal to the ~~sum~~ product of ~~their~~ individual complements.



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

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

=

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

→ It shows that the NOR gate is equivalent to a bubbled AND gate.

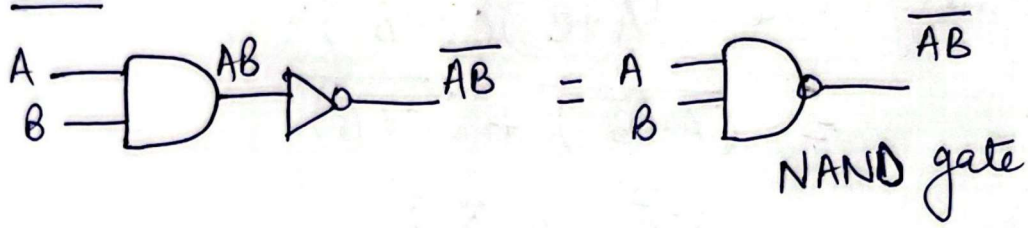
Scanned with CamScanner

Law 2 (Theorem 2) :-

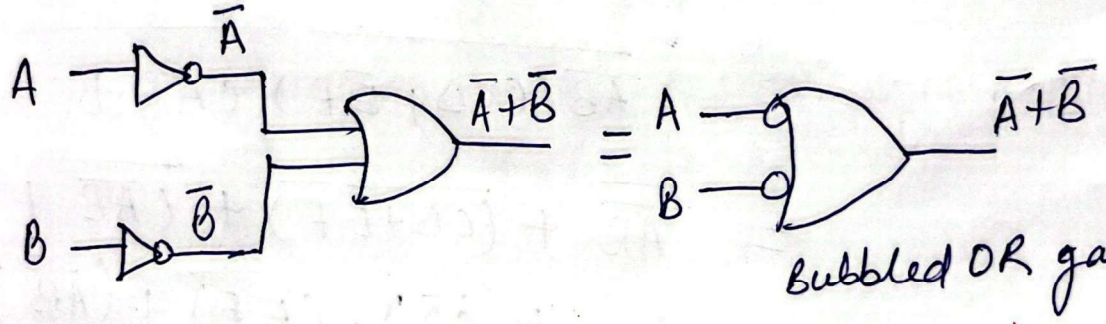
$$\boxed{\overline{AB} = \bar{A} + \bar{B}}$$

This law states that complement of the product of variables is equal to the sum of their individual complements.

LHS



RHS



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

=

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

⇒ It shows that the NAND gate is equivalent to a bubbled OR gate.

$$\boxed{\begin{aligned} \overline{A+B} &= \bar{A}.\bar{B} \\ \overline{AB} &= \bar{A}+\bar{B} \end{aligned}}$$

Scanned with CamScanner

Process of Demorganization → {Break the line / Change the sign}

1. Complement the entire given function.
2. Change all the ANDs to ORs and all the ORs to ANDs.
3. Complement each of the individual variables.
4. Change all 0's to 1's and 1's to 0's.

Ex. Demorganize $\overline{(A+\bar{B})(C+\bar{D})}$

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

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

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

Ex: Demorganize $\overline{\overline{AB}\,(CD+\bar{E}F)\,(\overline{AB}+\overline{CD})}$

$$= \overline{\overline{AB}} + \overline{(CD+\bar{E}F)} + \overline{(\overline{AB}+\overline{CD})}$$

$$= AB + (\overline{CD}).(\overline{\bar{E}F}) + (\overline{\overline{AB}}).(\overline{\overline{CD}})$$

$$= AB + (\bar{C}+\bar{D}).(\bar{\bar{E}}+\bar{F}) + \cancel{(\bar{A}+\bar{B})}\,AB.\cancel{(\bar{C}+\bar{D})}\,CD$$

$$= AB + (\bar{C}+\bar{D})(E+\bar{F}) + AB.CD.$$

$$= (AB + \bar{C}E + \bar{C}\bar{F} + \bar{D}E + \bar{D}\bar{F} + ABCD).$$

Scanned with CamScanner

Ex: Reduce the expression $\overline{\bar{A}B + \bar{A} + AB}$

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

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

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

$$= 0 + 0 = 0 \qquad A\bar{A} = 0$$

Ex: Show that $AB + A\bar{B}C + B\bar{C} = AC + B\bar{C}$

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

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

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

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

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

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

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

$$= AC(B+1) + B\bar{C}(A+1) \qquad \because B+1 = 1, \; A+1 = 1$$

$$= AC + B\bar{C} \quad R.H.S.$$

Ex: Prove that $(A+\bar{B}+AB)(A+\bar{B})(\bar{A}B) = 0$

$$L.H.S \quad (A+\bar{B}+AB)(A+\bar{B})(\bar{A}B) = 0$$

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

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

$$= (A+\bar{B}) \cdot 1 \cdot \bar{A}B \qquad \because (A+\bar{B}) \cdot (A+\bar{B}) = (A+\bar{B}), \; \because A+1 = 1$$

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

$$= 0$$

Scanned with CamScanner

Ex: Prove that $\overline{A \oplus B} = \bar{A} \oplus B = A \oplus \bar{B}$

L.H.S $= \overline{A \oplus B} = \overline{(\bar{A}B + A\bar{B})}$

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

R.H.S $= \bar{A} \oplus B = \bar{\bar{A}}.B + \bar{A}\bar{B}$

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

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

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

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

$= (\overline{A \oplus B})$

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

using distribution law :-

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

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

$= (1+B)(1+A)(A+B+C) = 1.1.(A+B+C)$

$= (A+B+C)$

Scanned with CamScanner

Ex:- Apply Demorgan's theorem to the expression

$$F = \overline{\overline{AB}\,(CD+\overline{E}F)(\overline{AB}+\overline{CD})}$$

$$= \overline{\overline{AB}} + \overline{(CD+\overline{E}F)} + \overline{(\overline{AB}+\overline{CD})}$$

→ Break the line and change the sign.

$$= AB + (\overline{CD}\cdot\overline{\overline{E}F}) + (\overline{\overline{AB}}\cdot\overline{\overline{CD}})$$

$$= AB + (\overline{CD}\cdot\overline{E}F) + (AB\cdot CD)$$

$$= AB + (\overline{C}+\overline{D})(E+\overline{F}) + ABCD$$

Ex:-

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

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

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

$$= 0$$

OR

Break the lower line b/w AB and change the sign b/w them and simply.

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

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

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

$$= AB\cdot\overline{AB}$$

$$= 0.$$

Scanned with CamScanner

Ex:. Reduce the expression $xyz + yz + y'z + xy'$

$$F = xyz + yz + y'z + xy'$$
$$= xyz + z(y+y') + xy'$$
$$= xyz + z + xy'$$
$$= z(xy+1) + xy' \qquad \because xy+1 = 1$$
$$= z + xy'$$
$$= (z+x)(z+y') \qquad \because z \cdot z = z$$
$$= xy' + xz + z \cdot z + zy'$$
$$= xy' + z(x+1+y')$$
$$= xy' + z(1) = xy' + z.$$

Ex: Show that $AB'C' + A'BC + ABC = AB + AC$

LHS $F = AB'C' + A'BC + ABC$

$$= \cancel{AB'C' + BC(A'+A)} \qquad \because A'+A = 1$$
$$= \cancel{AB'C' + BC}$$
$$= \cancel{AB'C'} + \cancel{BC}CA$$

R.H.S:

$$\cancel{AB + AC}$$
$$\cancel{= AB(C+C') + AC(B+B')}$$
$$\cancel{= ABC + ABC' + ABC + ACB'}$$
$$\cancel{= ABC + ABC' + AB'C}$$

$$F = AB'C' + A'BC + ABC$$
$$= A(B'C' + BC) + A'BC \qquad \because A + \bar{A}B = A + B$$
$$= A(1) + A'BC$$
$$= A + BC$$
$$= A \cdot (B+C) \qquad \text{by Duality theorem}$$
$$= AB + AC.$$

Scanned with CamScanner

DUALS

Expression	Dual
1. $\bar{0} = 1$	$\bar{1} = 0$
2. $0 \cdot 1 = 0$	$1+0 = 1$
3. $0 \cdot 0 = 0$	$1+1 = 1$
4. $1 \cdot 1 = 1$	$0+0 = 0$
5. $A \cdot 0 = 0$	$A+1 = 1$
6. $A \cdot 1 = A$	$A+0 = A$
7. $A \cdot A = A$	$A+A = A$
8. $A \cdot \bar{A} = 0$	$A+\bar{A} = 1$
9. $A \cdot B = B \cdot A$	$A+B = B+A$
10. $A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A+(B+C) = (A+B)+C$
11. $A \cdot (B+C) = AB + AC$	$A+(BC) = (A+B)(A+C)$
12. $A(A+B) = A$	$A+AB = A$
13. $A \cdot (A \cdot B) = A \cdot B$	$A+A+B = A+B$
14. $\overline{AB} = \bar{A} + \bar{B}$	$\overline{A+B} = \bar{A}\bar{B}$
15. $(A+B)(\bar{A}+C)(B+C) = (A+B)(\bar{A}+C)$	$AB + \bar{A}C + BC = AB + \bar{A}C$
16. $(A+C)(\bar{A}+B) = AB + \bar{A}C$	$AC + \bar{A}B = \cancel{(A+C)(\bar{A}+B)}$ $(A+B)(\bar{A}+C)$
17. $A + \bar{B}C = (A+\bar{B})(A+C)$	$A(\bar{B}+C) = A\bar{B} + AC$
18. $(A+B)(C+D) = AC + AD + BC + BD$	$(AB+CD) = (A+C)(A+D)$ $(B+C)(B+D)$
19. $A+B = AB + \bar{A}B + A\bar{B}$	$AB = (A+B)(\bar{A}+B)(A+\bar{B})$
20. $A + B\overline{(C+DE)} = A + B\bar{C}DE$	$A[B + \overline{(C \cdot D+E)}] = A(B+\bar{C}+D+E)$
21. $\overline{\overline{AB} + \bar{A} + AB} = 0$	$\overline{\overline{A+B} \cdot \bar{A} \cdot (A+B)} = 1$
22. $AB + \overline{AC} + A\bar{B}C(AB+C) = 1$	$(A+B)(\overline{A+C}) \cdot [(A+\bar{B}+C) + (A+B) \cdot C] = 0$
23. $ABD + ABCD = ABD$	$(A+B+D) \cdot (A+B+C+D) = (A+B+D)$
24. $\overline{A\bar{B} + ABC} + A(B+A\bar{B}) = 0$	$\overline{(A+\bar{B}) \cdot (A+B+C)} \cdot (A + [B(A+\bar{B})]) = 1$
25. $A + \bar{B}C(A+\bar{B}C) = A + \bar{B}C$	$A \cdot [(\bar{B}+C) + A \cdot (\bar{B}+C)] = A \cdot (\bar{B}+C)$

Scanned with CamScanner

Ex: Reduce the expression

$$F = A + B[AC + (B+\bar{C})D]$$

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

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

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

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

$$= A.1 + BD.1$$

$$= A + BD.$$

Ex. $F = \underbrace{\overline{(A+\bar{B}C)}}_{\text{demorganize}} (A\bar{B} + ABC)$

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

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

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

$$= 0 + 0 = 0$$

Ex: Show that $AB + A\bar{B}C + B\bar{C} = AC + B\bar{C}$

L.H.S: $AB + A\bar{B}C + B\bar{C}$

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

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

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

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

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

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

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

$$= AC + B\bar{C} \quad \text{R.H.S.}$$

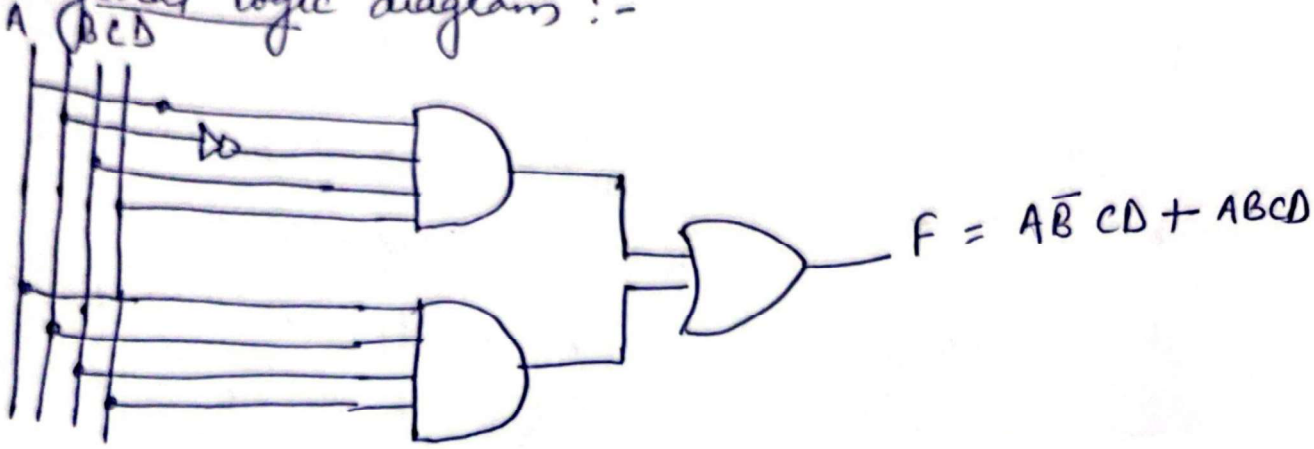
put $\because 1 = C+\bar{C}$

Scanned with CamScanner

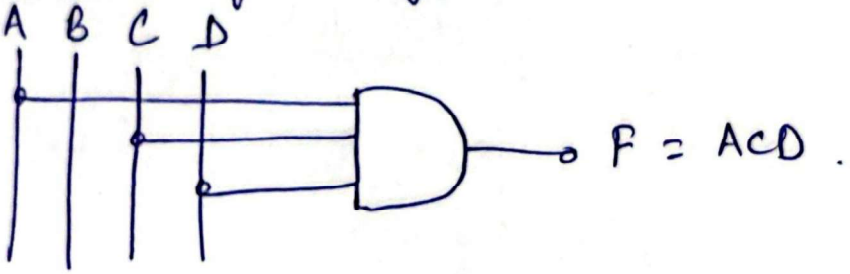
5. Simply the equation $F = A\bar{B}CD + ABCD$ and draw truth table and logic diagram.

$$F = A\bar{B}CD + ABCD$$
$$= ACD(\bar{B}+1)$$
$$= ACD.$$

Original logic diagram :-



Simplified logic diagram :-



A C D	F = ACD
0 0 0	0
0 0 1	0
0 1 0	0
0 1 1	0
1 0 0	0
1 0 1	0
1 1 0	0
1 1 1	1

Truth table from equation.

Scanned with CamScanner

Ex: Develop a Boolean equation from truthtable and its simplification.

A	B	C	F
0	0	0	0
0	0	1	1 ✓
0	1	0	0
0	1	1	0
1	0	0	1 ✓
1	0	1	1 ✓
1	1	0	1 ✓
1	1	1	1 ✓

The function F where the output = 1 is given as -

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

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

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

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

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

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

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

$$F = A + \bar{B}C \quad \text{Ans.}$$

Scanned with CamScanner

LOGIC GATES

Logic gates are the fundamental building blocks of digital systems. The name logic gate is derived from the ability of such a device to make decisions, in the sense that it produces one output level when some combinations of input levels are present and a different output level when other combinations of input levels are present.

→ Inputs and Outputs of logic gates can occur in two levels →

HIGH ~~and~~ LOW
OR TRUE and FALSE
OR ON and OFF
OR 1 and 0

→ A table which lists all the possible combinations of input variables and the corresponding outputs is called a truth table. It shows how logic circuit's output responds to various combinations of logic levels at the inputs.

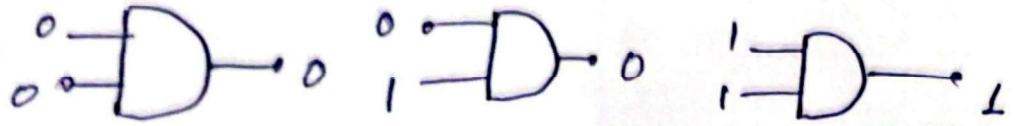
→ Voltage levels represent logic 1 and logic 0. level logic may be - positive logic or negative logic.

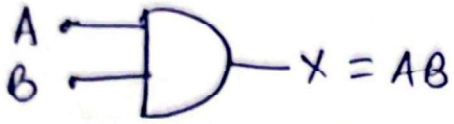
Scanned with CamScanner

The AND GATE :-

Input = 2 or more
Output = 1. when all input is at logic 1.

AND → All or Nothing gate.





a) Two-input AND gate

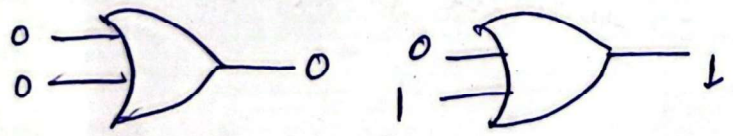
Inputs		Output
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

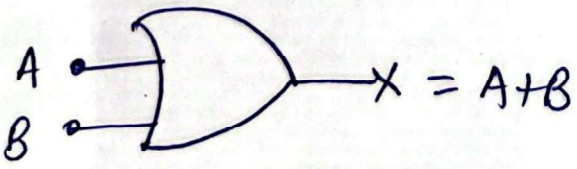
b) Truth table

The OR Gate :-

Input = 2 or more
Output = 1 , if any of its input is in logic 1.

OR → any or all gate





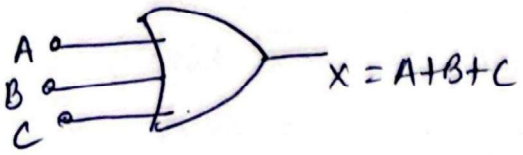
a) logic Symbol

Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

b) Truth Table

Scanned with CamScanner

of three inputs :-

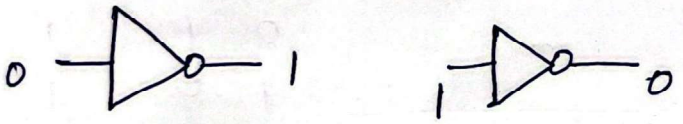


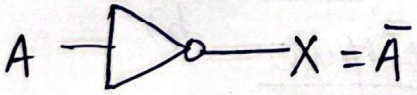
Inputs			Output
A	B	C	X = A+B+C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Truth Table

The NOT Gate (Inverter) -

Input = 1
Output = 1 , complements the input





a) logic symbol

Input	Output
A	X
0	1
1	0

b) Truth table (Inverter)

Scanned with CamScanner

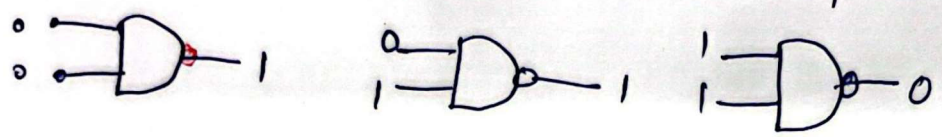
The Universal Gates –

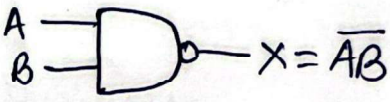
NAND and NOR gates are called Universal gates.

– Both NAND and NOR gates can perform all the three basic logic functions (AND, OR and NOT). AOI logic can be converted to NAND or NOR logic.

The NAND Gate –

NAND → NOT AND → AND output is NOTed.



= 

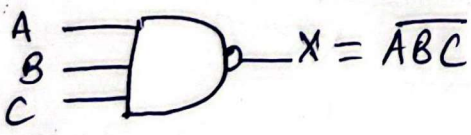
a) A two input NAND gate

Truth table

Inputs		Output
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

b)

Three inputs –



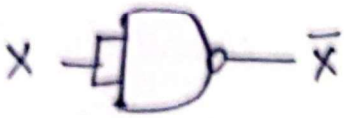
a) logic symbol

Inputs			Outputs
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

b) Truth table

Scanned with CamScanner

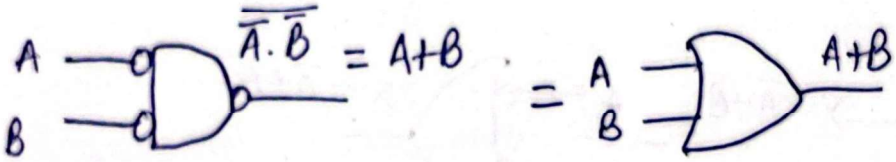
NAND gate as an Inverter: A NAND gate can also be used as an inverter by tying all its input terminals together and applying the signal to be inverted to the common terminal or by connecting all its input terminals except one, to logic 1 and applying the signal to be inverted to the remaining terminal.



a) NAND gate as an inverter

b) NAND gate as a controlled inverter

Bubbled NAND gate: The bubbled NAND gate is equivalent to OR gate.



Scanned with CamScanner

Bubbled OR gate :- Looking at the truth table of a two-input NAND gate, we see that the output $X = 1$, when either $A = 0$ or $B = 0$ or A and B $= 0$ i.e if either $\bar{A} = 1$ or $\bar{B} = 1$ or both $\bar{A}$ and B are equal to 1.

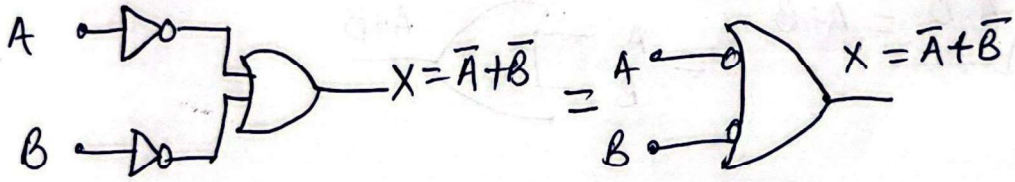
→ NAND gate can perform the OR function.

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

→ A NAND function can also be realized by first inverting the inputs and then ORing the inverted inputs.

→ NAND gate is a combination of two NOT gates and an OR gate.

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



Bubbled OR gate

Inputs		Inverted Inputs		Output
A	B	$\bar{A}$	$\bar{B}$	X
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

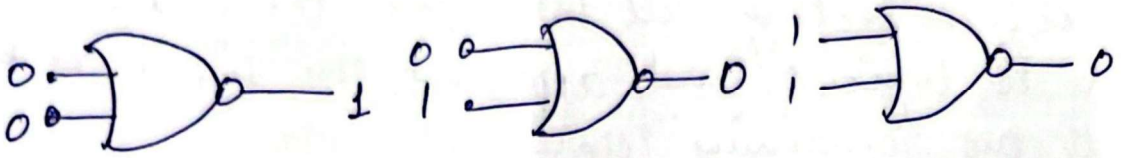
Scanned with CamScanner

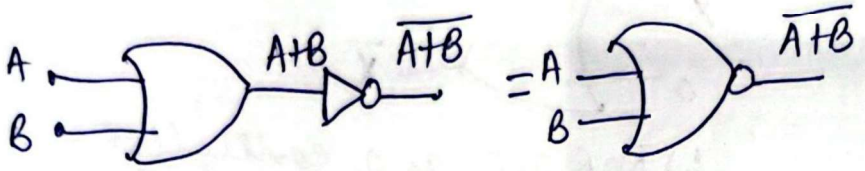
The NOR Gate :-

NOR → NOT-OR i.e the OR output is NOTed.

$$X = \overline{A+B+C+\cdots}$$

Output = 1, only when ~~one or~~ each of its input is at logic 0.



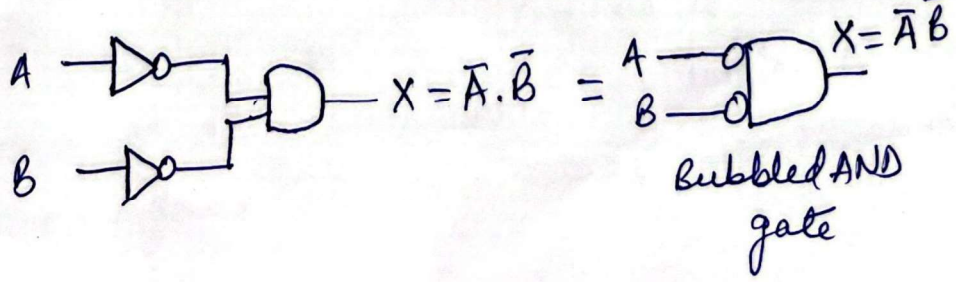


Inputs A	B	Output X
0	0	1
0	1	0
1	0	0
1	1	0

Bubbled AND gate :- looking at the truth table of a two-input NOR gate, we see that the output X is 1 only when both A and B are equal to 0 i.e only when both $\bar{A}$ and $\bar{B}$ are equal to 1. That means, a NOR gate is equivalent to an AND gate with inverted inputs and output, $X = \bar{A}.\bar{B}$.

for NOR, $X = \overline{A+B}$

$$X = \bar{A}.\bar{B}$$

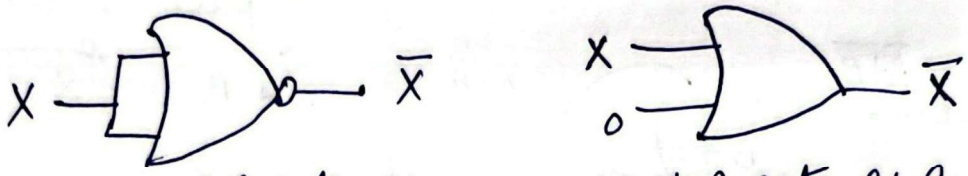


Inputs AB	Inverted Inputs $\bar{A}$	$\bar{B}$	Output X
00	1	1	1
01	1	0	0
10	0	1	0
11	0	0	0

Scanned with CamScanner

→ A NOR gate is equivalent to the same truth table.

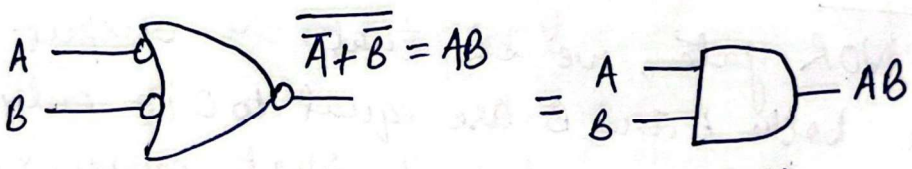
<u>NOR gate as an inverter</u>:- A NOR gate can be used as an inverter, by tying all its input terminals and applying the signal to be inverted to the common terminal or by connecting all its input terminals except one to logic 0, and applying the signal to be inverted to the remaining terminal.



a) NOR gate as an inverter

b) NOR gate as a controlled inverted

<u>Bubbled NOR gate</u>:



Bubbled NOR gate

AND

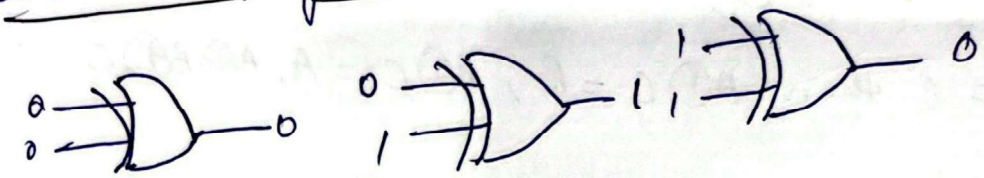
Scanned with CamScanner

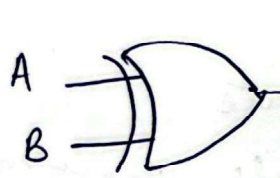
The Exclusive-OR (X-OR) Gate :-

input = 2

output = 1, whose output assumes a logic 1 state when one and only one of its two inputs assumes a logic 1 state. When both the inputs assume the logic 0 state or when both the inputs assume the logic 1 state, the output assumes a logic 0 state.

→ X-OR gate produces an output 1 only when the inputs are not equal, it is called an anti-coincidence gate or inequality detector.





$X = A \oplus B$

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

a) logic symbol

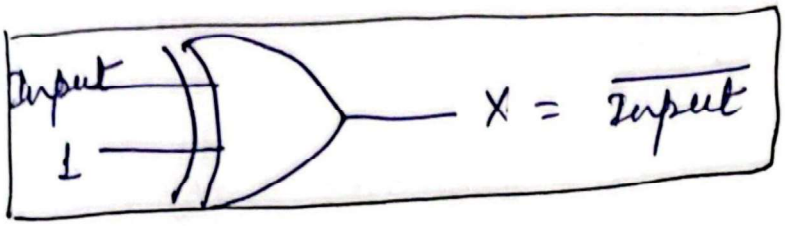
Inputs A B	Output $X = A \oplus B$
0 0	0
0 1	1
1 0	1
1 1	0

X-OR Gate as an Inverter —

If input bit is 0, the output is $0 \oplus 1 = 1$
if input bit is 1, the output is $1 \oplus 1 = 0$

- X-OR gate can be used as a controlled inverter, i.e one of its inputs can be used to decide whether the signal at the other input will be inverted or not.

Scanned with CamScanner



Properties of X-OR –

1. $A \oplus 1 = \bar{A}$
   $= A \cdot \bar{1} + \bar{A} \cdot 1 = A \cdot 0 + \bar{A} = \bar{A}$
2. $A \oplus 0 = A$
3. $A \oplus A = 0$
4. $A \oplus \bar{A} = 1$
5. $AB \oplus AC = A(B \oplus C)$
6. If $A \oplus B = C$, then $A \oplus C = B$, $B \oplus C = A$, $A \oplus B \oplus C = 0$.

The Exclusive-NOR (X-NOR) Gate :-

X-NOR gate = X-OR + NOT gate

Output = 1, when both the inputs assume same 0 state or when both inputs assume a 1 state.

Output = 0, when one of the inputs assume 0 state.

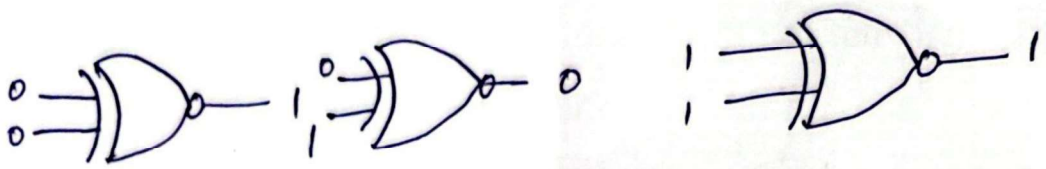
→ It is also called <u>Coincidence gate</u>, because its output is 1 only when its inputs coincide.

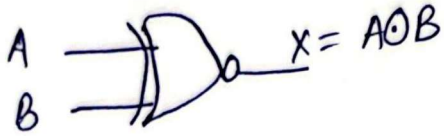
→ It is used as an <u>Equality detector</u>, because its output is 1 only when its inputs are equal.

Scanned with CamScanner

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

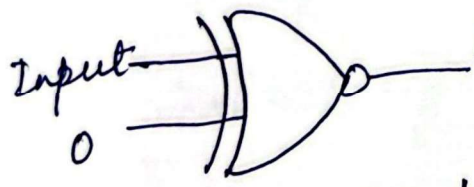
X = A ex-nor B





Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

X-NOR gate as an Inverter –



– X-NOR gate can be used as an inverter by connecting one of the two terminals to logic 0 and feeding the input sequence to be inverted to the other terminal.

If input = 0, output is $0 \odot 0 = 1$
input = 1, output is $1 \odot 0 = 0$

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

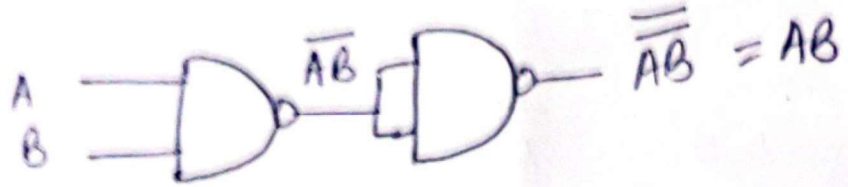
$$A \odot B \odot C \neq \overline{A \oplus B \oplus C}$$

Scanned with CamScanner

NAND and NOR gates as Universal Gates -

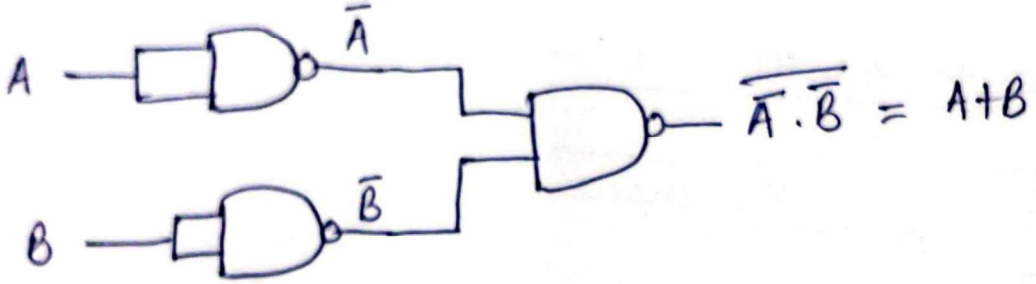
1. Realization of AND function using only NAND gates.

$$AB = \overline{\overline{AB}} = (\overline{\overline{AB}})$$



2. NAND → OR

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

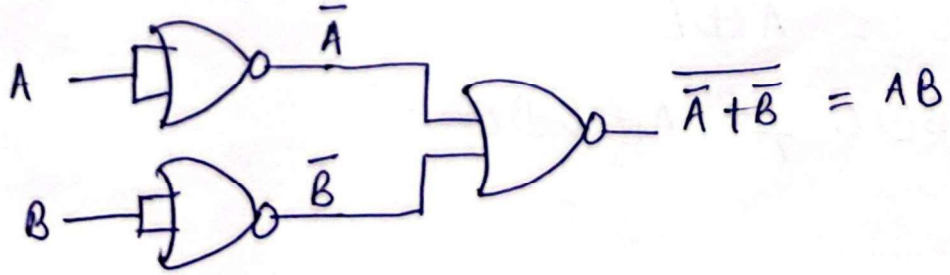


3. NAND → NOT gate

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

4. NOR gate → AND gate

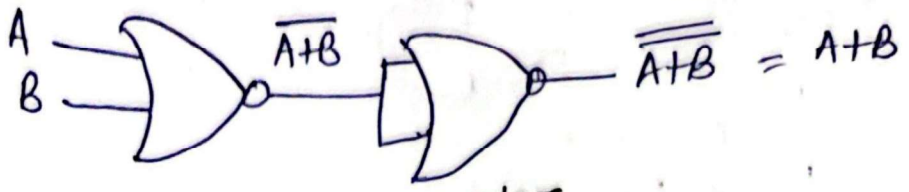
$$AB = \overline{\overline{AB}} = \overline{\overline{A}+\overline{B}}$$



Scanned with CamScanner

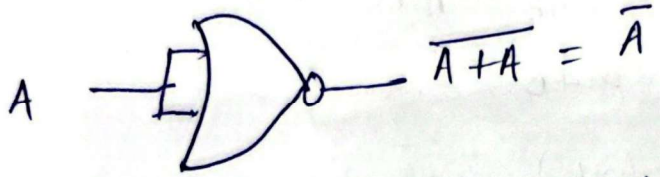
NOR gate → OR gate

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



6. NOR gate → ~~OR~~ NOT gate

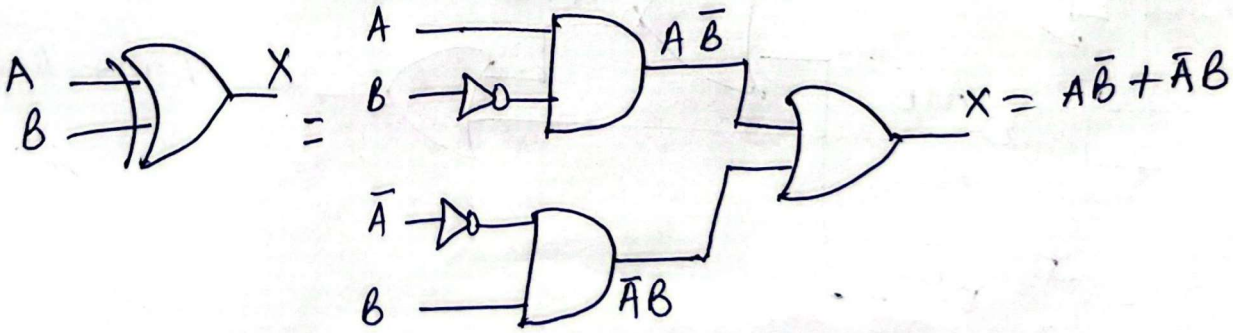
~~A+B~~ $\bar{A} = \overline{A+A}$



Ex1. Realize the X-OR function using a AOI logic b) NAND logic and c) NOR logic.

a) Using AOI logic:- AND-OR-Inverter (NOT)

X-OR function, $X = A\bar{B} + \bar{A}B$



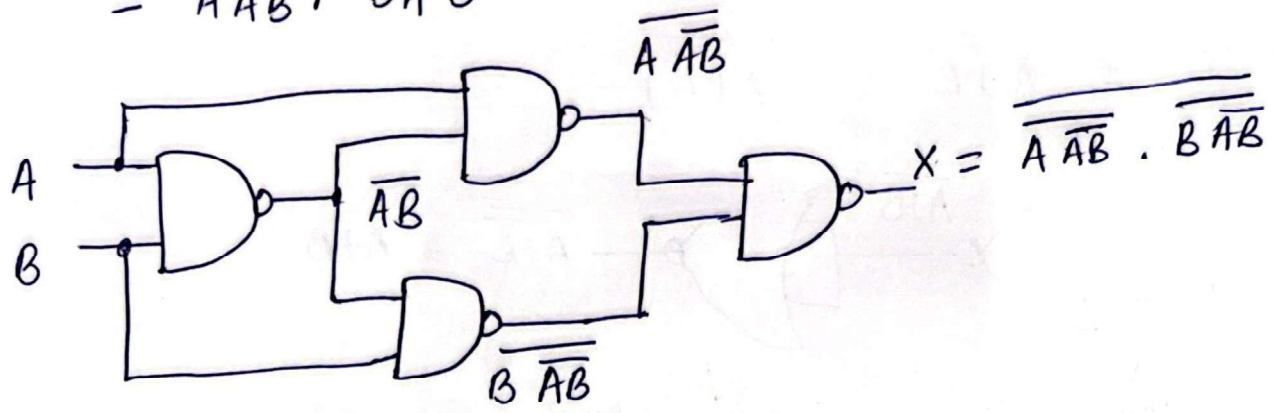
b) Using NAND logic:-

$$X = A\bar{B} + \bar{A}B$$
$$= A\bar{A} + A\bar{B} + \bar{A}B + B\bar{B}$$
$$= A(\bar{A}+\bar{B}) + B(\bar{A}+\bar{B})$$
$$= A\cdot\overline{AB} + B\,\overline{AB}$$
$$= \overline{\overline{A\cdot\overline{AB} + B\,\overline{AB}}}$$

reverse Demorgans

Scanned with CamScanner

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



c) Using NOR logic :-

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

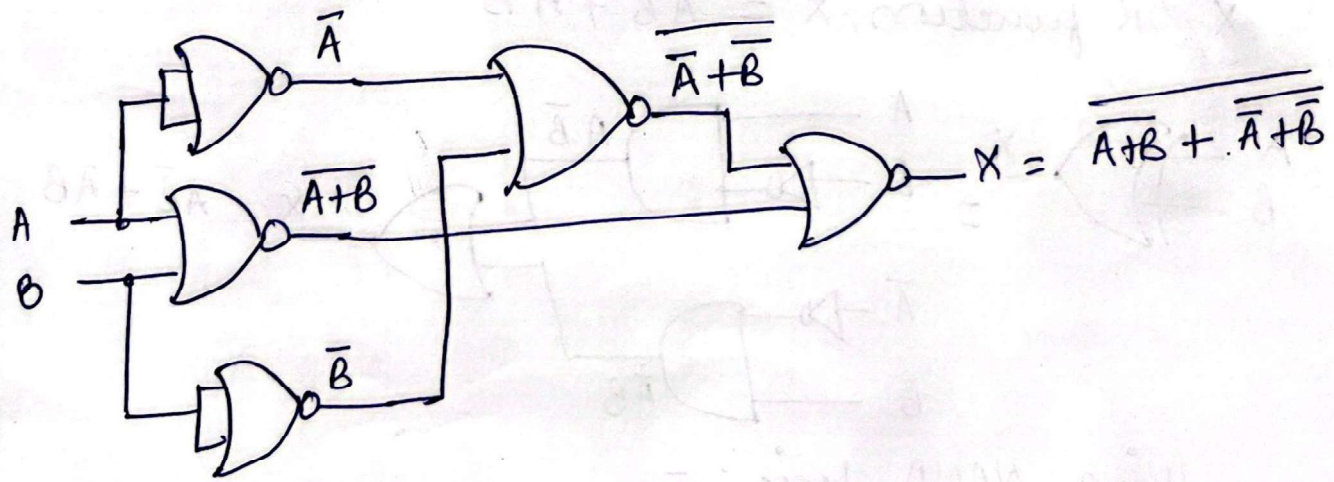
$$= A\bar{A} + A\bar{B} + \bar{A}B + B\bar{B}$$

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

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

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

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



Scanned with CamScanner

Standard Representations for Logical Functions -

- Logical functions are expressed in terms of logical variables and takes one of the two possible values 0 and 1. The logic function can be expressed in the following canonical forms :-

1. Standard Sum-of-Product (SOP) form :- This form is also called the Disjunctive Normal Form (DNF).

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

- A standard SOP form is one in which a number of product ~~of~~ terms, each one of which contains all the variables of the function either in complemented or non-complemented form are summed together. Each of the product terms in SOP form is called a minterm.

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

A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

E.g.: $Y = A'B + A'B' + AB'$ is a function in standard SOP form. It has two variable A & B. It can be represented as -

A	B	
0	0	$A'B'$
0	1	$A'B$
1	0	AB'
1	1	AB

$Y = 01 + 00 + 10$

$= m_1 + m_0 + m_2$

$Y = \sum m(0,1,2)$

Steps to convert a boolean function into SOP form :-

1. If any or more variables are missing in any product term, expand that term by multiplying it with the sum of each one of missing variable and its complement.
2. Drop out the redundant terms.

Scanned with CamScanner

(22) → A minterm assumes the value 1 only for one combination of the variables. An n variable function can have in all 2^n minterms.
The sum of the minterms whose value is equal to 1 is the standard SOP form of the function.

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

Decimal Code	A	B	C	f(A,B,C)	
0	0	0	0	0	
1	0	0	1	1	$\bar{A}\bar{B}C$
2	0	1	0	1	$\bar{A}B\bar{C}$
3	0	1	1	1	$\bar{A}BC$
4	1	0	0	0	
5	1	0	1	1	$A\bar{B}C$
6	1	1	0	0	
7	1	1	1	0	

$f(A,B,C) = m_1 + m_2 + m_3 + m_5$ where $f=1$

$f(A,B,C) = \Sigma m(1,2,3,5)$

→ Minterms –

Complemented variable = 0
Non complemented variable = 1

Scanned with CamScanner

(2) Standard Product-of-Sums (POS) form :-

This form is also called Conjunctive Canonical form. It is also called Expanded Product-of-Sums form or Canonical POS form. This is ~~derived~~ derived by considering the combinations for which $f=0$. Each term is a sum of all the variables.

- A standard POS form is one in which a number of sum terms, each one of which contains all the variables of the function either in complemented or non complemented form are multiplied together. Each of the sum terms in the standard POS forms is called a Maxterm.

E.g:- $Y = (A+B)(A+\bar{B})(\bar{A}+\bar{B})$ in POS form.

$Y = (00).(01).(11).$

$Y = M_0 \cdot M_1 \cdot M_3 = \prod M(0,1,3)$

Variable Representations -

Complemented variable = 1
Non-complemented variable = 0

Scanned with CamScanner

Steps of conversion of a boolean function in standard POS form –

1. If one or more variables are missing in any sum term, expand that term by adding the products of each of the missing term and its complement.
2. Drop out redundant term (repeating terms).

Ex: Convert $F = A' + B'$ into its standard SOP form.

1st term, 2nd term

B missing, A missing

$$\Rightarrow \quad F = A'(B+B') + (A+A')B'$$

$$= A'B + \underline{A'B'} + AB' + \underline{A'B'}$$

redundant term

$$= A'B + A'B' + AB'$$

$$= 01 + 00 + 10$$

$$= m_1 + m_0 + m_2$$

$$F = \sum m(0,1,2)$$

→ Missing terms in minterms = Maxterms

and Missing terms in maxterms = minterms.

Scanned with CamScanner

Ex: Convert Boolean function $F = A(A+\bar{B})B$ to standard POS form.

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

1st term A has missing B variable $= A + BB' = (A+B)(A+B')$

2nd term $(A+\bar{B})$ — no missing variable.

3rd term B has missing variable A $= B + AA' = (B+A)(B+A')$
$= (A+B)(A'+B)$

$$\Rightarrow F = (A+B)(A+B')(A+B')(A+B)(A'+B)$$

$$= (A+B)(A+B')(A'+B)$$

$$= (00).(01).(10).$$

$$F = M_0 \cdot M_1 \cdot M_2$$

$$F = \prod M(0,1,2).$$

Ex: $Y = (A+BC)(B+C'A)$

i) Convert it into standard SOP and POS forms.

ii) Implement the SOP form using NAND gates and POS expression using NOR gates.

Sol. i) $Y = (A+BC)(B+C'A)$

SOP form:-

$$Y = AB + A \cdot C' \cdot A + B \cdot B \cdot C + BC \cdot C'A$$

$$\therefore Y = AB + AC' + BC + 0$$

Scanned with CamScanner

$$Y = AB + AC' + BC$$

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

$$= ABC + ABC' + \cancel{ABC'} + AB'C' + \cancel{ABC} + A'BC$$

$$= ABC + ABC' + AB'C' + A'BC$$

$$Y = (111) + (110) + (100) + (011)$$

$$= m_7 + m_6 + m_4 + m_3$$

$$Y = \Sigma m(3, 4, 6, 7)$$

<u>POS form :-</u> $Y = (A+BC)(B+C'A)$

$$= (A+B)(A+C)(B+C')(\cancel{B+A})$$

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

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

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

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

$$= (000) \cdot (001) \cdot (010) \cdot (101)$$

$$Y = M_0 \cdot M_1 \cdot M_2 \cdot M_5$$

$$Y = \Pi M(0, 1, 2, 5)$$

Scanned with CamScanner

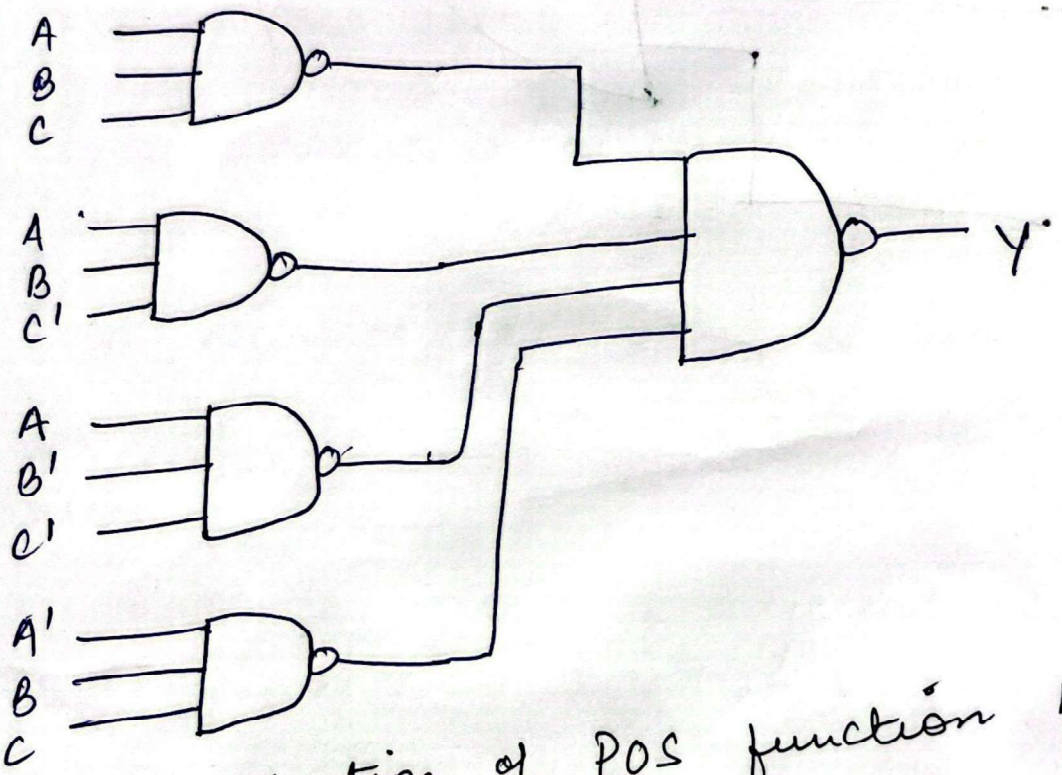
ii) Implementation of SOP function by NAND gates:-

$$Y = ABC + ABC' + AB'C' + A'BC$$

$$= \overline{\overline{ABC + ABC' + AB'C' + A'BC}}$$

→ break the inner line and change the sign

$$Y = \overline{(\overline{ABC}) \cdot (\overline{ABC'}) \cdot (\overline{AB'C'}) \cdot (\overline{A'BC})}$$



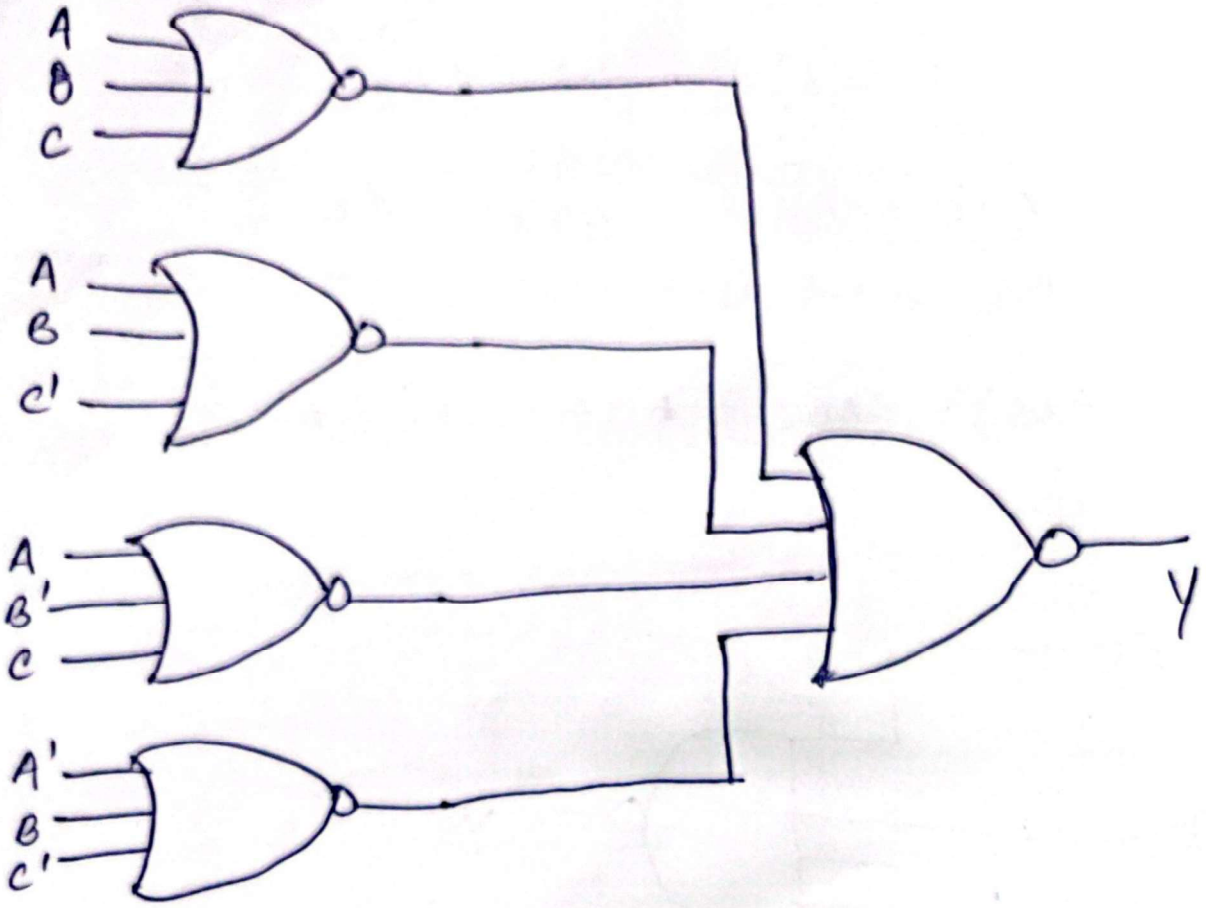
Implementation of POS function by NOR gate.

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

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

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

Scanned with CamScanner



Scanned with CamScanner

K-MAP (KARNAUGH-MAP)

- The boolean expressions can be simplified algebraically, but it can appear in many different forms. This is not a systematic method to ensure whether the minimal expression obtained is the real minimal. There is a lack of specific rules to predict each succeeding step in the manipulative process.
- The K-map is a systematic method of simplifying boolean expressions. The K-map is a chart or a graph composed of an arrangement of adjacent cells, each representing a particular combinations of variables in sum or product form.

n-variables has 2^n no. of cells.

Rules:-

1. Make the group of adjacent 1's (the group of power of 2 i.e 2, 4, 8 etc).
2. Try to cover all the 1s using minimum number of groups.
3. Overlapping of the group is allowed during minimization.
4. Consider those minterms individually, which cannot be grouped with other minterms.

Scanned with CamScanner

Two-Variable K-MAP – SOP form

Variables = 2

no. of minterms = $2^2 = 4$

A \ B	0	1
0	m_0	m_1
1	m_2	m_3

=

A \ B	0	1
0	$A'B'$	$A'B$
1	AB'	AB

A	B	minterms
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

A \ B	0	1
0		
1		1

a) AB

A \ B	0	1
0		1
1	1	1

b) $A+B$ (take sum of terms)

A \ B	0	1
0	1	1
1	1	1

$F = 1$

POS form:-

$F = \prod M(0,1)$ – In POS form, put 0 in the corresponding maxterm place / cell.

A \ B	0	1
0	0 (0)	0 (1)
1	(2)	(3)

$F = A$

A \ B	0	1
0		0
1		0

$F = \prod M(1,3)$

$F = \overline{B}$

A \ B	0	1
0		
1	0	0

$F = \prod M(2,3)$

$F = \overline{A}$

A \ B	0	1
0	0	0
1	0	0

$F = 0$

Scanned with CamScanner

Ex: Reduce the expression $f = (A+B)(A+\bar{B})(\bar{A}+\bar{B})$ using mapping.

$\downarrow$

$(00) \cdot (01)\ (11)$

$f = \Pi(0,1,3)$

A \ B	0	1
0	1	1
1		1

$f = A\bar{B}$ (take product of terms).

Three variable K-map –

A \ BC	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$ m_0 (0)	$\bar{A}\bar{B}C$ m_1 (1)	$\bar{A}BC$ m_3 (3)	$\bar{A}B\bar{C}$ m_2 (2)
1	$A\bar{B}\bar{C}$ m_4 (4)	$A\bar{B}C$ m_5 (5)	ABC m_7 (7)	$AB\bar{C}$ m_6 (6)

a) Minterms

A \ B	00	01	11	10
0	$A+B+C$ M_0	$A+B+\bar{C}$ M_1	$A+\bar{B}+\bar{C}$ M_3	$A+\bar{B}+C$ M_2
1	$\bar{A}+B+C$ M_4	$\bar{A}+B+\bar{C}$ M_5	$\bar{A}+\bar{B}+\bar{C}$ M_7	$\bar{A}+\bar{B}+C$ M_6

b) Maxterms

Ex:- $f = \underset{001}{\bar{A}\bar{B}C} + \underset{101}{A\bar{B}C} + \underset{010}{\bar{A}B\bar{C}} + \underset{110}{AB\bar{C}} + \underset{111}{ABC}$

$f = \Sigma m(1,2,5,6,7)$

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

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

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

$= (000) \cdot (001)(111)(011)(110)$

$= M_0 \quad M_5 \quad M_7 \quad M_3 \quad M_6$

$f = \Pi M(0,3,5,6,7)$

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

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

Scanned with CamScanner

Ex: Reduce, $f = \sum m(0,2,3,4,5,6)$ using K-map, implement it using NAND gate.

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

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

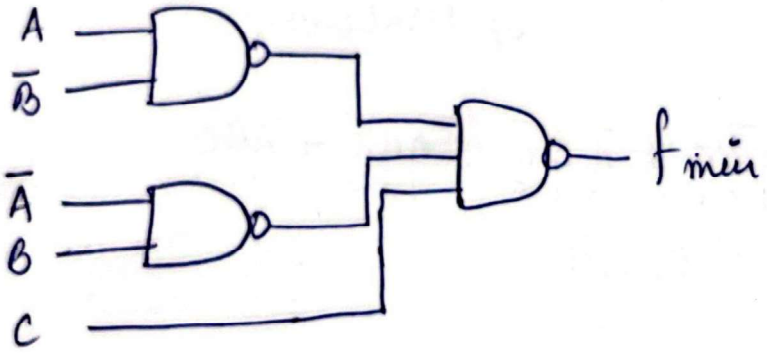
$$f_{min} = A\bar{B} + \bar{A}B + \bar{C}$$

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

Break inner line and change the sign.

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

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



Scanned with CamScanner

Ex:- Reduce the $f = \Pi M(0, 1, 2, 3, 4, 7)$ using mapping and implement in NOR logic.

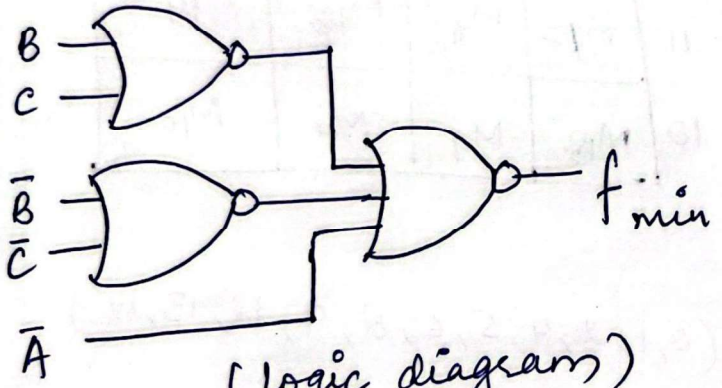
A \ BC	00	01	11	10
0	0	0	0	0
1	0		0	

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

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

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

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



(logic diagram)

Scanned with CamScanner

Ex.

$F(x,y,z) = \Sigma m(0,2,4,5,6)$

x \ yz	00	01	11	10
0	1			1
1	1	1		1

$f = z' + xy'$

Four Variable K-map

— $2^4 = 16$ combinations of input variables ABCD or wxyz.

AB \ CD	00	01	11	10
00	m_0 0	m_1 1	m_2 3	m_3 2
01	m_4 4	m_5 5	m_6 7	m_7 6
11	m_{12} 12	m_{13} 13	m_{15} 15	m_{14} 14
10	m_8 8	m_9 9	m_{11} 11	m_{10} 10

SOP form

AB \ CD	00	01	11	10
00	$(A+B+C+D)$ M_0 0	M_1 1	M_2 3	M_3 2
01	M_4 4	M_5 5	M_6 7	M_7 6
11	M_{12} 12	M_{13} 13	M_{15} 15	M_{14} 14
10	M_8 8	M_9 9	M_{11} 11	M_{10} 10

Ex:- $F(w,x,y,z) = \Sigma(0,1,2,4,5,6,8,9,12,13,14)$

wx \ yz	00	01	11	10
00	1	1		1
01	1	1		1
11	1	1		1
10	1	1		

$f = y' + w'y' + xz'$ Ans

Octave = 1

Quad = 2

Scanned with CamScanner

Ex:- $F(A,B,C,D) = \sum m(0,1,2,3,4,5) + d(10,11,12,13,14,15)$ express in POS form and minimize.

$\Rightarrow F(A,B,C,D) = \prod M(6,7,8,9) + d(10,11,12,13,14,15)$

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

quad

Octave

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

Scanned with CamScanner

Ex:- $F(A,B,C,D) = \sum(0,2,3,5,7,8,9,10,11,13,15)$

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

$F = BD + \bar{B}\bar{D} + AD + CD$

Ex: $f(A,B,C,D) = \sum m(0,1,2,3,5,7,8,9,10,12,13)$

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

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

SOP form

Ex:- $f(A,B,C,D) = \prod M(4,6,11,14,15)$

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

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

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

POS form

Scanned with CamScanner

Don't care Combinations :- It often occurs that for certain input combinations, the value of the output is unspecified either because the input combinations are invalid or because the precise value of the output is of no consequence. The combinations ~~or optional combinations~~ for which the values of the expression are not specified are called 'don't care' combinations or optional combinations. The output is a don't care for these invalid combinations e.g: in 8421 (BCD) code, the binary states 1010, 1011, 1100, 1101, 1110 and 1111 are invalid and the corresponding outputs are don't cares.

Ex :- Reduce the expression $f = \Sigma m(1,5,6,12,13,14) + d(2,4)$.

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

SOP form

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

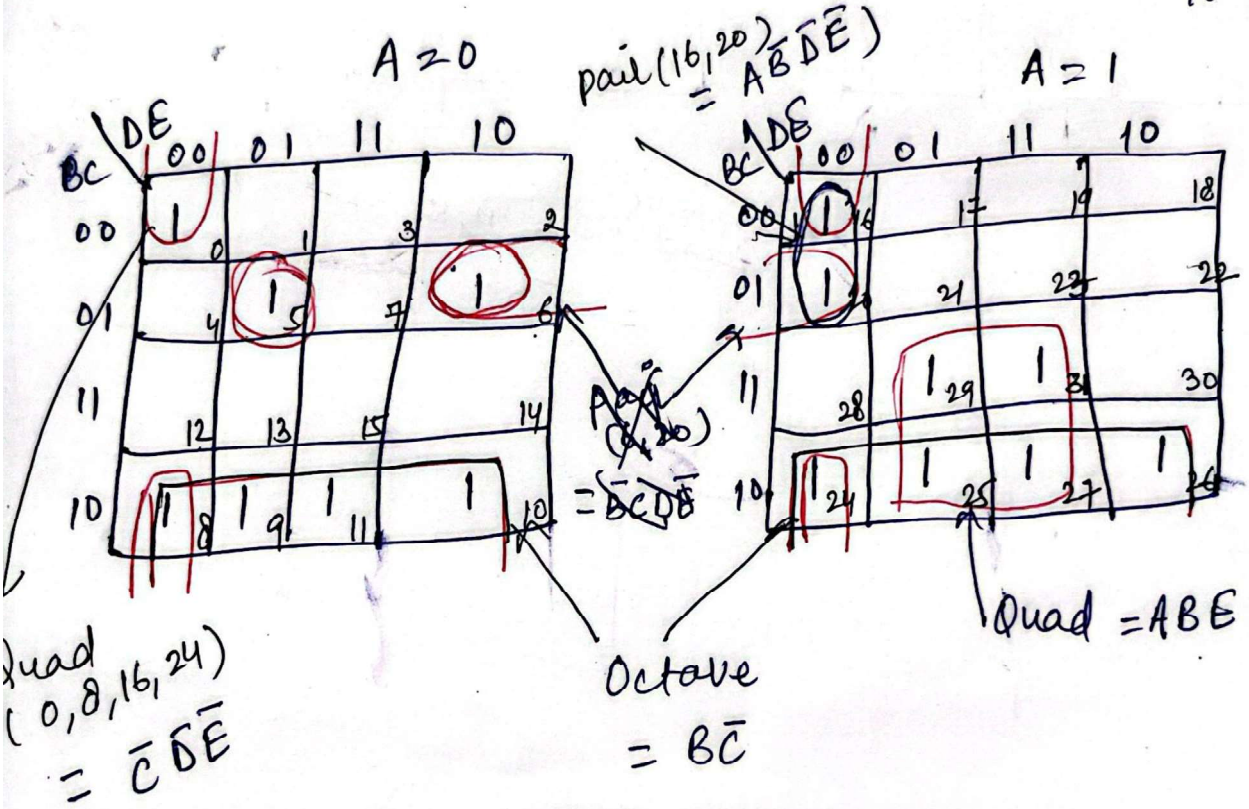
Scanned with CamScanner

Five variable K-Map – $2^5 = 32$ distinct possible terms in a five variable.

– We consider the two four-variable K-maps superimposed one one another.

Ex: Simplify the logic function:-

$F(A,B,C,D,E) = \Sigma m(0,5,6,8,9,10,11,16,20,24,25,26,27,29,31)$ using K-map.



Invidual $(m_5) = \bar{A}\bar{B}C\bar{D}E$

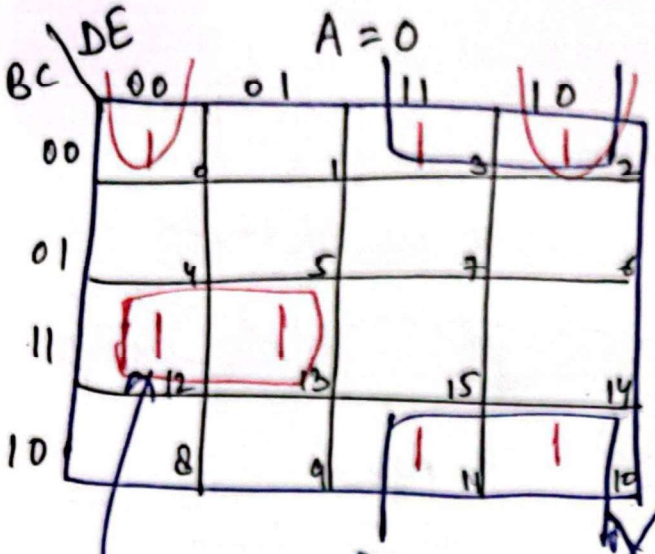
$m_6 = \bar{A}\bar{B}CD\bar{E}$

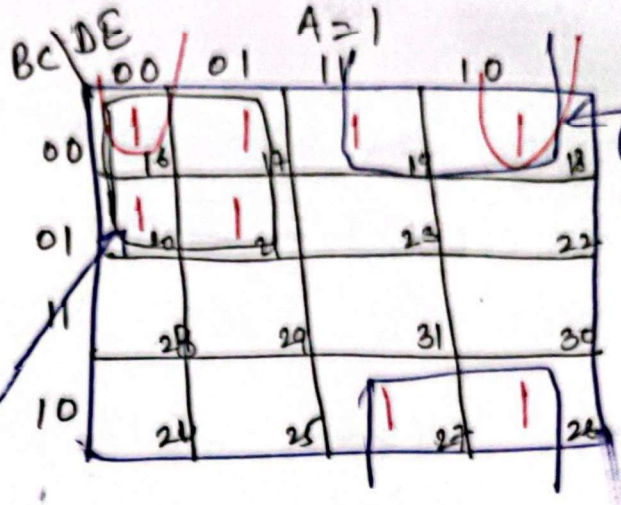
$$F = \bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}C\bar{D}E + \bar{A}\bar{B}CD\bar{E} + B\bar{C} + ABE$$

Scanned with CamScanner

Ex: Reduce the function:-

$f = \Sigma m(0,2,3,10,11,12,13,16,17,18,19,20,21,26,27)$





pair (12,13) $= \bar{A}BC\bar{D}$

quad (16,17,20,21) $= A\bar{B}\bar{D}$

quad (0,2,16,18) $= \bar{B}\bar{C}\bar{E}$

octave ~~quad~~ (3,2,11,10,19,18,27,26) $= \bar{C}D$

$$f = \bar{A}BC\bar{D} + A\bar{B}\bar{D} + \bar{B}\bar{C}\bar{E} + \bar{C}D.$$

Scanned with CamScanner

QUINE - Mc CLUSKEY METHOD (TABULAR METHOD)

- The minimization of Boolean expressions using K-maps is limited to a maximum of six variables.
- The Quine - Mc cluskey method is also known as tabular method, is a more systematic method of minimizing expressions of even larger number of variables.
- The fundamental of this tabulation procedure is based is that, repeated application of the combining theorem $PA + P\bar{A} = P$ (P is a set of literals) on all adjacent pairs of terms, yields the set of all prime implicants, from which a minimal sum may be selected.

E.g:- Consider the minimization of the expression -

$$\sum m(0,1,4,5) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C$$

The first two terms, and the 3rd & 4th terms can be combined to yield -

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

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

- Here if we consider binary representation of minterms, m_0 (000) and m_1 (001), they differ only one position. The result is 00 - i.e C variable is absorbed.

Scanned with CamScanner

- similarly m_4 (100) and m_5 (101)

i.e 10 – (Result).

Now, 00 – and 10 – can be combined b'coz they differ in only one position.

The result is – 0 – . $\Rightarrow \bar{B}$.

Steps :-

(1) List all the minterms

(2) Arrange all minterms in groups of the same number of 1s in their binary representation in column 1. Start with the least no. of 1's group and continue with the groups of increasing number of 1s. The number of 1s in a term is called the <u>index</u> of the term. The number of 1s in the binary form of a minterm is also called its <u>weight</u>.

(3) Compare each term of the lowest index group with every term in the succeeding group. Two terms from adjacent groups are combinable, if their binary representations differ by just a single digit in the same position, the differing term is replaced by a dash (–). Place a check mark (✓) next to every term and write the combined terms in column 2.

Repeat this by comparing each term in a group of index i with every term in the group index i+1, until all possible combinations have been exhausted.

Scanned with CamScanner

Compare the terms generated in step 2 in the same way. Write the new terms in column 3 and put a check mark next to each term which has combined in column 2. Continue the process with terms in columns 3, 4 etc. until no further combinations are possible.

The remaining unchecked terms constitute the set of prime implicants of the expression. They are called <u>prime implicants</u> b'coz they are not covered by any other term with fewer literals.

(5) List all the prime implicants and draw the prime implicant chart. (The don't cares if any should not appear in the prime implicant chart).

(6) Obtain the <u>Essential prime implicants</u> and write the minimal expression.

* <u>Prime Implicants</u> - In K-map, prime implicants are the groups of adjacent minterms that form squares or rectangles. They represent all the possible groups that can be formed in a K-map.

* <u>Essential prime Implicants</u> - These are groups that cover at least one minterm that cannot be covered by any other prime implicants. They are usually part of the final solution.

→ Atleast one '1', not participating in any other group.

Scanned with CamScanner

Ex:- $F(A,B,C,D) = \Sigma m(0,1,3,7,8,9,11,15)$

ABCD
0 - 0000
1 - 0001
3 - 0011
7 - 0111
8 - 1000
9 - 1001
11 - 1011
15 - 1111

Step-1 — Column 1

Index (group)	Minterm	Binary Rep.
index 0	0	0000 ✓
index 1	1	0001 ✓
	8	1000 ✓
index 2	3	0011 ✓
	9	1001 ✓
index 3	7	0111 ✓
	11	1011 ✓
index 4	15	1111 ✓

Step-2 — column 2

Index	Matched Pairs	ABCD	Mat.P
0	0,1	000 - ✓	(0,1,8,9) - 00 -
	0,8	- 000 ✓	(0,8,1,9) - 00 -
1	1,3	00 - 1 ✓	(1,3,9,11) - 0 - 1
	1,9	- 001 ✓	(1,9,3,11) - 0 - 1
	8,9	100 - ✓	
2	3,7	0 - 11 ✓	(3,7,11,15) - - 11
	3,11	- 011 ✓	(3,11,7,15) - - 11
	9,11	10 - 1 ✓	
3	7,15	- 111 ✓	
	11,15	1 - 11 ✓	

Step 3 — column 3

→ Matched pair, if there is a difference of only one term.

→ Tick the terms, those are in matched pair.

Step 4

→ There is no matched pair in step 3.

index 0 → $\bar{B}\bar{C}$

index 1 → $\bar{B}D$

index 2 → CD

Scanned with CamScanner

Sol:-

Prime implicant table

Prime Implicants	Minterms	0	1	3	7	8	9	11	15
$\bar{B}\bar{C}$ →	0, 1, 8, 9	X	X			X	X		
$\bar{B}D$ →	1, 3, 9, 11		X	X			X	X	
CD →	3, 7, 11, 15			X	X			X	X
		$\bar{B}\bar{C}$			CD				

→ * Circle the column having single cross.

$$\boxed{F = \bar{B}\bar{C} + CD}$$ Ans.. Essential Implicants.

Ex 1 - $f = \Sigma m(6,7,8,9) + d(10,11,12,13,14,15)$

Sol^n:- Table on next page.

Prime Implicant chart -

PIs / Minterms	6	7	8	9
8, 9, 10, 11, 12, 13, 14, 15 → A			X	X
6, 7, 14, 15 → BC	X	X		

↑ Both are essential prime implicants.

$$\boxed{F = A + BC}$$ Ans.

Scanned with CamScanner

Index	Minterm	Binary Rep.
Index 1	8 ✓	1000
	6 ✓	0110
Index 2	9 ✓	1001
	10 ✓	1010
	12 ✓	1100
	7 ✓	0111
Index 3	11 ✓	1011
	13 ✓	1101
	14 ✓	1110
Index 4	15 ✓	1111

Pairs	ABCD	Quads	ABCD	Octets	ABCD
8, 9	1 0 0 – ✓	8,9,10,11	1 0 – – ✓	8,9,10,11	1 – – –
8,10	1 0 – 0 ✓	8,9,12,13	1 – 0 – ✓	12,13,14,15	
8,12	1 – 0 0 ✓	8,10,12,14	1 – – 0 ✓		
6,7	0 1 1 – ✓	6,7,14,15	– 1 1 – ✗		
6,14	– 1 1 0 ✓	9,11,13,15	1 – – 1 ✓		
9,11	1 0 – 1 ✓	10,11,14,15	1 – 1 – ✓		
9,13	1 – 0 1 ✓	12,13,14,15	1 1 – – ✓		
10,11	1 0 1 – ✓				
10,14	1 – 1 0 ✓				
12,13	1 1 0 – ✓				
12,14	1 1 – 0 ✓				
7,15	– 1 1 1 ✓				
11,15	1 – 1 1 ✓				
13,15	1 1 – 1 ✓				
14,15	1 1 1 – ✓				

Scanned with CamScanner

Q:- Obtain the set of prime implicants for the boolean expression -

$f = \Sigma m(0, 1, 6, 7, 8, 9, 13, 14, 15)$ using tabular method.

Solⁿ:-

	column 1			column 2			column 3	
Index	Minterm	Binary Repres.		Pairs	ABCD		Quads	ABCD
Index 0	0	0000	✓	0,1	000 –	✓	0,1,8,9	– 00 –
				0,8	– 000	✓		
Index 1	1	0001	✓	1,9	– 001	✓		
	8	1000	✓	8,9	100 –	✓		
Index 2	6	0110	✓					
	9	1001	✓	6,7	011 –	✓	6,7,14,15	– 11 –
Index 3	7	0111	✓	6,14	– 110	✓		
	13	1101	✓	9,13	1 – 01	×		
	14	1110	✓	7,15	– 111	✓		
Index 4	15	1111	✓	13,15	11 – 1	×		
				14,15	111 –	✓		

→ The terms 9,13 and 13,15 in column 2 and (0,1,8,9) and (6,7,14,15) cannot be combined with any other terms. So, these four terms are the prime implicants.

Scanned with CamScanner

Prime implicant chart

PIs | minterms	0	1	6	7	8	9	13	14	15
9, 13						X	X		
13, 15							X		X
0, 1, 8, 9	X	X			X	X			
6, 7, 14, 15			X	X				X	X

$f = \bar{B}\bar{C} + BC$ (13 minterm is missing in these terms so we can consider 9,13 or 13,15 also).

$F = \bar{B}\bar{C} + BC + ABD$ Ans.

OR

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

Scanned with CamScanner

1. $f = \sum m(1,2,3,5,6,7,8,9,12,13,15)$ using the tabular method.

Index	Minterm	Binary Rep.	Pairs	ABCD	Quads	ABCD	
Index 1	1	0001 ✓	1,3	00-1 ✓	1,3,5,7	0--1	P
	2	0010 ✓	1,5	0-01 ✓	1,5,9,13	--01	Q
	8	1000 ✓	2,3	001- ✓	2,3,6,7	0-1-	R
			2,6	0-10 ✓	8,9,12,13	1-0-	S
Index 2	3	0011 ✓	8,9	100- ✓			
	5	0101 ✓	8,12	1-00 ✓	5,7,13,15	-1-1	T
	6	0110 ✓					
	9	1001 ✓	3,7	0-11 ✓			
	12	1100 ✓	5,7	01-1 ✓			
Index 3	7	0111 ✓	5,13	-101 ✓			
	13	1101 ✓	6,7	011- ✓			
Index 4	15	1111 ✓	9,13	1-01 ✓			
			12,13	110- ✓			
			7,15	-111 ✓			
			13,15	11-1 ✓			

Scanned with CamScanner

Prime Implicant table -

PIs | Minterms	1	2	3	5	6	7	8	9	12	13	15
1,3,5,7 P	X		X	X		X					
1,5,9,13 Q	X			X				X		X	
2,3,6,7 R		X	X		X	X					
8,9,12,13 S							X	X	X	X	
5,7,13,15 T				X			X			X	X

Essential Implicants :- R , S and T.

= (2, 3, 6, 7, 8, 9, 12, 13, 5, 15)

Only minterm 1 is not covered, so, we can include P or Q.

$P = \bar{A}D$, $Q = \bar{C}D$, $R = \bar{A}C$, $S = A\bar{C}$, $T = BD$

~~$F = \bar{A}D + \bar{A}B + A\bar{C} + BD$~~

Two solutions :-

$F = P + R + S + T = \bar{A}D + \bar{A}C + A\bar{C} + BD$

or $F = Q + R + S + T = \bar{C}D + \bar{A}C + A\bar{C} + BD$

Scanned with CamScanner

Combinational & Sequential Circuit

Digital Electronics

Combinational Circuit

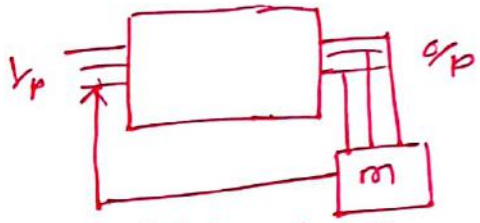
→ O/p is only depending on present I/p.

- No Feedback
- No Memory

Eg
- HA / FA
- HS / FS
- Mux / Demux
- Encoder / Decoder
- Code Converter.

Sequential Circuit

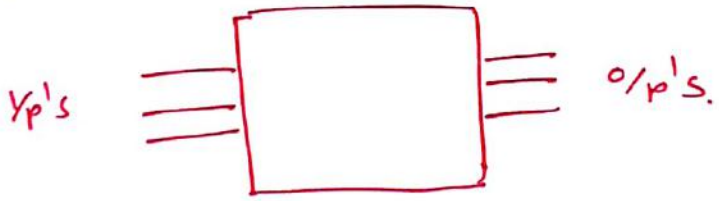
→ O/p is depending on present I/p and past O/p.



Posibility of memory & feedback.

Eg.
- Counter
- Shift register
- Flip Flop.

- Designing of combinational Circuits.



Step 1 - Determine & Define total I/p's and total O/p's of the Circuit.

Step-2 - Make truth table that defines relationship in bet.^{n} I/p's and O/p's.

Step-3 - Determine boolean eq^n using K-map.

Step-4 - Based on boolean eq^n, we can form circuit.

Engineering Funda YouTube Channel

The minimal function that can detect "divisible by 2" with 8421 BCD [$D_8 D_4 D_2 D_1$] is given by $\overline{D_1}$

	D_8	D_4	D_2	D_1	Y
0 –	0	0	0	0	1
1 –	0	0	0	1	0
2 –	0	0	1	0	1
3 –	0	0	1	1	0
4 –	0	1	0	0	1
5 –	0	1	0	1	0
6 –	0	1	1	0	1
7 –	0	1	1	1	0
8 –	1	0	0	0	1
9 –	1	0	0	1	0
10 –	1	0	1	0	X
⋮			⋮		⋮
15 –	1	1	1	1	X

Y: D_2D_1 \ D_8D_4	00	01	11	10
00	1 (0)	1 (4)	X (12)	1 (8)
01	0 (1)	0 (5)	X (13)	0 (9)
11	0 (3)	0 (7)	X (15)	X (11)
10	1 (2)	1 (6)	X (14)	X (10)

$\boxed{\overline{D_1}}$

The minimal function that can detect "divisible by 3" with 8421 BCD [$D_8 D_4 D_2 D_1$] is given by

$$\boxed{Y = D_1 D_8 + \overline{D_1} D_2 D_4 + D_1 D_2 \overline{D_4} + \overline{D_1}\,\overline{D_2}\,\overline{D_4}\,\overline{D_8}}$$

	D_8	D_4	D_2	D_1	Y
0 –	0	0	0	0	1
1 –	0	0	0	1	0
2 –	0	0	1	0	0
3 –	0	0	1	1	1
4 –	0	1	0	0	0
5 –	0	1	0	1	0
6 –	0	1	1	0	1
7 –	0	1	1	1	0
8 –	1	0	0	0	0
9 –	1	0	0	1	1
10 –	1	0	1	0	X
⋮			⋮		⋮
15 –	1	1	1	1	X

Y: D_2D_1 \ D_8D_4	00	01	11	10
00	1 (0)	0 (4)	X (12)	0 (8)
01	0 (1)	0 (5)	X (13)	1 (9)
11	1 (3)	0 (7)	X (15)	X (11)
10	0 (2)	1 (6)	X (14)	X (10)

IV ($\overline{D_1}\,\overline{D_2}\,\overline{D_4}\,\overline{D_8}$)

I ($D_8 D_1$)

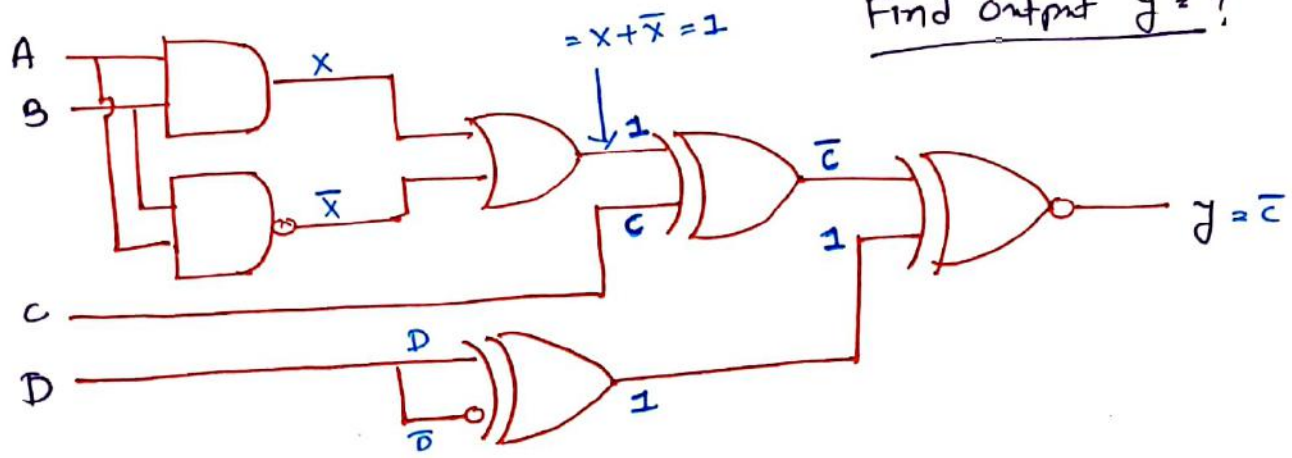
III ($D_1 D_2 \overline{D_4}$)

II ($D_2 \overline{D_1} D_4$)

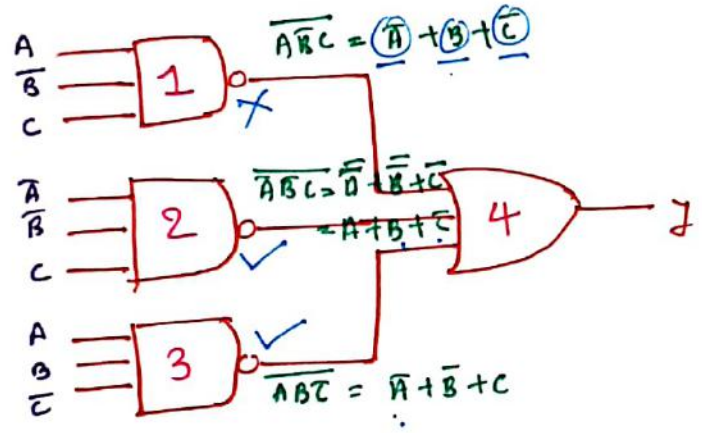
Scanned with CamScanner

Combinational Circuit Examples. 85

1) Find output $y = ?$

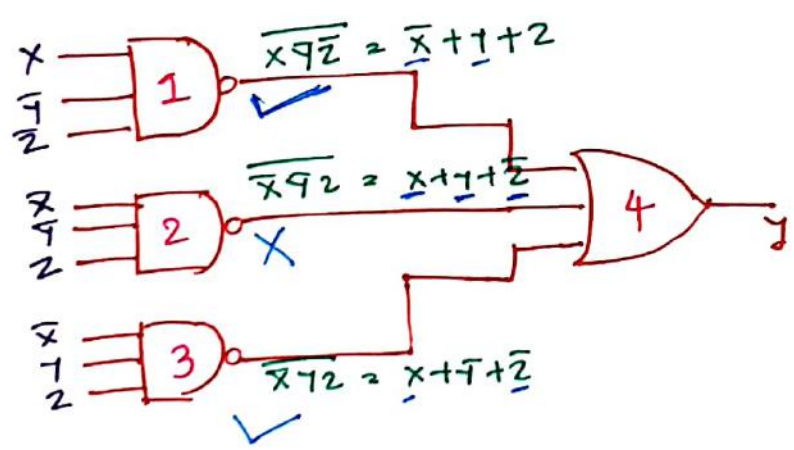


2)



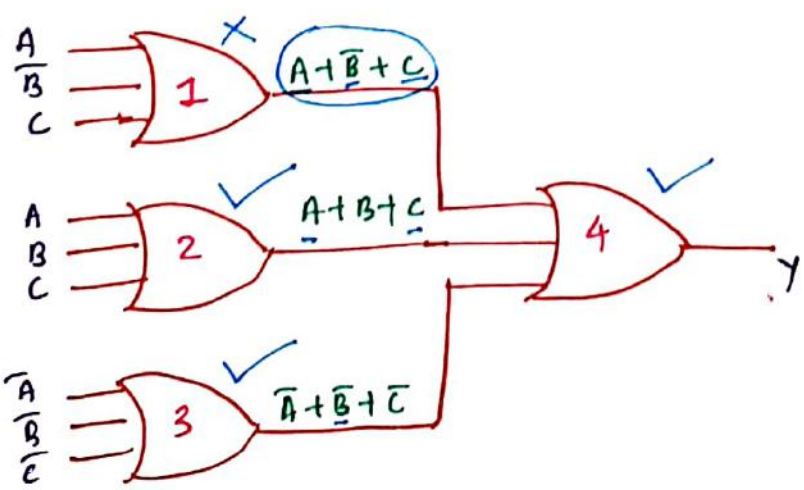
what is redundent gate in given combinational circuit ?

3)



What is redundant gate in given combinational circuit ?

4)

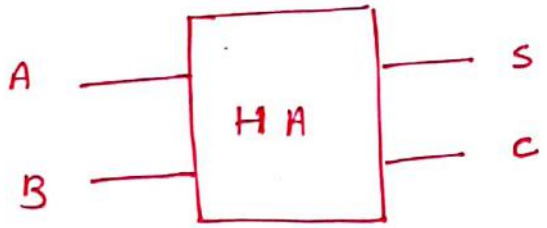


What is redundant gate in given combinational circuit ?

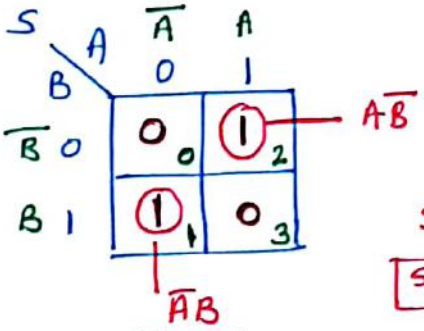
Scanned with CamScanner

HALF ADDER

→ Half Adder we perform two bit addition.

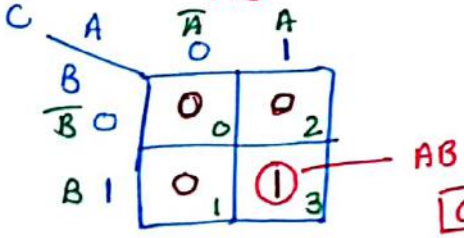


A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

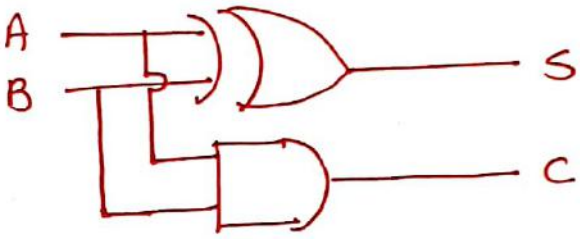


$S = A\bar{B} + \bar{A}B$

$\boxed{S = A \oplus B}$



$\boxed{C = A.B}$

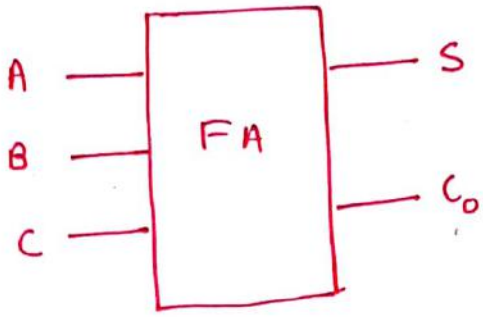


Engineering Funda YouTube Channel

Full Adder

87

- Full Adder Circuit is used to perform 3 bits addition.



A	B	C	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S: C \ AB	$\bar{A}\bar{B}$ 00	$\bar{A}B$ 01	AB 11	$A\bar{B}$ 10
$\bar{C}$ 0	0 (0)	1 (2)	0 (6)	1 (4)
C 1	1 (1)	0 (3)	1 (7)	0 (5)

$\bar{A}\bar{B}C$ $\quad$ $\bar{A}B\bar{C}$ $\quad$ ABC $\quad$ $A\bar{B}\bar{C}$

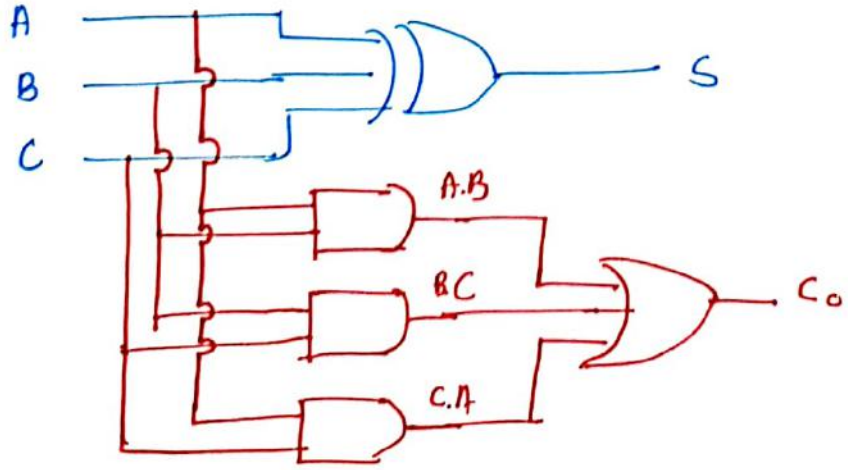
$$\rightarrow S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$$
$$= \bar{A}(\bar{B}C + B\bar{C}) + A(BC + \bar{B}\bar{C})$$
$$= \bar{A}(B \oplus C) + A(B \odot C)$$
$$= \bar{A}(B \oplus C) + A\overline{(B \oplus C)}$$

$$\boxed{S = A \oplus B \oplus C}$$

C_o: C \ AB	$\bar{A}\bar{B}$ 00	$\bar{A}B$ 01	AB 11	$A\bar{B}$ 10
$\bar{C}$ 0	0 (0)	0 (2)	1 (6)	0 (4)
C 1	0 (1)	1 (3)	1 (7)	1 (5)

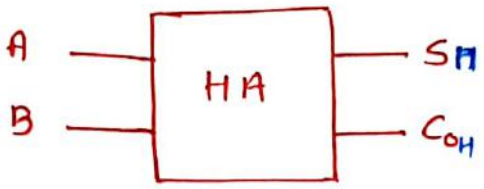
CB $\quad$ AB $\quad$ AC

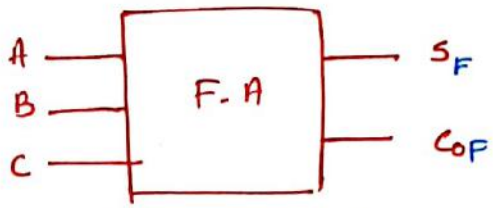
$$\boxed{C_o = AB + B.C + A.C}$$



Engineering Funda YouTube Channel
Scanned with CamScanner

Full Adder circuit using Half Adder.



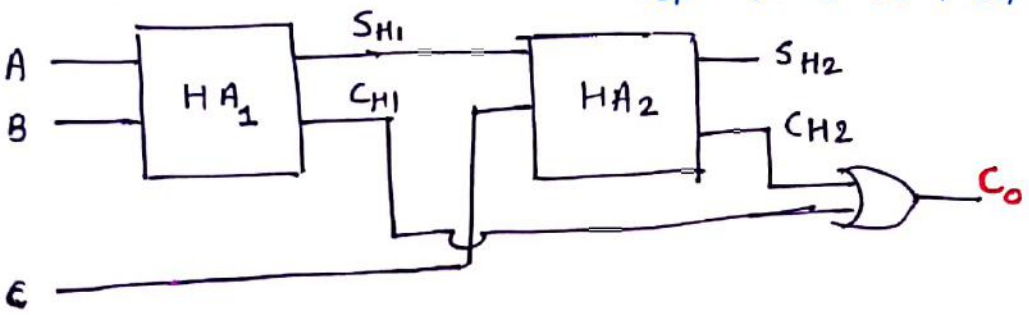


$S_H = A \oplus B$

$C_{OH} = A.B$

$S_F = A \oplus B \oplus C$

$C_{OF} = AB + BC + CA$



- $S_{H_1} = A \oplus B = A\bar{B} + \bar{A}B$

  $C_{H1} = A.B$

$S_{H2} = S_{H1} \oplus C = A \oplus B \oplus C$

$C_{H2} = S_{H1} . C$

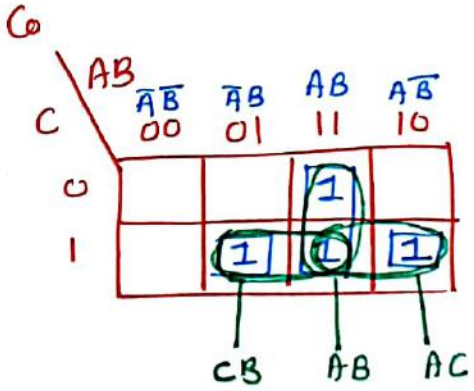
$= (A\bar{B} + \bar{A}B) . C$

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

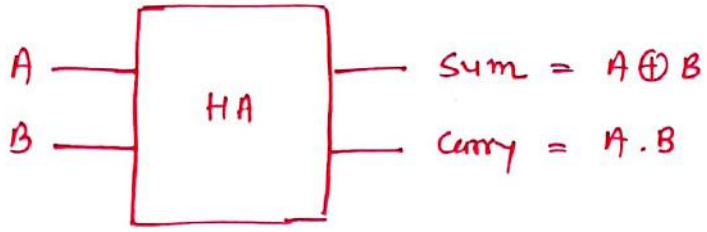
- $C_o = C_{H_2} + C_{H1}$

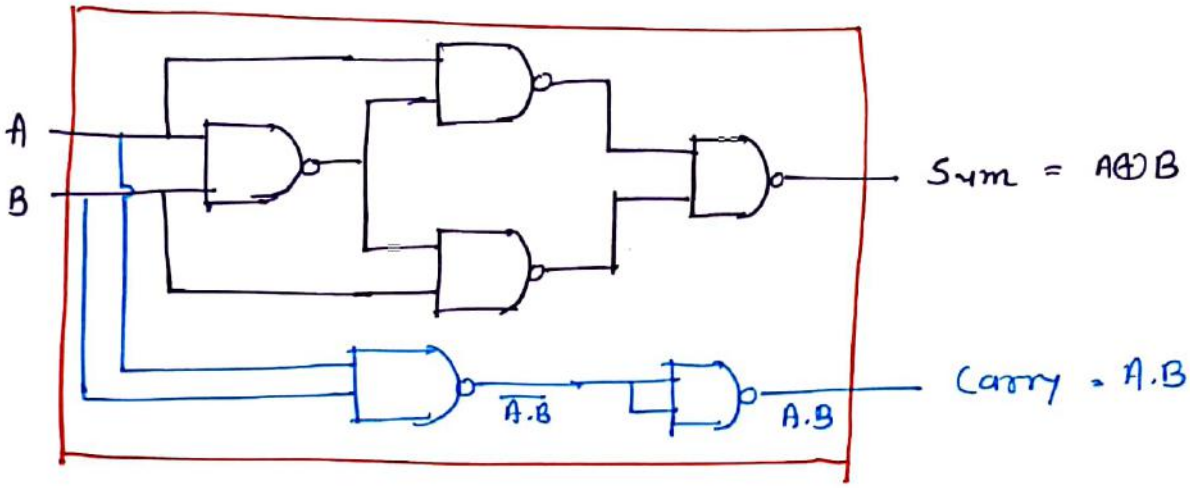
  $= \underline{A.B} + \underline{A\bar{B}C} + \underline{\bar{A}BC}$

  $= A.B + BC + AC$



HALF ADDER BY NAND GATES





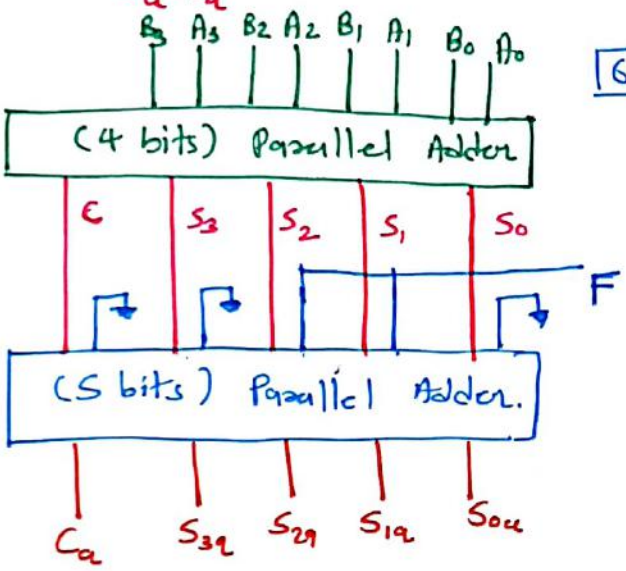
Engineering Funda YouTube Channel

BCD Adder.

A ← BCD data
\+ B ← BCD data

C S [If S < 9]
\+ 6 [If S > 9]

C_a S_a



$\boxed{6 \rightarrow 00110}$

C	S_3	S_2	S_1	S_0		
0	0	0	0	0	→	0
0	0	0	0	1	→	1
0	0	0	1	0	→	2
0	0	0	1	1	→	3
0	0	1	0	0	→	4
0	0	1	0	1	→	5
0	0	1	1	0	→	6
0	0	1	1	1	→	7
0	1	0	0	0	→	8
0	1	0	0	1	→	9
0	1	0	1	0	→	10
0	1	0	1	1	→	11
0	1	1	0	0	→	12
0	1	1	0	1	→	13
0	1	1	1	0	→	14
0	1	1	1	1	→	15
1	0	0	0	0	→	16
1	0	0	0	1	→	17
1	0	0	1	0	→	18

[S<9]

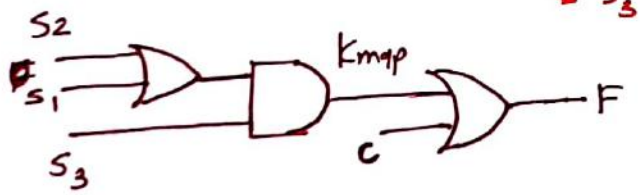
(S>9) (C=0)

(S>9) (C=1)

→ Cond.^{n} 1 → C = 1

→ Cond.^{n} 2 → find Kmap

$K_{map} = S_2S_3 + S_1S_3$
$= S_3(S_2 + S_1)$



Kmap

S_1S_0 \ S_3S_2	00	01	11	10
00	0	4	1 12	8
01	1	5	1 13	9
11	3	7	1 15	1 11
10	2	6	1 14	1 10

S_2S_3

S_1S_3

S < 9	15 < S < 9	S > 9 (C = 1)
A = 4	A = 5	
B = 4	B = 6	
8	11	
01000	01011	
↑↑↑↑↑	↑↑↑↑↑	
C S_3 S_2 S_1 S_0	C S_3 S_2 S_1 S_0	

Scanned with CamScanner

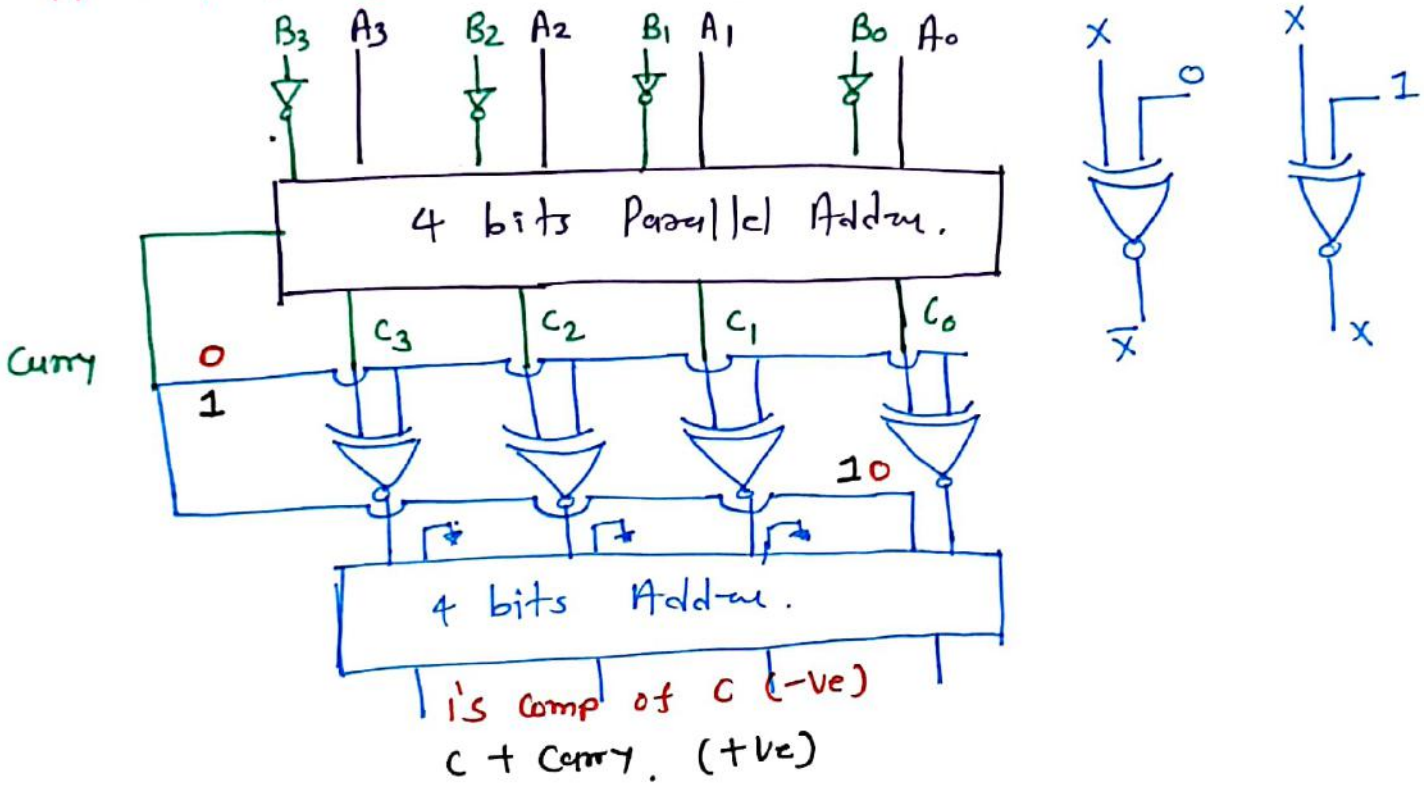
1's compliment Subtraction using parallel Adder

$$\begin{array}{r} A \\ -\,B \\ \hline R \end{array} \xrightarrow{\text{Binary}} A \qquad \xrightarrow{\text{1's Comp}} \begin{array}{r} (-B) \\ \hline \text{Carry } C \end{array}$$

Step 1

Step-2

- If Carry = 1 [Result is +ve] ⇒ R = C + Carry
  If Carry = 0 [Result is −ve] ⇒ R = 1's comp. of C.]



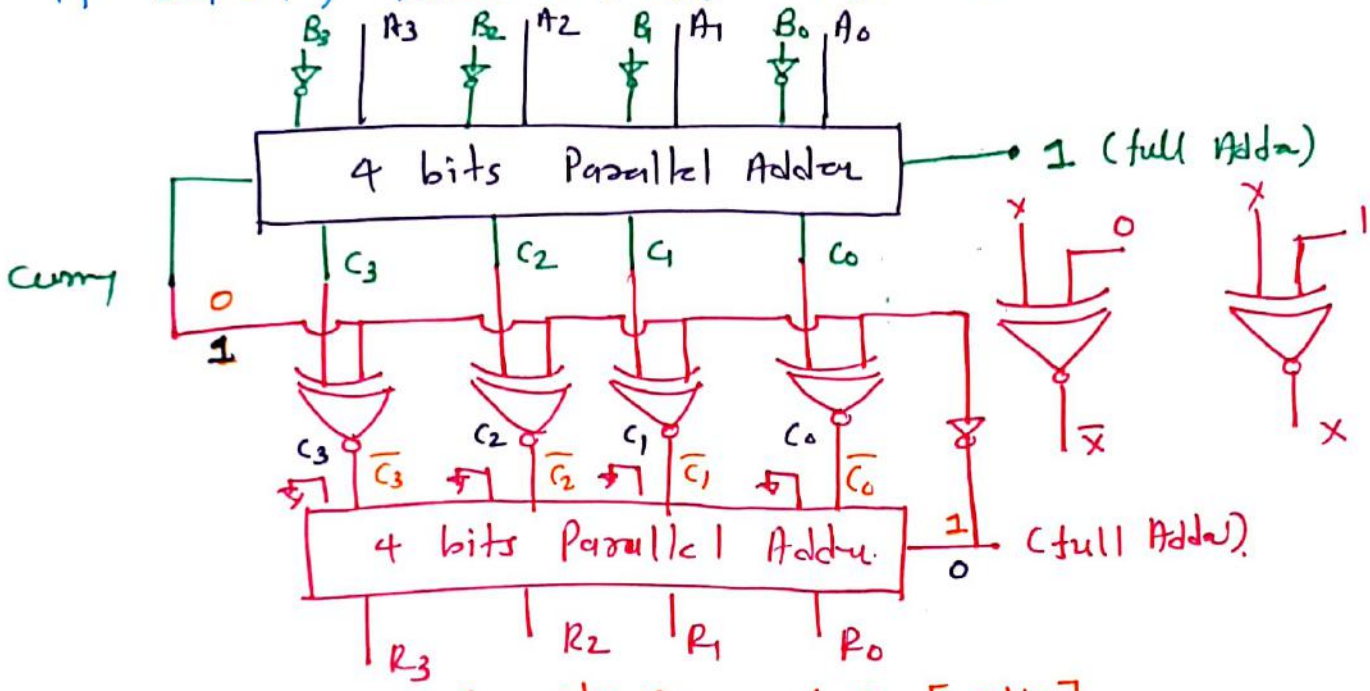
Engineering Funda YouTube Channel

2's Compliment Substruction using parallel Adder

$$\begin{array}{c} A \\ -B \\ \hline R \end{array} \quad \begin{array}{l} \xrightarrow{\text{Binary}} A \\ \xrightarrow{\text{2's Comp.}} -B \\ \hline \text{Carry } C \end{array} \Bigg] \text{ — Step-1}$$

→ If Carry = 0, Result is (−Ve), ⇒ R = 2's Comp. of C] → Step 2

If Carry = 1, Result is (+Ve), ⇒ R = S



R = 2's Comp of C [−Ve].

R = C [+Ve].

Half Subtractor 94

→ Half subtractor is performing two bits subtraction.

$$\begin{array}{r} X \\ -\ Y \\ \hline B\ D \end{array}$$

(X − Y)

X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

D	$\bar{X}$ (X = 0)	X (X = 1)
$\bar{Y}$ (Y = 0)	0	1
Y (Y = 1)	1	0

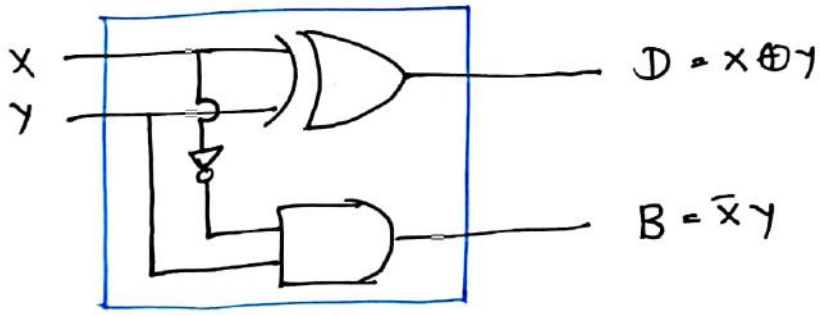
$$D = X\bar{Y} + \bar{X}Y$$
$$= X \oplus Y$$

B	$\bar{X}$ (X = 0)	X (X = 1)
$\bar{Y}$ (Y = 0)	0	0
Y (Y = 1)	1	0

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

→ for Y − X

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

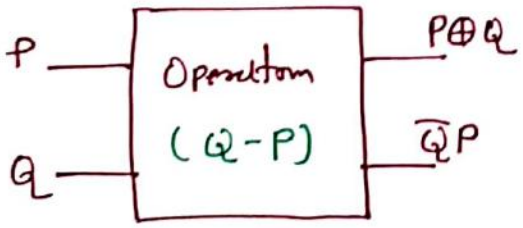


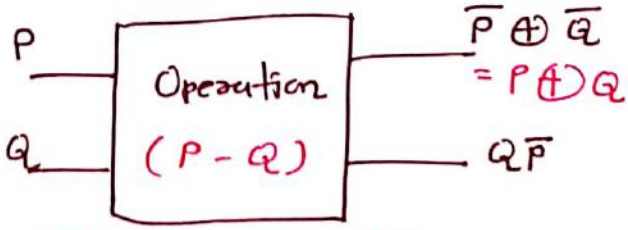
Half Sub.

Half Adder

$S = X \oplus Y$

$C = XY$



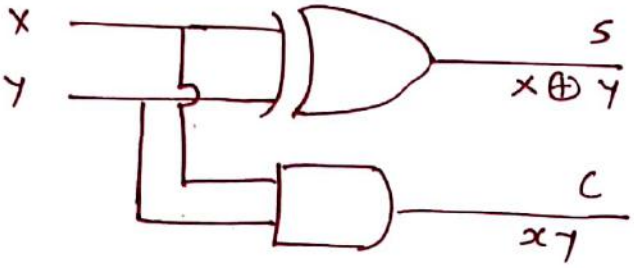


$$\bar{P} \oplus \bar{Q} = \bar{P}\bar{\bar{Q}} + \bar{\bar{P}}\bar{Q}$$
$$= \bar{P}Q + P\bar{Q}$$
$$= P \oplus Q$$

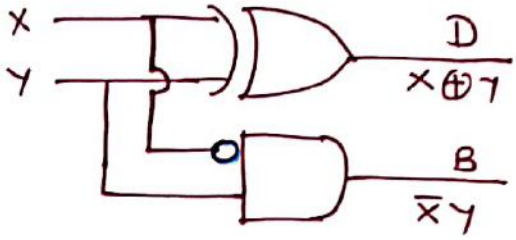
Scanned with CamScanner

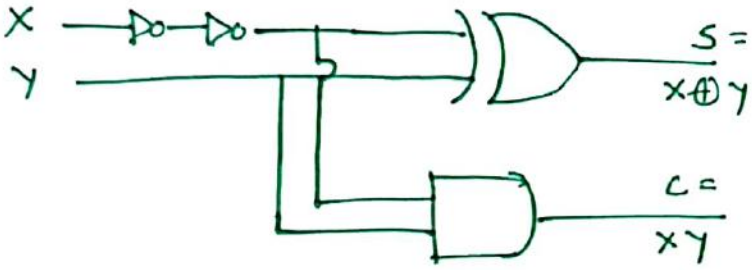
Half Adder using Half Subtractor

→ Half Adder

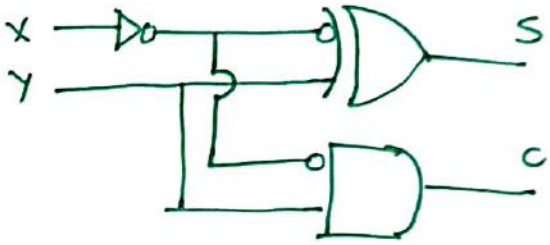


→ Half Subtractor

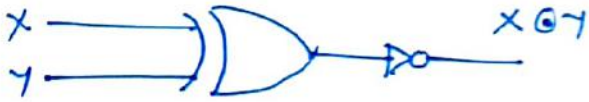


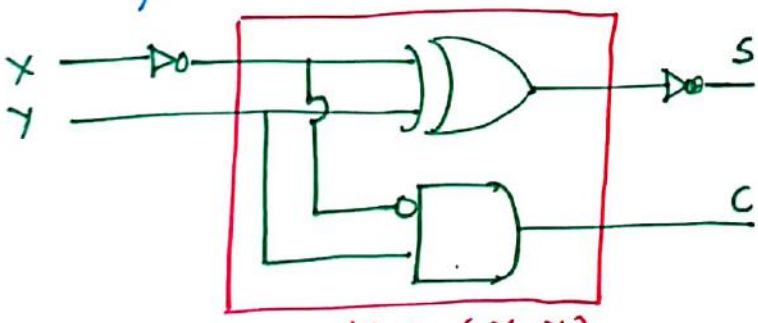


⇒



⇒





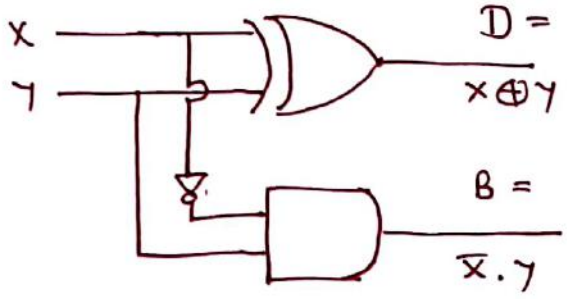
H.S (X−Y)

H.S (Y−X)

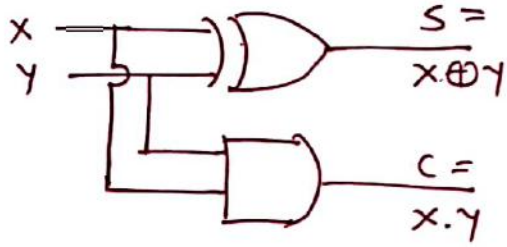
Engineering Funda YouTube Channel

Half Subtractor using Half Adder. 96

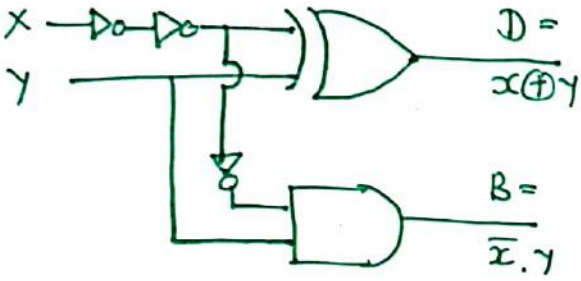
→ Half Subtractor → Half Adder



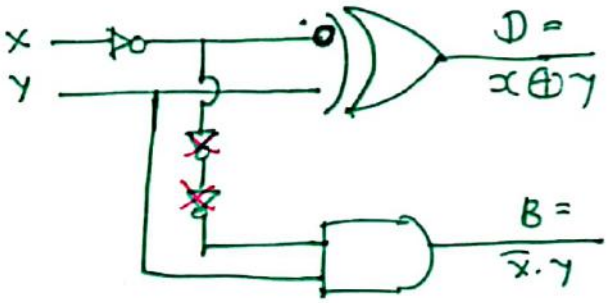
(X − Y)



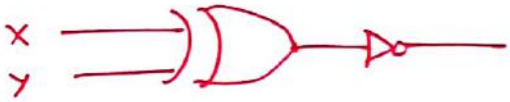
(X + Y)

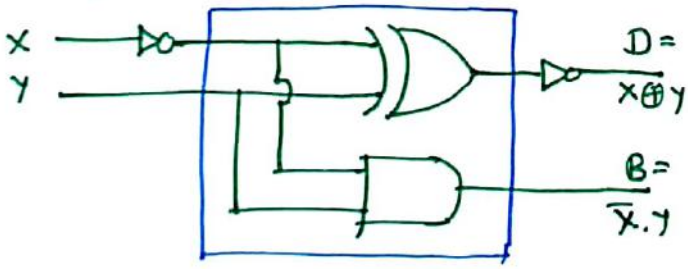


⇒

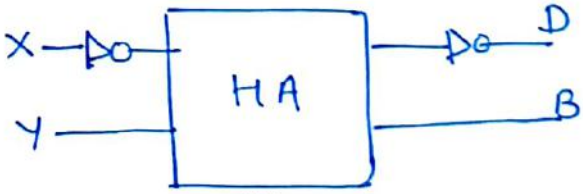


⇒

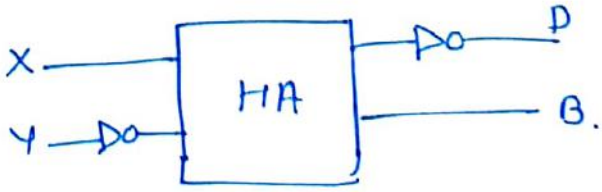




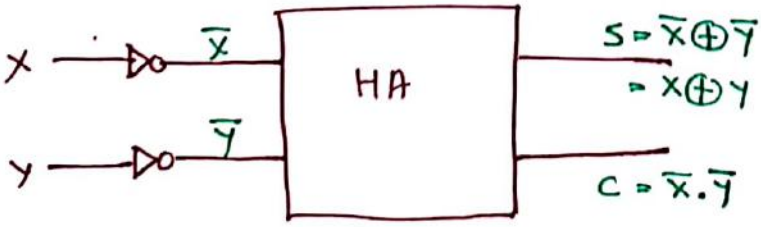
Half Adder.



(X − Y)



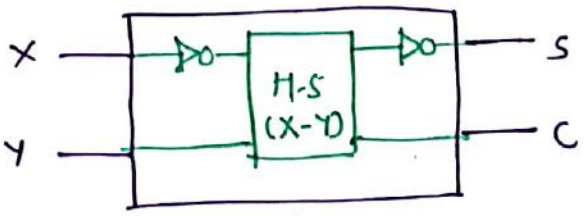
(Y − X).



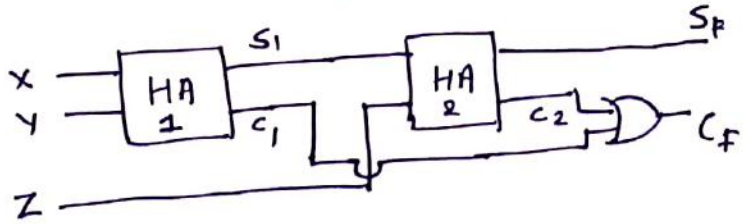
Engineering Funda YouTube Channel

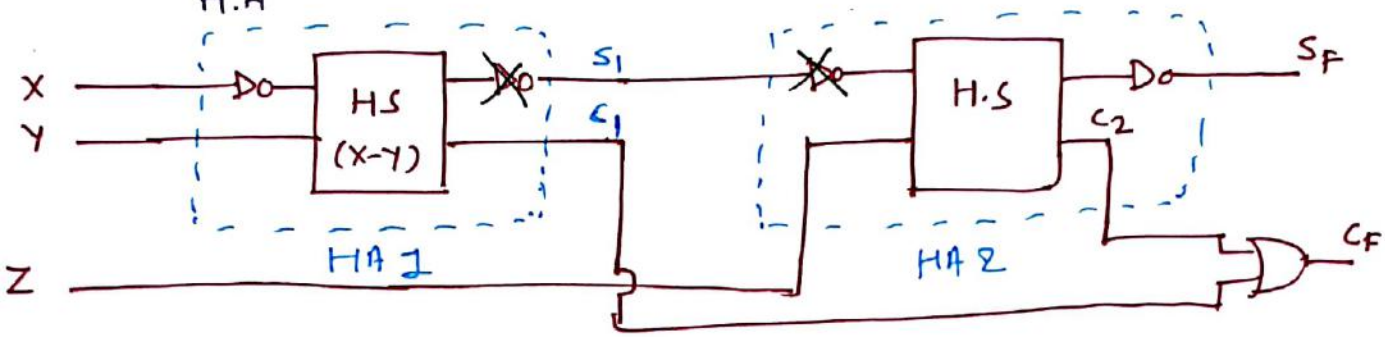
Full Adder using Half Subtractor

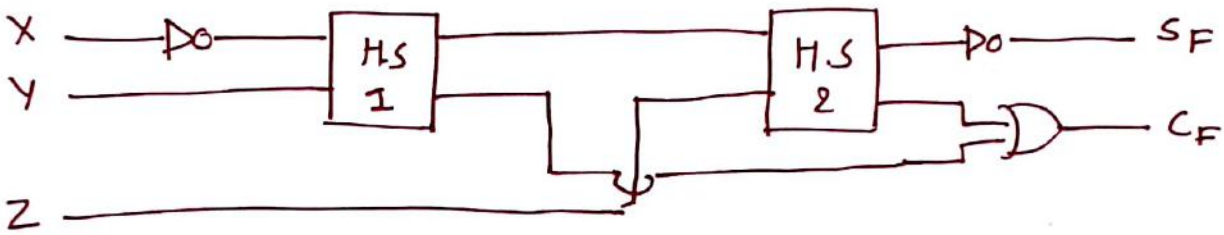
97



H.A







Engineering Funda YouTube Channel

Full Subtractor

98

→ Full Subtractor performs 3 bits Subtraction.

$(A - B - C)$ or $A - (B + C)$

A	B	C	D	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

D

C \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

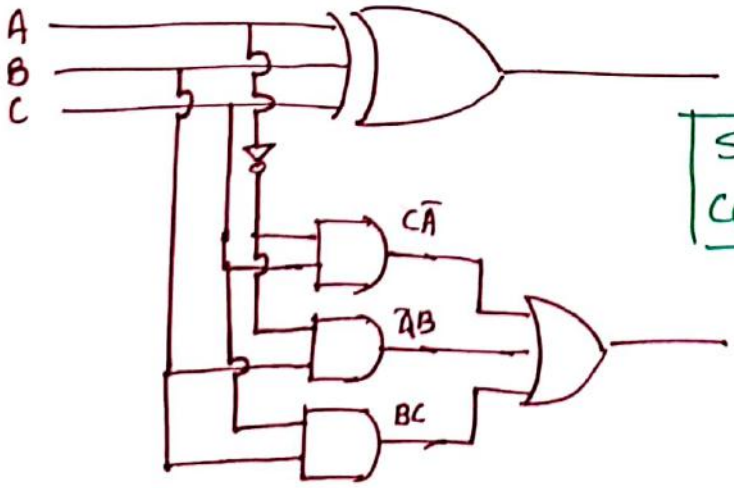
$$D = \bar{A}\bar{B}C + \bar{A}B\bar{C} + ABC + A\bar{B}\bar{C}$$
$$= \bar{A}(\bar{B}C + B\bar{C}) + A(BC + \bar{B}\bar{C})$$
$$= \bar{A}(B \oplus C) + A(B \odot C)$$
$$= A \oplus B \oplus C$$

B_o

C \ AB	00	01	11	10
0	0	1	0	0
1	1	1	1	0

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

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



$D = A \oplus B \oplus C$

Sum = $A \oplus B \oplus C$	Full Adder.
Carry = $AB + BC + CA$	

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

$(A - B - C)$

Engineering Funda YouTube Channel

Full Adder using Full Subtractor

→ Full Adder.

$[A + B + C]$

$S = A \oplus B \oplus C$

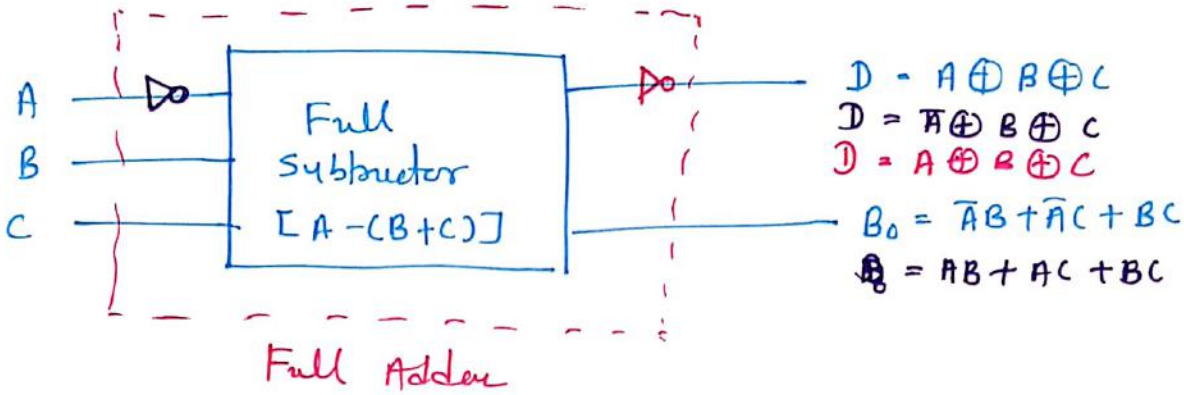
$C_0 = AB + BC + CA$

→ Full Subtractor

$[A - (B + C)]$

$D = A \oplus B \oplus C$

$B_0 = \bar{A}B + \bar{A}C + BC$



Scanned with CamSca

Full subtractor using Half Subtractor

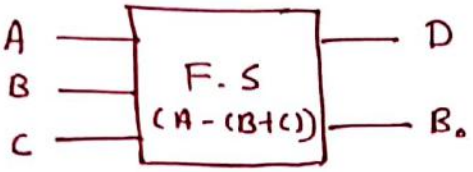
→ Full subtractor

[A, B, C]

[A − (B + C)]

→ $D = A \oplus B \oplus C$

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



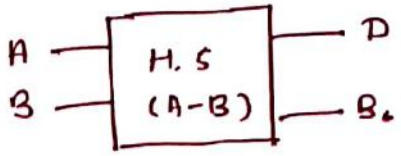
→ Half subtractor

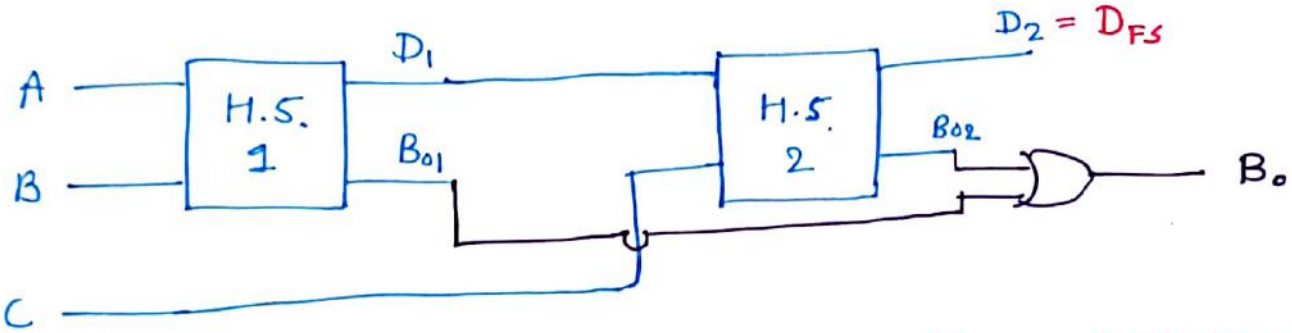
[A, B]

[A − B]

→ $D = A \oplus B$

$B_o = \bar{A}B$





H.S.1

$D_1 = A \oplus B$

$B_{01} = \bar{A}B$

H.S.2

$D_2 = D_1 \oplus C = A \oplus B \oplus C$

$B_{02} = \overline{D_1} \cdot C$

→ $B_o = B_{01} + B_{02}$

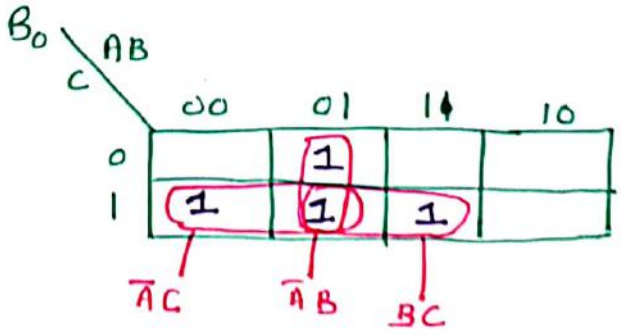
$= \bar{A}B + \overline{D_1} \cdot C$

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

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

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

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



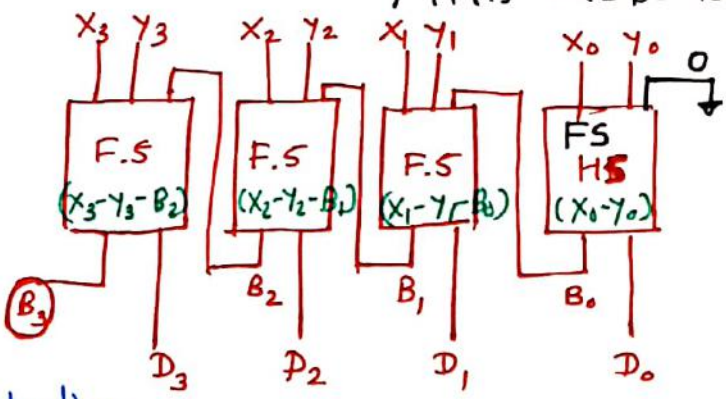
Engineering Funda YouTube Channel

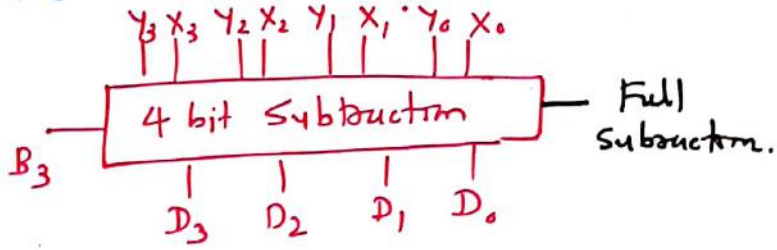
Parallel Subtractor using Full Subtractor / Half subtractor.

	B_3	B_2	B_1	B_0	0
X		X_3	X_2	X_1	X_0
$-\ Y$		Y_3	Y_2	Y_1	Y_0
$B_0\ D$	B_3	D_3	D_2	D_1	D_0

$(X-Y-B)$
Full Subtractor.

(X_0-Y_0)
Half Subtraction

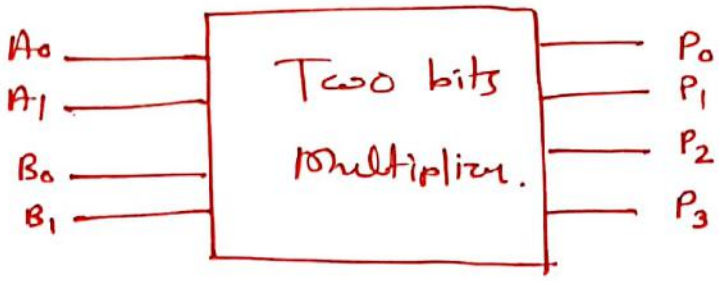




102

Engineering Funda YouTube Channel

2 Bits Multiplier using Half Adder 103



$$\begin{array}{r} 11 \\ 10 \\ \hline 0110 \end{array}$$

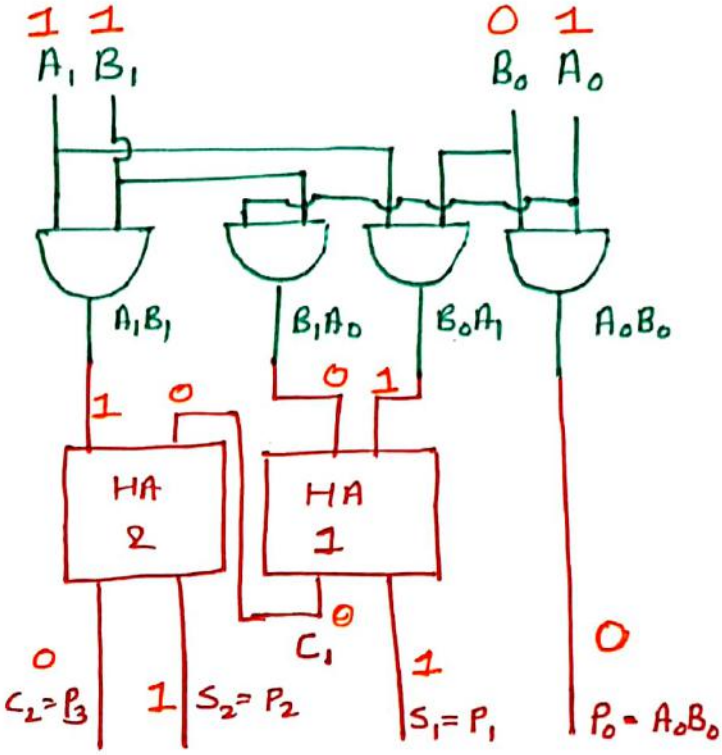
		A_1	A_0
		B_1	B_0
	C_1	B_0A_1	B_0A_0
C_2	A_1B_1	B_1A_0	X
P_3	P_2	P_1	P_0

→ $P_0 = B_0A_0$

$P_1 = HA_1[B_0A_1, B_1A_0]$

$P_2 = HA_2[C_1, A_1B_1]$

$P_3 = C_2$



Excess-3 Addition by Parallel Adder

Step-1 – take two Excess-3 data A, B

Step-2 – Add Two data A & B

$A + B = S$ (carry C)

Step-3 – Check

If $C = 0$

Ans = $S - 3$

$-3 \rightarrow -0011$

$\downarrow$ 1's Comp

1100

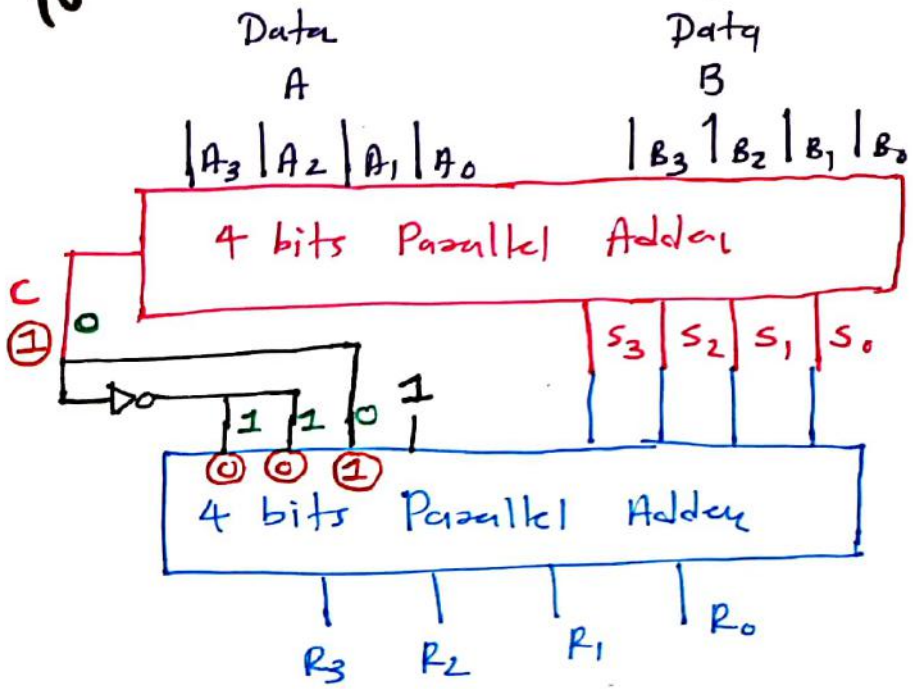
$+1 \downarrow$ 2's Comp

$\boxed{1101}$

If $C = 1$

Ans = $S + 3$

$+3 \rightarrow 0011$

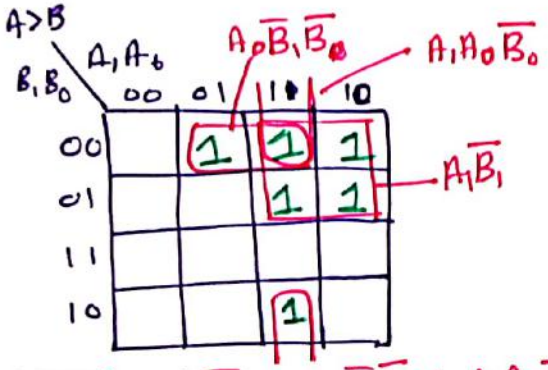


Engineering Funda YouTube Channel

2 bits Comparator 105

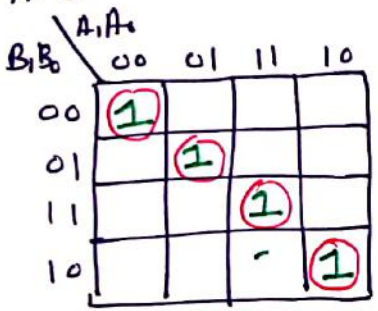
A_1	A_0	B_1	B_0		A>B	A=B	A<B
0	0	0	0	→	0	1	0
0	0	0	1	→	0	0	1
0	0	1	0	→	0	0	1
0	0	1	1	→	0	0	1
0	1	0	0	→	1	0	0
0	1	0	1	→	0	1	0
0	1	1	0	→	0	0	1
0	1	1	1	→	0	0	1
1	0	0	0	→	1	0	0
1	0	0	1	→	1	0	0
1	0	1	0	→	0	1	0
1	0	1	1	→	0	0	1
1	1	0	0	→	1	0	0
1	1	0	1	→	1	0	0
1	1	1	0	→	1	0	0
1	1	1	1	→	0	1	0

A>B



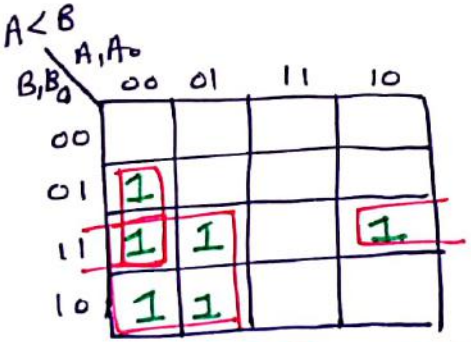
$$(A>B) = A_1\bar{B}_1 + A_0\bar{B}_1\bar{B}_0 + A_1A_0\bar{B}_0$$

A=B

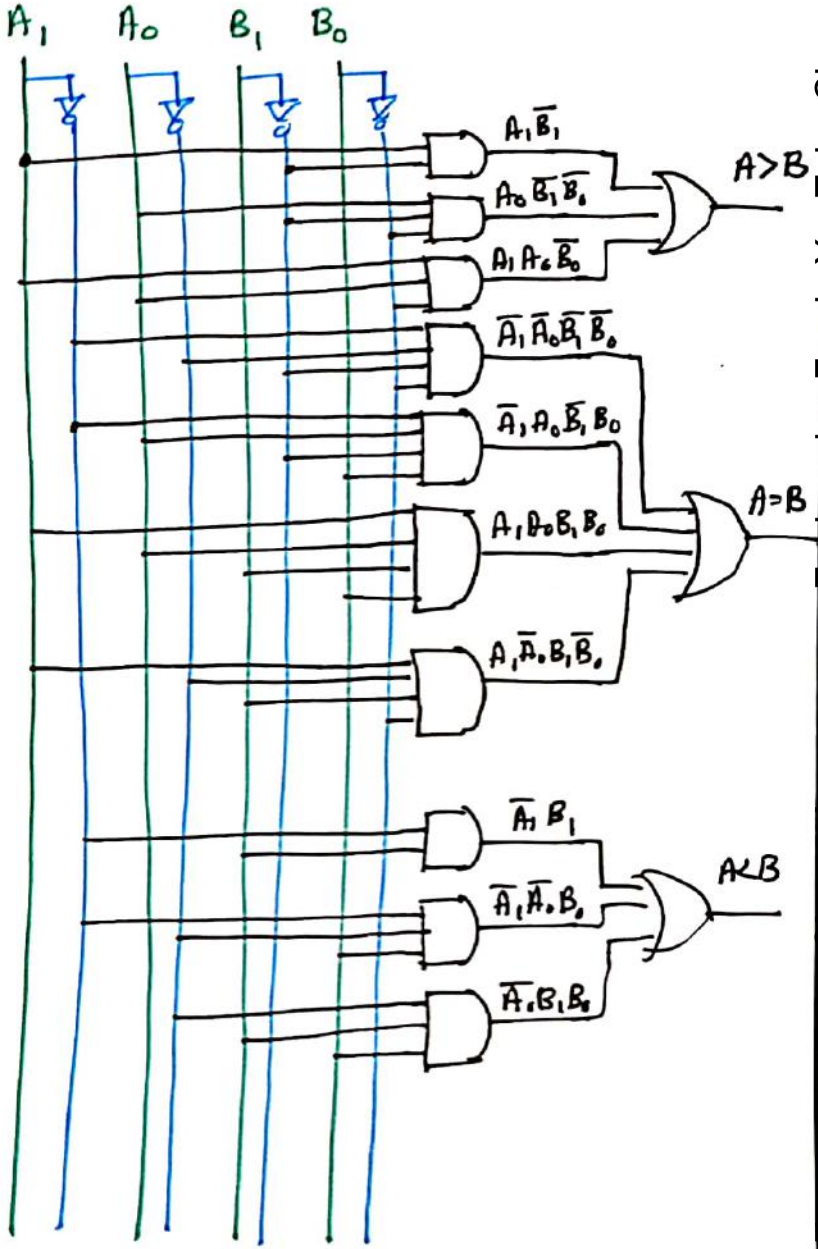


$$(A=B) = \bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 + \bar{A}_1A_0\bar{B}_1B_0 + A_1A_0B_1B_0 + A_1\bar{A}_0B_1\bar{B}_0$$

A<B



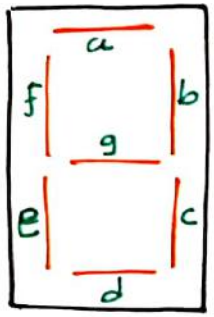
$$(A<B) = \bar{A}_1B_1 + \bar{A}_1\bar{A}_0B_0 + \bar{A}_0B_1B_0$$



Engineering Funda YouTube Channel

Seven Segment Display Decoder

106



Digit	b_3	b_2	b_1	b_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	1	0	1	0	x	x	x	x	x	x	x
other	1	1	1	1	x	x	x	x	x	x	x

a

b_1b_0 \ b_3b_2	00	01	11	10
00	1_0	0_4	X_{12}	1_8
01	0_1	1_5	X_{13}	1_9
11	1_3	1_7	X_{15}	X_{11}
10	1_2	1_6	X_{14}	X_{10}

$a = b_1 + b_3 + b_0b_2 + \overline{b_0}\,\overline{b_2}$

b

b_1b_0 \ b_3b_2	00	01	11	10
00	1_0	1_4	X_{12}	1_8
01	1_1	0_5	X_{13}	1_9
11	1_3	1_7	X_{15}	X_{11}
10	1_2	0_6	X_{14}	X_{10}

$b = \overline{b_2} + \overline{b_1}\,\overline{b_0} + b_1b_0$

c

b_1b_0 \ b_3b_2	00	01	11	10
00	1_0	1_4	X_{12}	1_8
01	1_1	1_5	X_{13}	1_9
11	1_3	1_7	X_{15}	X_{11}
10	0_2	1_6	X_{14}	X_{10}

$c = \overline{b_1} + b_0 + b_2$

D

b_1b_0 \ b_3b_2	00	01	11	10
00	1_0	0_4	X_{12}	1_8
01	0_1	1_5	X_{13}	1_9
11	1_3	0_7	X_{15}	X_{11}
10	1_2	1_6	X_{14}	X_{10}

$d = b_3 + b_1\overline{b_0} + b_1\overline{b_2} + \overline{b_1}b_0b_2 + \overline{b_0}\,\overline{b_2}$

e

b_1b_0 \ b_3b_2	00	01	11	10
00	1_0	0_4	X_{12}	1_8
01	0_1	0_5	X_{13}	0_9
11	0_3	0_7	X_{15}	X_{11}
10	1_2	1_6	X_{14}	X_{10}

$e = b_1\overline{b_0} + \overline{b_0}\,\overline{b_2}$

f

b_0b_1 \ b_3b_2	00	01	11	10
00	1_0	1_4	X_{12}	1_8
01	0_1	1_5	X_{13}	1_9
11	0_3	0_7	X_{15}	X_{11}
10	0_2	1_6	X_{14}	X_{10}

$f = b_3 + \overline{b_0}b_2 + \overline{b_0}\,\overline{b_1} + \overline{b_1}\,\overline{b_2}$

g

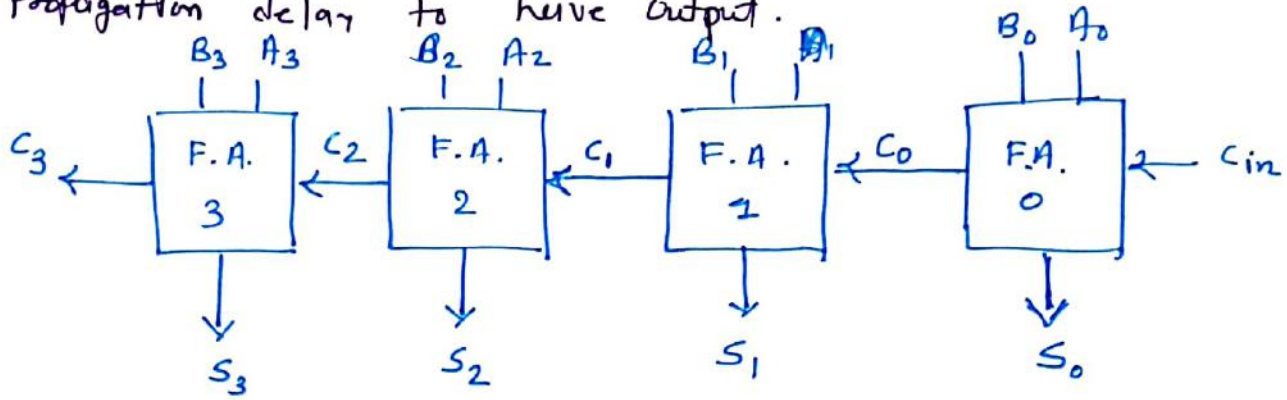
b_1b_0 \ b_3b_2	00	01	11	10
00	0_0	1_4	X_{12}	1_8
01	0_1	1_5	X_{13}	1_9
11	1_3	0_7	X_{15}	X_{11}
10	1_2	1_6	X_{14}	X_{10}

$g = b_3 + \overline{b_1}b_2 + b_1\overline{b_0} + b_1\overline{b_2}$

Engineering Funda YouTube Channel

Carry Look Ahead Adder [CLA Adder]

- CLA Adder is better then Full Adder in terms of Speed.
- By Full Adder when we have parallel Adder, it takes Propagation delay to have output.



A_0	B_0	C_{in}	C_0	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	1	$A_0\bar{B}_0C_{in}$
1	1	0	1	A_0B_0
1	1	1	1	

$\rightarrow C_0 = A_0B_0 + A_0\bar{B}_0C_{in} + \bar{A}_0B_0C_{in}$

$= A_0B_0 + C_{in}(A_0\bar{B}_0 + \bar{A}_0B_0)$

$= A_0B_0 + C_{in}(A_0 \oplus B_0)$

$\rightarrow C_i = \underbrace{A_iB_i}_{\text{Carry generator } G_i} + C_{i-1}\underbrace{(A_i \oplus B_i)}_{\text{Carry Propagator } P_i}$

$\rightarrow \boxed{C_i = [G_i + C_{i-1}P_i]}$

$\rightarrow i = 0$

$C_0 = G_0 + C_{-1}P_0$

$\rightarrow i = 1$

$C_1 = G_1 + C_0P_1 = G_1 + [G_0 + C_{-1}P_0]P_1$

$= G_1 + P_1G_0 + P_1P_0C_{-1}$

$\rightarrow i = 2$

$C_2 = G_2 + C_1P_2$

$= G_2 + P_2[G_1 + P_1G_0 + P_1P_0C_{-1}]$

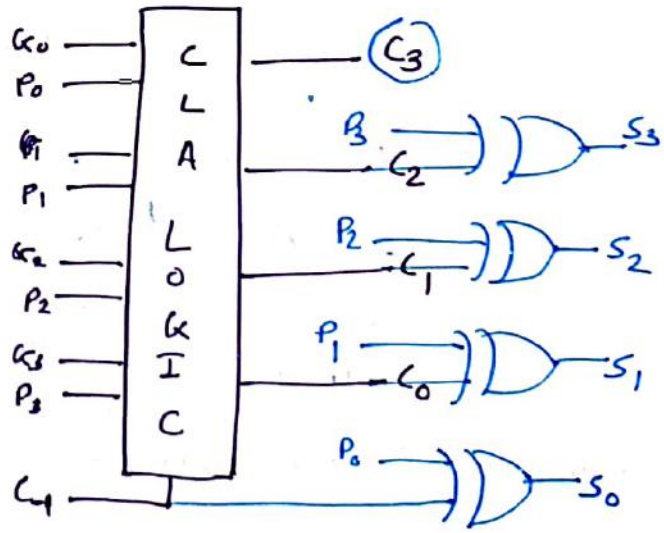
$= G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_{-1}$

$\rightarrow i = 3$

$C_3 = G_3 + C_2P_3$

$= G_3 + P_3[G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_{-1}]$

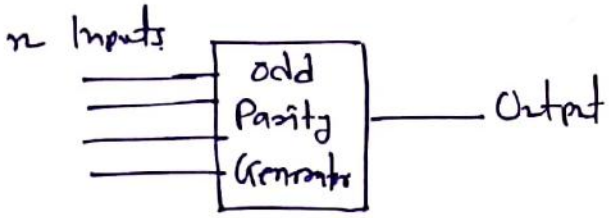
$= G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_{-1}$



Scanned with CamScanner

Even Parity Generator / Odd Parity Generator

– For even parity generator output = 1 when number of 1's at I/P is odd.



– For odd parity generator output = 1, when number of 1's at O/P is even.

b_3	b_2	b_1	b_0	O/P (Even parity) P
0	0	0	0	0 (0)
0	0	0	1	1 (1)
0	0	1	0	1 (2)
0	0	1	1	0 (3)
0	1	0	0	1 (4)
0	1	0	1	0 (5)
0	1	1	0	0 (6)
0	1	1	1	1 (7)
1	0	0	0	1 (8)
1	0	0	1	0 (9)
1	0	1	0	0 (10)
1	0	1	1	1 (11)
1	1	0	0	0 (12)
1	1	0	1	1 (13)
1	1	1	0	1 (14)
1	1	1	1	0 (15)

$$Q_{odd} = \overline{b_3 \oplus b_2 \oplus b_1 \oplus b_0}$$

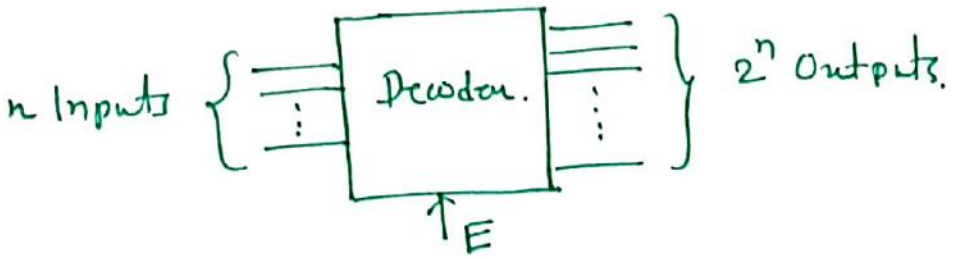
P: b_1b_0 \ b_3b_2	00	01	11	10
00	0 (0)	1 (4)	0 (12)	1 (8)
01	1 (1)	0 (5)	1 (13)	0 (9)
11	0 (3)	1 (7)	0 (15)	1 (11)
10	1 (2)	0 (6)	1 (14)	0 (10)

$$P(b_0b_1b_2b_3) = b_0 \oplus b_1 \oplus b_2 \oplus b_3$$

Engineering Funda YouTube Channel

Decoder Basics and 2X4 Decoder

- Decoder decodes n Inputs to 2^n Outputs.

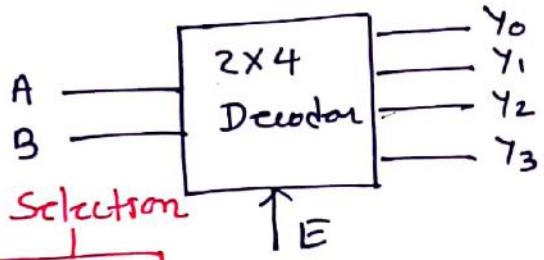


E = Enable.

If E = 0, Decoder is desabled.

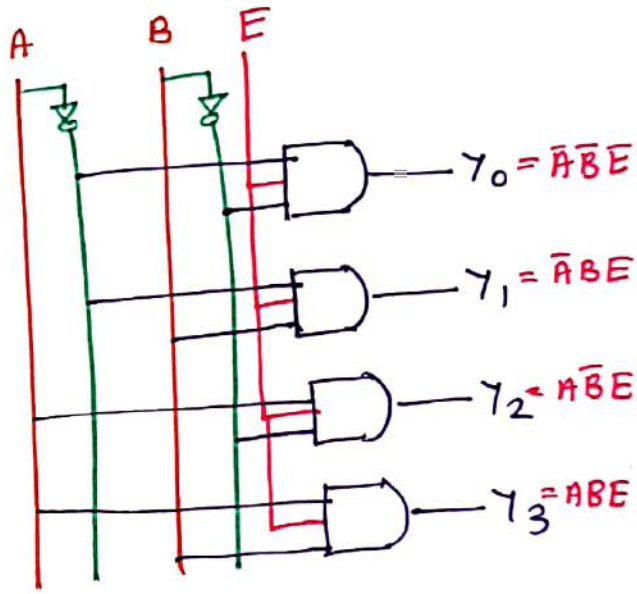
If E = 1, Decoder is Enabled.

- 2X4 Decoder.



Selection

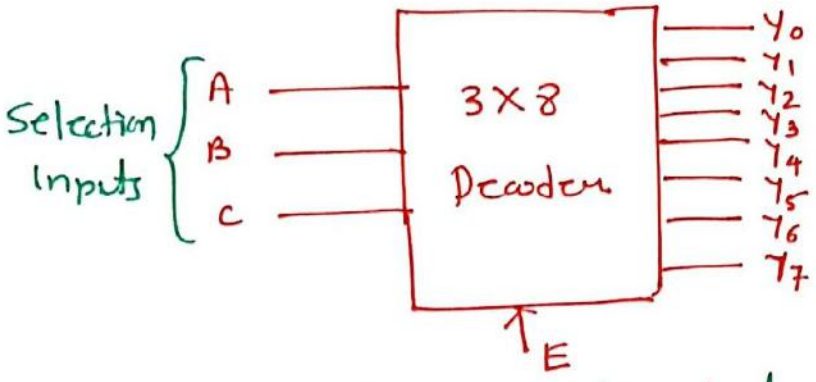
E	A	B	Y_0	Y_1	Y_2	Y_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



Engineering Funda YouTube Channel

3 X 8 Decoder

110



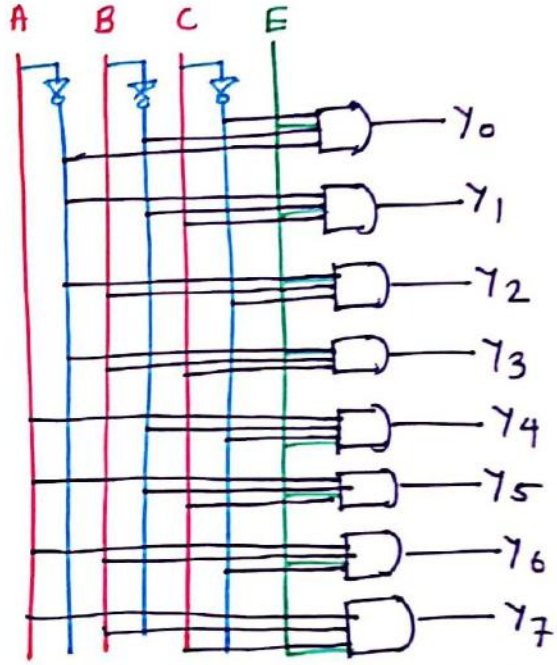
E is enable terminal

E = 0 , Decoder is desabled.

E = 1 , Decoder is Enabled.

E	A	B	C	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	X	X	X	0	0	0	0	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	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

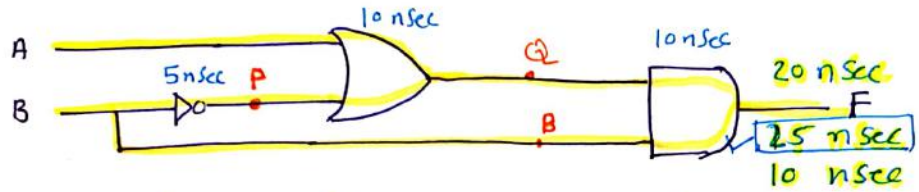
Selection Inputs — Outputs.



$Y_7 = ABCE$

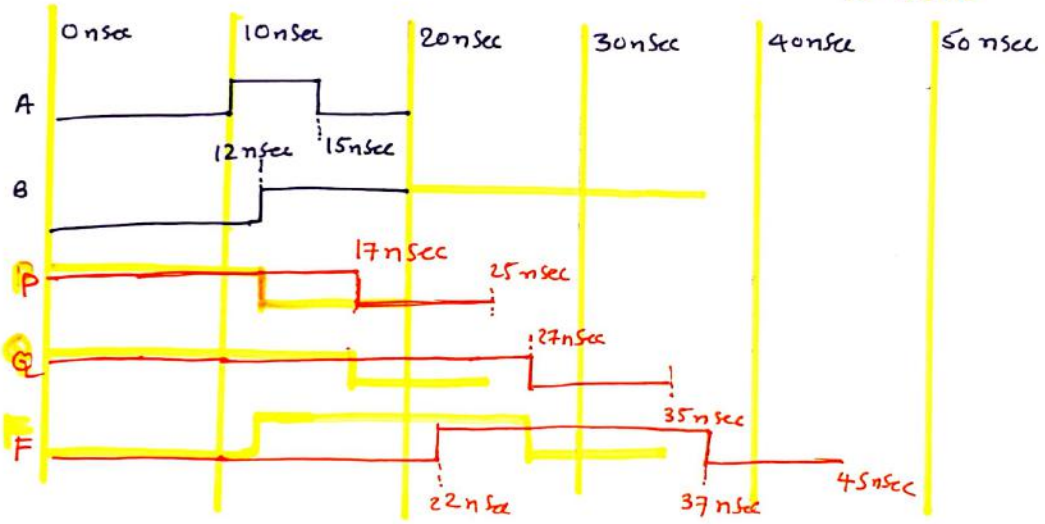
Engineering Funda YouTube Channel

Combinational Circuit Output waveform with delay at gates. [Part 1]



1] Find output waveform F.

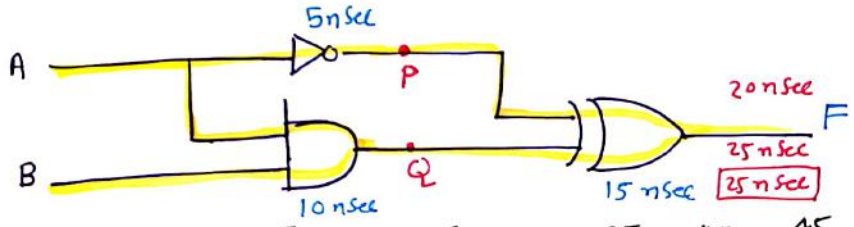
2] Find total delay of given combinational circuit.



Scanned with CamSca

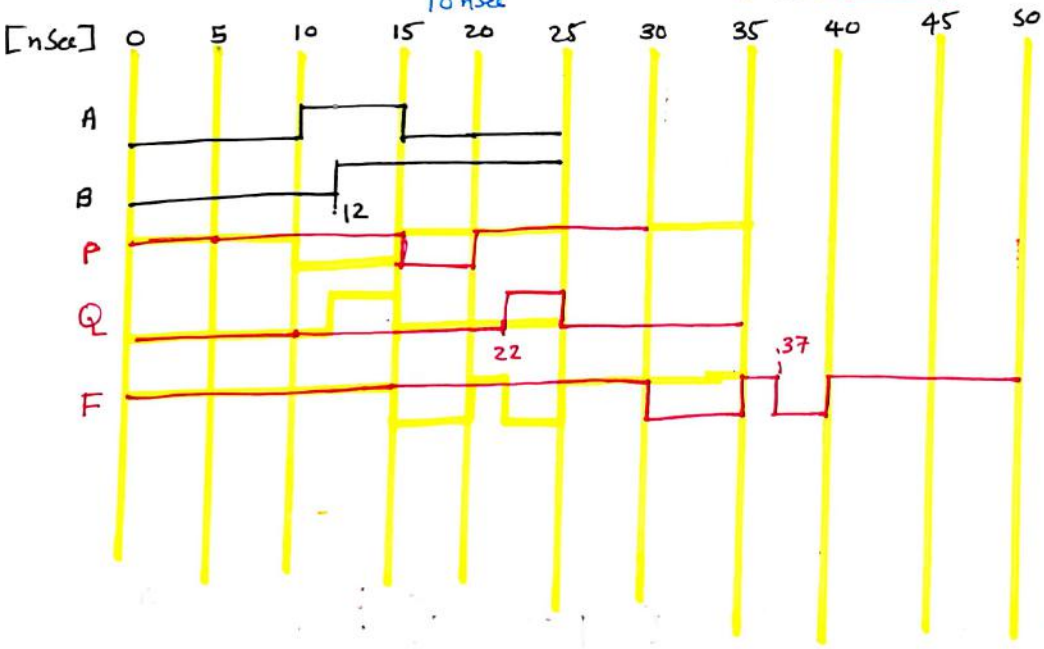
Combinational Circuit Output waveform with delay at gates [Part 2].

112



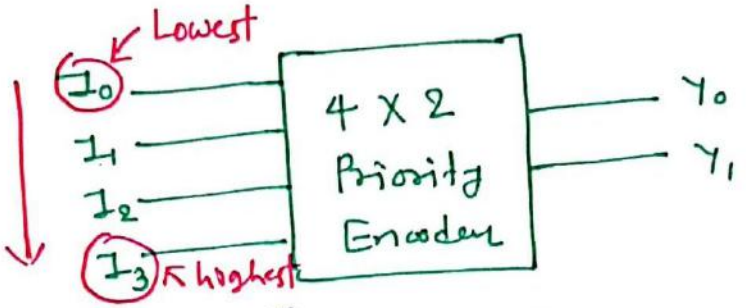
1] Find output waveform F.

2] Find total delay of given Combinational circuit.



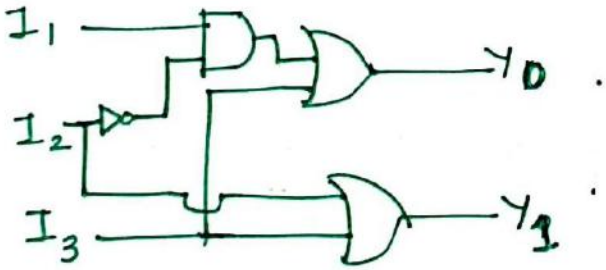
Scanned with CamSca

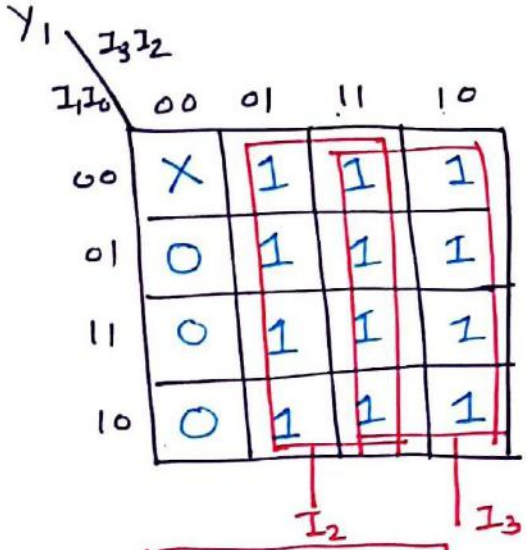
Priority Encoder 11E



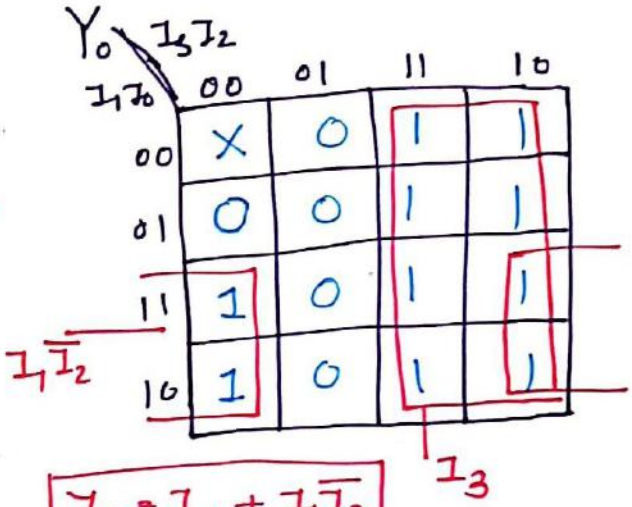
$\rightarrow n = 4 = 2^m \Rightarrow m = 2$

I_3	I_2	I_1	I_0	Y_1	Y_0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1





$$\boxed{Y_1 = I_3 + I_2}$$

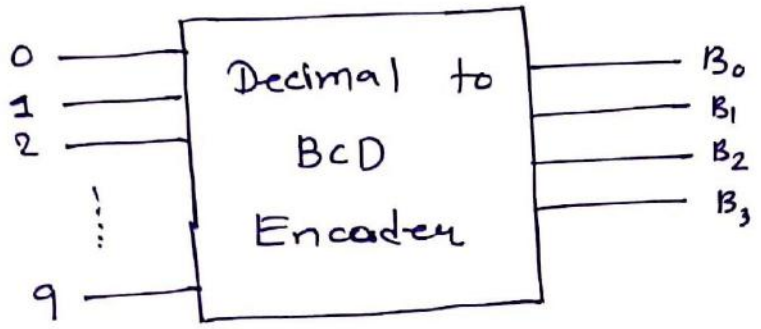


$$\boxed{Y_0 = I_3 + I_1\overline{I_2}}$$

Engineering Funda YouTube Channel

Decimal to BCD Encoder

→ Decimal [0 to 9] to BCD [0000 to 1001] Encoder.



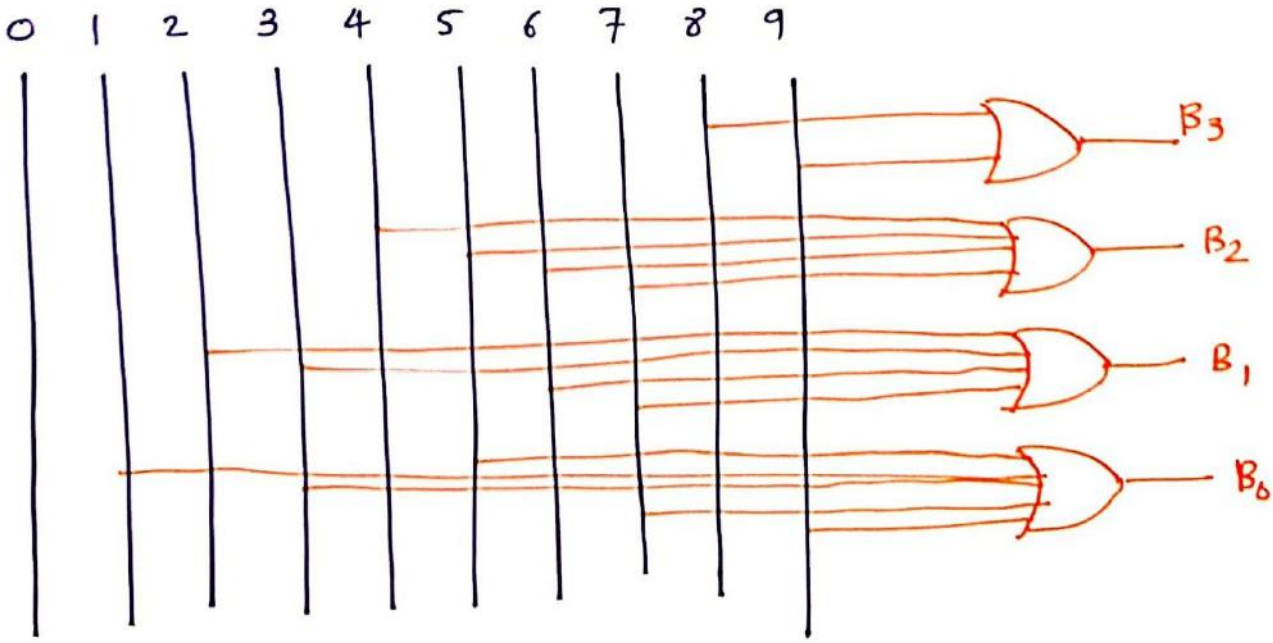
Input Decimal Number	O/P B_3	B_2	B_1	B_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

$B_3 = 8 + 9$

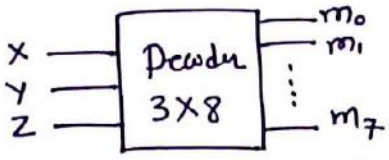
$B_2 = 4 + 5 + 6 + 7$

$B_1 = 2 + 3 + 6 + 7$

$B_0 = 1 + 3 + 5 + 7 + 9$

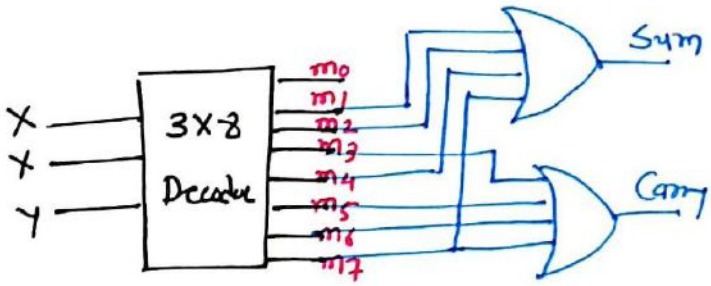


Full Adder using Decoder



$m = 2^n = 2^3 = 8$

X	Y	Z	S	C	o/p
0	0	0	0	0	m_0
0	0	1	1	0	m_1
0	1	0	1	0	m_2
0	1	1	0	1	m_3
1	0	0	1	0	m_4
1	0	1	0	1	m_5
1	1	0	0	1	m_6
1	1	1	1	1	m_7

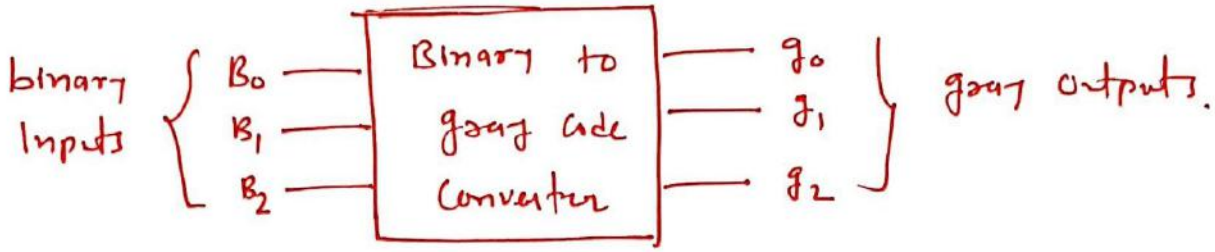


$S = m_1 + m_2 + m_4 + m_7$

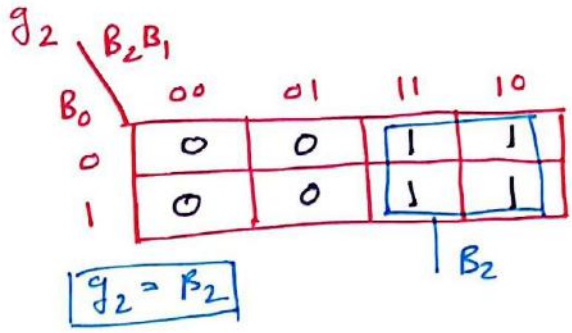
$C = m_3 + m_5 + m_6 + m_7$

Engineering Funda YouTube Channel

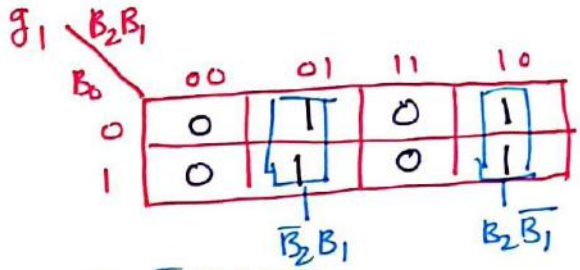
Binary to Gray Code Converter



Inputs			Outputs		
B_2	B_1	B_0	g_2	g_1	g_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

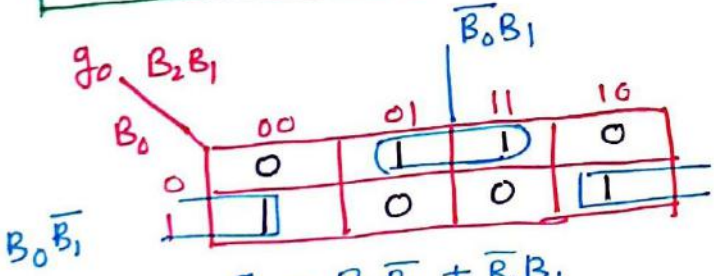


$$\boxed{g_2 = B_2}$$



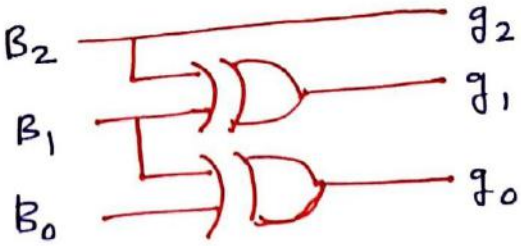
$$g_1 = \overline{B_2}B_1 + B_2\overline{B_1}$$

$$\boxed{g_1 = B_2 \oplus B_1}$$



$$g_0 = B_0\overline{B_1} + \overline{B_0}B_1$$

$$\boxed{g_0 = B_0 \oplus B_1}$$



Engineering Funda YouTube Channel

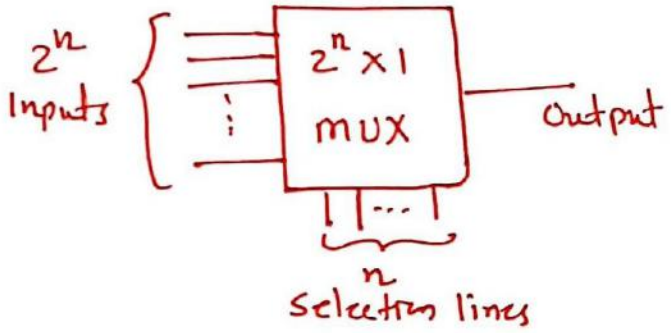
Multiplexer [MUX]

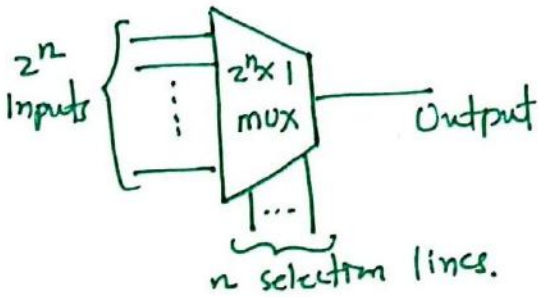
→ $[2^n \times 1]$ MUX

2^n ↑ Inputs

1 ↑ Output

→ n is number of selection lines.





→ MUX IC's ⇝ MSI scale IC's.

Advantages

1) Complexity is less
2) Cost is less
3) less wiring.

Applications

1) Data Selector
2) Switch
3) combinational circuits.

Types → $[2^n \times 1]$ MUX

[2×1] MUX	[4×1] MUX	[8×1] MUX	[16×1] MUX
$n=1$	$n=2$	$n=3$	$n=4$

[32×1] MUX
$n=5$

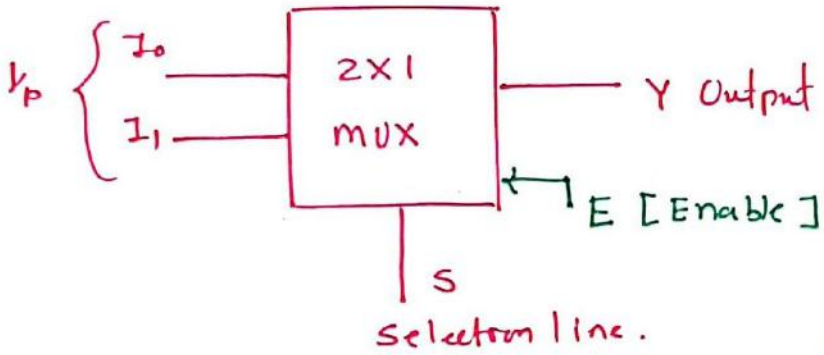
Engineering Funda YouTube Channel

2 × 1 Multiplexer

→ [2^n × 1] MUX

→ n = 1 , 1 selection line

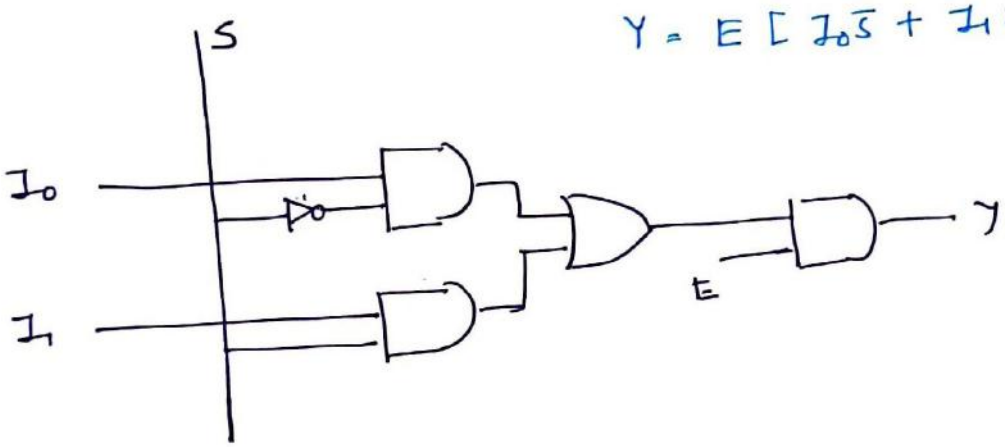
→ 2 Inputs , 1 Output .



E	S	Y
0	X	0
1	0	I_0
1	1	I_1

$$Y = I_0 \bar{S} E + I_1 S E$$

$$Y = E [I_0 \bar{S} + I_1 S]$$

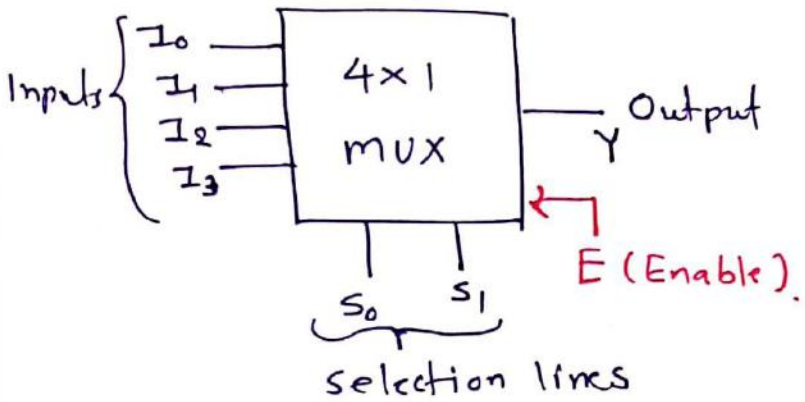


Engineering Funda YouTube Channel

4x1 mux

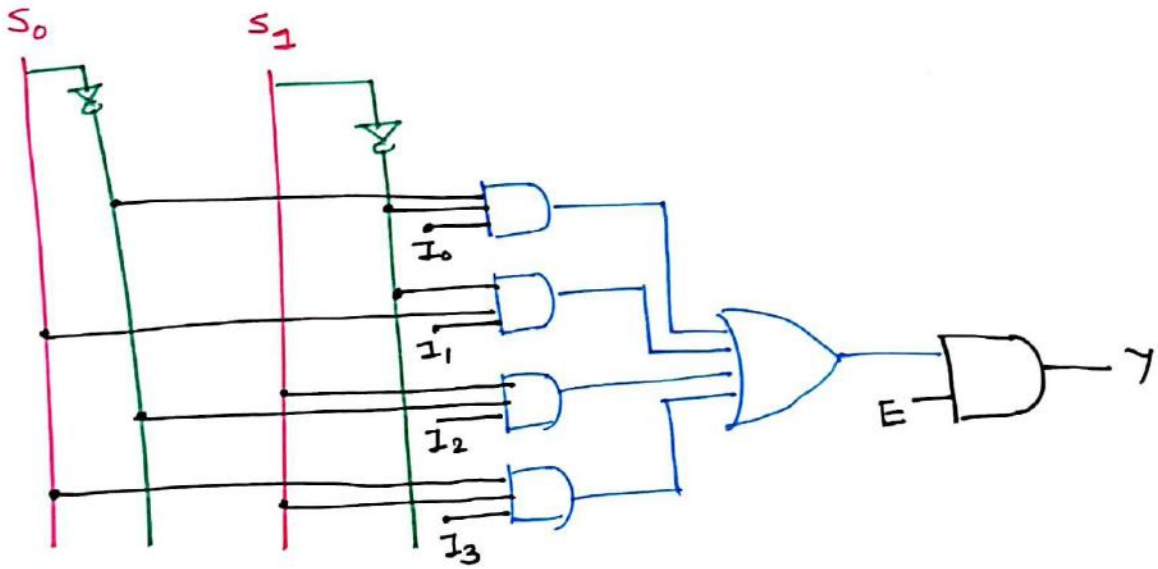
→ $[2^n \times 1]$ mux

⇒ $n = 2$, 2 selection lines.



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

$$Y = E.[\bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3]$$



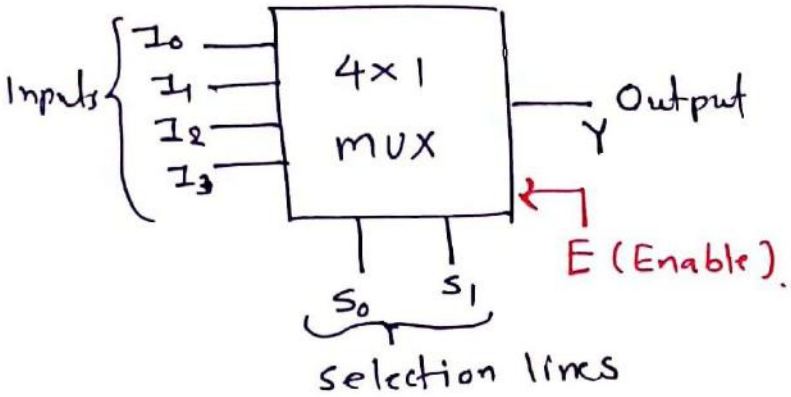
Engineering Funda YouTube Channel
Scanned with CamSca

4x1 mux

119

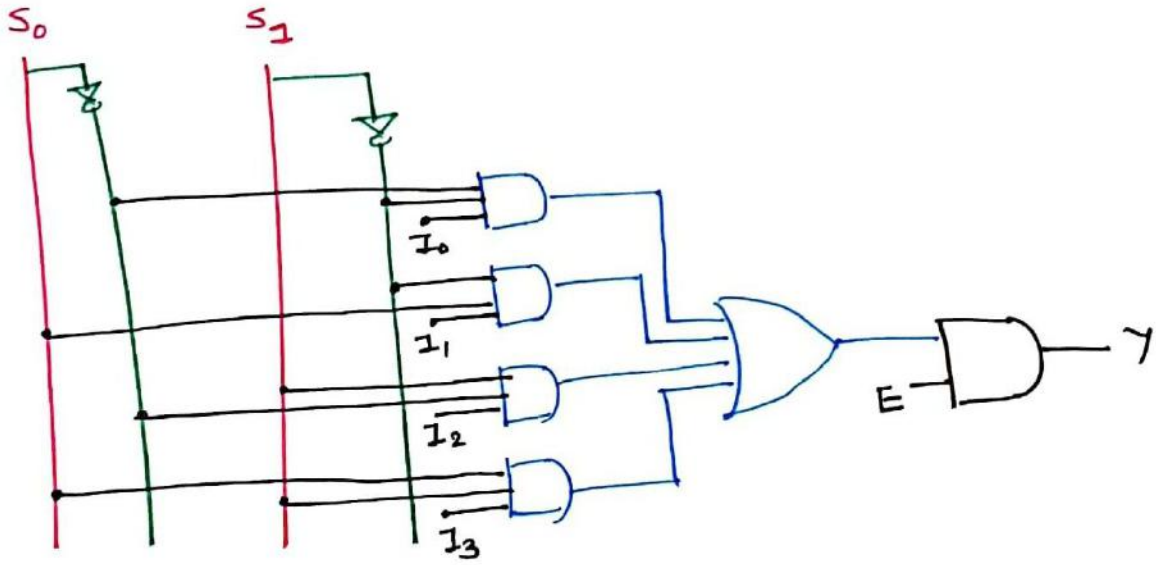
→ $[2^n \times 1]$ mux

⇒ $n = 2$, 2 selection lines.



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

$$Y = E.[\bar{S_1}\bar{S_0}I_0 + \bar{S_1}S_0I_1 + S_1\bar{S_0}I_2 + S_1S_0I_3]$$



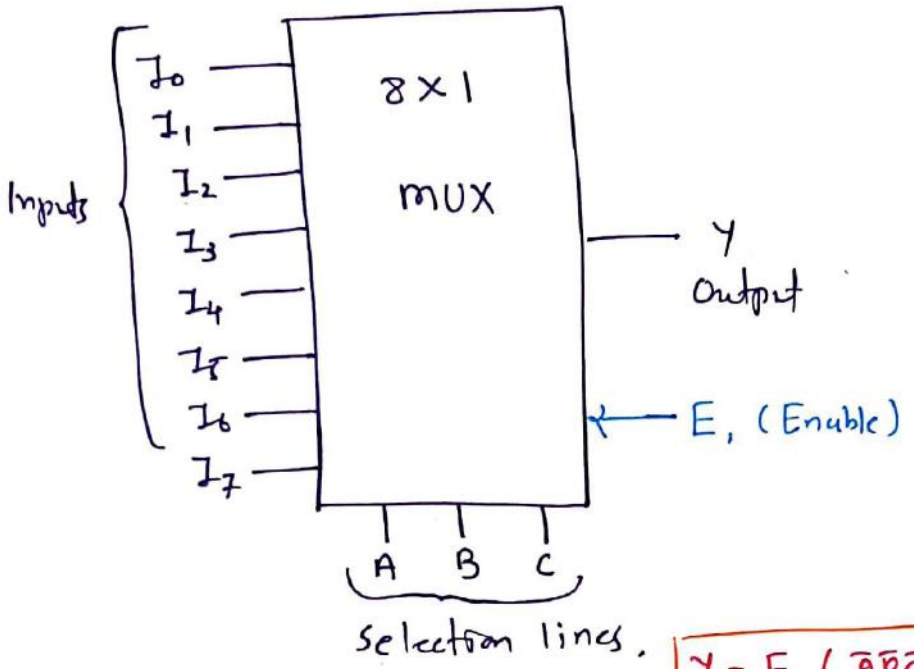
Engineering Funda YouTube Channel

8 X 1 mux

120

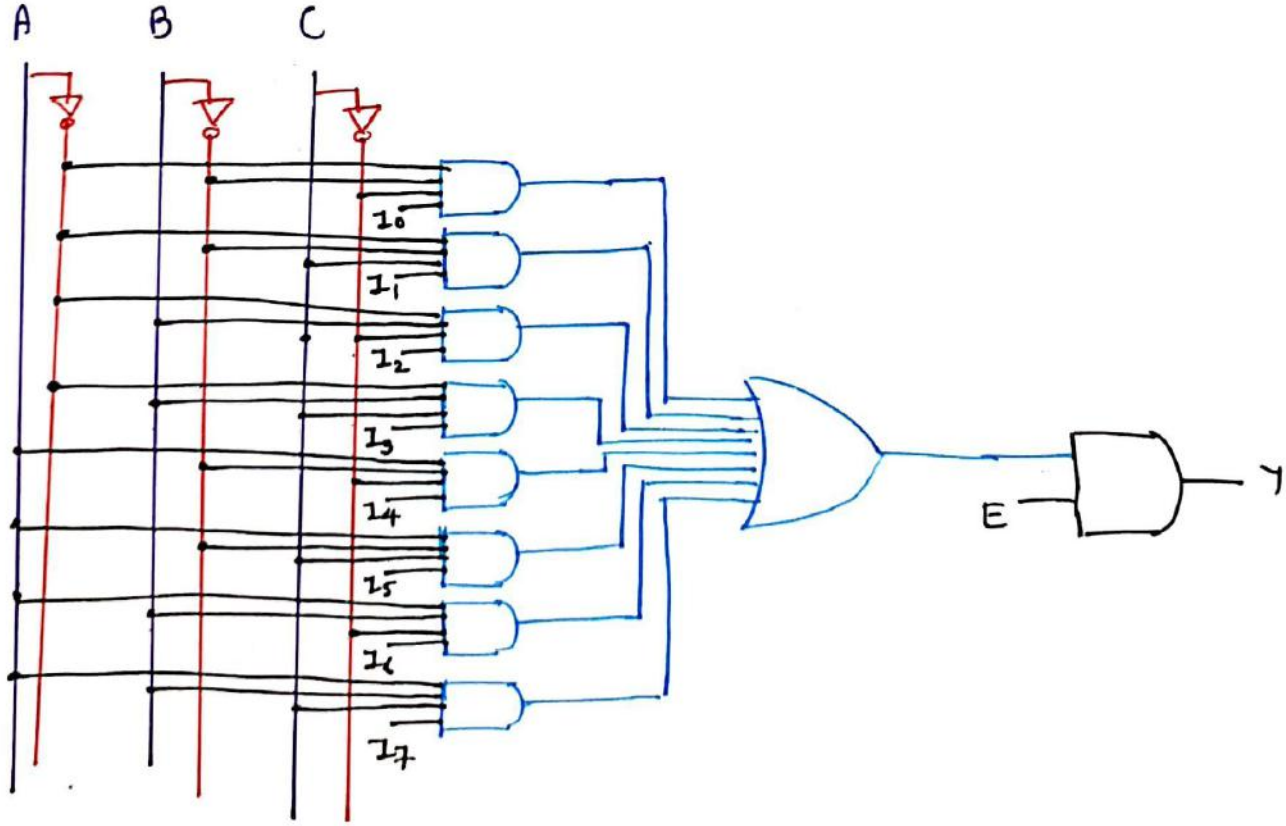
→ $[2^n \times 1]$ mux

→ $n = 3$, 3 selection lines.



E	A	B	C	Y
0	X	X	X	0
1	0	0	0	I_0
1	0	0	1	I_1
1	0	1	0	I_2
1	0	1	1	I_3
1	1	0	0	I_4
1	1	0	1	I_5
1	1	1	0	I_6
1	1	1	1	I_7

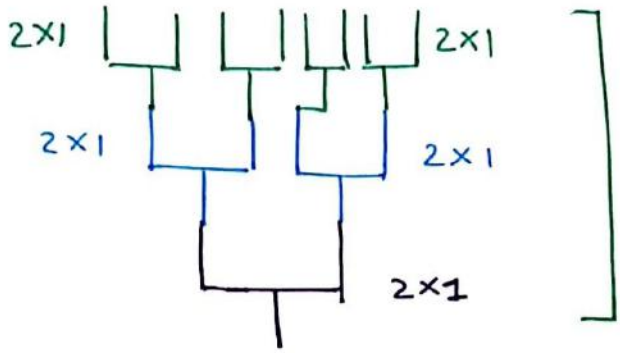
$$Y = E.(\bar{A}\bar{B}\bar{C}I_0 + \bar{A}\bar{B}CI_1 + \bar{A}B\bar{C}I_2 + \bar{A}BCI_3 + A\bar{B}\bar{C}I_4 + A\bar{B}CI_5 + AB\bar{C}I_6 + ABCI_7)$$



Engineering Funda YouTube Channel

MUX Tree

→ MUX Tree is used to obtain higher order MUX using Lower Order MUX.



8×1 MUX By 2×1 MUX

MUX Tree

1] How many 4×1 MUX is req.^d to make 32×1 MUX.

$\frac{32}{4} = 8$

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

$\frac{2}{4} = 0.5 < 1$

$= 8 + 2 + 1$
$= 11$

2] How many 2×1 MUX is req.^d to get 64×1 MUX.

$\frac{64}{2} = 32$

$\frac{32}{2} = 16$

$\frac{16}{2} = 8$

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

$\frac{4}{2} = 2$

$\frac{2}{2} = 1$

$= 32 + 16 + 8 + 4 + 2 + 1$
$= 63$

$\frac{64}{8} = 8$

$\frac{8}{8} = 1$

$= 8 + 1$
$= 9$

3] How many 8×1 MUX req^d to get 64×1 MUX.

Engineering Funda YouTube Channel

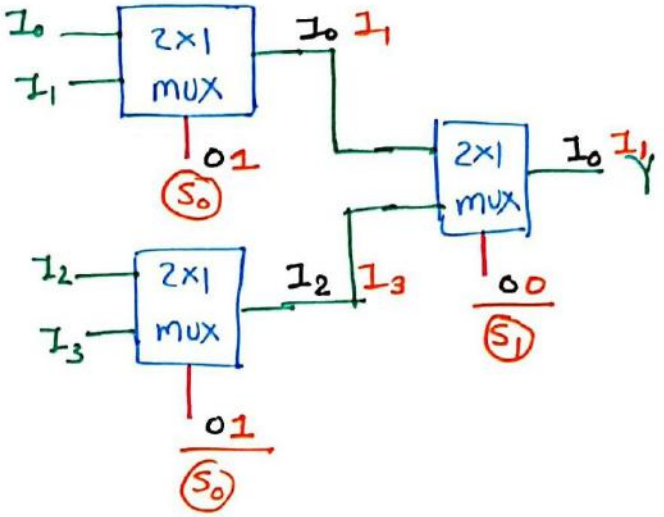
Design 4X1 MUX using 2X1 MUX. 122

→ 1st find, How many 2X1 MUX req.d to get 4X1 MUX.

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

$\frac{2}{2} = 1$ $\quad = 3$

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



Engineering Funda YouTube Channel

Design 8X1 MUX using 2X1 MUX

123

→ 1st Find, How many 2X1 MUX is req.cd to get 8X1 MUX

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

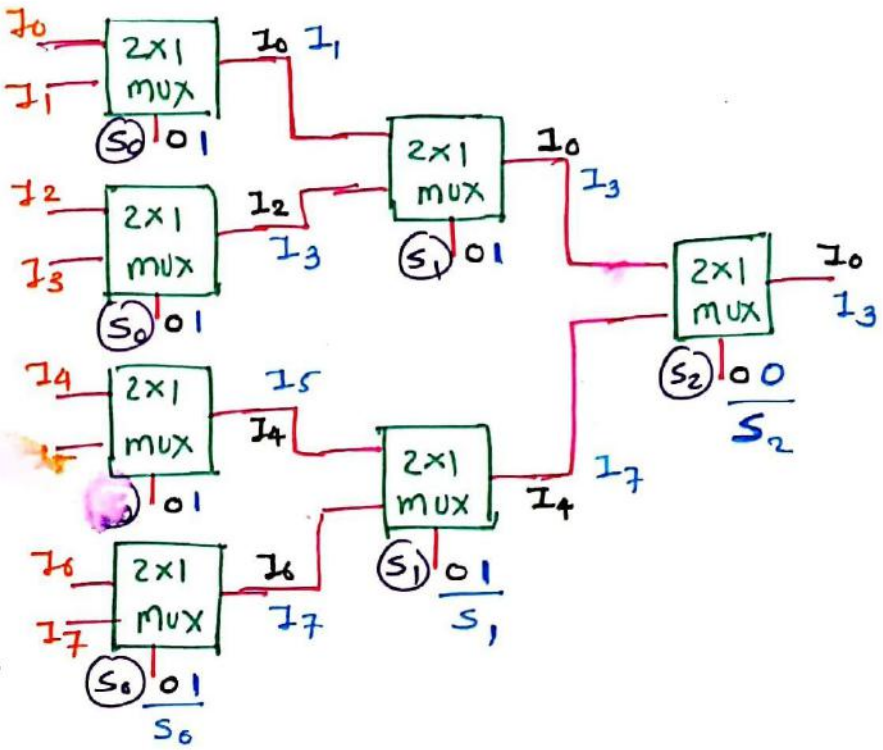
$\frac{4}{2} = 2$

$\frac{2}{2} = 1$

$= 4 + 2 + 1$

$= 7$

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



8×1 MUX using 4×1 MUX 12h

→ 1st step is to find, How many 4×1 MUX is req.d to get 8×1 MUX

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

$\frac{2}{4} = 0.5 < 1$

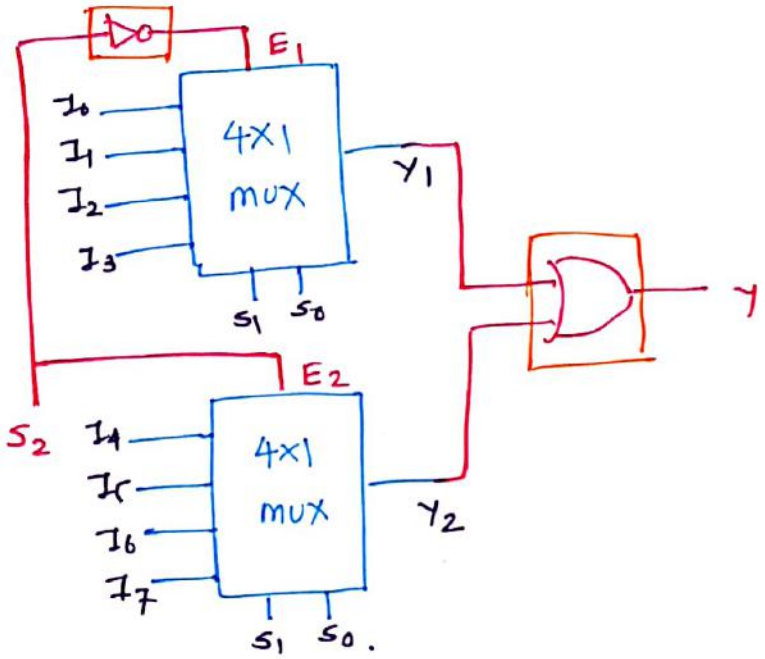
$= 2 + 0.5$

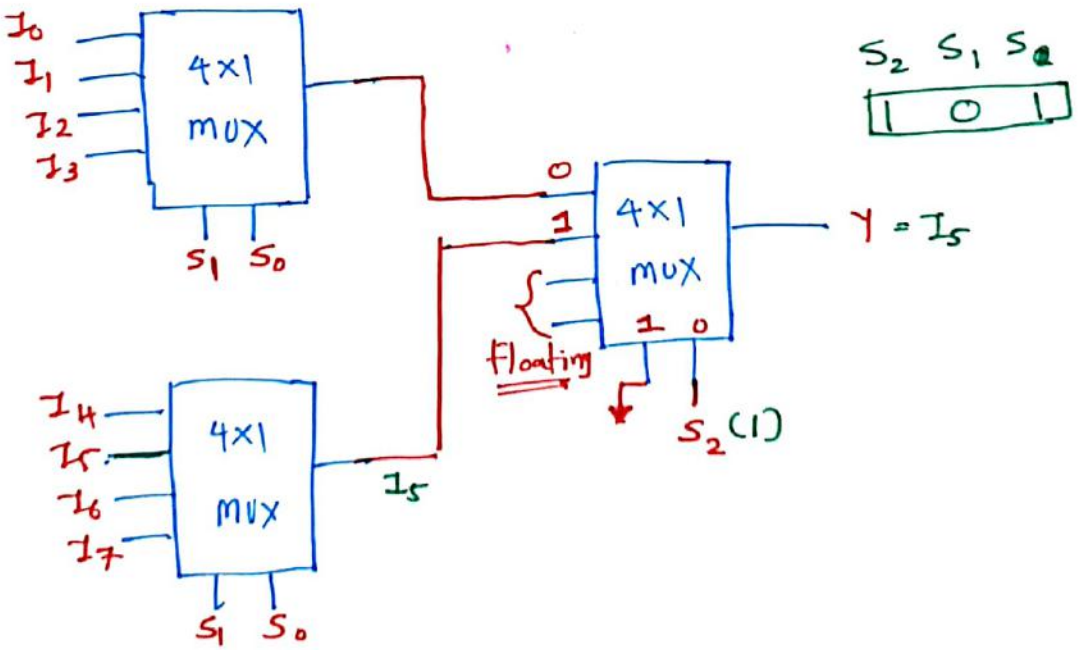
$= 2.5$

$= 3$ [without Additional gates]

→

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



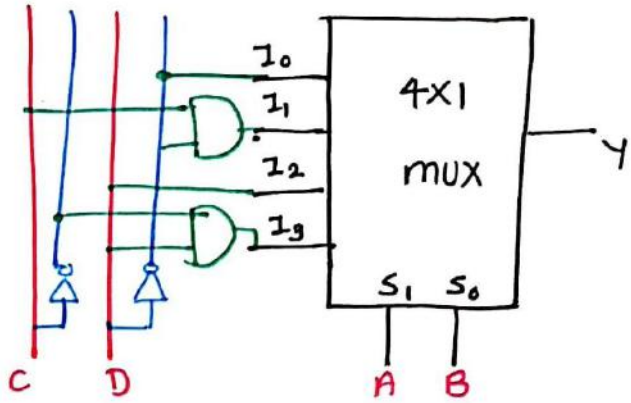


SOP Implemantation by MUX 125

Function $F(A,B,C,D) = \Sigma_m(0,2,6,9,11,13)$ make it by 4x1 MUX.

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

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



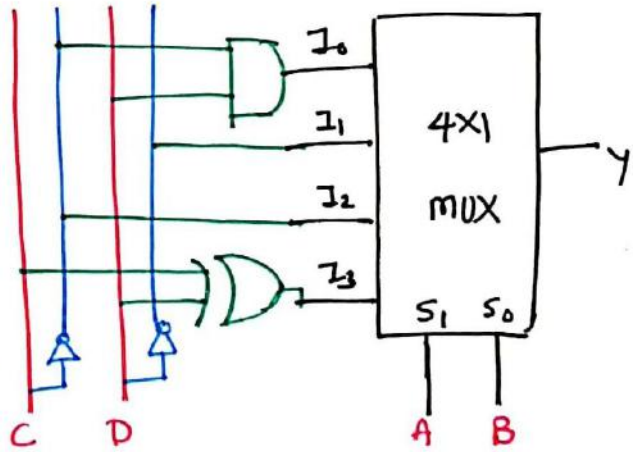
$I_0 = \overline{D}$ | $I_2 = D$

$I_1 = C\overline{D}$ | $I_3 = \overline{C}D$

Function $F(A,B,C,D) = \Sigma_m(1,4,6,8,9,13,14)$ make it by 4x1 MUX.

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

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



$I_0 = \overline{C}D$ | $I_2 = \overline{C}$

$I_1 = \overline{D}$ | $I_3 = \overline{C}D + C\overline{D}$

$= C \oplus D$

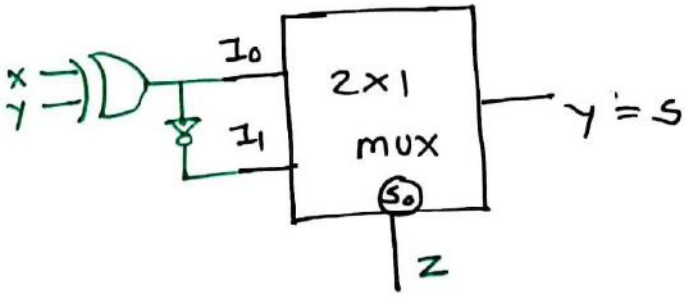
Engineering Funda YouTube Channel

Full Adder by 2X1 mux

X	Y	Z	S	C_0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S \ Z \ XY	00	01	11	10
0		1		1
1	1		1	

S_0	Y $= X \oplus Y$
0	$I_0 = \bar{X}Y + X\bar{Y}$
1	$I_1 = \bar{X}\bar{Y} + XY$ $= \overline{X \oplus Y}$

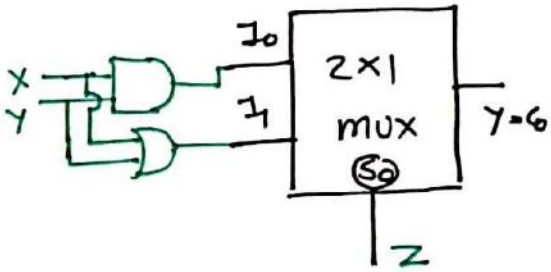


C_0 \ Z \ XY	00	01	11	10
0			1	
1		1	1	1

$I_0 = x.y$

$I_1 = Y + X$

S_0	$Y = C_0$
0	I_0
1	I_1



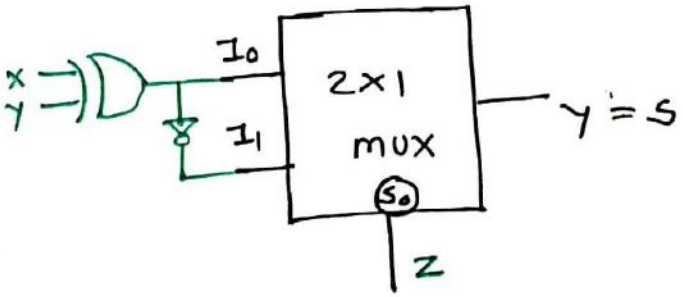
Engineering Funda YouTube Channel

Full Adder by 2X1 MUX

X	Y	Z	S	C_0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S	XY	00	01	11	10
Z	0		1		1
	1	1		1	

S_0	$Y = X \oplus Y$
0	$I_0 = \bar{X}Y + X\bar{Y}$
1	$I_1 = \bar{X}\bar{Y} + XY$
	$= \overline{X \oplus Y}$

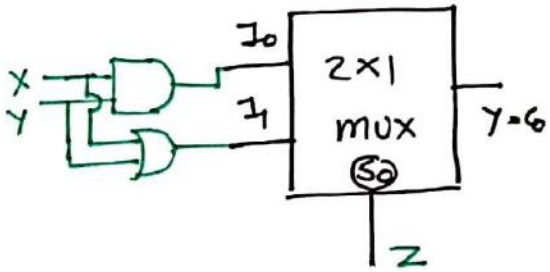


C_0	XY	00	01	11	10
Z	0			1	
	1		1	1	1

$I_0 = x.y$

$I_1 = y + x$

S_0	$Y = C_0$
0	I_0
1	I_1



Engineering Funda YouTube Channel

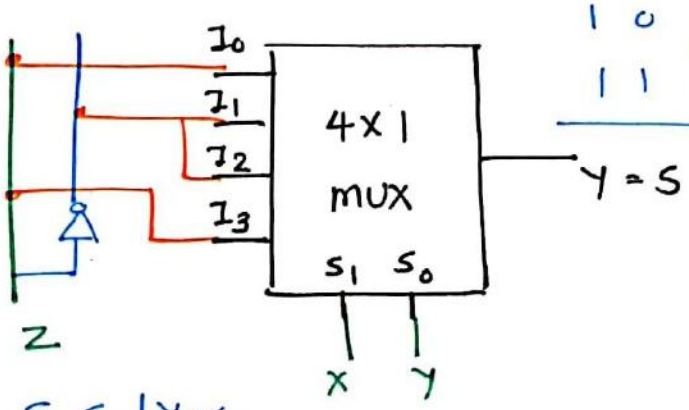
Full Adder using 4X1 MUX

X	Y	Z	S	C_0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S

Z \ XY	00	01	11	10
0		1		1
1	1		1	

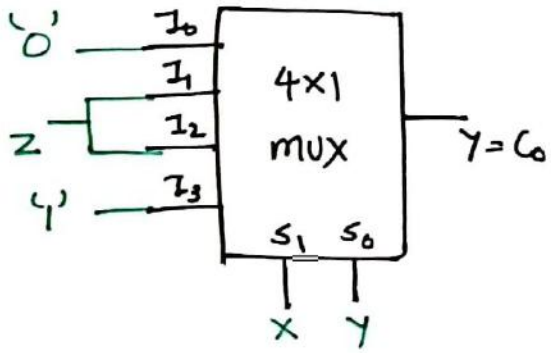
S_1	S_0	Y = S
0	0	$I_0 = Z$
0	1	$I_1 = \bar{Z}$
1	0	$I_2 = \bar{Z}$
1	1	$I_3 = Z$



C_0

Z \ XY	00	01	11	10
0			1	
1		1	1	1

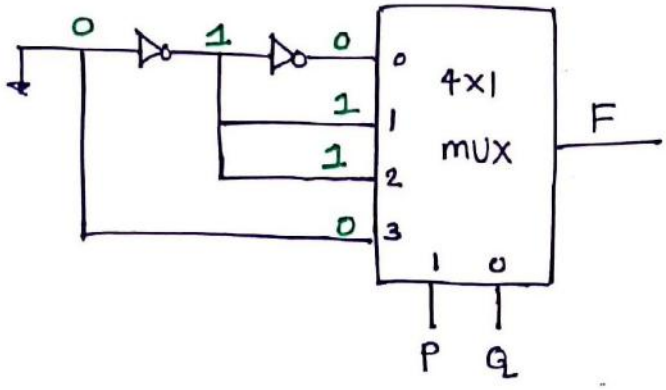
S_1	S_0	$Y = C_0$
0	0	$I_0 = 0$
0	1	$I_1 = Z$
1	0	$I_2 = Z$
1	1	$I_3 = 1$



Scanned with CamScanner

Boolean function from Multiplexer Circuit 128

1)

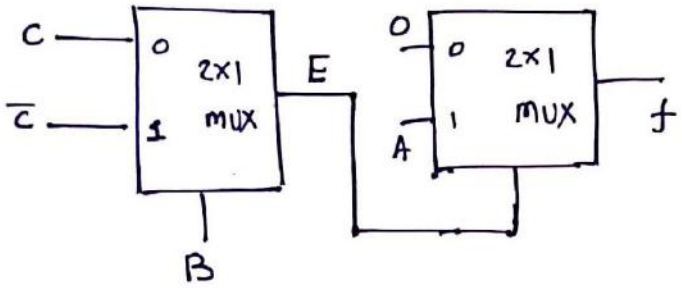


$F = \overline{P}\overline{Q}.0 + \overline{P}Q.1 + P\overline{Q}.1 + PQ.0$

$F = \overline{P}Q + P\overline{Q}$

$\boxed{F = P \oplus Q}$

2)



$\rightarrow E = \overline{B}C + B\overline{C}$

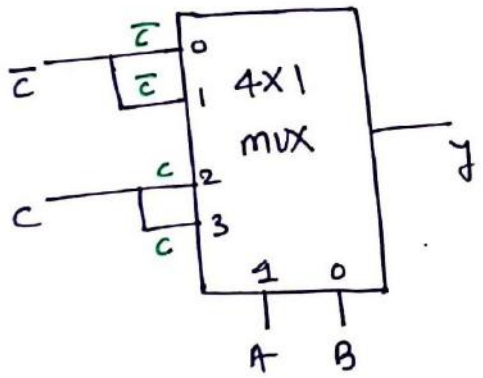
$= B \oplus C$

$\rightarrow f = \overline{E}.0 + EA$

$= A.E$

$= A.(B \oplus C)$

3)



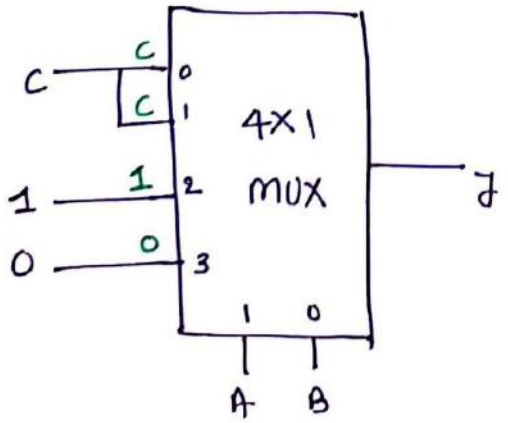
$\rightarrow y = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}C + ABC$

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

$= \overline{A}\overline{C} + AC$

$= \overline{A \oplus C}$ or $A \odot C$

4)



$\rightarrow y = \overline{A}\overline{B}C + \overline{A}BC + A\overline{B}.1 + AB.0$

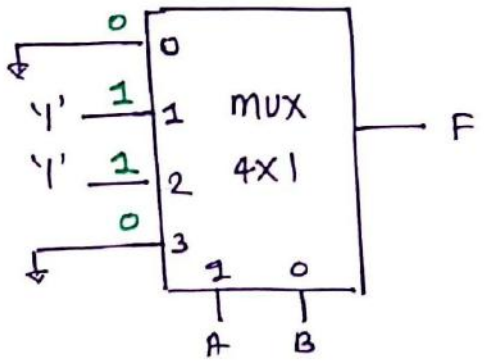
$= \overline{A}C(\overline{B}+B) + A\overline{B}$

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

Engineering Funda YouTube Channel

Identification of logic gate from multiplexer Circuit

1]

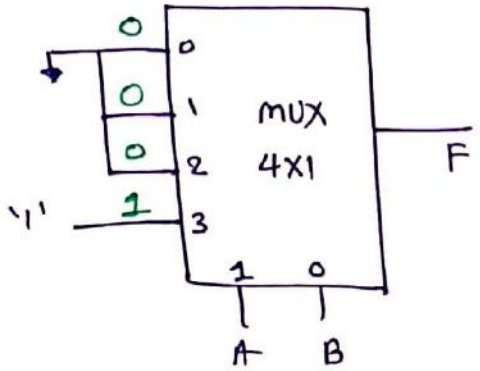


$F = \overline{A}\,\overline{B}.0 + \overline{A}B.1 + A\overline{B}.1 + AB.0$

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

$= A \oplus B$ [XOR gate]

2]

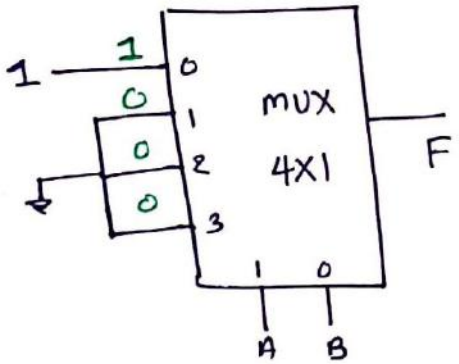


$F = \overline{A}\,\overline{B}.0 + \overline{A}B.0 + A\overline{B}.0 + AB.1$

$= A.B$

[AND gate].

3]



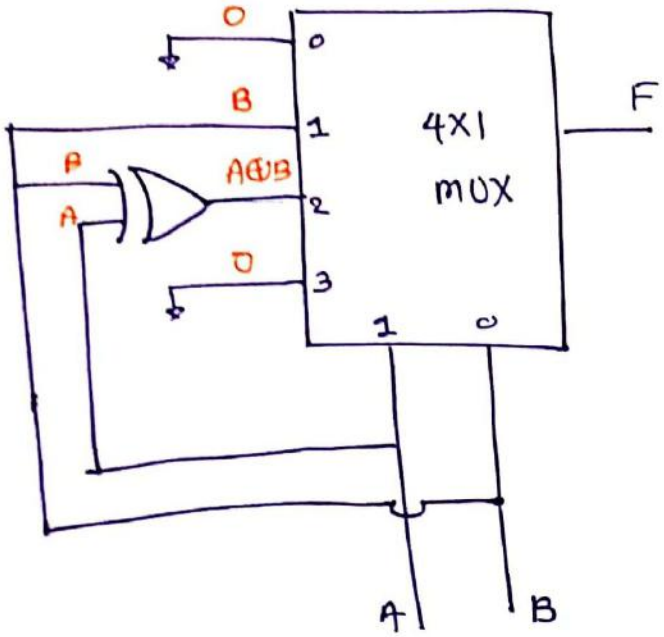
$F = \overline{A}\,\overline{B}.1 + \overline{A}B.0 + A.\overline{B}.0 + AB.0$

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

$= \overline{(A+B)}$ [NOR gate]

Engineering Funda YouTube Channel

Examples based on Multiplexer 130



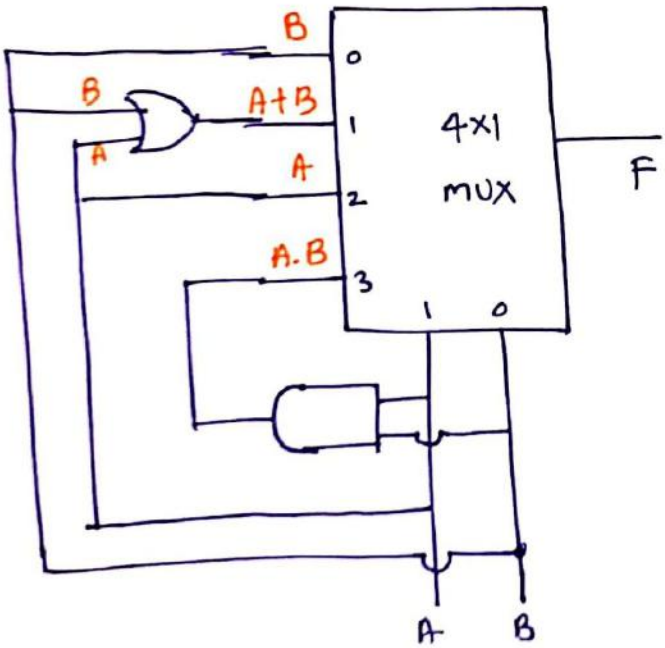
$$F = \overline{A}\,\overline{B}.0 + \overline{A}B.B + A\overline{B}\,(A\overline{B} + \overline{A}B) + AB.0$$

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

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

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

$$= A \oplus B$$



$$F = \overline{A}\,\overline{B}.B + \overline{A}B.(A+B) + A\overline{B}.A + AB.AB$$

$$= 0 + \overline{A}BA + \overline{A}BB + A\overline{B} + AB$$

$$= \underset{1}{\overline{A}B} + \underset{2}{A\overline{B}} + \underset{3}{AB}$$

$$= \Sigma_m(1, 2, 3)$$

$$- [0, 1, 2, 3]$$

$$= \Pi_m[0]$$

Engineering Funda YouTube Channel

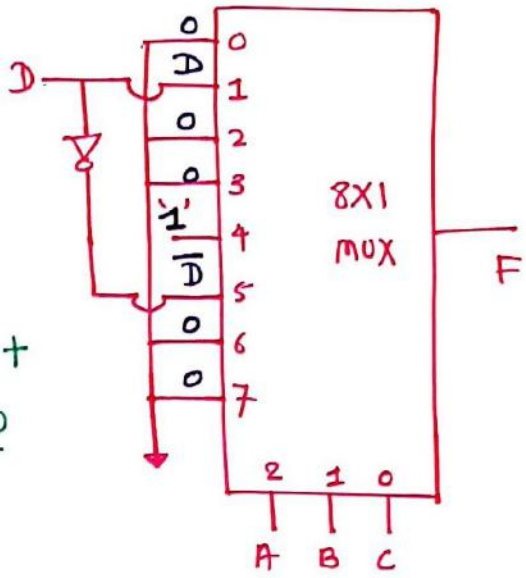
1] Function F(A, B, C, D) can be expressed as

a] $\Sigma_m(3, 8, 9, 10)$

b] $\Sigma_m(3, 8, 10, 14)$

c] $\Pi(0, 1, 2, 4, 5, 6, 7, 11, 12, 13, 15)$

d] $\Pi(0, 1, 2, 4, 5, 6, 7, 10, 12, 13, 15)$



$\rightarrow F = \underline{\bar{A}\bar{B}\bar{C}.0} + \bar{A}\bar{B}C.D + \underline{\bar{A}B\bar{C}.0} + \underline{\bar{A}BC.0} +$
$A\bar{B}\bar{C}.1 + A\bar{B}C.\bar{D} + \underline{AB\bar{C}.0} + \underline{ABC.0}$

$= \underline{\bar{A}\bar{B}CD} + \underline{A\bar{B}\bar{C}} + \underline{A\bar{B}C\bar{D}}$

0011	1000	1010
3	1001	10
	8, 9	

$= \Sigma_m(3, 8, 9, 10)$

2] Find the correct statements.

A Multiplexer.

a] Selects one of many Inputs and transmits on single Output.

X b] Routs the data from single Input to many of outputs.

c] It Converts Parallel data to Serial data

d] is a Combinational Circuit.

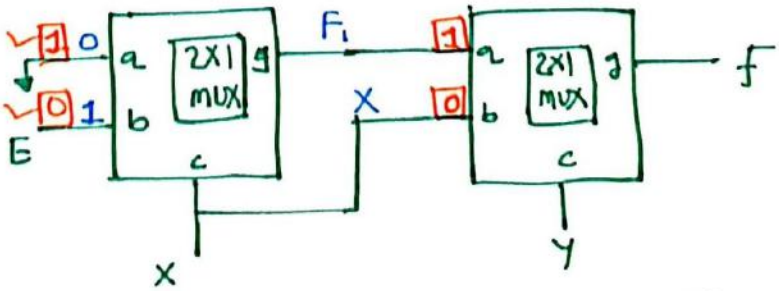
3] If function $g = ac + b\bar{c}$ then find the Output f

(a) $X \oplus Y$

(b) $XE + X\bar{Y}$

(c) $X\bar{Y} + E\bar{X}Y$

(d) None of these



$F_1 = \bar{X}E + X.0$
$= \bar{X}E$

$\rightarrow f = \bar{Y}X + YF_1$
$= \bar{Y}X + Y\bar{X}E$

- $g = \bar{c}a + cb$ x
- $g = \bar{c}b + ca$ ✓

Engineering Funda YouTube Channel

Coincidence Logic 132

- To get coincidence logic following Cond^n should get satisfied.

Cond.^{n} ① Inputs terminals ≥ 2, Output terminal $= 1$

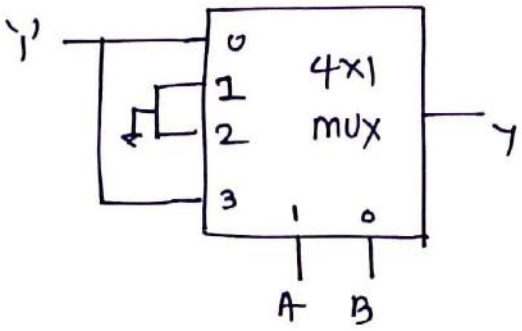
Cond.^{n} ② If all V_p's $= 0$, $O/p = 1$

If all V_p's $= 1$, $O/p = 1$

e.g. Two terminal XNOR gate.

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

e.g.



Engineering Funda YouTube Channel

Demultiplexer and 1 to 2 Demultiplexer

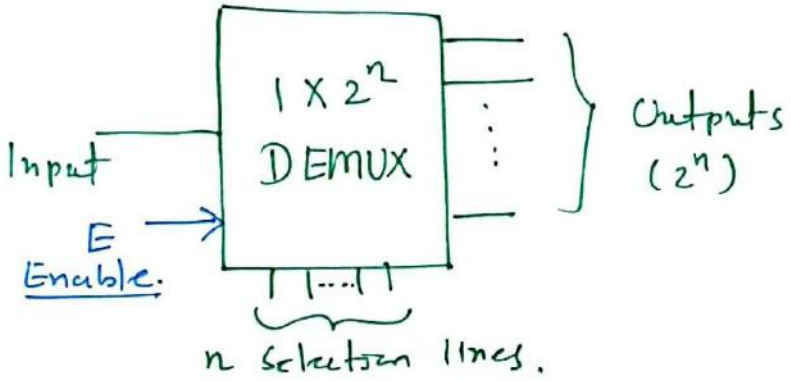
→ It is having reverse operation to that of multiplexer.

→ Basic form of Demultiplexer is $[1 \times 2^n]$ DeMUX.

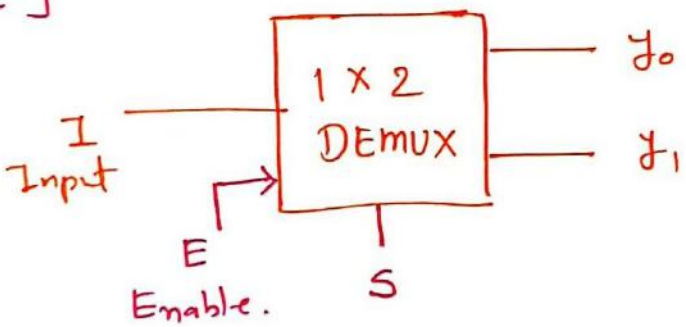
Input — Outputs $= 2^n$

[n = No of selection lines].

→ DeMultiplexer is serial to Parallel converter.



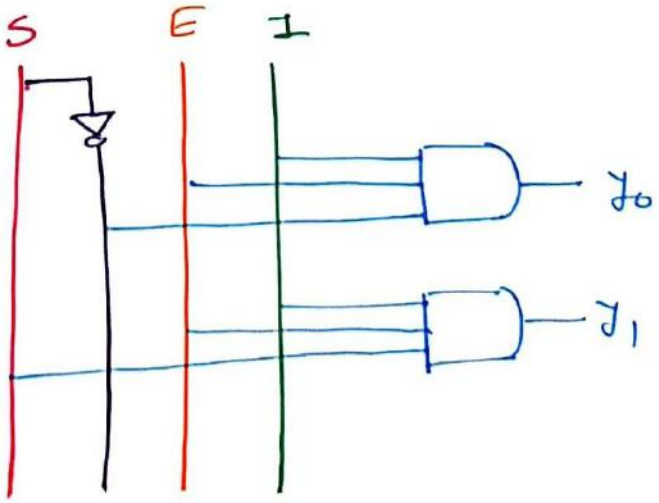
- 1 to 2 Demultiplexer
- [1 to 2^n]
- $n = 1$



E	S	Y_0	Y_1
0	X	0	0
1	0	I	0
1	1	0	I

- $y_0 = I E \bar{S}$

  $y_1 = I E S$



Applications

- Data Distribution
- Serial to parallel Converter.
- One to many Circuit.

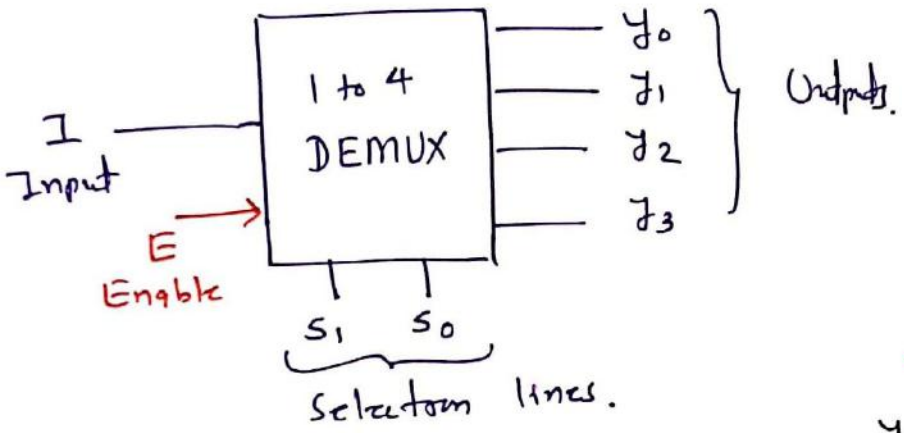
1 to 4 Demultiplexer

134

1 Input 4 Outputs.

→ [1 × 2^n] DeMux

→ n = 2, Selection lines.



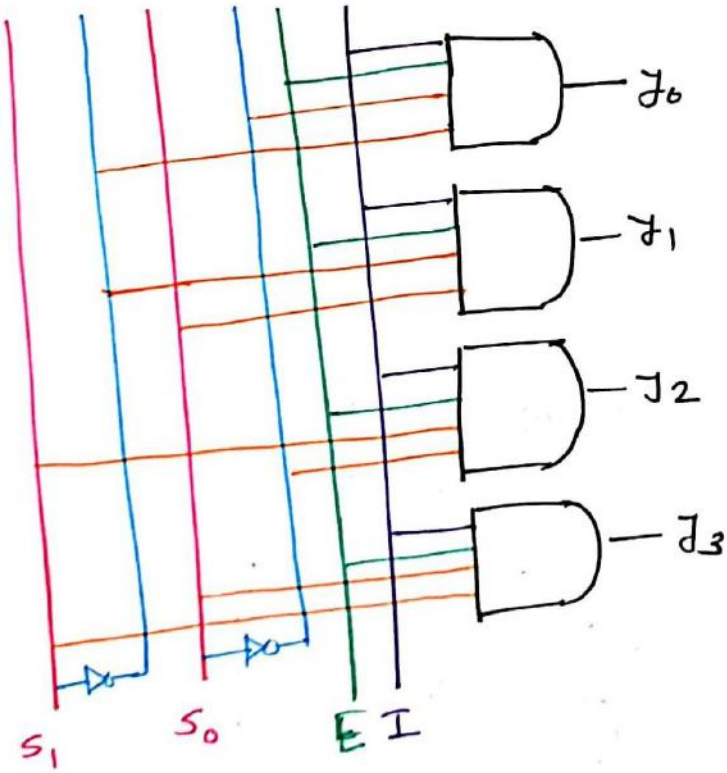
E	S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	X	X	0	0	0	0
1	0	0	I	0	0	0
1	0	1	0	I	0	0
1	1	0	0	0	I	0
1	1	1	0	0	0	I

$y_0 = I E \bar{S_1} \bar{S_0}$

$y_1 = I E \bar{S_1} S_0$

$y_2 = I E S_1 \bar{S_0}$

$y_3 = I E S_1 S_0$



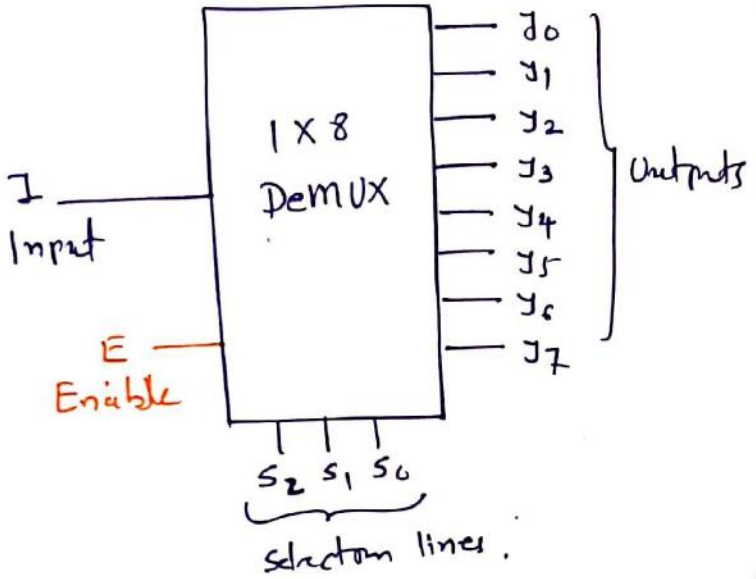
Engineering Funda YouTube Channel

1 to 8 Demultiplexer

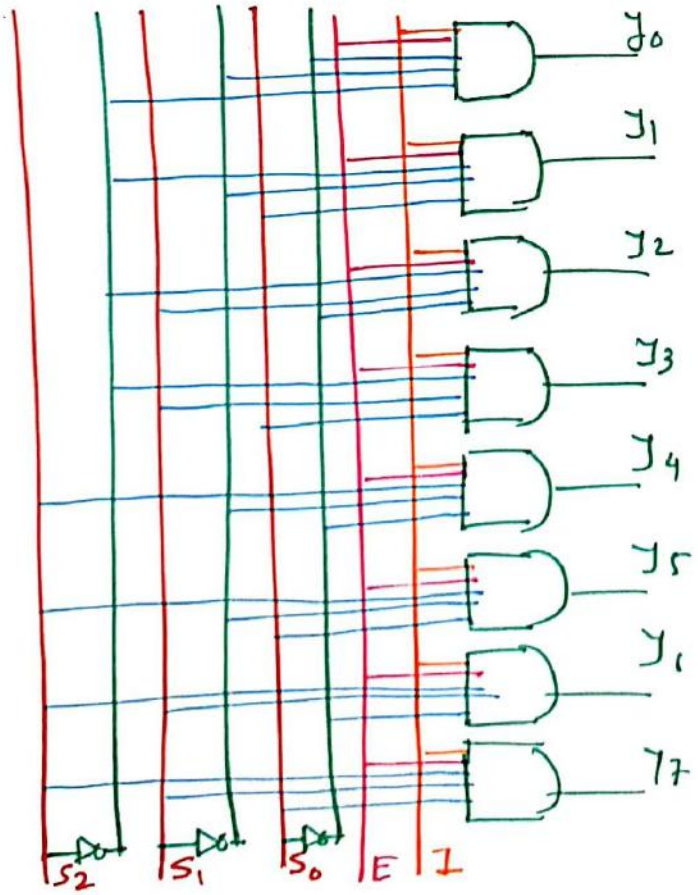
1 Input 8 Outputs.

→ [1 × 2^n] DeMUX

→ n = 3, Selection lines.



E	S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	I	0	0	0	0	0	0	0
1	0	0	1	0	I	0	0	0	0	0	0
1	0	1	0	0	0	I	0	0	0	0	0
1	0	1	1	0	0	0	I	0	0	0	0
1	1	0	0	0	0	0	0	I	0	0	0
1	1	0	1	0	0	0	0	0	I	0	0
1	1	1	0	0	0	0	0	0	0	I	0
1	1	1	1	0	0	0	0	0	0	0	I



$y_0 = I E \overline{S_2}\,\overline{S_1}\,\overline{S_0}$

$y_1 = I E \overline{S_2}\,\overline{S_1} S_0$

$y_2 = I E \overline{S_2} S_1 \overline{S_0}$

$y_3 = I E \overline{S_2} S_1 S_0$

$y_4 = I E S_2 \overline{S_1}\,\overline{S_0}$

$y_5 = I E S_2 \overline{S_1} S_0$

$y_6 = I E S_2 S_1 \overline{S_0}$

$y_7 = I E S_2 S_1 S_0$

Engineering Funda YouTube Channel
Scanned with CamScanner

Full Subtractor using 1:8 DeMUX.

(A − B − Bin)

1 Input 8 outputs → [1 X 2^n] Demux.

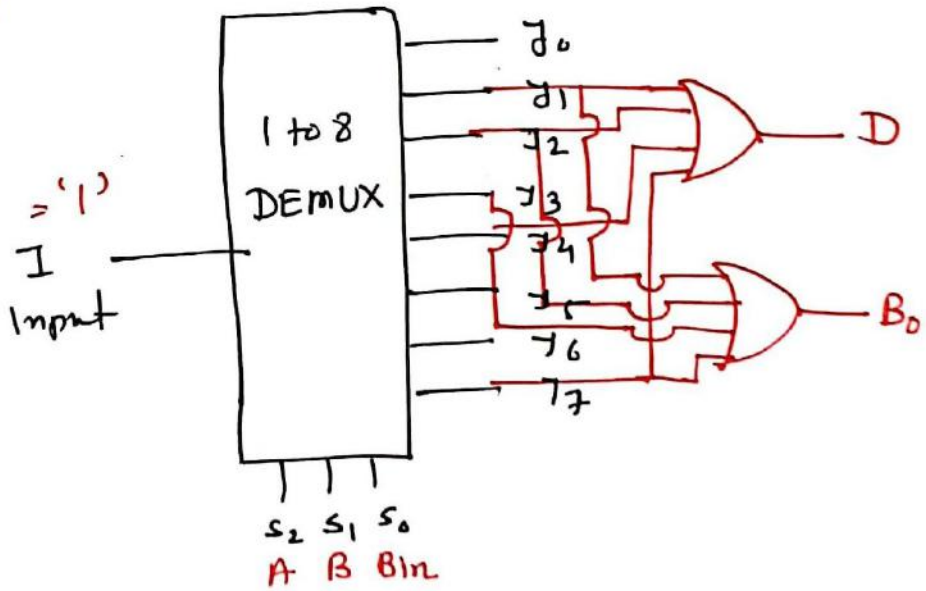
→ n = 3, Selection lines.

	A	B	Bin	D	B_o
m_0	0	0	0	0	0
m_1	0	0	1	1	1
m_2	0	1	0	1	1
m_3	0	1	1	0	1
m_4	1	0	0	1	0
m_5	1	0	1	0	0
m_6	1	1	0	0	0
m_7	1	1	1	1	1

→ $D = \Sigma m(1,2,4,7)$

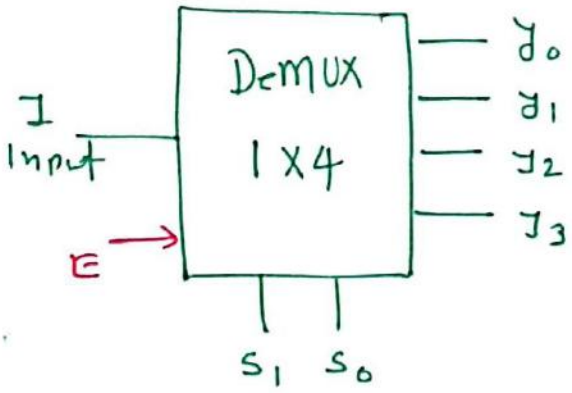
$B_o = \Sigma m(1,2,3,7)$

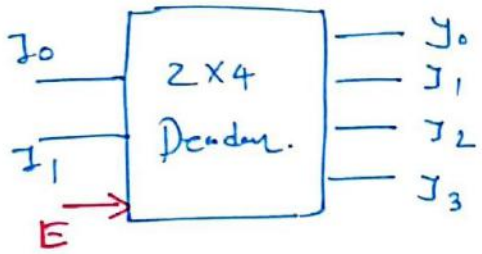
S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



Scanned with CamScanner

DeMUX as Decoder





→ If $I = '1'$

→ $S_1 S_0 = I_0 I_1$

Engineering Funda YouTube Channel

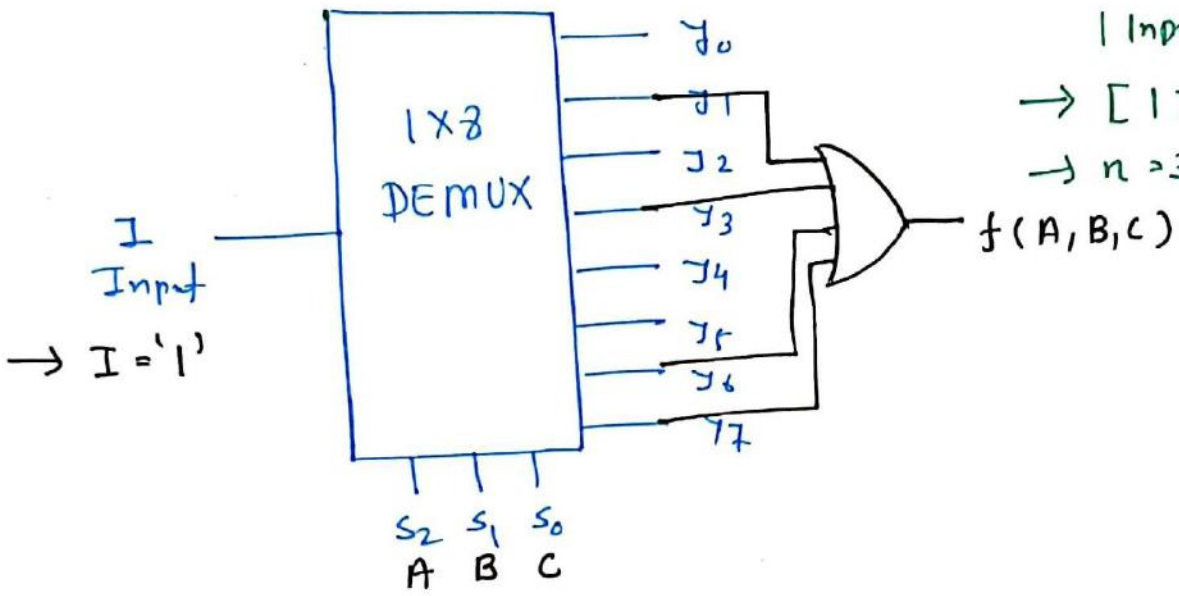
Implementation of boolean expression using DeMUX.

Implement $f(A, B, C) = \Sigma m(1, 3, 6, 7)$ using 1×8 DeMUX.

1 Input 8 Outputs.

→ $[1 \times 2^n]$ DeMUX

→ $n = 3$, selection lines.



- Implement $f(A, B, C) = AB + B\bar{A} + C$ using 1×8 DeMUX

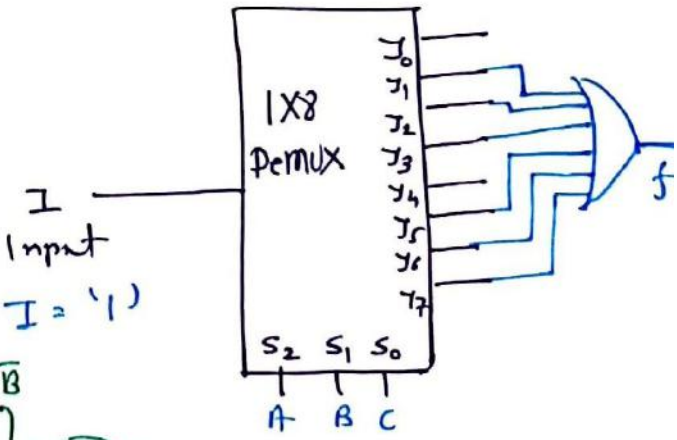
1 Input 8 Outputs.

→ $[1 \times 2^n]$ DeMUX

→ $n = 3$ selection lines

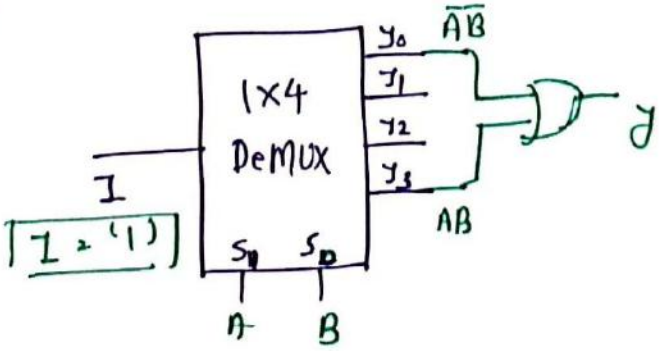
f	AB: 00	01	11	10
C = 0	0	1 (2)	1 (6)	4
C = 1	1 (1)	1 (3)	1 (7)	1 (5)

$f(A, B, C) = \Sigma m(1, 2, 3, 5, 6, 7)$



- If $y = \underset{3}{AB} + \underset{0}{\bar{A}\bar{B}}$, Implement using 1×4 DeMUX

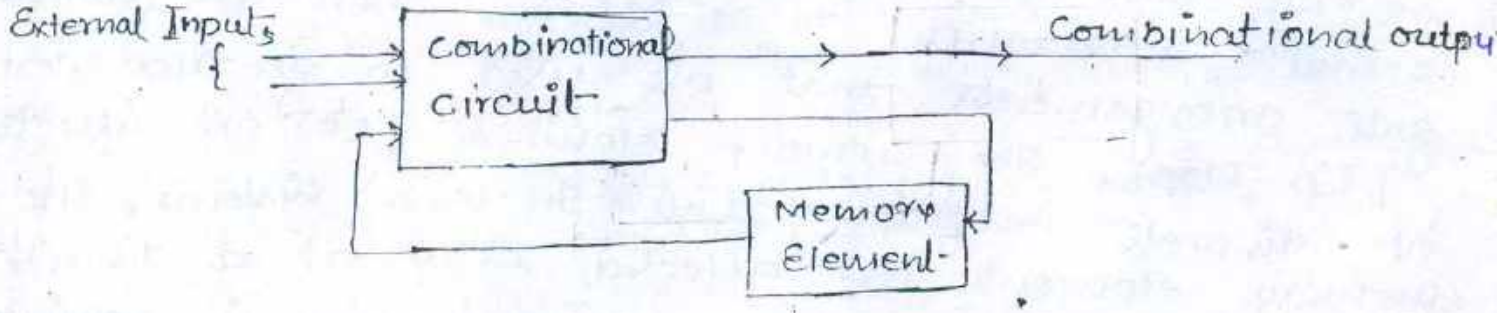
→ $y = \Sigma m(0, 3)$



UNIT - III

SEQUENTIAL LOGIC CIRCUITS

Introduction – So far, we have studied about the design and analysis of combinational digit circuits whose outputs at any instant of time entirely depends on the present inputs. Any prior input-level conditions have no effect on the present outputs because combinational logic circuits have no memory (storage device). Most digital systems are made up of both combinational circuits and memory elements. In practice there are many applications in which digital outputs are required to be generated in accordance with the sequence in which the input signals are received. These applications require outputs to be generated that are not only dependent on the present input conditions but they also depend upon the past history of these inputs. Such type of digital circuits is known as sequential circuit. Following figure shows the block diagram of sequential circuits



(Block Diagram of Sequential circuits)

The combinational circuit accepts external logic signals from external inputs and from the outputs of memory elements. The memory elements are devices

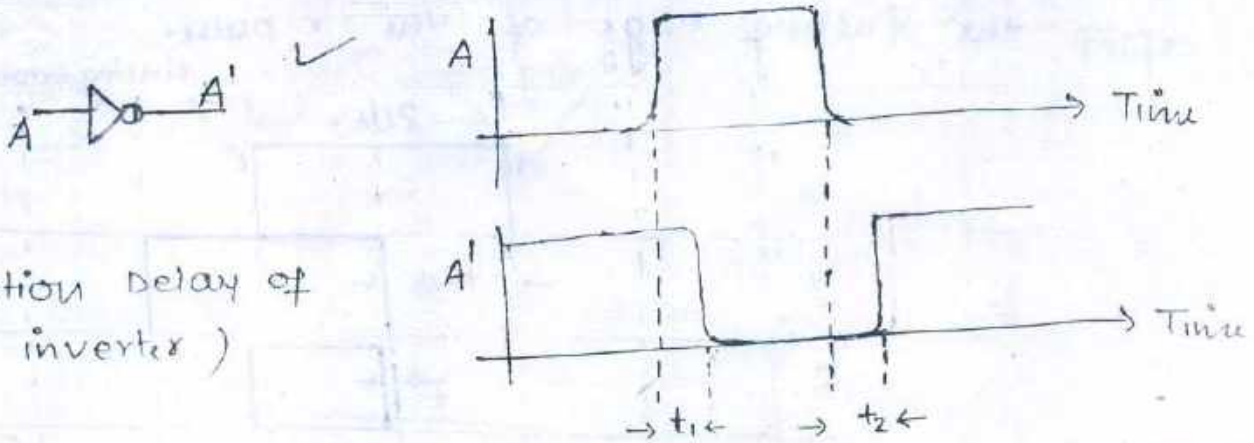
capable of storing binary information with the
The binary information stored in the memory elem
at any given time difines the state of the Sequential
circuit. External inputs, together with the present-
state of the memory elements, determine the binary
value at the output terminals. They also determine
the condition for changing the state in the
memory element. The next state of the memory
elements is also a function of external inputs
and the present state. Thus a sequential circuit
is specified by a time sequence of inputs, output
and internal states.

Sequential circuits are classified in two
main categories known as synchronous and
Asynchronous sequential circuits. This classification
depends on the timing of their signals.
A sequential circuit whose behaviour can be defined
from the knowledge of its signal at discrete instant
of time is referred to as a synchronous sequential
circuits. The memory elements used in synchronous
sequential circuits are called Flip-Flops which is made
up of an assembly of logic gates, and is capable
of storing one bit of information. Even though
a logic gate, by itself, has no storage capability,
several can be connected together in ways that
permit information to be stored. There are several different
gate arrangements that are used to produce these
flip-flops. The memory elements (F.F.) are affected
at discrete instants of time. In these systems, the
memory elements are affected only at a discrete
instants of time. This synchronization is achieved
by a timing device known as a clock, which
generates a periodic train of clock pulses as
shown below

The behaviour of Asynchronous sequential circuits depends upon the sequence in which input signal change and can be affected at any instant of time. The commonly used memory elements in asynchronous sequential circuit are time delay devices. The memory capability of a time-delay device is to the finite time it takes for the signal to propagate through the device. Generally the internal propagation delay of logic gates is of sufficient duration to produce the needed delay, so that physical time delay units may be unnecessary. Thus an asynchronous sequential circuit may be regarded as a combinational circuit with feedback.

5.1 GATE DELAYS AND TIMING DIAGRAM —

When the input to a logic gate is changed, the output will not change instantaneously. The transistors or other switching elements within the gate take a finite time to react to a change in input, so that the change in the gate output is delayed with respect to the input change. Following figure shows possible input and output waveforms for an inverter.



(Propagation Delay of an inverter)

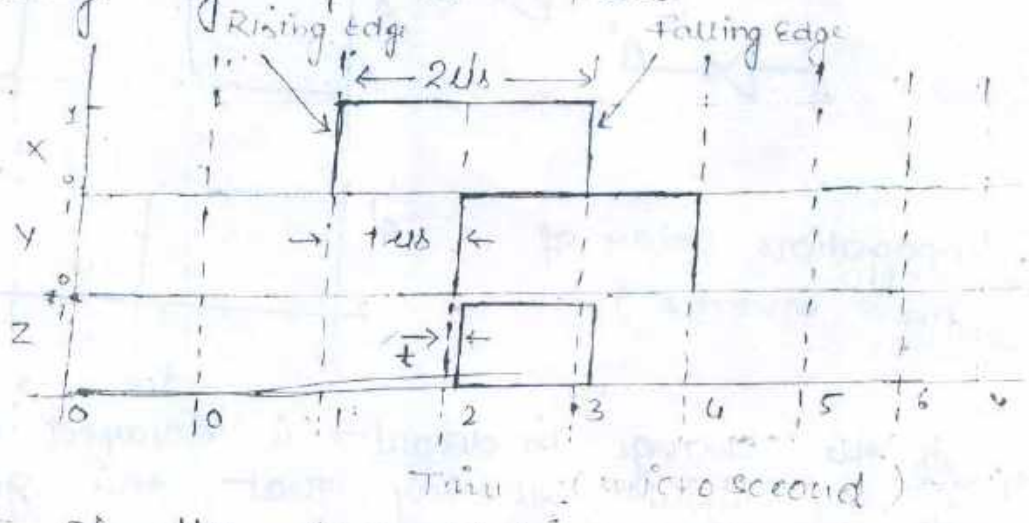
If the change in output is delayed by time t with respect to the input, we say that this gate has a propagation delay of t. In practice, the propagation delay for a 0 to 1 output change may be different than the delay for a 1 to 0 change. Propagation delays

for integrated circuit gates may be as short as a few nanoseconds, and in many cases these delays can be neglected.

Timing diagrams are frequently used in the analysis of sequential networks. These diagrams sh various signals in the network as a function of tir several variables are usually plotted with the sa time scale so that the times at which these variables change with respect to each other cans easily observed.

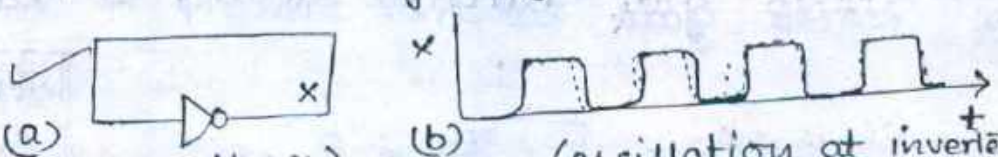
Consider the figure given below, shows a tim diagram for a network with an added delay element.

The input x consists of two pulses, the first of whic is two microseconds (2×10^{-6} second) wide and the secon is three microseconds wide. The delay element has an output y which is same as the input except that i is delayed by one microsecond. That is, y changes to 1 value one microsecond after the rising edge of the x pulse and returns to 0 one microsecond after the falling edge of the x pulse.



The output of the AND gate (z) should be 1 in the time interval in which both x and y are 1. We assume a small propagation delay in the AND ga (t). then z will be as shown in the above diagram.

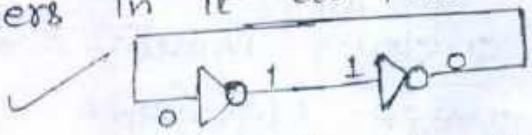
The switching networks we have studied so far have not had feedback connections. In order to construct a switching network that has memory such as Flip-Flop, we must introduce feedback into the network. By feedback we mean that the output of one of the gates is connected back into the input of another gate in the network so as to form a closed loop. In simple cases, we can analyze networks with feedback by tracing signals through the network. For example, consider the network in the following figure.



(a) (Inverter with feedback) (b) (oscillation at inverter output)

If at some instance of time, the inverter input is 0, the 0 will propagate through the inverter and cause the output to become 1. This 1 is fed back into the input, so after the propagation delay, the inverter output will become 0. When this 0 feedback into the input, the output will again switch to 1 and so forth. The inverter output will continue to oscillate back and forth between 0 and 1 as shown in above fig (b) and it will never reach a stable condition. The rate at which the network oscillates is determined by the propagation delay in the inverter.

Next consider a feedback loop which has two inverters in it as shown in following figure

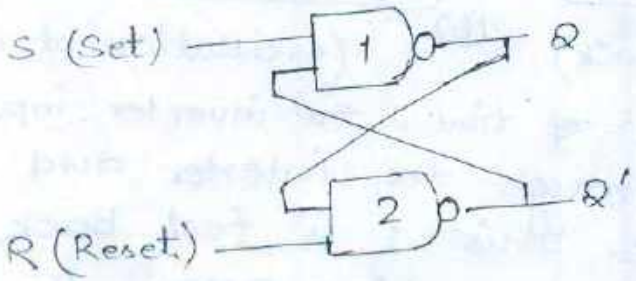


In this case the network has two stable conditions often referred to as stable states. If the input to the first inverter is 0, its output will be 1. The input to the second inverter will be 1, and its output will be 0. This 0 will feedback into the

first inverter, but since this input is already no change will occur. The network is then in stable state.

5.2 1-Bit Memory Unit —

The basic memory unit is known as Flip-Flop. The Flip-flop is known by other names, including latch and bistable multivibrator. Flip-flop has two stable states known as 1 s and the 0 state. It can be designed by using NAND or NOR gates. The NAND gate version call a NAND gate latch or simply a latch shown bel



(a) (Basic Flip-Flop with NAND gates)

Set S	Reset R	output Q
1	1	No chan
0	1	1
1	0	0
0	0	Invalid

(b) Truth

The two NAND gates are cross coupled so tha the output of NAND-1 is connected to one of the inputs of NAND-2 and the output of NAND-2 is connected to the one of inputs of NAND-1 gate constitutes a feedback path. Each Flip-Fl has two outputs labeled Q and Q' and two inputs set (S) and Reset (R). Under normal condition these outputs will always be the inverse of each The set input is the input that sets Q to the 1 state, & the Reset (or clear) input clears Q to the 0.

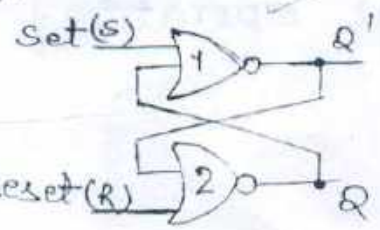
The NAND basic flip-flop circuit operates with both inputs normally at 1 unless the state of the flip-flop has to be changed. As set input is pulsed LOW (0), the output Q becomes high and Q' go to 0, putting the flip-flop into the state.

When $Q=1$ and $Q'=0$ prior to the application of set pulse, since $Q'=0$ is already keeping the NAND-1 output high (1), the low pulse at set will not change anything. Thus when set input returns to 1 state the, the latch outputs are still in $Q=1$ and $Q'=0$ state.

When set input is 1 and 0 is applied to the reset input, the output $Q'=1$, since Q' is connected as the input of NAND-1, hence the output of NAND-1 (Q) is 0. When clear input CR returns to the 1 state, the latch outputs are at $Q=0$ and $Q'=1$. This operation is called clearing or resetting the latch. and $Q=0$ state is also called the cleared or reset state.

When both the inputs (Set and Reset) are simultaneously 0 (low) will produce high outputs (1) at the both NAND gates that is $Q=Q'=1$, this is an undesired condition, since the two outputs Q and Q' are supposed to be complement of each other. When the set and reset inputs return high (1) state, the resulting output state will depend on which input returns 1 state first. Simultaneous transitions back to the 1 state will produce unpredictable results. Due to this reason the set = Reset = 0 condition is normally prohibited and not allowed for the NAND latch.

5.3 NOR GATE LATCH - As we know that the output of a NOR gate is 0 if any one input is 1 and the output is high (1) only when all inputs are 0. The NOR gate latch (or memory element) can be constructed using two cross coupled NOR gates as shown below.



(a)

(Basic Flip-Flop circuit with NOR gates)

(b)

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after S=1, R=0)
0	1	0	1	
0	0	0	1	(after S=0, R=0)
1	1	Invalid		

Truth Table

The circuit diagram of NOR latch is similar to the NAND latch except that the Q and Q' outputs have reversed positions. The analysis of the operation of the NOR latch can be performed exactly in the same manner as for the NAND latch. The results are given in the truth table.

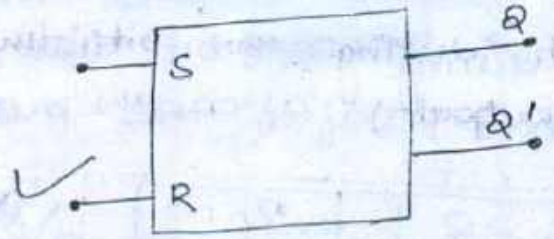
(I.) Assume When Set = Reset = 0. It is the normal resting state for the NOR latch and it has no effect on the output state. The outputs Q and Q' will remain in whatever state they were prior to the occurance of this input condition.

(II) When set input is 1 and reset input is 0. Since NOR gate-1 has an input of 1, its output Q' must be 0 which puts both inputs of gate 2 at 0 (low), so that output Q is 1 (high). When the set input is returned to zero (0), the outputs remain the same, because output Q remains a 1, leaving one input of gate 1 at 1, which results Q' as 0, which leaves both inputs of gate 2 at 0, so that output Q is 1.

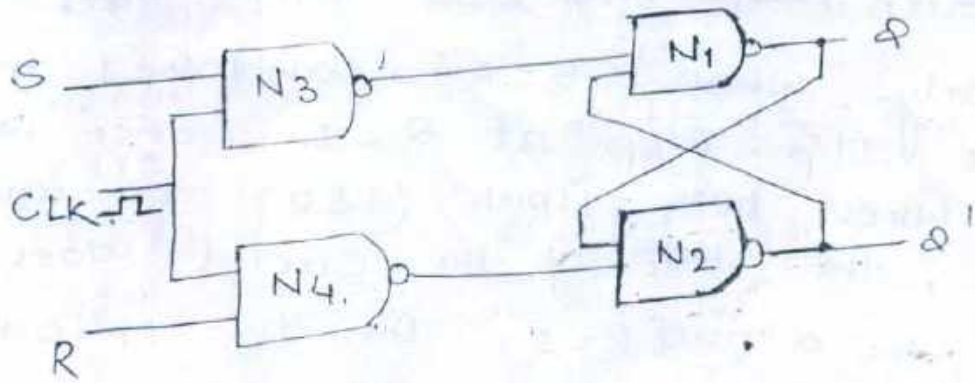
(III) When set input is 0 and reset input is 1. Since NOR gate-2 has an input of 1, its output Q must be 0, which puts both inputs of gate 1 at 0. so that output Q' is high 1. When the reset input returned to 0 state, the outputs remain the same.

(IV) When Set = Reset = 1.
This condition tries to set and reset (clear) the latch at the same time, that is when a 1 is applied to both the set and clear inputs, both Q and Q' outputs go to 0. This condition violates the assumption that Q and Q' are inverse of each other, hence in normal operation this condition must be avoided.

The NOR gate latch operates exactly like the NAND latch except that the set and reset inputs are active-high rather than active low, and the normal resting state is Set = Reset = 0. Q will be set High by a high pulse (1) on the set input and it will be cleared by a high pulse on the clear input. The block symbol for the NOR latch is shown below.



5.4 S-R Flip-Flop — Generally it is required to set or reset the memory unit in synchronism with CLOCK pulses (abbreviated as CLK, CK, CP), the exact times at which the output can change states. A S-R Flip-flop consists of a basic Flip-flop with a clock pulse and two additional NAND gates. The additional NAND gates are added for entering the input information at discrete instant of time. The circuit diagram of S-R Flip-Flop is shown below.



The clock pulse acts as an enable signal for the other inputs (S and R). When clock pulse is not present (that is CLK = 0) the NAND gates N3 and N4 are inhibited that is the outputs of gate N3 and N4

stay at logic 1, without the consideration of inputs S and R. When clock pulse is 1, the input information from the inputs is allowed to reach the output. The operation of the Flip-Flop can be expressed in the form of a table known as characteristic Table, shown below. In the following table S and R denotes the inputs, Q the present state during the bit time n and Q_{n+1} denotes the next state (output) after the pulse passes.

(Logic symbol of clocked S-R Flip-Flop)

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

Characteristic Table

R \ QS	00	01	11	10
0		1	1	x
1		x	x	

$Q_{n+1} = S + QR'$

$SR = 0$

Characteristic Equation

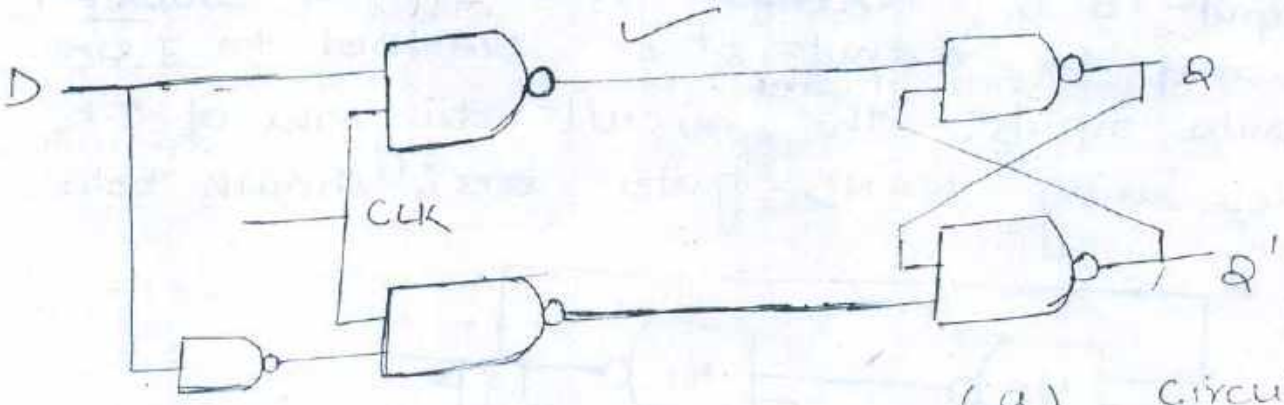
When $S = R = 0$ & $Q = 0$ on the application of clock pulse, the output of gate N3 and N4 go to 1, and the output of the Flip-Flop Q_{n+1} go to 0 similarly, when $S = R = 0$ and $Q = 1$ the output of the Flip-Flop at $Q = 1$. Hence we can say that when both inputs (S&R) are equal to 0 and CLK = 1, the state of the circuit does not change. When $S = 0$ and $R = 1$. On the application of clock pulse the output of gate N3 go to 1 and N4 go 0 and the output of Flip-Flop at $Q = 0$ whatever may be the value of present state (that $Q_n = 0$ or 1).

When $S=R=1$. on the application of clock pulse (CLK=1) the outputs of gate N_3 and N_4 are 0, Hence one of the inputs of NAND gate N_1 and N_2 is zero, resulting Q and Q' both outputs as 1 which is inconsistent with our assumption of complementry outputs. When the clock pulse has passed away (CLK=0), (while set and Reset inputs are maintained at same level 1) it is difficult to determine the next stable state (that is $Q_{n+1}=1$ or $Q_{n+1}=0$). It means the state of the circuit is indeterminate or undefined. Hence the condition $S=R=1$ is prohibited and it must not be allowed to occur.

In the characteristic equation of S-R flip-flop binary informations of characteristic table are algebraically expressed, showing that output (Q_{n+1}) depends upon inputs (S&R) as well as present state (Q). Both S and R inputs, equal to 1 simultaneously are not allowed is specified by a term $SR=0$.

5.5 D-Flip-Flop

It has only one input referred to as D-input or data input. The circuit diagram and characteristic table are shown below.



(a) Circuit diagram

Q	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

(b) Characteristic Table

D \ Q	0	1
0		
1	1	1

$Q_{n+1} = D$

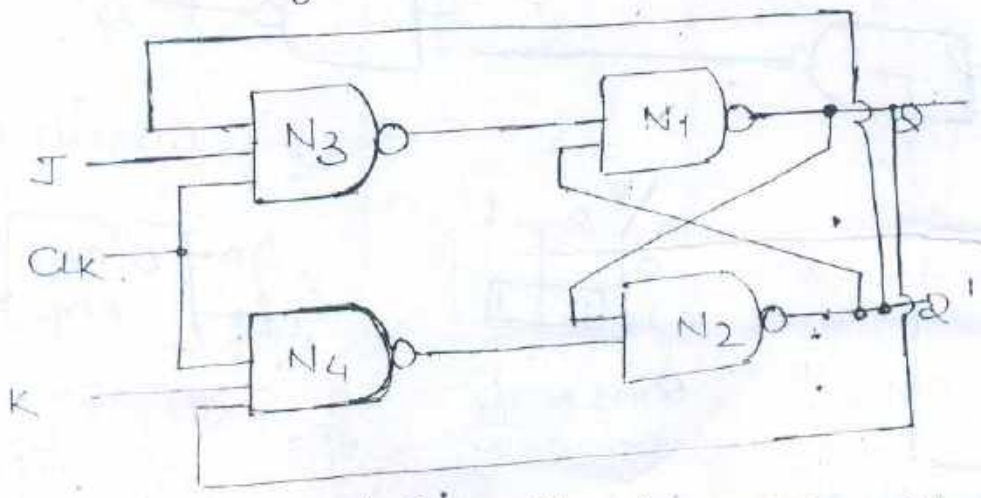
(c) Characteristic Equation

(d) Logic Symbol

From the characteristic table it is clear that the next state (output) Q_{n+1} at the end of clock pulse equals the input data D_n before the clockpulse. Thus the transfer of data from the input to the output is delayed and hence the name Delay (D) flip-flop. For example if D=1 before the clock pulse, Q = 1 after the clock pulse, regardless of the present state of Q. Sometimes D-Flip-Flop is called a gated D-latch. The clock pulse input is designated as Gate to indicate that this input enables the gated latch to make possible data entry into the circuit. The D-Flip-flop is either used as a latch to store one bit of data or as a delay device.

5.6 J.-K Flip-Flop —

The J-K flip-flop has two inputs J and K and a clock pulse. Indeterminate (undefined) state of an S-R Flip-flop when S = R = 1, is defined in J-K Flip-flop. A J-K Flip Flop is constructed with two cross-coupled NAND gates and two additional NAND gates. The output Q is NANDed with K and clock pulse inputs, and the output Q' is NANDed to J and clock pulse inputs. The circuit diagram of J-K Flip-flop using NAND gates is shown below



(Circuit diagram of J-K Flip-Flop)

The behaviour of the J-K Flip-flop is shown in the characteristic table given below.

Input			output (Next state)
Q	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

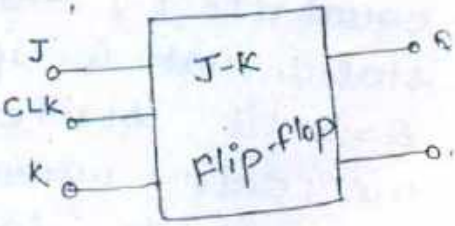
(a) (Characteristic Table)

K \ QJ	00	01	11	10
0		1	1	1
1		1		

$Q_{n+1} = JQ' + K'Q$.

(b) Characteristic (Next state) Equation

(c)



(Logic Diagram)

When J=0 and K=0. On the application of clock pulse (that is CLK=1) the output (next state) of the Flip-Flop at Q is equal to the present state; hence we can say that when J=K=0 the state of the circuit does not change.

When J=0 and K=1, during the clock pulse, the next state of the circuit at Q will be set to 0 after the clock pulse, whatever may be the value of present state (0 or 1). For example; Let J=0, K=1 and Q=0 (IInd row of characteristic table). In the presence of clock pulse, the output of NAND gate $N_3 = 1$ and $N_4 = 1$. Since Q'(=1) is connected as an input to the NAND N_1, which results the output of N_1 (that is Q_{n+1}) as 0. (In the case of NAND latch 1 as feed back is considered first, because 0 as feed back makes other input don't care)

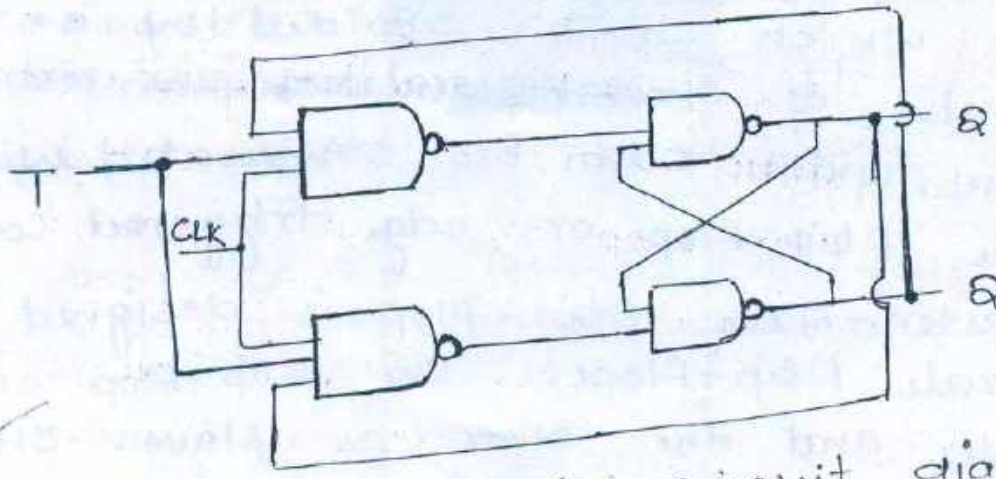
When $J = 1$ and $K = 0$ the Flip-Flop output (next is set to $Q = 1$ what ever may be the present state value (0 or 1). For example, consider the case when $Q = 1$, $J = 1$ and $K = 0$ (VIIth row of table). The output of NAND gate $N_3 = 1$ and $N_4 = 1$. Q (=1) is connected as an input to the NAND gate N_2 making both inputs equal to 1. Hence output of gate N_2, $Q' = 0$, making one input of NAND gate N_1 as zero, which results the output of gate N_1 $(Q_{n+1}) = 1$.

When $J = K = 1$. When both inputs J and K are equal to 1, the flip-flop switches to its complement state, that is if $Q = 0$ it switches to $Q = 1$ and if $Q = 1$ it switches to $Q = 0$. For example, consider the case when $J = K = 1$ and $Q = 1$ (VIIIth row of characteristic table). The output of NAND gate $N_3 = 1$ and $N_4 = 0$. One input of NAND gate N_2 is zero as results its output $Q' = 1$. Q' is connected as an input of NAND gate N_1 making its both inputs equal to 1. Hence output (Q_{n+1}) of gate $N_1 = 0$ which is complement of its present state.

Due to the feedback connection, when clock pulse remains high and both J and K inputs are high (1), the output complements again and again untill the clock pulse goes back to 0. Hence output becomes uncertain. This undesirable operation can be avoided by the application of a clock pulse of time duration shorter than the propagation delay time of the flip-flop.

T - Flip - Flop —

T Flip-Flop has one input referr to as T input. When both the inputs of J-K Flip-Fl are tied together, T Flip-Flop is obtained. The circu diagram and function table of T Flip-Flop is give below.



(a) circuit diagram

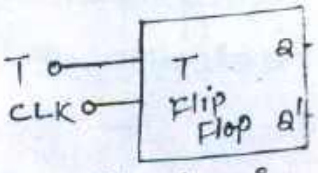
Input		output
Q	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

(b) Function Table

T \ Q	0	1
0		1
1	1	

$Q_{n+1} = Q'T + QT'$

(c) Characteristic Equation



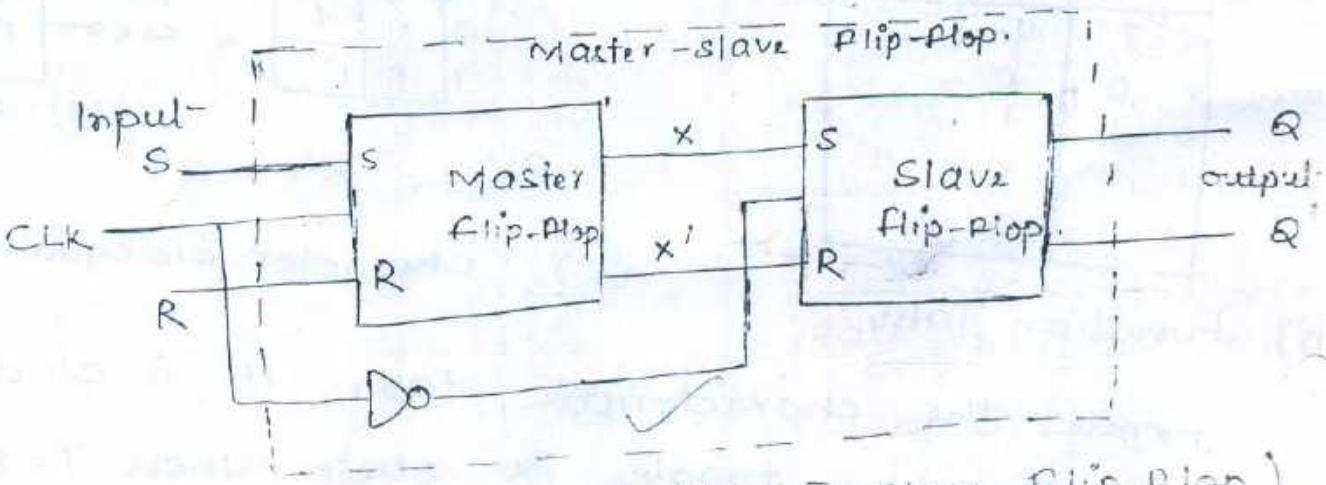
(d) Logic symbe

From the characteristic table, it is clear tha the T-Flip-Flop toggles its state when T=1. — regardless of the present state, in the presence of clock pulse. When T is low (0), the Flip-Flop remain in whatever state it was before. For example, whei T=0 (Ist and IIIrd row of the table) the next stat (Q_{n+1}) is equal to present-state (Q). When T=1 — (IInd and IVth row) the next state Q_{n+1} = (Present-state) Due to this ability of Flip-Flop to 'toggle' its state it is designated as T Flip-Flop.

MASTER - SLAVE S-R FLIP-FLOP —

In the J-K Flip-flop, because of the feedback connection, when J and K inputs are equal to 1, the output (next) complements its state and repeat complementing until the pulse goes back to 0. This is restrictive requirement which can be avoided by taking a clock pulse of short duration. The restriction on the pulse width can be eliminated with a master-slave flip-flop or edge triggered. Construc

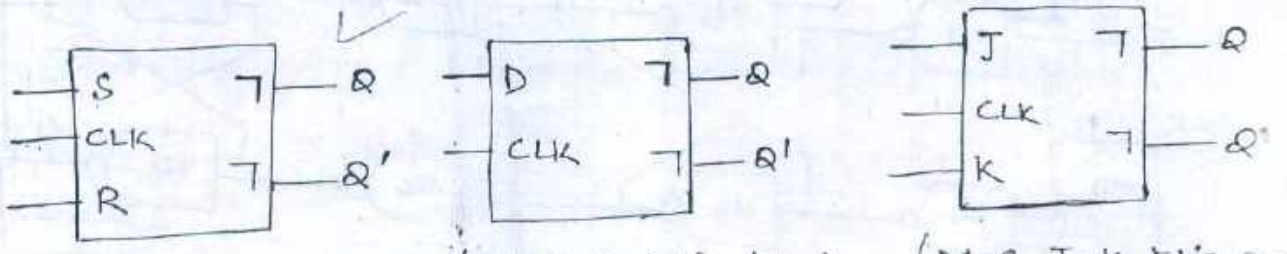
A master-slave flip-flop is designed from two seperate flip-flops. One flip-flop serves as a master and the other as slave. The logi diagram of a S-R master-slave flip-flop is shown below.



(Logic Diagram of S-R Master slave flip-flop)

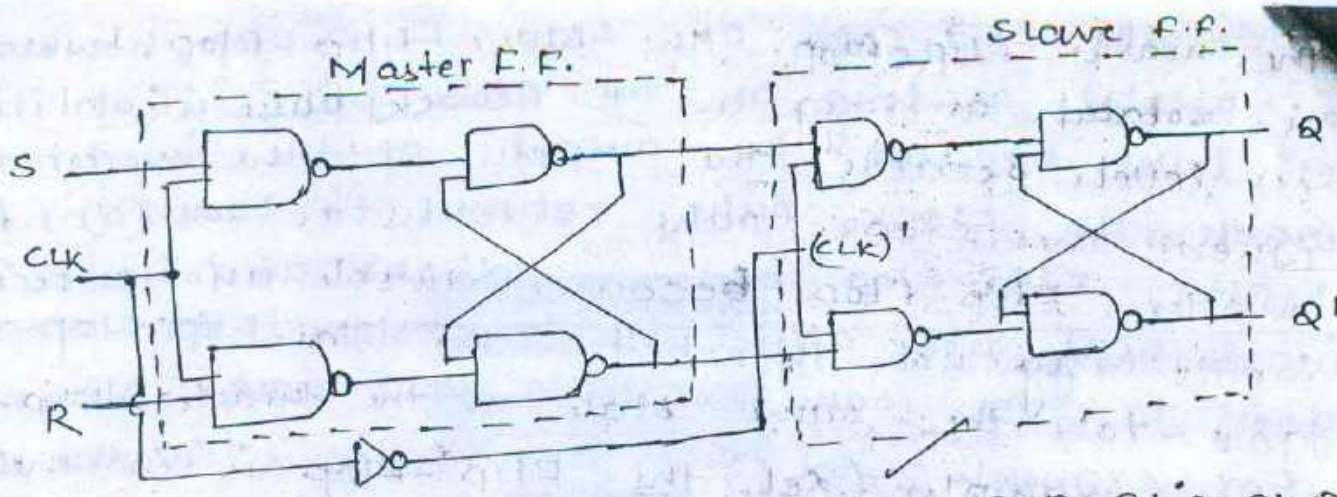
It consists of a master flip-flop, a slave flip flop and an inverter. When the clock pulse is 0 the master flip-flop is disabled, while the slave flip-flop (Because CLK = 1) is enabled since the clock input of the slave is 1 (the is slave flip-flop is enabled) the output Q is equ to x and Q' is equal to x'. When the clock pulse become 1 the input information at S and R is then accepted by

the master flip-flop. The slave flip-flop, however is isolated as long as the clock pulse is at its hig (1) level, because the output of the inverter is 0. When the clock pulse returns to low (0), the master flip-flop becomes disable and hence exter inputs may not affect it. The slave flip-flop then goes to the same state as the master flip-flop. For example, let the flip-flop is in the clear state before the application of a clock pulse, so that $x=0$ and $Q=0$ The input conditions ar $S=0$ and $R=1$. During the pulse transition from 0 to 1 the master flip-flop is reset and x equals to 0 (no change). The slave flip-flop is unaffected because its clock pulse $=0$. When pulse returns to 0, the information fr master-flip-flop (that is x & x') is allowed to pa through to the slave flip-flop, resulting the exter output $Q=0$. There are three types of master-slav flip-flops - S-R, J-K and D. The J-K master-slav flip-flop is most commonly available in IC form Logic symbols of master-slave flip-flops are shown below.



(M-S S-R Flip-flop) (M-S D Flip-flop) (M-S J-K Flip-f

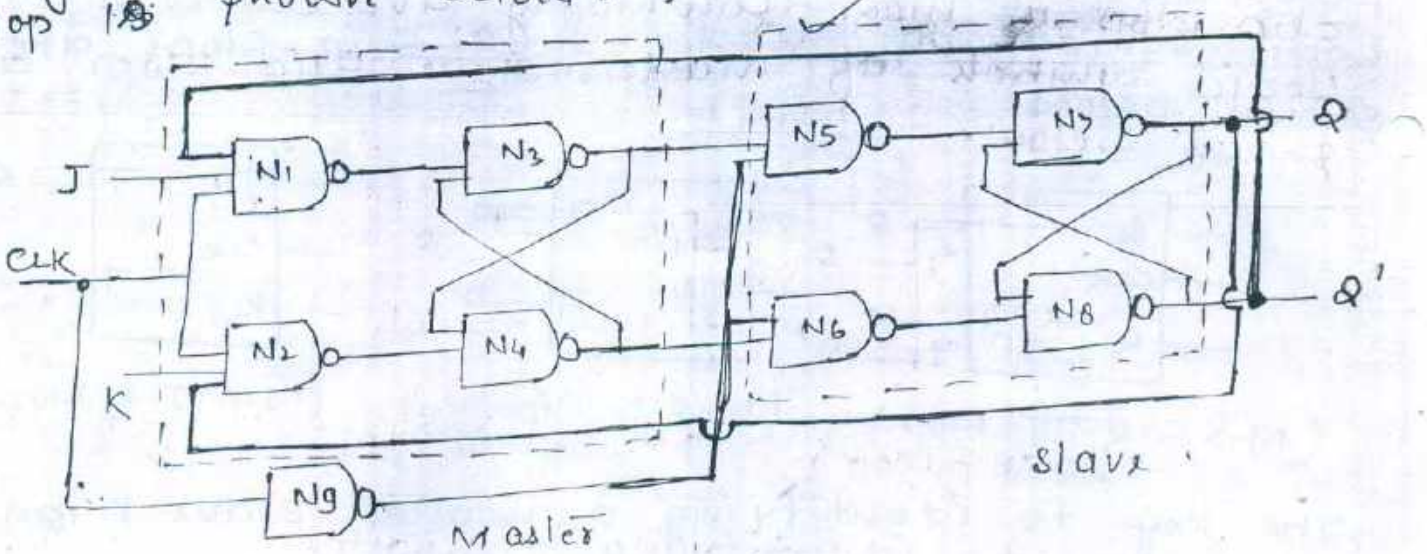
The key to identifying a master slave flip-flop by its logic symbol is the postponed output symbol (⌉) at the outputs. There is no dynamic input indicator at the clock input. The circuit diagram of master-slave S-R flip-flop is shown below.



(The master-slave S-R flip-flop

5.9 Master-slave J-K flip-flop —

The master-slave J-K flip-flop is constructed with two seper Flip-Flops, similar to that of the master-sla S-R flip-flop. The difference is that the Q out is connected back to the input of N_1 and the Q' output is connected back to the input of N_2 and external inputs are designated as J and K. The log diagram and truth table of master-slave J-K fl Flop is shown below.



inputs		output
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Q_n'

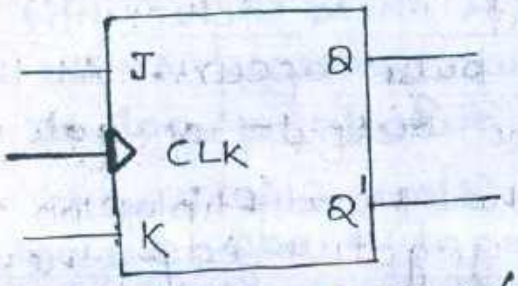
Gates N_1 through N_4 form the master flip-flop and gates N_5 through N_8 form the slave flip-flop. In the presence of positive edge of a clock pulse, the input information at the J&K inputs is transferred to the master flip-flop and is held there until the negative edge of the clock pulse occurs. At the negative edge of clock pulse the output of gate N_3 and N_4 are transmitted to the slave flip-flop. When the clock pulse is at negative edge the master flip-flop is not affected. Hence, the output is generated at the end of clock pulse that reduces the uncertainty at the output.

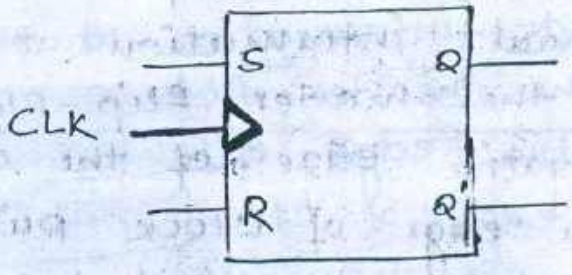
5.10 EDGE TRIGGERED FLIP-FLOPS

In synchronous systems, the exact times at which any output can change states are determined by a signal commonly called the clock. The flip-flops using the clock signal are called the clocked flip-flops. Control signals are effective only if they are applied in synchronization with the clock signal. The clock pulse is distributed to all parts of the system and most of the system outputs can change state only when the clock makes a transition. Clocked flip-flops may be positive edge triggered (↑) or negative edge triggered (↓).

Positive edge triggered flip-flops are those in which 'state transitions' take place only at positive-going edge (0 to 1 or low to high) of the clock pulse. Negative edge triggered flip-flops are those in which state transition take place only at the negative-going (1 to 0 or high to low) edge

of the clock signal. Positive-edge triggering is indicated by a triangle at the clock termina of the flip-flop.

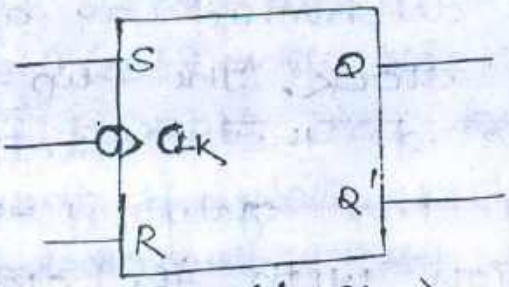


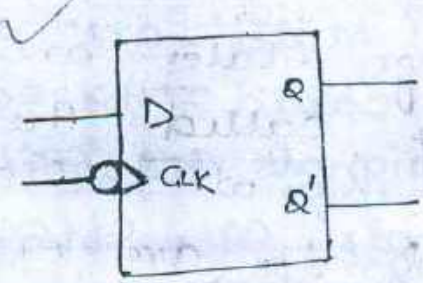


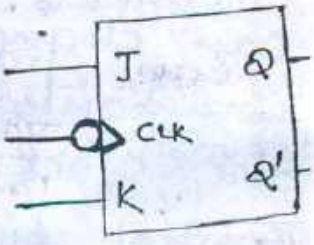
(S-R Flip-Flop)

(J-K Flip-Flop) (Logic symbol of positive edge-trigg Flip-Flops)

Negative-edge triggering is indicated by a triar with a bubble at the clock terminal of the Fli Flop.







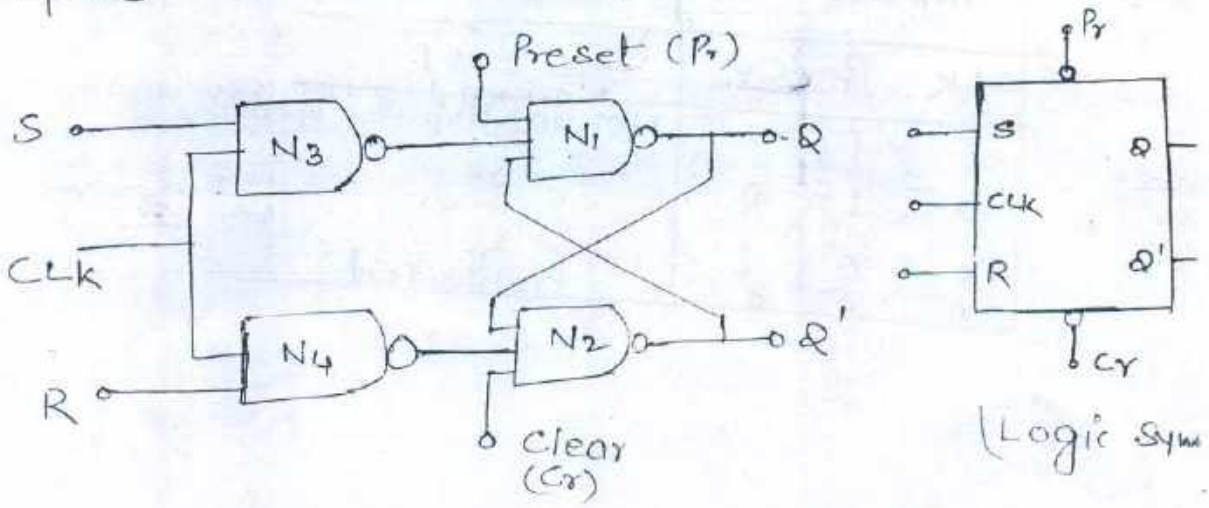
(S-R flip-flop) (D-Flip-Flop) (J-K flip-

(Logic symbol of Negative-edge triggered Flip-Flops)

The clocked Flip-Flops introduced earlier a triggered during the positive edge of the pulse and the state transition starts as soon as the pulse reaches the logic-1 level. As it is cl that, a sequential circuit has a feedback pat between the combinational circuit and memory elements. This feed back path can produce instability, in the case of level triggering. A better way to solve the feedback timing problem is to make the Flip-Flop sensitive to the pulse transition rather than the pulse duration.

Asynchronous Inputs —

In the clocked flip-flo the S, R, D and J-K inputs are called synchron inputs, because their effect on the Flip-Flop output is synchronized with the clock input These synchronous inputs must be used in conjun with a clock signal to trigger the Flip-Flop. In many applications it is desired to initially set(1) or reset(0) the flip-flops that is the initial sta of the Flip-flop is to be assigned. This is done b using the asynchronous inputs or direct inputs know as preset (Pr) and clear (Cr) inputs. These asynch nous inputs affect the Flip-Flop output independe of the synchronous inputs and the clock input. Th inputs may be applied at any time between clo pulses and are not in synchronism with the clock. An S-R Flip-Flop with preset and clear inputs is shown below.



(An S-R Flip-Flop with direct inputs)

These direct inputs can be used to set the Flip-flop to the 1 state or reset the Flip-flo to the 0 state at any time regardless of the condi at the other inputs. If Pr = Cr = 1 the circuit operates in normal way and Flip-Flop responds

to the S, R and clock inputs.

When $P_r = 0$ and $C_r = 1$, the output of NAND gate N_1 (Q) is immediately SET to 1, no matter what conditions are present at the S, R and clock inputs.

When $P_r = 1$ and $C_r = 0$ the Flip-Flop is clear (that is $Q = 0$).

The condition $P_r = C_r = 0$ may not be used, since it results an invalid state. By direct input once the state of Flip-flop is established, the asynchronous (direct) inputs P_r and C_r must be connected to logic 1 before the application of next clock pulse. The operation of S-R Flip-Flop with Preset and clear inputs is summarized below.

Inputs			output
CLK	P_r	C_r	Q_{n+1}
1	1	1	Normal Flip-Flop operation
0	1	0	0
0	0	1	1
0	0	0	Prohibited

Excitation Table —

The characteristic table of a Flip-flop specifies the operational characteristics of flip-flop. It specifies the next state when the present state of the Flip-flop and inputs are known. During the design process of sequential circuits, usually the present state and the next state of the circuit are specified, and we have to find the input condition that will cause the required state transition. A tabulation of these input conditions is known as excitation table. It is very important and useful design aid for sequential circuits. By the present state and the next state we mean that the state of the circuit before t_0 and after the clock pulse respectively. For example, the output of an S-R Flip-flop before the clock pulse is $Q_n = 1$ and it is desired that the output does not change on the application of clock pulse what input conditions (the values of S_n & R_n) must exists to achieve this state transition?

From the characteristic table of an S-R Flip-flop we find the following input conditions

(a) $S_n = 0$, $R_n = 0$ (fifth row)

(b) $S_n = 1$ $R_n = 0$ (Seventh row)

From the above input conditions, we may conclude that the R_n must be zero whereas S_n input may be either 0 or 1 (don't care, x). Similarly for other possible transitions ($Q_n \rightarrow Q_{n+1}$) input conditions may be derived. The excitation table for the S-R Flip Flop is shown below.

Present state Q_n	Next state Q_{n+1}	S-R Flip Flop S_n	R_n
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

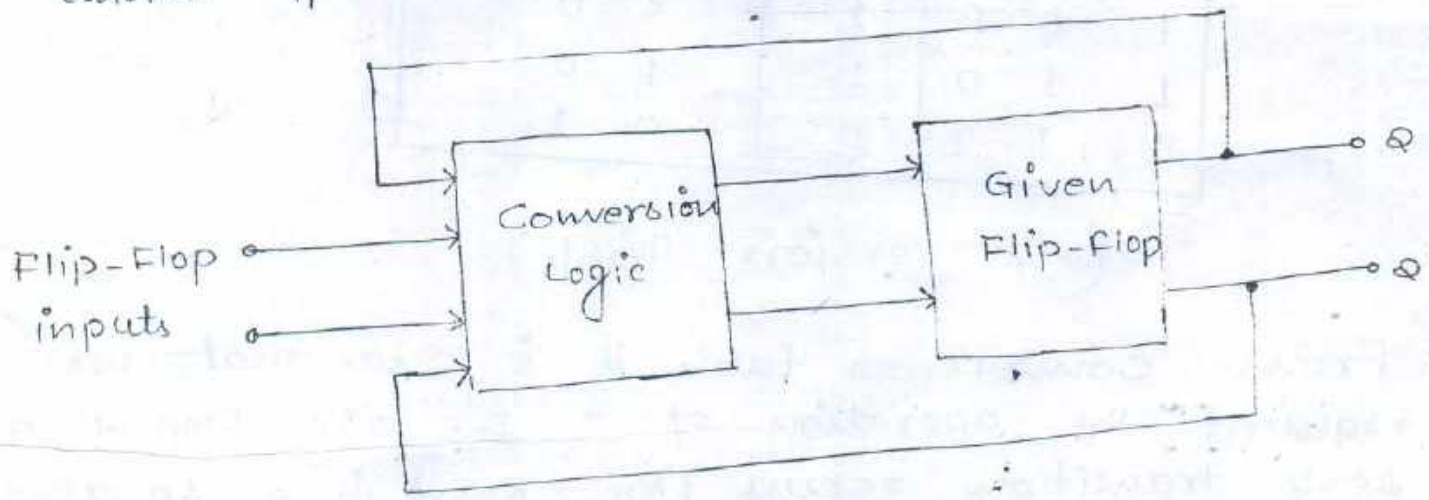
The first row shows that the Flip-Flop is in o
before the application of clock pulse, and it is desi
that output does not change when the clock pu
is applied. From the characteristic table of S-R flip
flip we find two input conditions (i) S = R = 0 & (ii) S = 0
R = 1, when the flip-flop will not change state. From
these input conditions we may conclude that
S must be 0 whereas R may be either 0 or 1, there
R is marked by the don't care condition x. /
If the flip-flop is in the 0-state and it is desi
to go it in the 1-state after the application of cloc
pulse, we must have S = 1 and R = 0. If the flip
is to have a transition from the (Q_n) 1-state t
the 0-state (Q_{n+1}), we must have S = 0 and R = 1

Excitation Table of J-K, T, and D Flip-Flops

Present state Q_n	Next state Q_{n+1}	J-K Flip Flop J_n	K_n	T Flip-Flop T_n	D Flip-Flop D_n
0	0	0	x	0	0
0	1	1	x	1	1
1	0	x	1	1	0
1	1	x	0	0	1

Conversion of Flip-Flops –

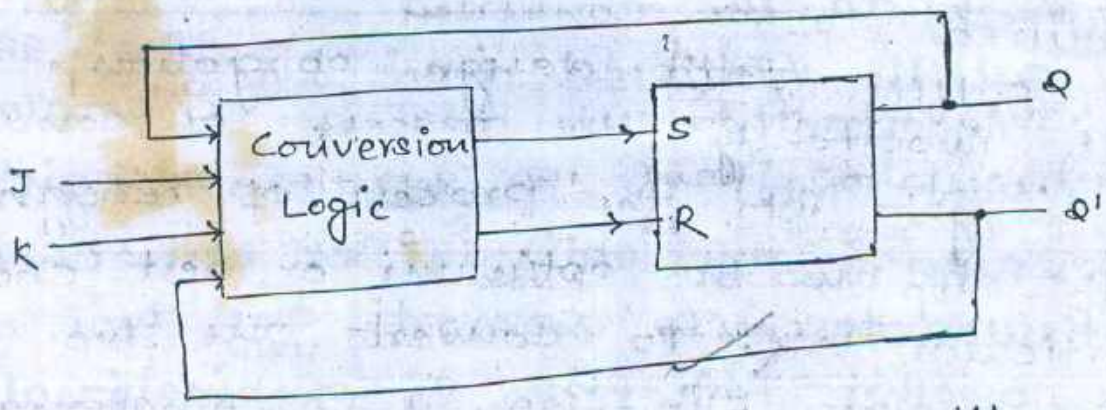
In previous section we studied about various type of Flip-Flops. There are times when one type of Flip-Flop is required to be converted to another in order to fulfill some design obligation. However, more importantly, the process of conversion provides an insight into the process of recognizing which Flip-Flop is the optimum or best choice for a particular task. To convert one type of Flip-Flop into another Flip-Flop, a combinational circuit is designed such that if the inputs of the required Flip-Flop are fed as inputs to the combinational circuit and the output of combinational circuit (conversion logic) is connected to the inputs of actual Flip-Flop then the output of the actual Flip-Flop is the output of required flip-flop. Excitation tables for both Flip-Flips are combined to prepare a truth table with data inputs and Q as inputs and input of the given Flip-Flop as the outputs. Generally model for conversion from one type of flip-flop to another type is shown below.



(General Model used for Conversion of Flip-Flops)

5.14 Conversion of S-R Flip-Flop to a J-K Flip-

General model used to convert an S-R Flip-F into J-K Flip-Flop is shown below.



In the conversion procedure, we will prepare a truth table with J, K, Q_n, Q_{n+1}, S and R. Where Q_n is the present state of the Flip-Flop and Q_{n+1} is the next state obtained when the particular J and K inputs are applied.

JK inputs			output	S-R inputs	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	x
0	0	1	1	x	0
0	1	0	0	0	x
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	x	0
1	1	0	1	1	0
1	1	1	0	0	1

(Conversion Table)

From conversion table it is clear that we required the operation of a J-K Flip-Flop, where state transition occurs ($Q_n \rightarrow Q_{n+1}$) in a specified manner. For such type of state transition what should be the values at S and R inputs. That is how much S and R inputs are responsible for such type of state transition.

Because S and R is the outputs of combinational logic, hence simplifying the truth table for S and R we get—

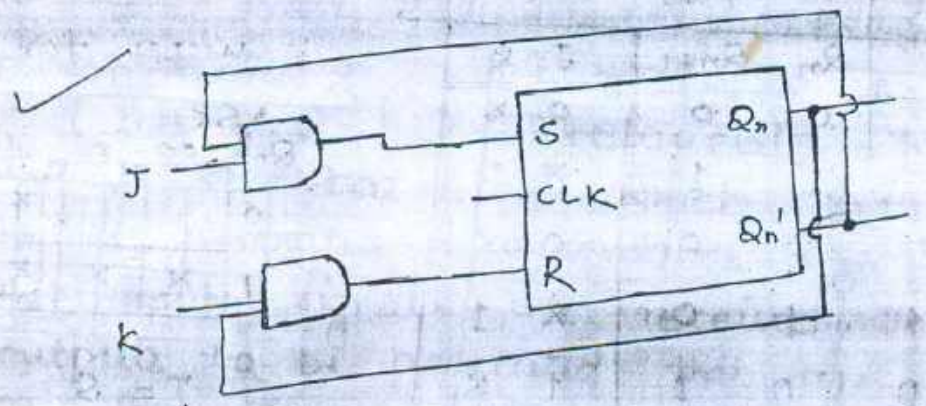
Q_n \ JK	00	01	11	10
0			1	1
1	X			X

$S = J Q_n'$

Q_n \ JK	00	01	11	10
0	X	X		
1		1	1	

$R = K Q_n$

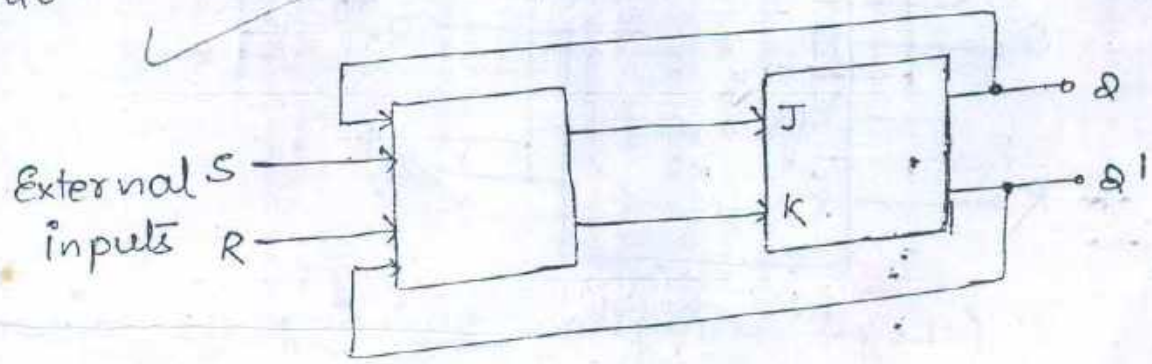
(K maps for S & R)



(Logic diagram of J-K flip-flop from S-R flip flop through conversion logic)

5.15 Conversion of J-K Flip-Flop to S-R Flip-Flop

General model for conversion of J-K flip-flop into S-R flip-flop is shown below.



In this case the outputs of combinational circ are the actual inputs to the J-K Flip-Flop an S, R becomes the external inputs. A truth tal is prepared using S, R, Q_n, Q_{n+1}, J and K. Here we get the values of J and K in terms of S. R and Q_n after simplification. In the S-R Flip Flop, the input values S=R=1 is not allowed, so the corresponding J and K are don't care. The corresponding conversion table, and the logi diagram are shown below.

External inputs		outputs		J-K inputs	
S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	0	X	X	X
1	1	1	X	X	X

(Conversion Table)

K map for J

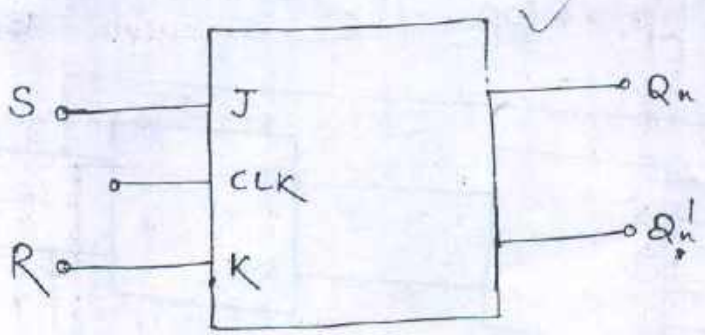
Q_n \ SR	00	01	11	10
0			X	1
1	X	X	X	X

J = S

K map for K

Q_n \ SR	00	01	11	10
0	X	X	X	X
1		1	X	

K = R



(Logic diagram showing the conversio from J-K Flip-Flop to S-R Flip-Flop)

5.16 Conversion From S-R Flip-Flop to D-Flip-Flop

In this case D is treated as external input to the Flip-Flop and S, R as the output of the conversion logic. Here S and R will be expressed in terms of D and Qn. The conversion procedure is shown below.

D	Qn	Qn+1	S	R
0	0	0	0	x
0	1	0	0	1
1	0	1	1	0
1	1	1	x	0

(Conversion Table)

K map for S

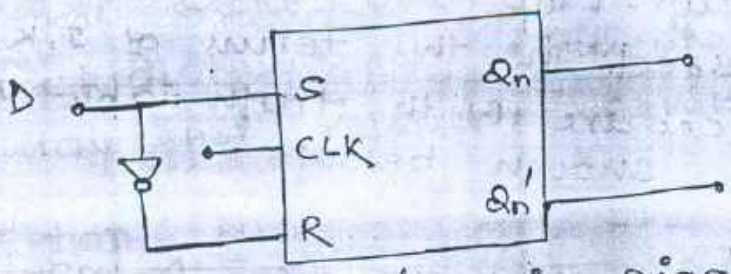
Qn \ D	0	1
0		1
1		x

$S = D$

K map for R

Qn \ D	0	1
0	x	
1	1	

$R = D'$



(Logic Diagram)

5.17 Conversion From J-K Flip-Flop to D flip-Flop

Here D is the external input and J, K are the output of conversion logic and actual input to the Flip-Flop. J and K are expressed in the terms of D input and Qn. The conversion table and logic diagram are shown below.

			J	K
0	0	0	0	x
0	1	0	x	1
1	0	1	1	x
1	1	1	x	0

K map for J

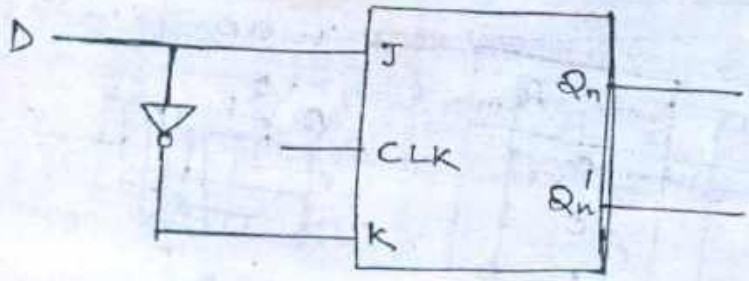
Q_n \ D	0	1
0		1
1	x	x

$J = D$

K map for K

Q_n \ D	0	1
0	x	x
1	1	

$K = D'$



(Logic diagram)

5.18 Conversion from D flip-flop to J-K flip-flop

Here D is the output of combinational logic & actual input to the flip-flop and J, K are external inputs. D is expressed in the terms of J, K & Q_n. The conversion procedure with truth table and logic diagram are shown below.

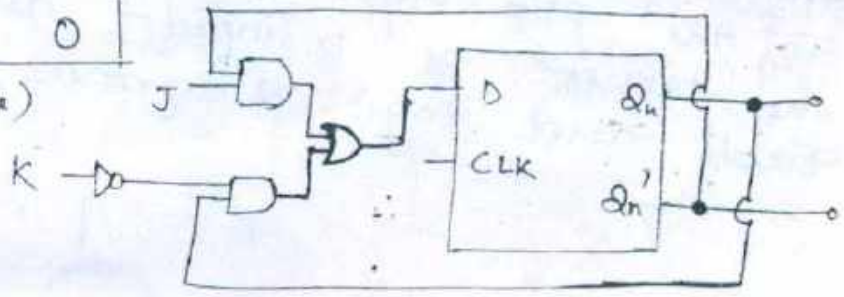
J	K	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

(Conversion Table)

K map for D input

Q_n \ JK	00	01	11	10
0			1	1
1	1			1

$D = J Q_n' + K' Q_n$

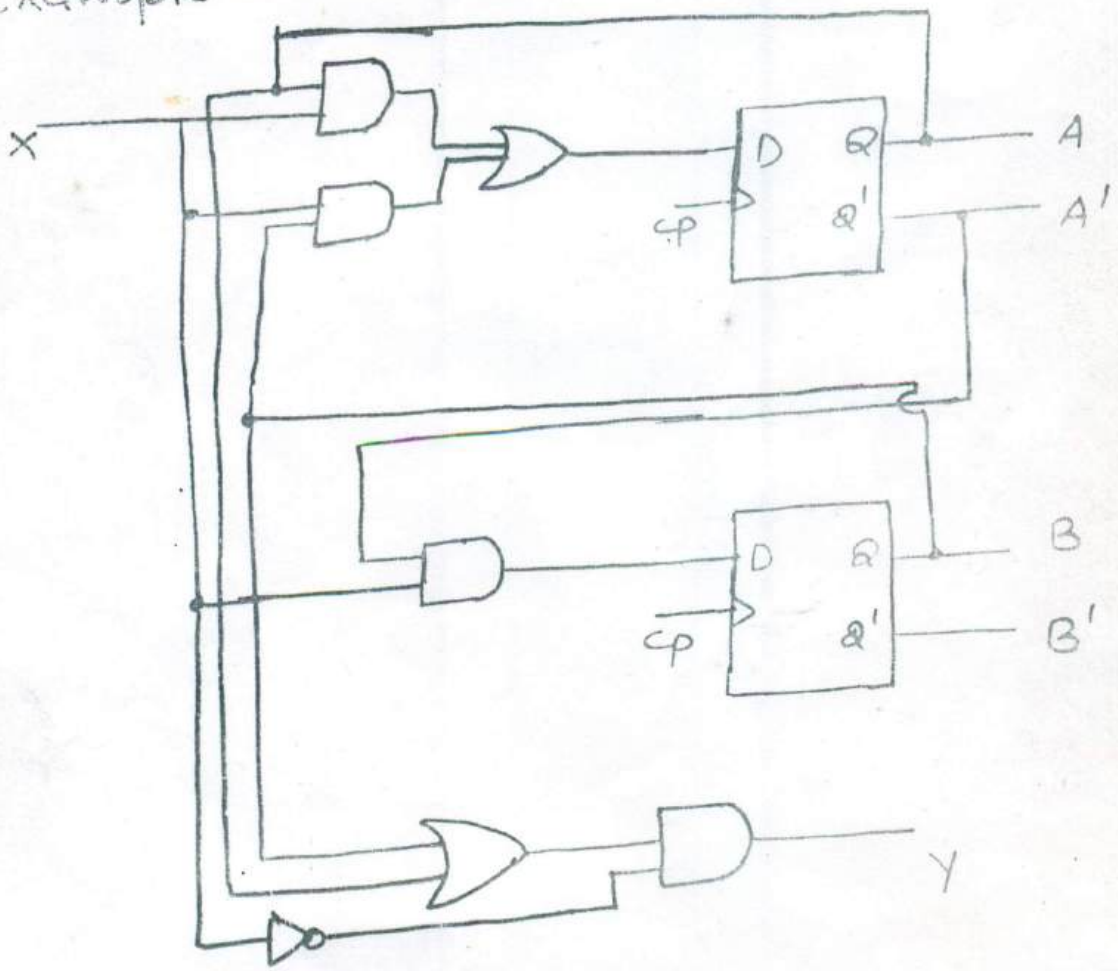


(Logic diagram)

Analysis of clocked Sequential Circuits —

The analysis of sequential ckt. consists of obtaining a Table or a diagram for the time sequence of inputs outputs and internal states.

Example —



Next state Eqn for the circuit is

$$A(t+1) = A(t)\,x(t) + B(t)\,x(t)$$

$$B(t+1) = A'(t)\,x(t)$$

state equation is an algebric expression that specifies the condition for a Flip-Flop state transition. state eqn can be written in more compact form as

$$A(t+1) = Ax + Bx$$

of sequential ckt for the time sequ

boolean function can be expressed as

$$Y = (A+B)X'$$

where X is external input

State Table – The state table is tabular representation of the relationship between the present state, input, the next state and output. The table consists of four sections labeled pr. state, input, next state and output. State table for the above circuit is prepared below.

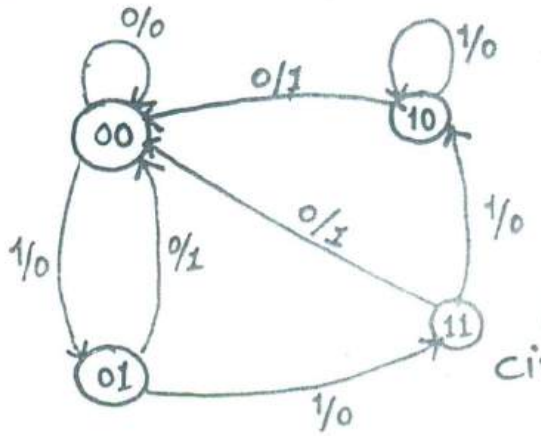
Present state A B	input X	Next state A* B*	output Y
0 0	0	0 0	0
0 0	1	0 1	0
0 1	0	0 0	1
0 1	1	1 1	0
1 0	0	0 0	1
1 0	1	1 0	0
1 1	0	0 0	1
1 1	1	1 0	0

A sequential ckt with m flip-flops and n inputs need 2^{m+n} rows in the state table. The binary values for the next state are derived directly fro the state equations. The binary values for outp variables is derived from the ckt or boolean express State table may be represented in another form where it has only three sections present state, next state and output.

Present state A B	Next state A* B* X=0	Next state A* B* X=1	output (Y) X=0 Y	output (Y) X=1 Y
0 0	0 0	0 1	0	0
0 1	0 0	1 1	1	0
1 0	0 0	1 0	1	0

of ... sequential ckt. consists of ... time sequence of inputs

STATE DIAGRAM — It is pictorial (graphical) representation of the information available in a state table. In state diagram, a state is represen by a circle and transition between states is indic by directed lines (or arcs) connecting the circles. Th directed lines are lebeled with two binary numbe seperated by a slash. The input value during t pr. state is lebeled first and the number after th slash gives the o/p during the pr. state. State diagr of above sequential ckt. is shown below.



The state diagram gives a pictorial view of state transit and is the form suitable for humai interpretation of th circuit operation.

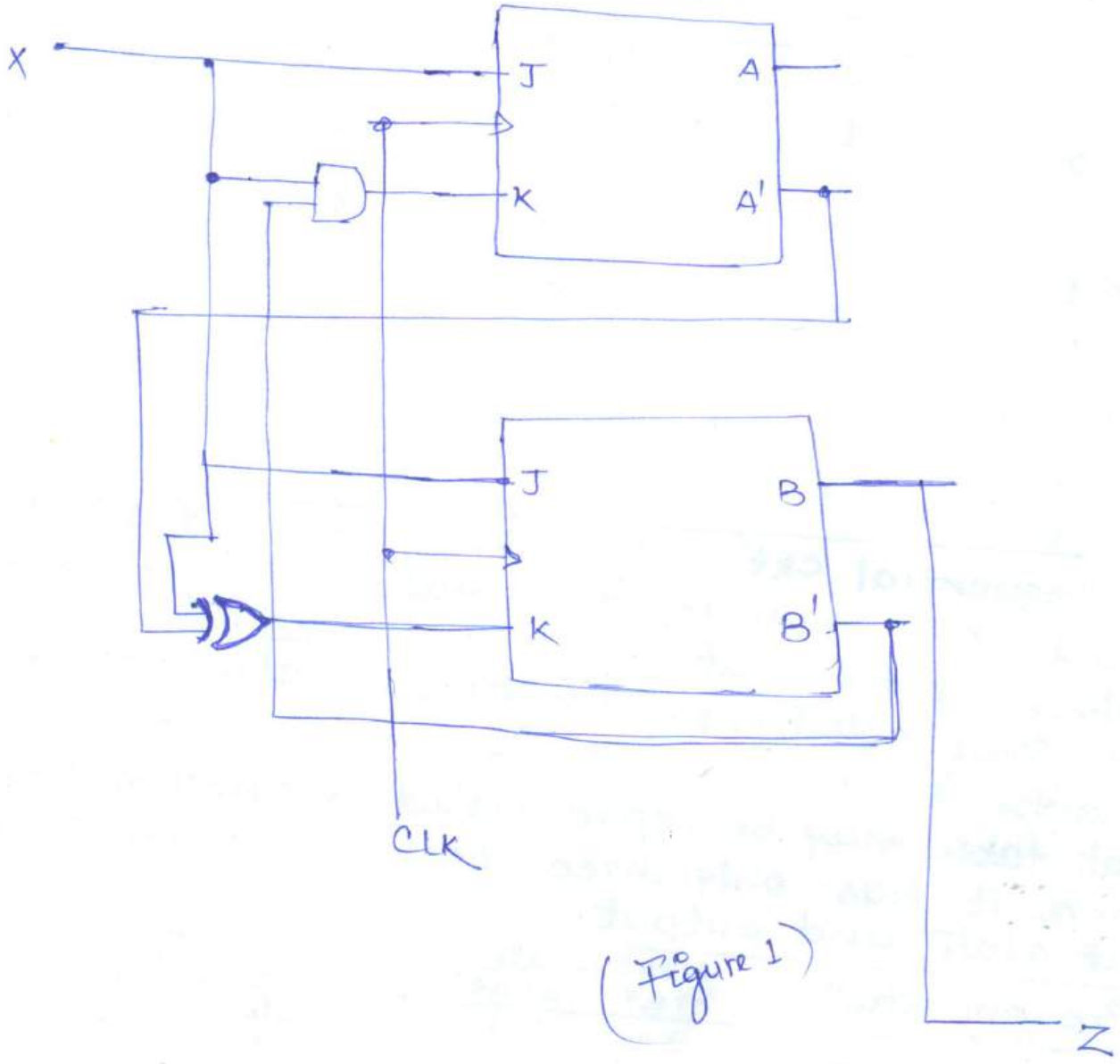
(Figure 1)

Analysis with J-K Flip-flops:— Next state val of a sequential circuit with D-flip-flops can derived directly from the next state equations (Eq In the sequential circuits designed with other flip- the next state can be derived by obtaining the bin values of each flip-Flop input functions in term of the present state and input variables. Then b using characteristic table of corresponding Flip-Fl next state value is determined.

consider an example — A sequential ckt. with two J-K flip-flops A and B, one input x, and output Z. The circuit can be specified by the fol F.F. input functions.

$J_A = x$ $\quad J_B = x$ $\quad \& \; Z = B$

$K_A = B'x$ $\quad K_B = x \oplus A' = A'x + Ax'$



(Figure 1)

of clock ... 1 cut consists of obtaining

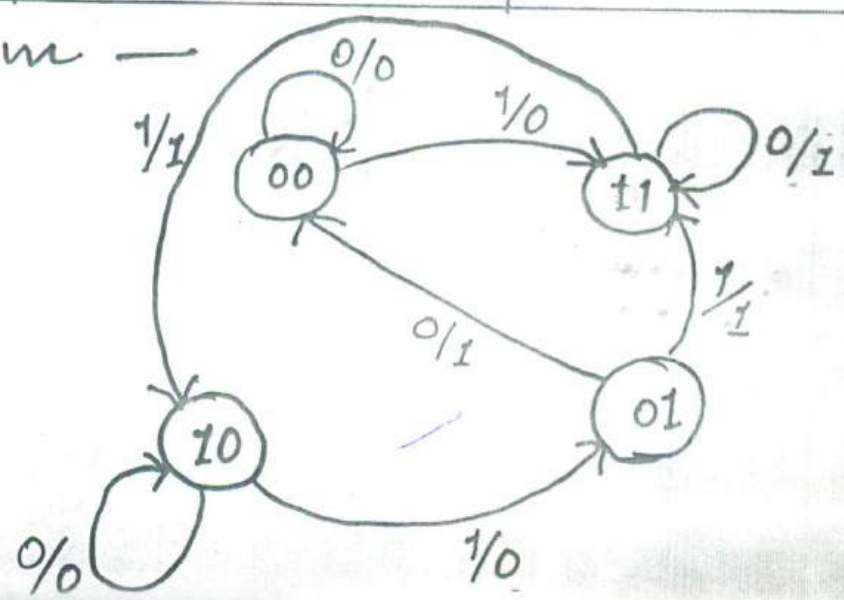
State Table for the ckt with J-K flip-flop —

Pr state A	B	Inputs X	Next state A^*	B^*	o/p Z	Flip-Flop inputs J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	0	0	1
0	0	1	1	1	0	1	1	1	0
0	1	0	0	0	1	0	0	0	1
0	1	1	1	1	1	1	0	1	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	1	0	1	1	1	1
1	1	0	1	1	1	0	0	0	0
1	1	1	1	0	1	1	0	1	1

Simplified state table

Present state AB	Next state A^* B^* X=0	X=1	output (Z) X=0	X=1
00	00	11	0	0
01	00	11	1	1
10	10	01	0	0
11	11	10	1	1

State Diagram —



Counters — *A sequential ckt that goes through a prescribed sequence of states upon the application of input pulses is called a counter. The input pulses called count pulses, may be clock pulses or they may originate from an external source and may occur at prescribed intervals of time or at random.

: counters are used for counting the number of occurrences of an event.

* Counters are used as frequency divider to obtain waveforms with frequencies that are specific fractions of the clock frequency.

Type — (a) Synchronous (b) Asynchronous

a) In a synchronous counter, the input pulses are applied to all CP inputs of all flip-flops. The common pulse triggers all the Flip-flops simultaneously. The next state of the flip-flop is determined from the values of flip-flop inputs at the time of clock pulse.

b) Asynchronous counters are also called as ripple counters. Ripple counters are easiest to design and requires less hardware. Flip-Flops within the ripple counter are not trigge simultaneously. In it first stage (L.S.B) causes the second to Flip (start), the second causes the third to Flip, and the third causes to fourth Flip and so on. Ripple counters are also calle series counter.

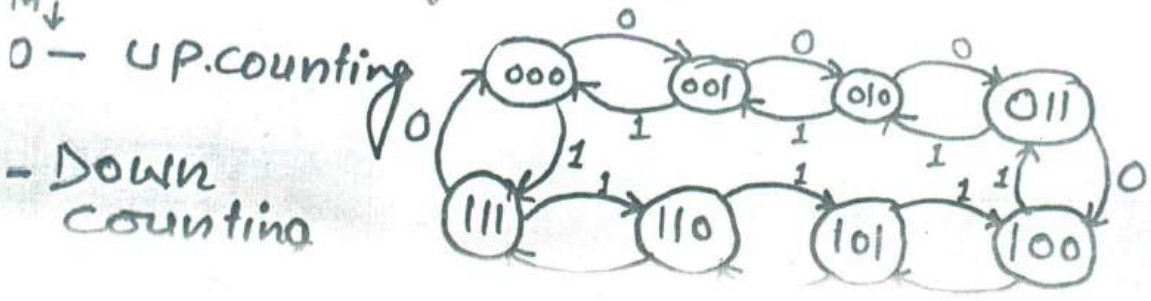
Ex:— Design a 3-bit binary UP/Down Counter (7) with a direction control M using J-K Flip-Flop

When direction control input M=0, it acts as an up counter. When M=1 it acts as down counter.

Excitation Table

Control input M	Present state Q_2	Q_1	Q_0	Next state Q_2^*	Q_1^*	Q_0^*	flip-flop inputs J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	1	0	x	0	x	1	x
0	0	0	1	0	1	0	0	x	1	x	x	1
0	0	1	0	0	1	1	0	x	x	0	1	x
0	0	1	1	1	0	0	1	x	x	1	x	1
0	1	0	0	1	0	1	x	0	0	x	1	x
0	1	0	1	1	1	0	x	0	1	x	x	1
0	1	1	0	1	1	1	x	0	x	0	1	x
0	1	1	1	0	0	0	x	1	x	1	x	1
1	0	0	0	1	1	1	1	x	1	x	1	x
1	1	1	1	1	1	0	x	0	x	0	x	1
1	1	1	0	1	0	1	x	0	x	1	1	x
1	1	0	1	1	0	0	x	0	0	x	x	1
1	1	0	0	0	1	1	x	1	1	x	1	x
1	0	1	1	0	1	0	0	x	x	0	x	1
1	0	1	0	0	0	1	0	x	x	1	1	x
1	0	0	1	0	0	0	0	x	0	x	x	1
	0	0	0									

State Diagram

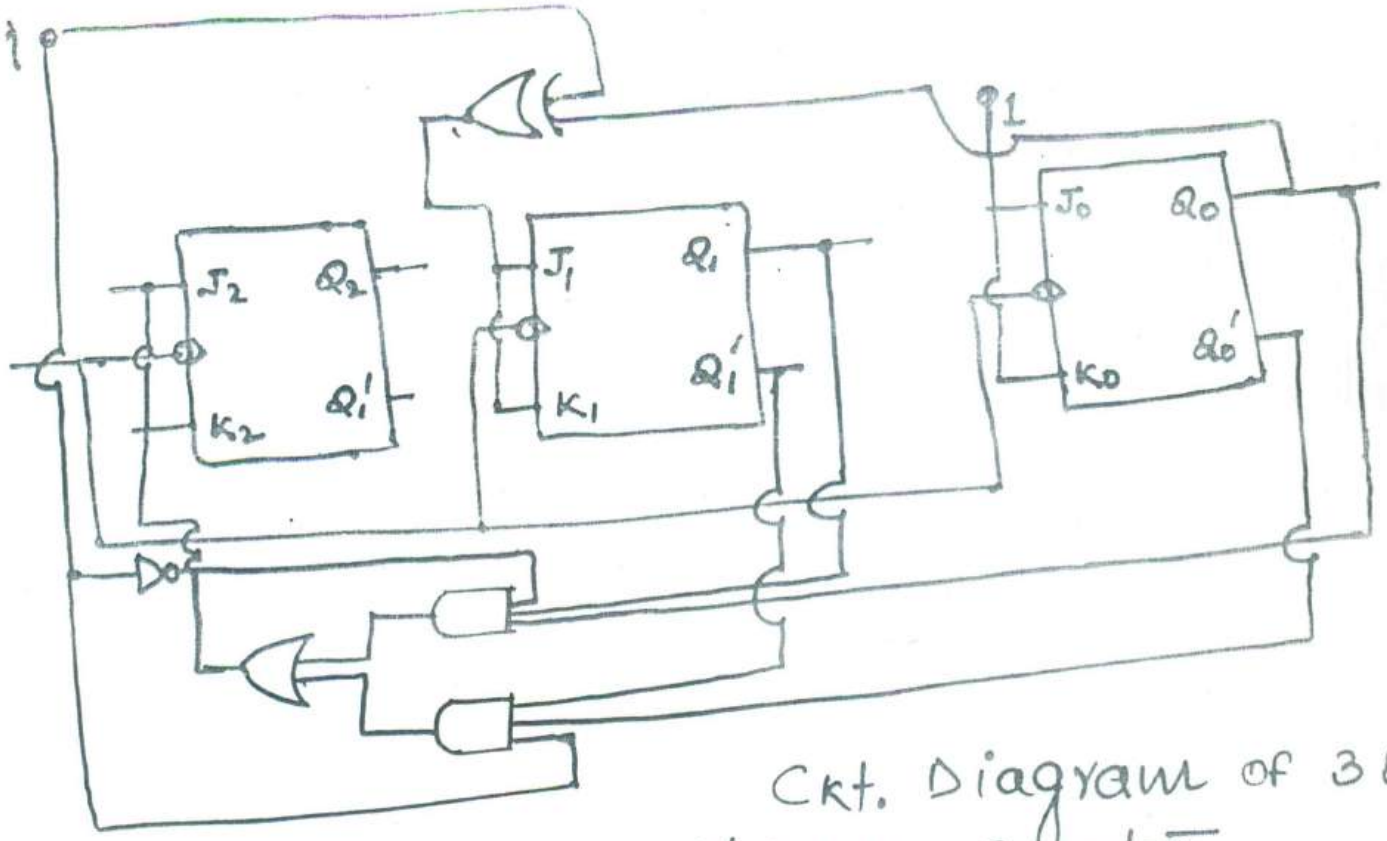


From Table, we can find after simplication the values of Flip-flop inputs as

$$J_0 = K_0 = 1$$

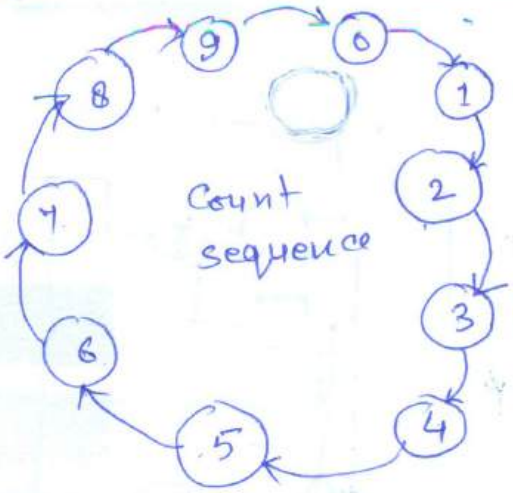
$$J_1 = K_1 = Q_0 M' + Q_0' M$$

$$J_2 = K_2 = M' Q_1 Q_0 + M Q_1' Q_0'$$



Ckt. Diagram of 3 bit - up/ Down counter.

BCD Counter



Excitation Table for BCD counter

Pr. state	Next state	output	Flip-flop inputs			
$Q_8\ Q_4\ Q_2\ Q_1$	$Q_8\ Q_4\ Q_2\ Q_1$	Y	T_8	T_4	T_2	T_1
0 0 0 0	0 0 0 1	0	0	0	0	1
0 0 0 1	0 0 1 0	0	0	0	1	1
0 0 1 0	0 0 1 1	0	0	0	0	1
0 0 1 1	0 1 0 0	0	0	1	1	1
0 1 0 0	0 1 0 1	0	0	0	0	1
0 1 0 1	0 1 1 0	0	0	0	1	1
0 1 1 0	0 1 1 1	0	0	0	0	1
0 1 1 1	1 0 0 0	0	1	1	1	1
1 0 0 0	1 0 0 1	0	0	0	0	1
1 0 0 1	0 0 0 0	1	1	0	0	1

obtaining minimal expression for T_8, T_4, T_2 & T_1 by K-map we get

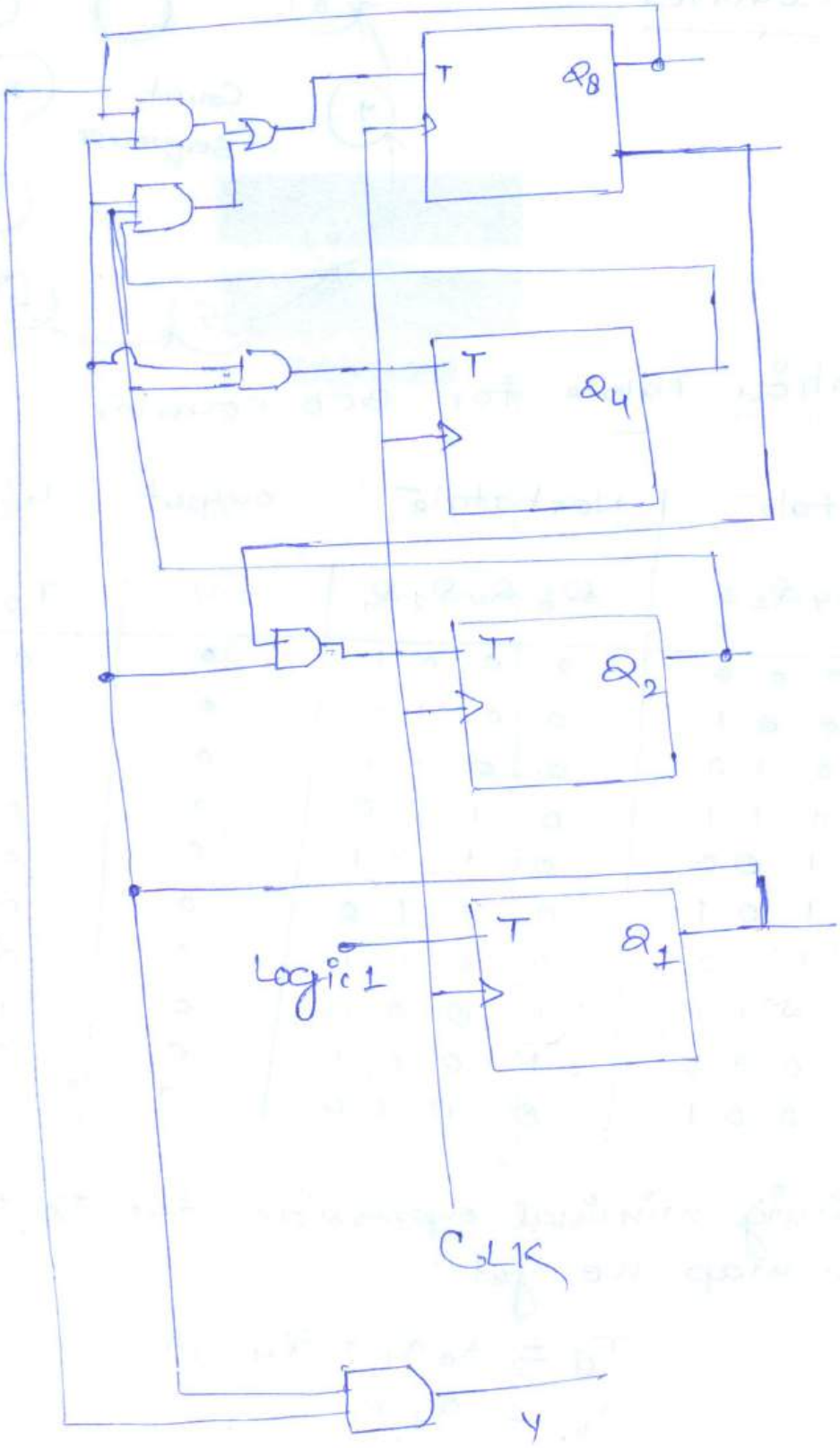
$$T_8 = Q_8 Q_1 + Q_4 Q_2 Q_1$$

$$T_4 = Q_2 Q_1$$

$$T_2 = Q_8' Q_1$$

$$T_1 = 1$$

$$Y = Q_8 Q_1$$



Logic diagram

Self correcting Counter

Design and Implement a BCD counter using J-K flip-flops. Is the counter self-correcting.

BCD counter or Mod-10 Counter.

Excitation requirements

Present state				Next state				Required Excitation F. F. inputs							
Q_4	Q_3	Q_2	Q_1	Q_4^*	Q_3^*	Q_2^*	Q_1^*	J_4	K_4	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	0	0	1	0	x	0	x	0	x	1	x
0	0	0	1	0	0	1	0	0	x	0	x	1	x	x	1
0	0	1	0	0	0	1	1	0	x	0	x	x	0	1	x
0	0	1	1	0	1	0	0	0	x	1	x	x	1	x	1
0	1	0	0	0	1	0	1	0	x	x	0	0	x	1	x
0	1	0	1	0	1	1	0	0	x	x	0	1	x	x	1
0	1	1	0	0	1	1	1	0	x	x	0	x	0	1	x
0	1	1	1	1	0	0	0	1	x	x	1	x	1	x	1
1	0	0	0	1	0	0	1	x	0	0	x	0	x	1	x
1	0	0	1	0	0	0	0	x	1	0	x	0	x	x	1

On solving, using four variable K-maps we get flip-flop input values as

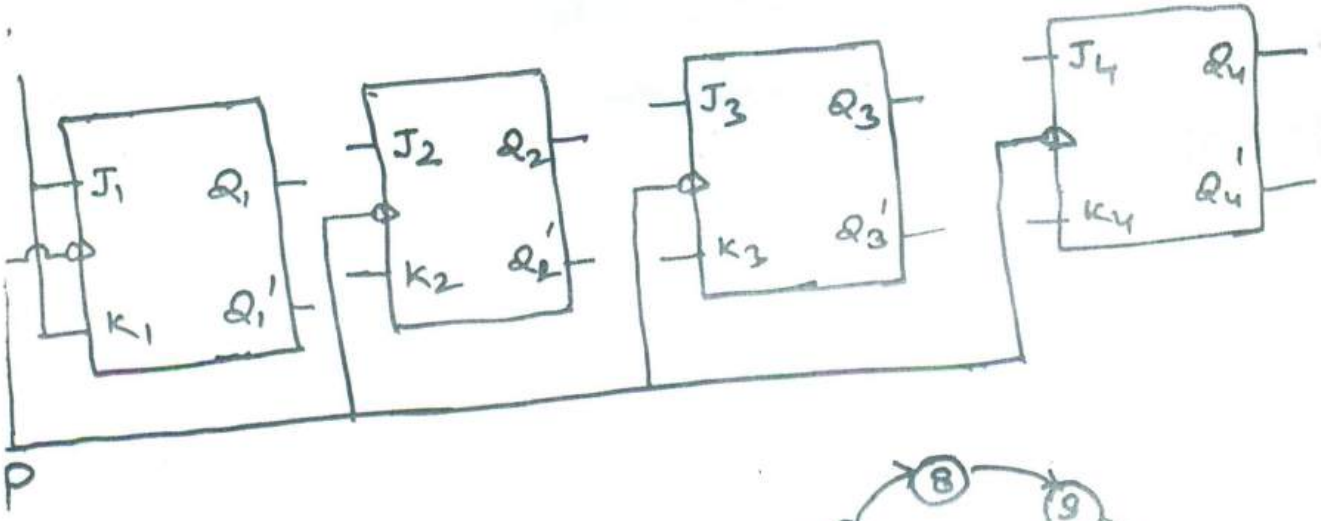
$J_4 = Q_3 Q_2 Q_1$ $\qquad$ $J_2 = Q_4' Q_1$

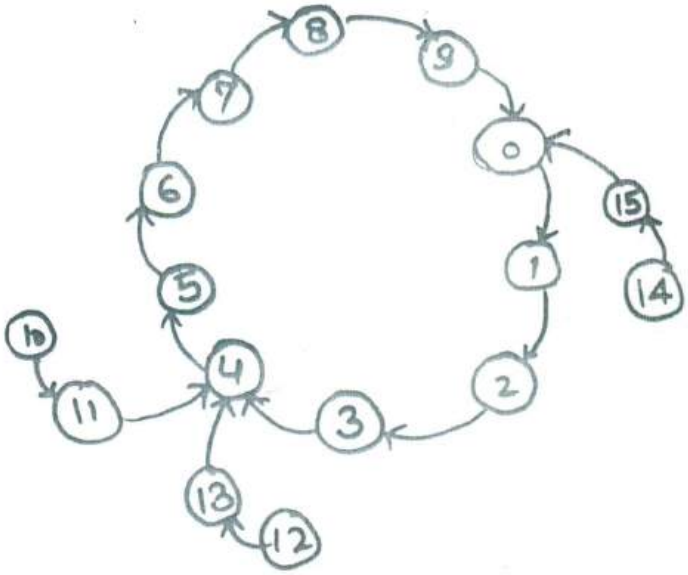
$K_4 = Q_1$ $\qquad$ $K_2 = Q_1$

$J_3 = Q_2 Q_1$ $\qquad$ $J_1 = K_1 = 1$

$K_3 = Q_2 Q_1$

Ckt. Diagram of BCD counter —



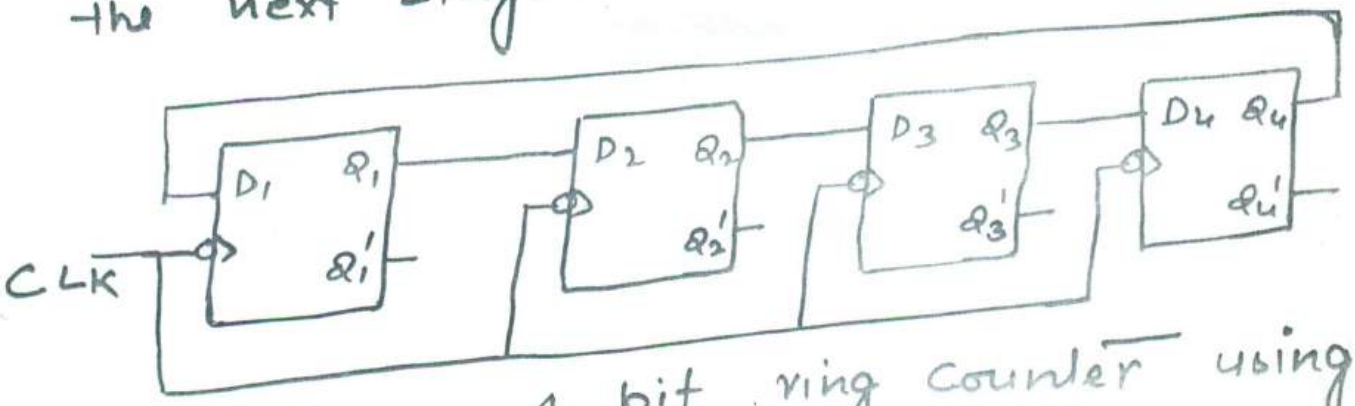


Q_4	Q_3	Q_2	Q_1	Present Inputs J_4K_4	J_3K_3	J_2K_2	J_1K_1	Next state Q_4	Q_3	Q_2	Q_1
1	0	1	0	0 0	0 0	0 0	1 1	1	0	1	1
1	0	1	1	0 1	1 1	0 1	1 1	0	1	0	0
1	1	0	0	0 0	0 0	0 0	1 1	1	1	0	1
1	1	0	1	0 1	0 0	0 1	1 1	0	1	0	0
1	1	1	0	0 0	0 0	0 0	1 1	1	1	1	1
1	1	1	1	1 1	1 1	0 1	1 1	0	0	0	0

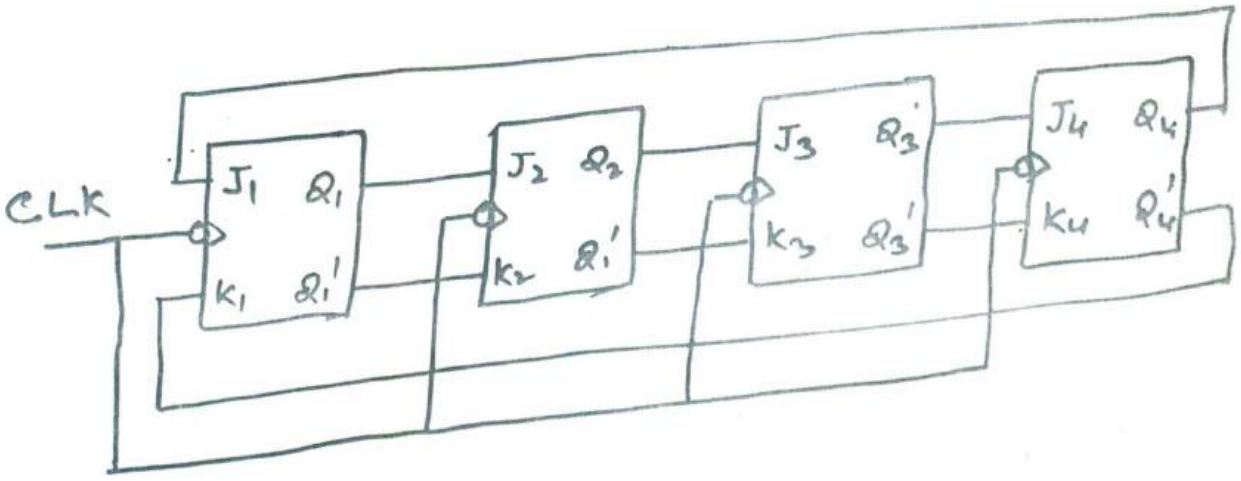
Skip counter —

Programmable Counter

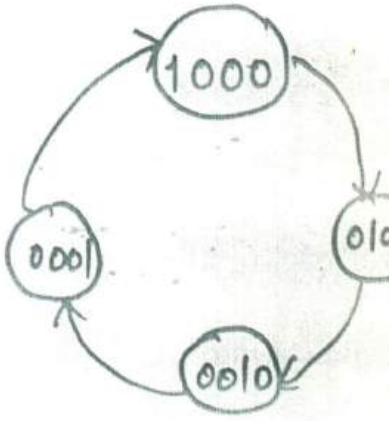
Ring counter → * simplest shift register counter
* o/p of each is connected to the input of the next stage.



4 bit ring counter using Dff.



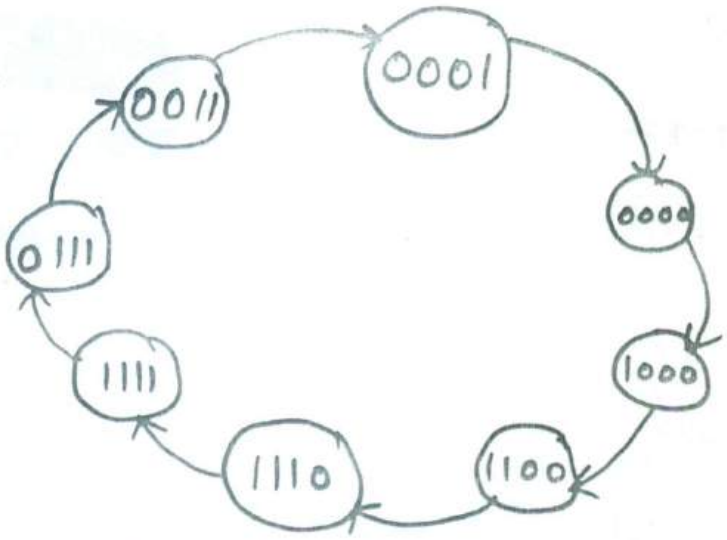
Q_1	Q_2	Q_3	Q_4	After CLK
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7



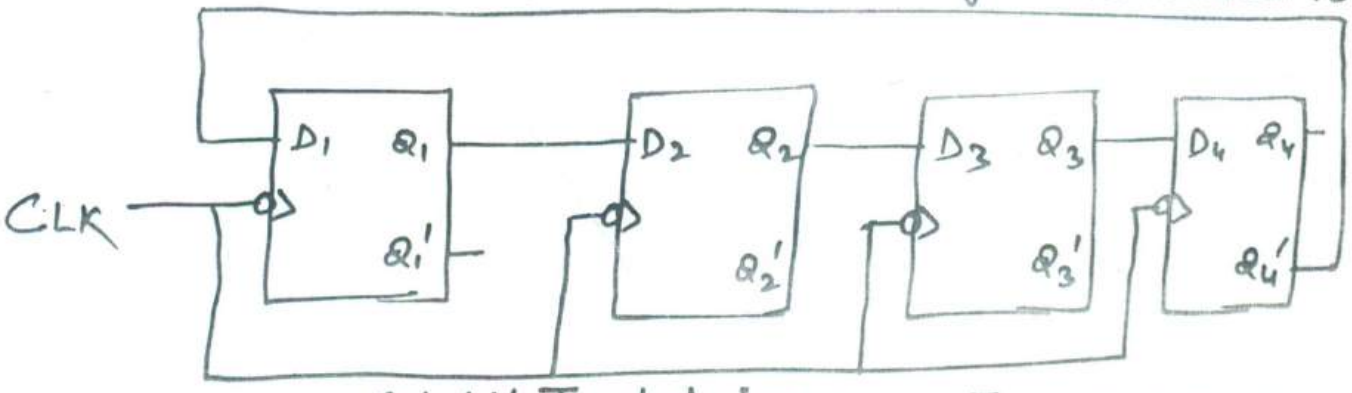
Twisted Ring counter - (Johnson Counter)

Sequence Table

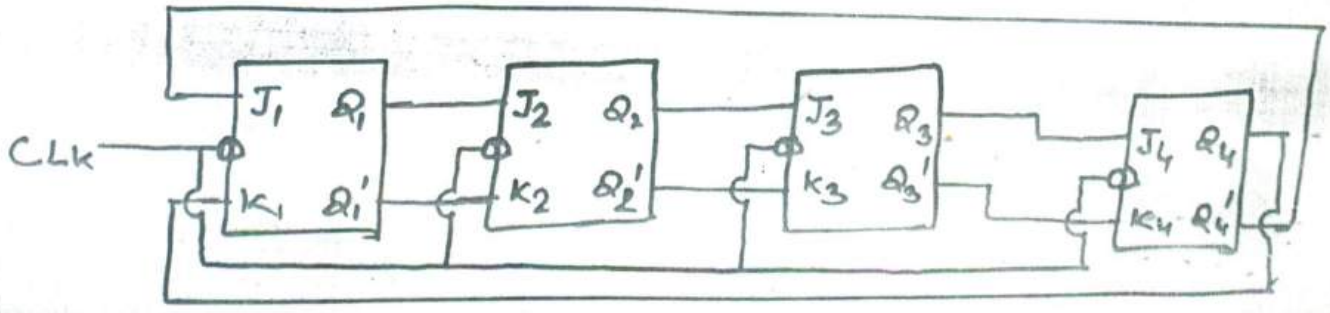
Q_1	Q_2	Q_3	Q_4	After CP
0	0	0	0	0
1	0	0	0	1
1	1	0	0	2
1	1	1	0	3
1	1	1	1	4
0	1	1	1	5
0	0	1	1	6
0	0	0	1	7
0	0	0	0	8
1	0	0	0	9



This counter is obtained from a serial-in, serial out shift register by providing a feed back from the inverted output of the last F.f. to the D-input of the first f.f. The Q output of each stage is connected to the D input of next stage but the Q' output of the last stage is connected to the D input of first stage, therefore the name Twisted ring Counter



(4-bit Twisted ring Counter with D f.f.)



An n f.f. Johnson counter can have 2n unique states and can count up to 2n pulses. So it is a mod-2n counter. An AND is used to avoid the

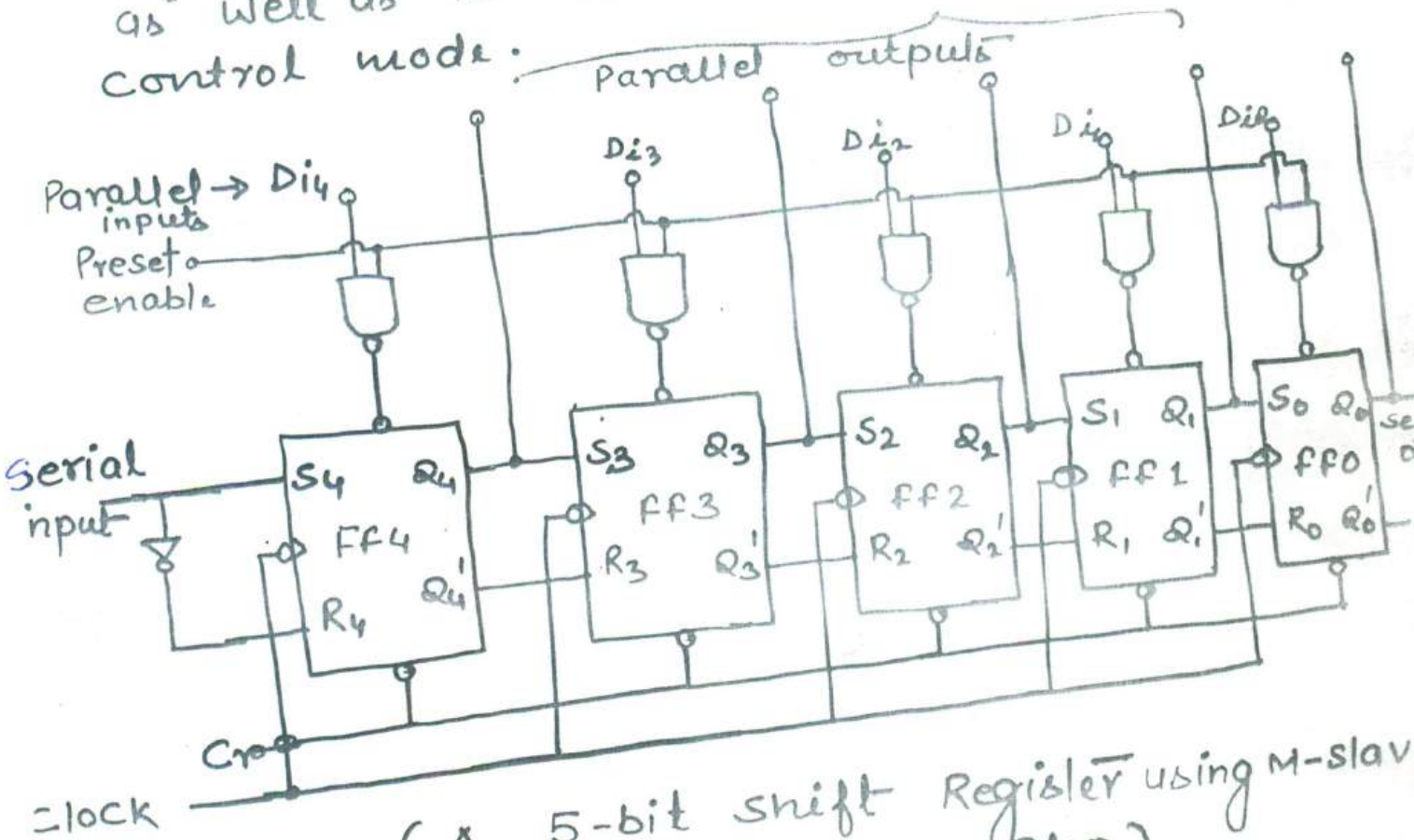
Shift Registers —

- Data may be available in parallel form or in serial form through registers.
* The data is said to be in serial form when the data bits appear sequentially (one after the other in time) at a single terminal.
* parallel data transfer is the simultaneous transmission of all bits of data from one device to another.
* A register is a set of flip-flops used to store binary data. The storage capacity of a register is the number of bits (1s and 0s) of digital data it can retain.

Shift registers are logic ckts used basically for the storage and transfer of digital data. The basic difference between a shift register and a counter is that, a shift register has no specified sequence of (inputs) states except in certain very specialized applications, whereas a counter has a specified sequence of states.

* A shift register also forms the basis for some very important arithematic operations, e.g. division, multiplication and complementation are frequently implemented by means of a register.
* A shift register can also be connected to form a number of different types of counters.
* Registers in which data are entered or/and taken out in serial form are referred to as shift registers

* Since bits are shifted in the flip-flops with the occurrence of clock pulses either in the right direction (right shift register) or in the left direction (left shift register). In the bi-directional shift register, data can be shifted from left to right as well as in the reverse direction using the control mode:



(A 5-bit shift Register using M-slave S-R flip-flop)

* This ckt can be used in any of the four modes (SISO, SIPO, PISO, PIPO).
* The operation can be explained by assuming the 5-bit data 10110.
* The process of entering the digital word starts with the data input corresponding to the LSB at the serial input and first clock pulse. At the falling edge of the first clock pulse the output of FF4 (Q4) will be 0 and the outputs of other flip

Next, the input corresponding to next bit is applied and, at the falling edge of the second clock pulse, the flip-flop outputs will be

$$Q_4 = 1$$
$$Q_3 = Q_2 = Q_1 = Q_0 = 0$$

The input corresponding to each bit is applied till the MSB and bits go on shifting from lef to right. At the end of fifth clock pulse the outputs of flip-flops are

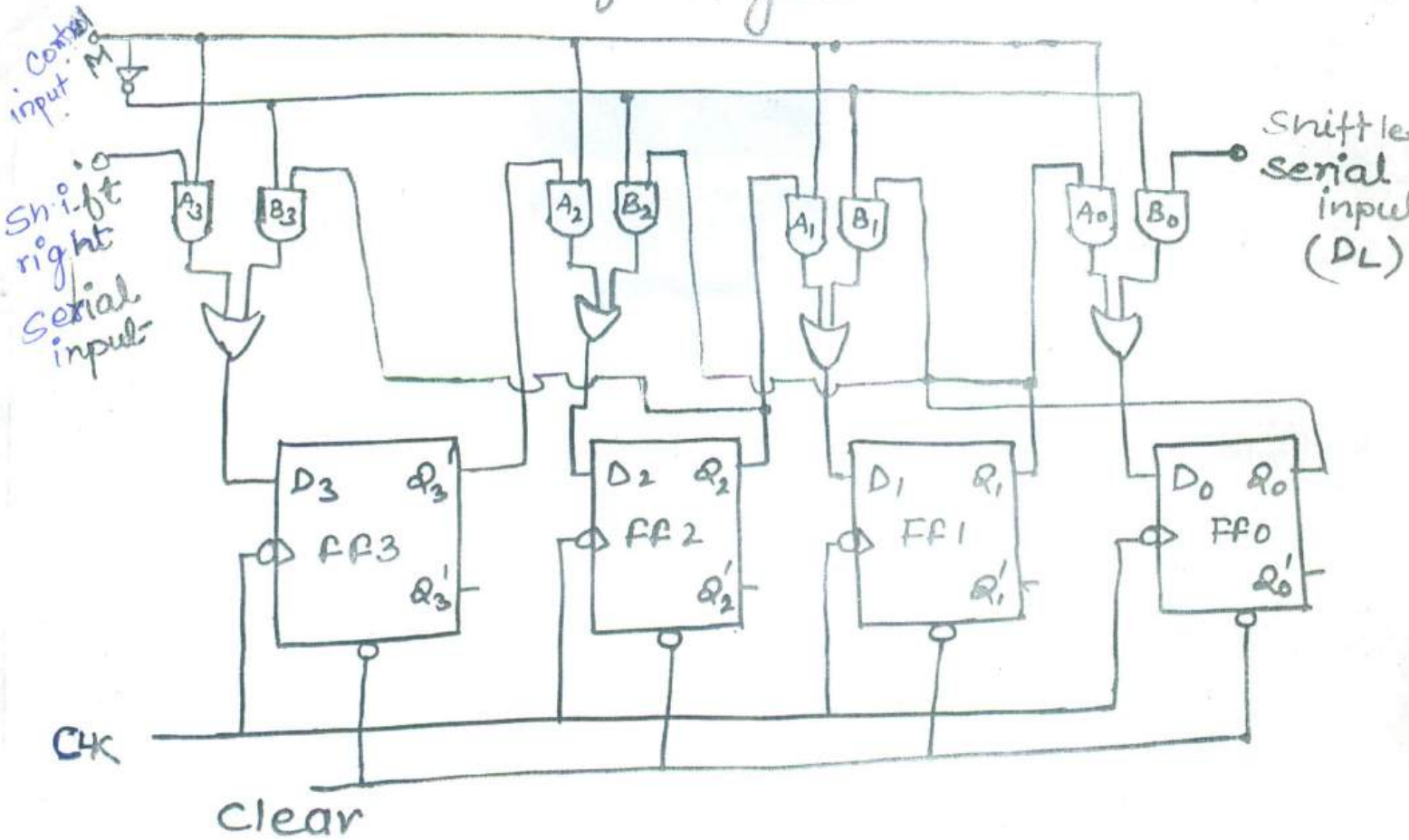
$$Q_4 = 1 \quad Q_3 = 0, \; Q_2 = 1 \; Q_1 = 1, \; Q_0 = 0$$

which is same as the number to be stored.

* The process of entering the data is also referre to as <u>writing</u> into the register.
* The data stored can be retrieved (reading) in tw ways <u>serial out</u> and <u>parallel out</u>.
* The no. of clock pulses required to obtain the da at Q_0 will be same as the no. of bits.
* In parallel form of reading the data is availab at Q_4 Q_3 Q_2 Q_1 Q_0 and extra clock is not required.

<u>Parallel input</u> – Data can be entered in the parallel form making preset input equal to 1. After clearing the Flip-flops, if the data lines a connected to the parallel inputs (D_{i4}, D_{i3}, D_{i2}, D and a 1 is applied to the preset input, the da are written into the register. It is referred to as asynchronous loading.

Bi-directional shift Register —



* When the control Mode M = 1, all the A AND gates are enabled and the data at D_R is shifted to the right when clock pulses are applied.
* When M = 0, A gates are inhibited and B gates are enabled allowing the data at D_L to be shifted to the left. M should be changed only when CLK = 0, otherwise the data stored in the register may be altered.

Application of shift Register —

1. Delay line
2. Serial to parallel converter
3. Parallel to serial converter
4. Ring counter
5. Twisted Ring Counter
6. Sequence Generator*

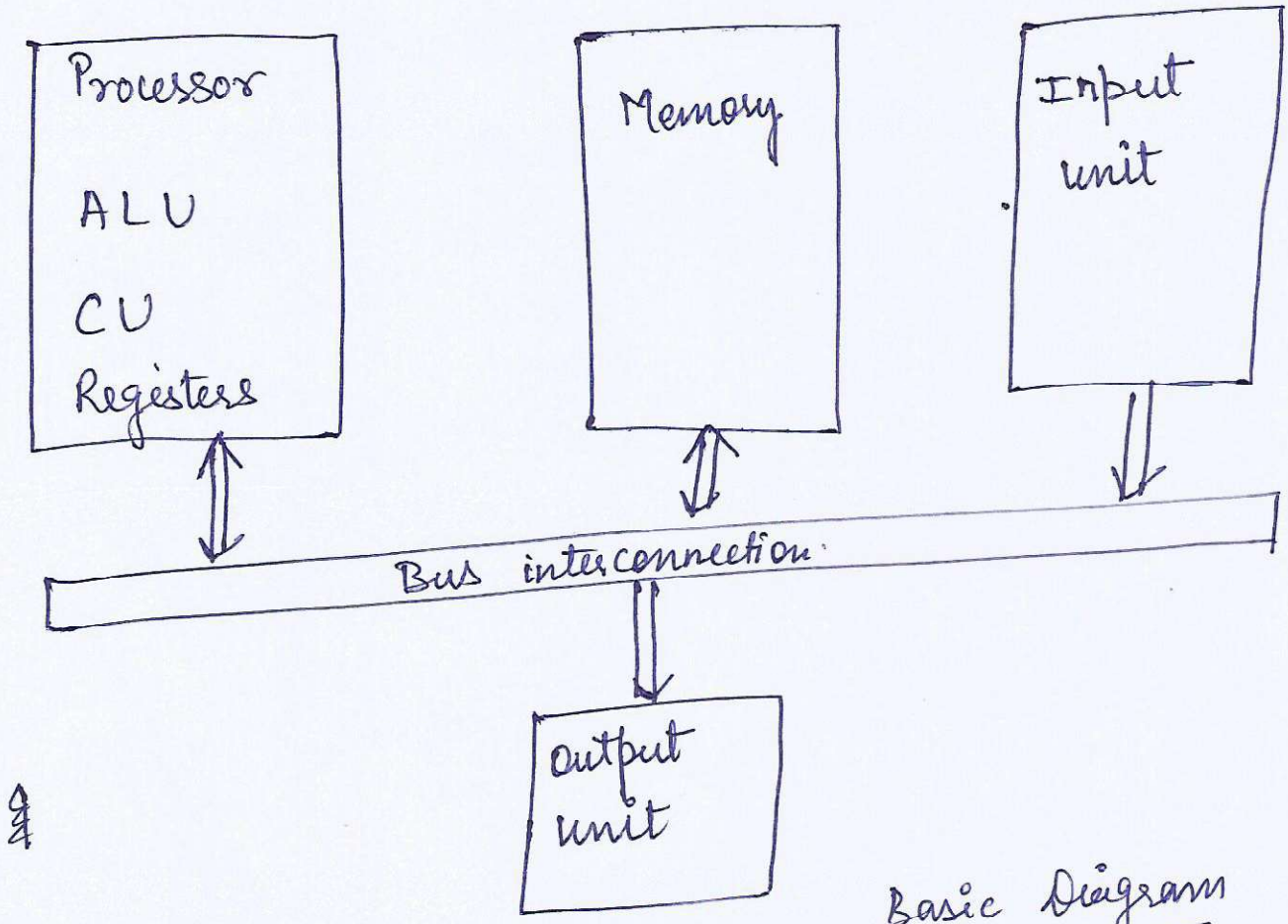
COA Unit-1 Notes

[redacted]

Functional units and their interconnections:

→ Functional units of a computer / digital system are:

- → Input unit
- → Output Unit
- → Central Processing Unit (CPU)
- → Memory
- → Bus structure.



Basic Diagram.

Note: Theory about these components are given in "Fundamental of COA" Notes and Unit 1 Notes (about bus)

Buses, bus Architecture, types of buses and bus arbitration

Bus:

→ Bus is a communication system that helps data transfer between different modules of the computer.

→ A bus is a communication pathway connecting two or more devices.

→ A bus is a group of electrical lines/wires that carry computer signals/bits.

→ A bus consists of multiple lines. Each line is capable of transmitting signals representing 1 or binary 0.

A bus that connects major computer components (processor, memory, I/O) is called a system bus.

→ On any bus lines can be classified into three function groups:-

1. Data Bus
2. Address Bus
3. Control Bus.

Data Bus:- Data lines provides path for moving data between components.

- Bidirectional

→ Width of databus is number of lines in data bus.

→ Each line can carry only one bit at a time.

Address Bus: (Unidirectional)

→ The address lines are used to designate (know) the source or destination of the data on the data bus.

For example: if the processor wishes to read or write a memory word, it puts the address of desired word on the address lines.

The width of address bus determines the maximum possible memory capacity of the system.

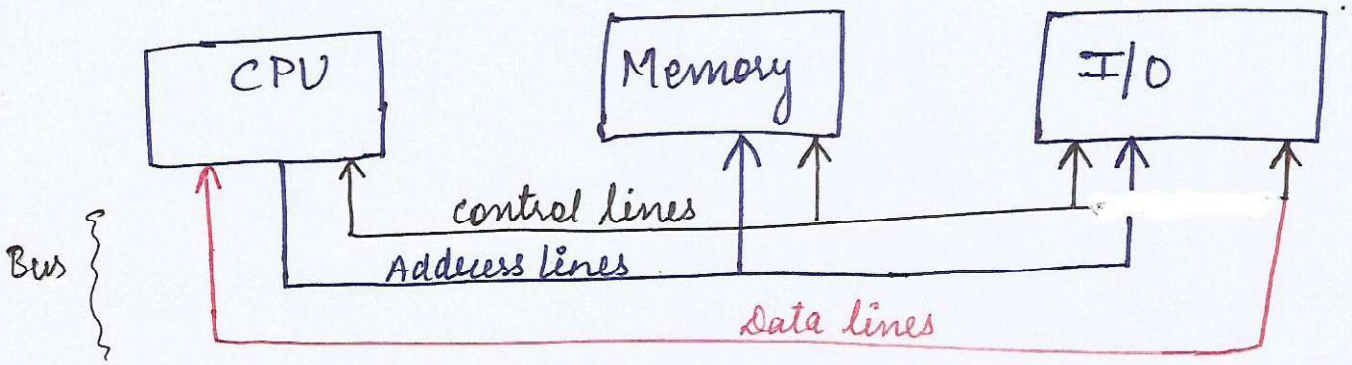
Control Bus: (Bidirectional)

- The control lines are used to control the access to and the use of the data and address lines.
- control signals transmit both command and timming information

Timing signals ⇒ indicate the validity of data and address information

command signals ⇒ specify operations to be performed.

- Typical control lines include:-
  - Memory write
  - Memory Read
  - I/O write
  - I/O Read
  - Transfer Ack
  - Bus Request
  - Bus Grant
  - Interrupt Request
  - Interrupt Ack
  - clock
  - etc.



Bus interconnection scheme.

Types of Buses:

- Dedicated
- Multiplexed

- Dedicated bus line is permanently assigned either to one function or to a physical subset of computer components.
- Example of functional dedication :- the use of separate dedicated address and data lines.
- Multiplexed: Address and data information may be transmitted over the same set of lines using an Address Valid control line.

For example:

At the beginning of a data transfer, the address is placed on the bus and Address valid line is activated. ~~At this point~~ The address is then removed from the bus and the same bus are used for subsequent read or write data transfer. (this known as time multiplexing)

Bus Arbitration:

→ More than one module may need control of the bus e.g CPU and DMA controller.

e.g. I/O module may need to read or write directly to the memory without sending the data to the processor.

The process by which multiple requests are recognized and given priority given to one of them is called arbitration

- Arbitration can be — Centralized / localized
  — Decentralized / Distributed

→ In a centralized scheme, a single hardware device, referred to as a bus controller or arbiter, is responsible for allocating time on the bus. The device may be a separate module or part of the processor.

- In distributed scheme, there is no central controller. Rather, each module contains access control logic and the modules act together to share the bus.

- With both methods of arbitration, the purpose is to designate one device, either the processor or an I/O module, as master. The master may initiate a data transfer (e.g. read or write) with some other device, which acts as slave for this particular exchange.

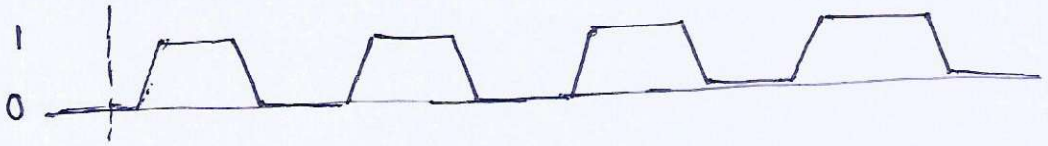
~~Timing~~

Timing Timing Refers to the way in which events are coordinated on the bus.

- Buses use — ① Synchronous Timing
  — ② Asynchronous Timing

Synchronous Timing:

- With synchronous timing the occurrence of the events on the bus is determined by a clock.
- The bus includes a clock line upon which a clock transmits a regular sequence of alternative 0's and 1's of equal duration. A single 1-0 transmission is referred to as a clock cycle or bus cycle and defines a time slot.
- All events starts at the beginning of a clock cycle.



clock.

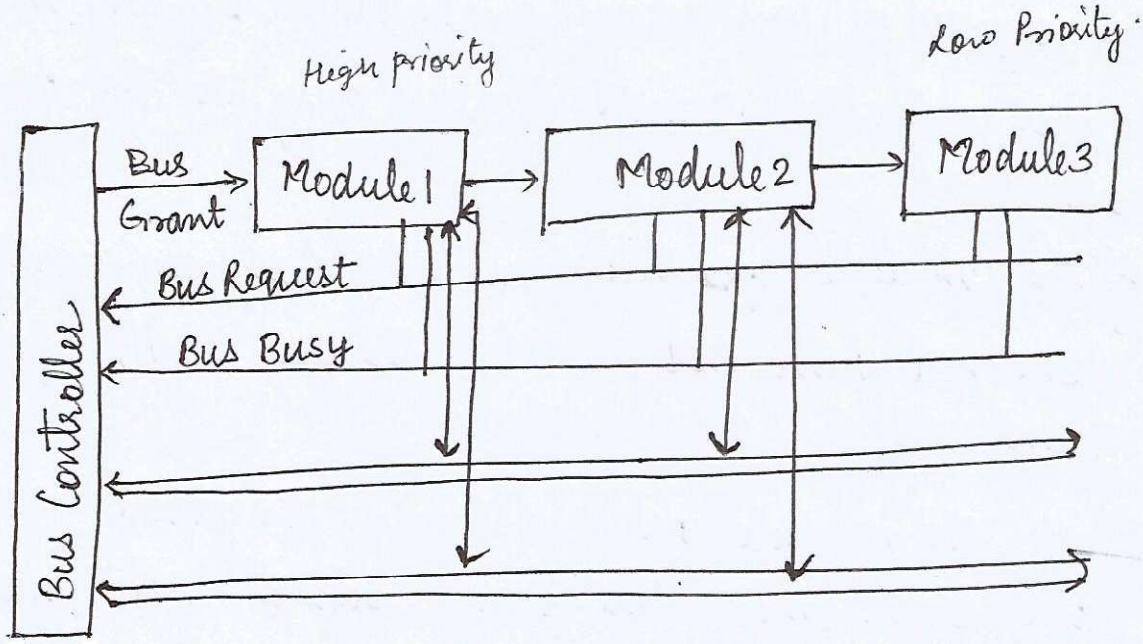
Asynchronous Timing:

→ with asynchronous timing, the occurrence of one event on a bus follows and depends on the occurrence of a previous event.

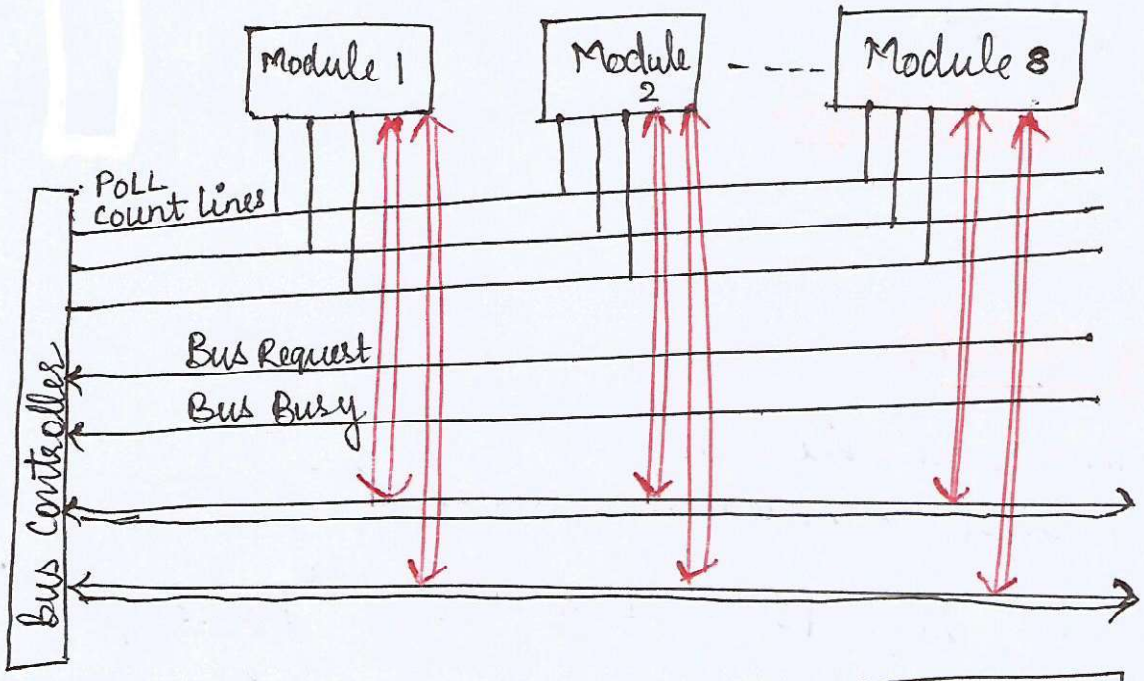
→ No clock.

Bus Arbitration Techniques:

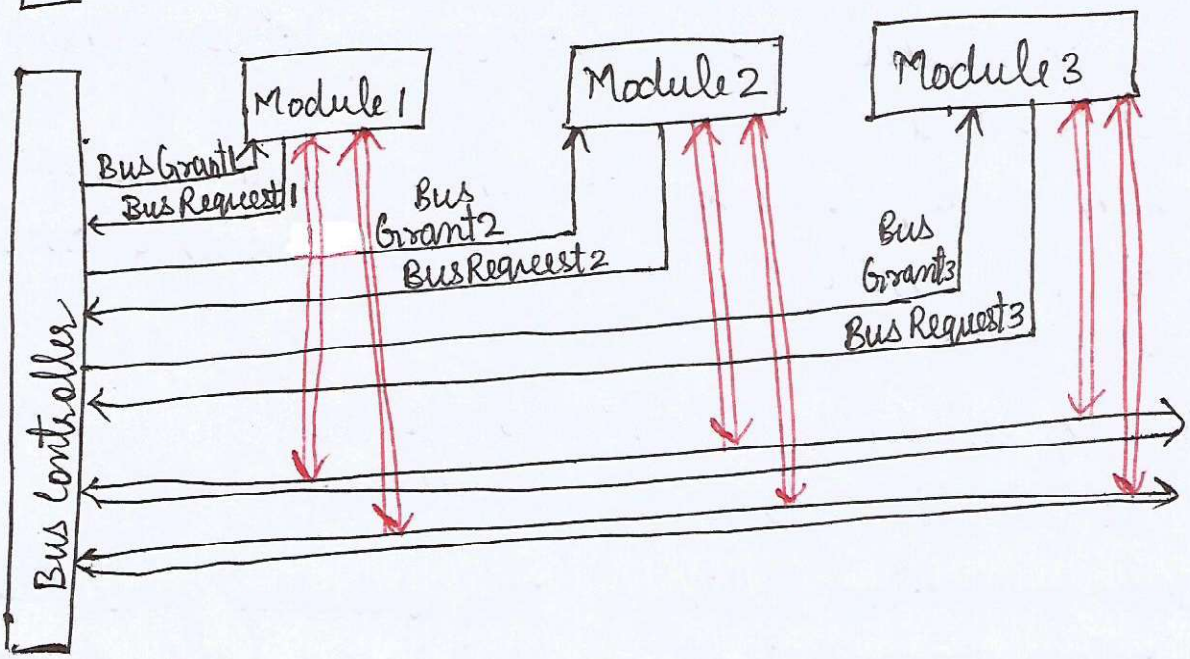
Daisy chain



Polling



Independent Requesting



Daisy-chain Method or Serial Method:

⇒ The request of bus usage can be placed by any module connected.

⇒ The bus controller grants the bus only to the first module connected.

⇒ ⇒ if the module ~~granted bus~~, who has been granted bus to, ~~does not~~ have not requested the bus, will forward the grant to the next module.

⇒ if the module who is having the grant wants to use bus, makes the busy signal 'set' and use the bus. After using bus, the module unset the busy signal.

Disadvantage: Modules has the fixed Priority.

Polling Method:

Polling count line = n

Max Number of connected modules ⇒ 2^n.

⇒ Modules requests the bus using 'Bus Request' line.

⇒ Bus controller places any one of 2^n sequences on the poll count lines.

⇒ The module, for which polling sequence is allocated by the bus controller, can set busy signal and use the bus.

⇒ Priority is set by the bus controller.

⇒ It is time consuming method.

⇒ Non-productive Polling ⇒ Bus controller generated the polling sequence for the module which did not request the bus

Independent Request Arbitration Technique:

→ In this method, every module has connected to the bus controller by separate 'Bus grant' and 'Bus Request'

⇒ Individual module can request to the bus controller by its own 'Bus Request' line. And, Bus controller can grant the bus to the module ~~to~~ which has requested for the bus.

⇒ But this method increases the cost of the system.

⇒ Better Performance.

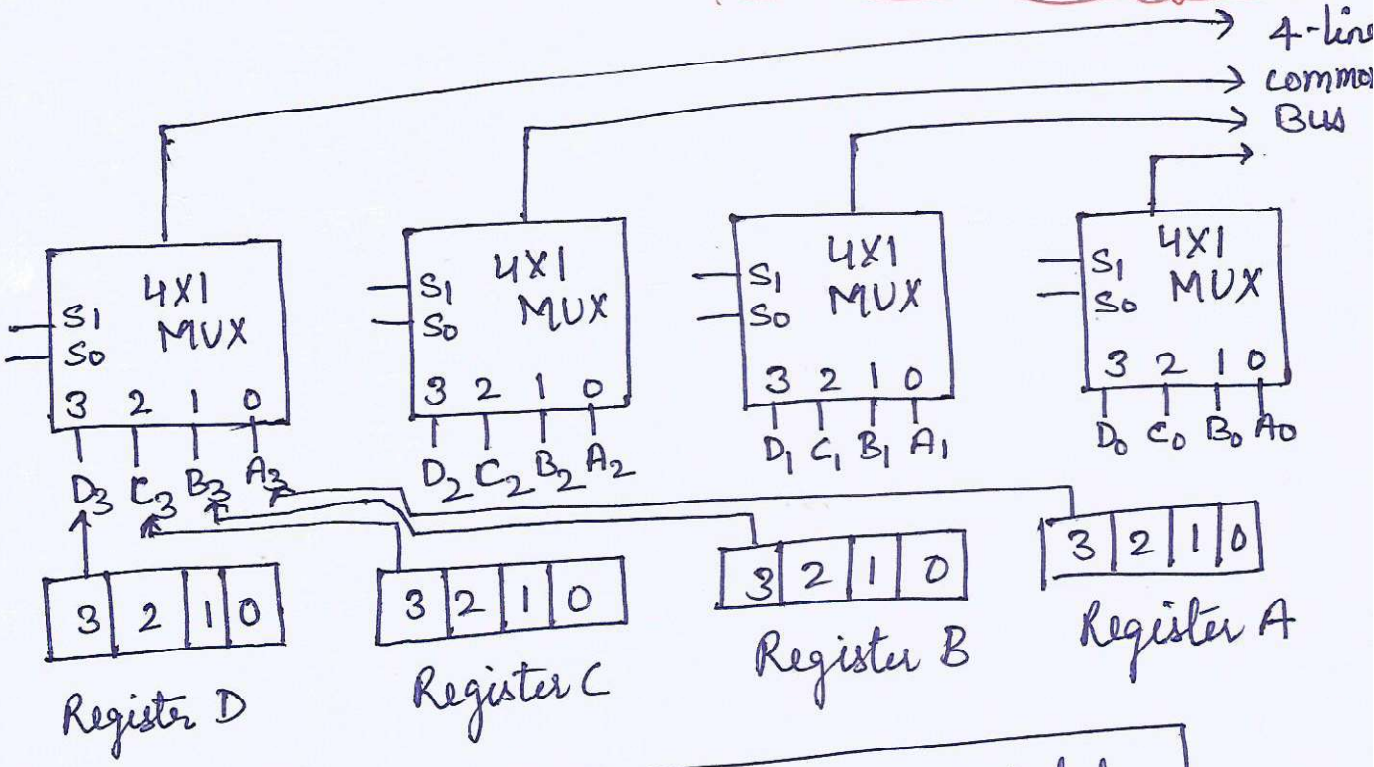
Bus and Memory Transfer:

- A typical digital computer has many registers.
- A path must be provided to transfer information from one register to another.
- common bus structure
  - → using Multiplexer.
  - → Using Tristate Buffer.

Bus system using Multiplexer:

Example ⇒ For four registers.

Number of multiplexers = size of register.
Size of multiplexer = Number of registers



Function Table:

S_1	S_0	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

Memory Transfer:

~~Memory Read: The transfer of information to be stored into memory is~~

Memory Read: The transfer of information from memory to the outside environment is called a read operation

Read: $DR \leftarrow M[AR]$

DR ⇒ Data Register

M[AR] ⇒ memory location specified by Address Register AR.

Memory Write: The transfer of new information to be stored into the memory is called a write operation

Write: $M[AR] \leftarrow DR$

Registers

- Registers are used to store data temporarily

→ The register in CPU performs two roles

- User Visible Register
  - General Purpose Register
  - Data Register
  - Address Register
  - Condition codes
- control and Status Register
  - Program Counter
  - Instruction Register
  - MAR
  - MDR

User Visible Register:

- A user visibe register may be referenced by means of machine language that the processor executes.
- categories
  - General Purpose
  - Data Register
  - Address Register
  - Condition Codes

General Purpose Registers:

- can be used by the programmer.
- contains operands for any opcode.
- can be used for addressing functions (e.g. Register indirect, displacement).

Data Register: Data Register may be used only to hold the data

Address Register:

Address Register may be devoted to a particular addressing modes. Examples
- → Segment Pointer
- → Index Register
- → Stack Pointer.

- Segment Pointer holds the address of the base of the segment in segmented addressing.

- Index Registers are used for indexed addressing and may be autoindexed.

- Stack Pointer points to the top of the stack. This allows implicit addressing that is push, pop and other stack instructions need not ~~sto~~ contain an explicit stack operand.

Condition codes (Also referred to as Flags):

condition codes are the bits set by the processor hardware as the result of operation.

Example:
- positive or negative result
- Zero Result
- Overflow
- etc.

control and status Register:

→ Registers employed to control the operation of processor.

→ Not visible to the user.

→ Four Registers are essential to instruction execution

1. Program counter
2. Instruction Register
3. Memory Address Register (MAR)
4. Memory Data Register (MDR)

- Program counter contains the address of the instruction to be fetched
- Instruction Register contains the instruction most ~~recently~~ recently fetched.

→ MAR contains the address of a location in the memory.

→ MDR or MBR (Memory Buffer Register) contains a word of the data to be written to memory or the word most recently read.

PSW : Program Status Word:

Many processors include a set of registers known as PSW, that contain the status information

- condition code and other status information

Sign: contains the sign bit of the result.

Zero: set when the result is zero.

Carry: set if an operation resulted in a carry into or borrow out of a higher order bit.

Equal: set if a logical compare result is equal.

Overflow: used to indicate arithmetic overflow.

Interrupt Enable/disable: used to enable or disable interrupts

Supervisor: – indicates whether the processor is executing supervisor or user mode.
Certain previledged instructions can be executed only in supervised mode, and certain memory areas can be accessed only in supervisor mode.

Topic

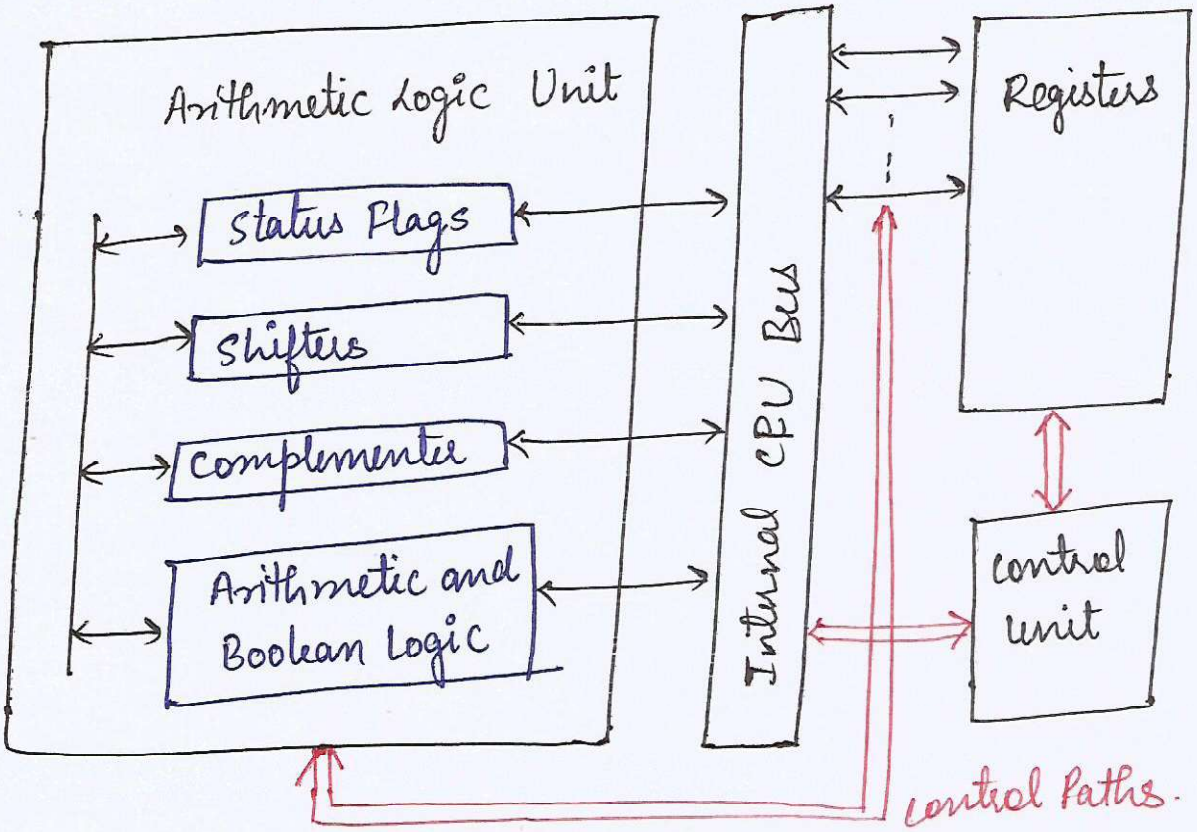
Processor Organization:

Processor contains ALU, CU and Registers

→ ALU ⇒ The ALU (Arithmetic Logic Unit) does the actual computation or processing of the data

→ CU: control unit:- CU controls the movement of data and instruction into and out of the processor and controls the operation of ALU.

→ Registers: minimal internal memory.



Internal structure of CPU

- Processor Organization means how the components of processor are connected and accomplish their tasks.

A processor does the following things.

- Fetch Instruction
- Interpret instruction
- Fetch Data
- Process Data
- Write Data

- <u>Fetch Instruction</u>: The processor reads an instruction from memory.

- <u>Interpret Instruction</u>: The instruction is decoded to determine what action is required.

- <u>Fetch data</u>: The execution of an instruction may require reading data from memory or an I/O module

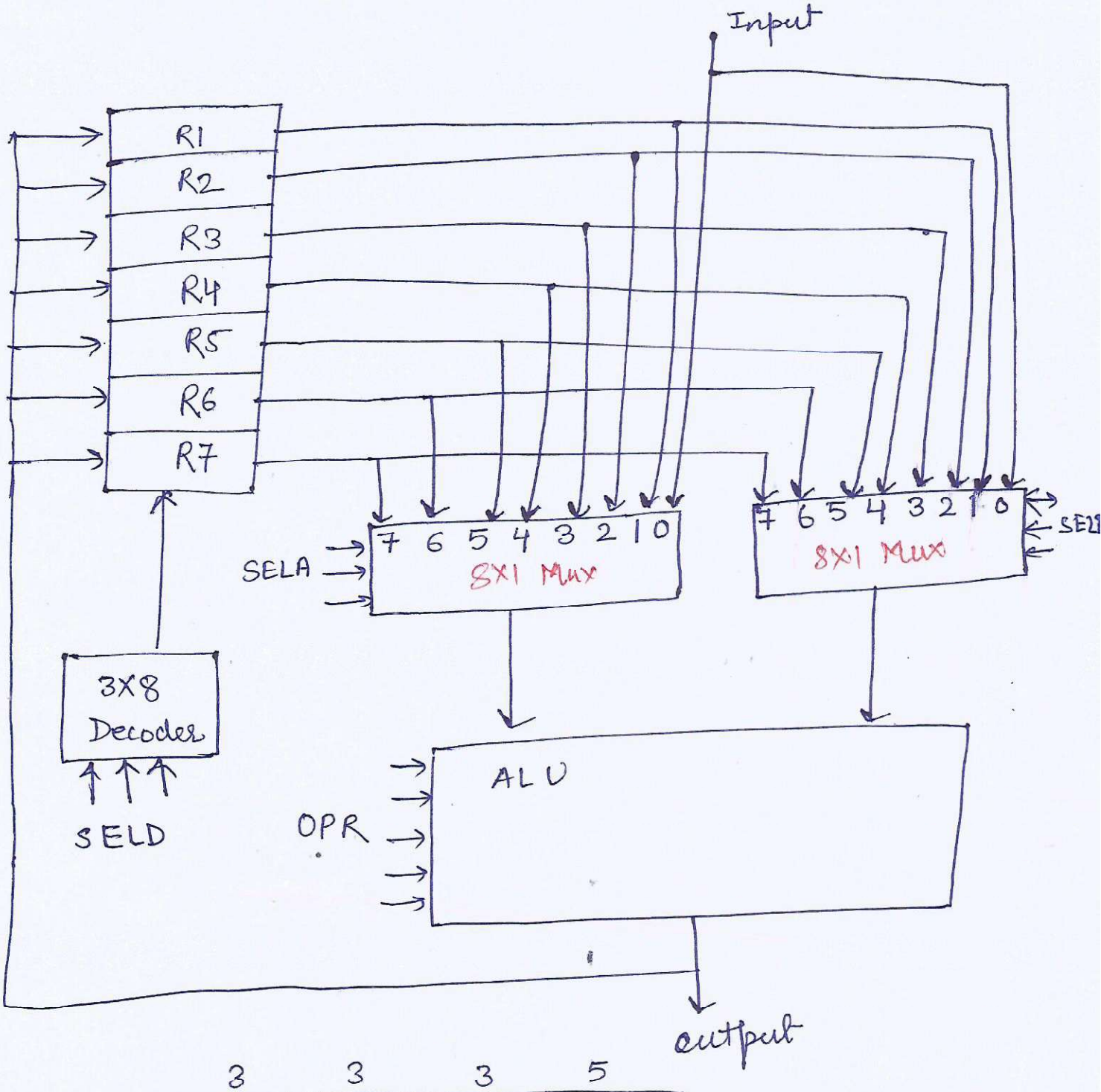
- <u>Process Data</u>:- The execution of an instruction may require performing some arithmetic or logical operation on the data.

- <u>Write Data</u>: The result of an execution may require writing data to memory or I/O module.

Topic

General Register Organization:

→ A bus organization for seven CPU registers.



3	3	3	5
SELA	SELB	SELD	OPR

control word.

Register set with common ALU

- The output of each Register is connected to two multiplexers to form the buses A and B.
- The selection lines in each multiplexer selects one register or input for the particular bus.

→ The buses A and B form the inputs to a common ALU.

→ The operation selected in the ALU determines the arithmetic or logic microoperation that is to be performed.

→ The result is available for output data and goes into the inputs of ~~the~~ all registers. Register for output is selected by a decoder.

→ The decoder activates one of the register load inputs.

Register Selection:

Binary Codes	SEL A	SEL B	SEL D
000	input	input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

Table of operations performed by ALU

	OPR select	Operation
0	00000	Transfer A
1	00001	Increment A
2	00010	Add A+B
5	00101	Subtract A−B
6	00110	Decrement A
8	01000	AND A, B
10	01010	OR A, B
12	01100	XOR A, B
14	01110	complement A
16	10000	shift Right A
24	11000	shift Left A

This table specifies some selected operants only ⇒ 11

five bits can represent total 32 operation

Topic.

Stack Organization:

→ stores information is LIFO order (Last In First Out)

→ The register that holds the address of the stack is called Stack Pointer.

→ Stack Pointer's value always points to the top element.

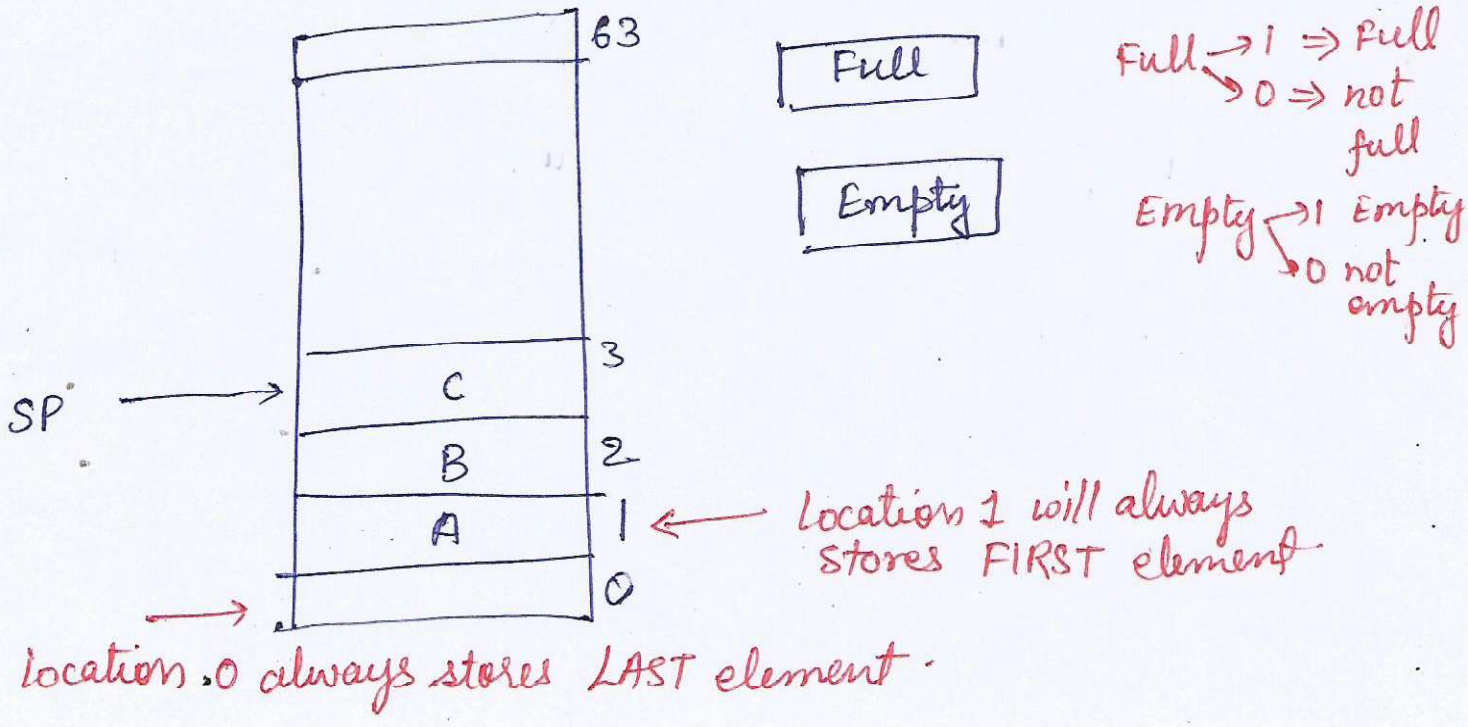
→ Two operations on stack ① Push
② Pop

⇒ Push ⇒ to insert an element in the stack
Pop ⇒ to delete an element from the stack

⇒ Two status Full ⇒ set to 1 when stack is full
Empty ⇒ set to 1 when stack is empty

⇒ Stack can be ⇒ collection of memory words or registers

⇒ Example 64 - word register stack



initially $SP \leftarrow 0$
$EMPTY \leftarrow 1$
$FULL \leftarrow 0$

if (Full = 0) ⇒ new item is inserted with push operation
if (EMPTY = 0) ⇒ an item can be deleted from stack with popoperation

<u>Push operation</u> :-

$SP \leftarrow SP+1$ increment stack Pointer
$M[SP] \leftarrow DR$ write item on the top of stack.
if (~~[illegible]~~ $SP=0$) then $FULL \leftarrow 1$
$EMPTY \leftarrow 0$

M[SP] denotes the memory word specified by the address presently ~~specify~~ available in the SP.

⇒ The first item is stored in stack at address 1.
The last item is stored in the stack at address 0.

<u>Pop operation</u> :

A new item is deleted from the stack if stack is not empty

$DR \leftarrow M[SP]$ ⇒ Read the data from the top of stack.
$SP \leftarrow SP-1$ ⇒ Decrement SP
if ($SP=0$) then $EMPTY \leftarrow 1$ Check if the stack is empty
$FULL \leftarrow 0$ ⇒ Mark the stack not full.

Ms. Minakshi

ADDRESSING MODES :

Two issues : ① How is the address of an operand specified ?
② How the bits of an instruction organized to define the operand addresses and operation of that instruction.

The operation field of an instruction specifies the operation to be performed. This operation must be executed on some data stored in the computer registers or memory words.
The way the operands are choosen during the program execution depends on the addressing mode of the instruction.
The addressing mode specifies a rule for interpreting or modifying the address field of the instruction before the operand is actually referenced.

Most Common addressing techniques :

- Immediate Addressing
- Direct Addressing
- Indirect addressing
- Register Addressing
- Register Indirect Addressing
- Displacement → Relative Addressing
  - → Base Register Addressing
  - → Indexing
- Stack addressing

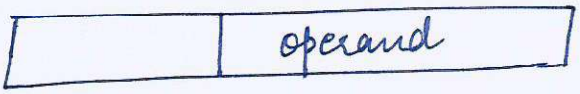
① Immediate Addressing:

→ In this addressing mode, the operand value is present in the instruction.

→ **Advantage:** no memory reference other than the instruction fetch is required to obtain the operand.

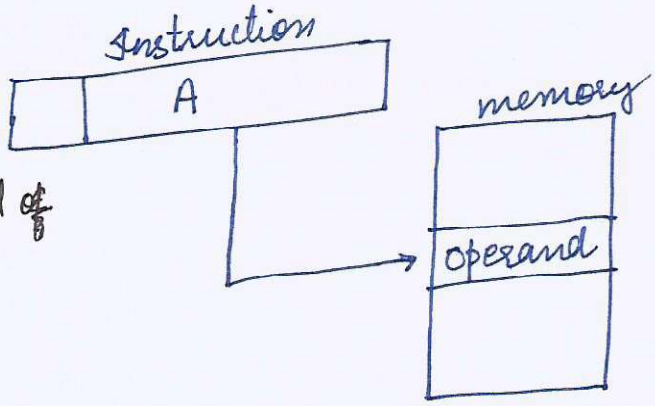
→ **Disadvantage:** Limited operand magnitude.

Instruction



② Direct Addressing:

In this addressing mode, the address field contains the actual/effective address of the operand.



$EA = A$

A ⇒ content of an address field in the instruction

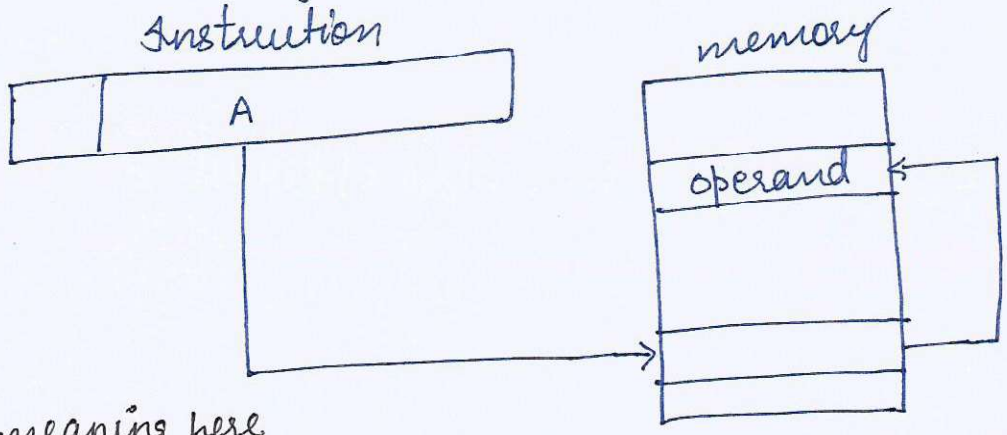
Advantage:

- It requires only one memory reference and no special calculation.
- Simple

Disadvantage: Limited address space.

3 Indirect Addressing:

In this addressing mode, address field refers to the address of a word in the memory.



EA = (A)

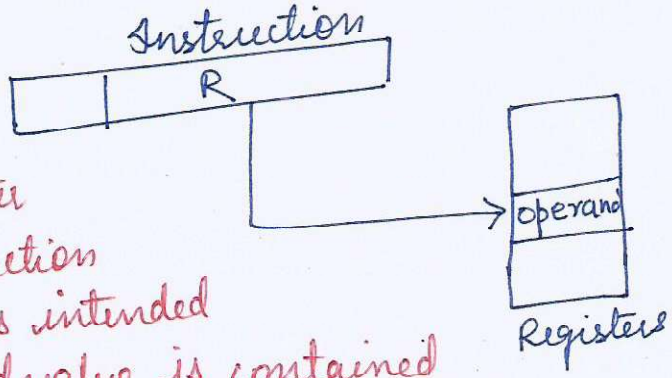
* parantheses meaning here
  ⇒ "content of"
* EA = Actual / effective address of the location containing the referenced operand.

Advantage: Large address space:
→ for a word length of N, an address space of 2^N is now available.

Disadvantage: requires two memory reference
→ one for address of operand.
→ one for to get its value (operand's)

4 Register Addressing:

Register addressing is ~~similar~~ similar to direct addressing. The only difference is that the address field refers to a register rather than a main memory address.



EA = R

if the content of a register address field in an instruction is 5, then register R5 is intended address, and the operand value is contained in R5.

26/

Advantages:

→ only a small address field in instruction is needed

→ no time consuming memory references.

Disadvantages:

→ very limited register address space.

"content of the register will be operand" in Register Addressing

⑤ Register Indirect Addressing:

- similar to indirect addressing
- But here, the address field of instruction specifies a register whose contents give the address of the operand in the memory.

$$EA = (R)$$

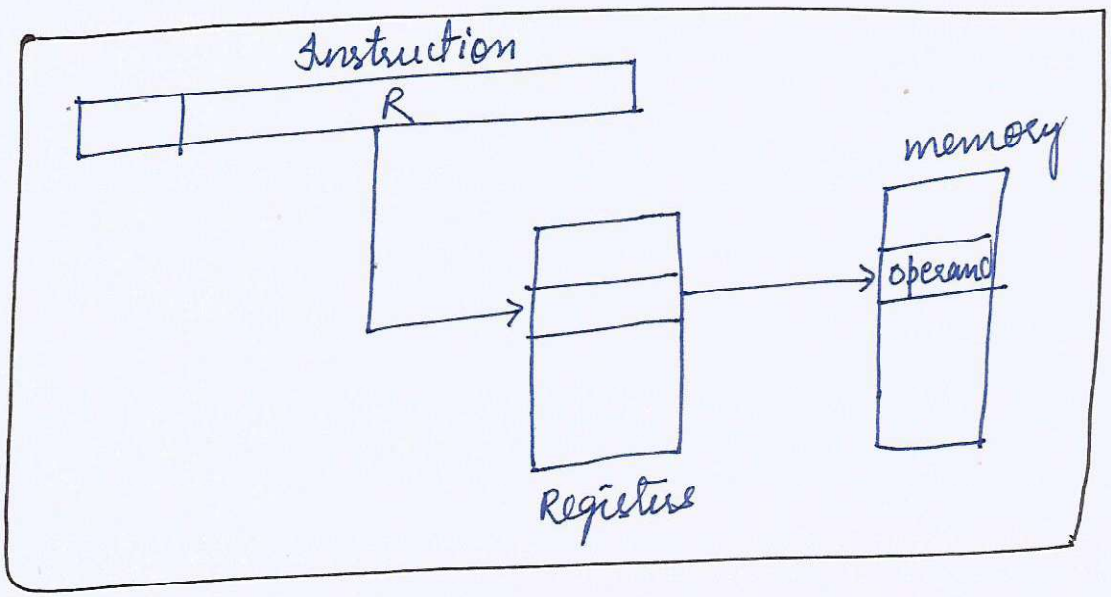
Advantage:

→ fewer bits to select a register than a memory address

→ Large address space.

Disadvantage:

- Extra memory Reference.



Meenakshi

COA Unit-5 Notes

Peripheral Devices:-

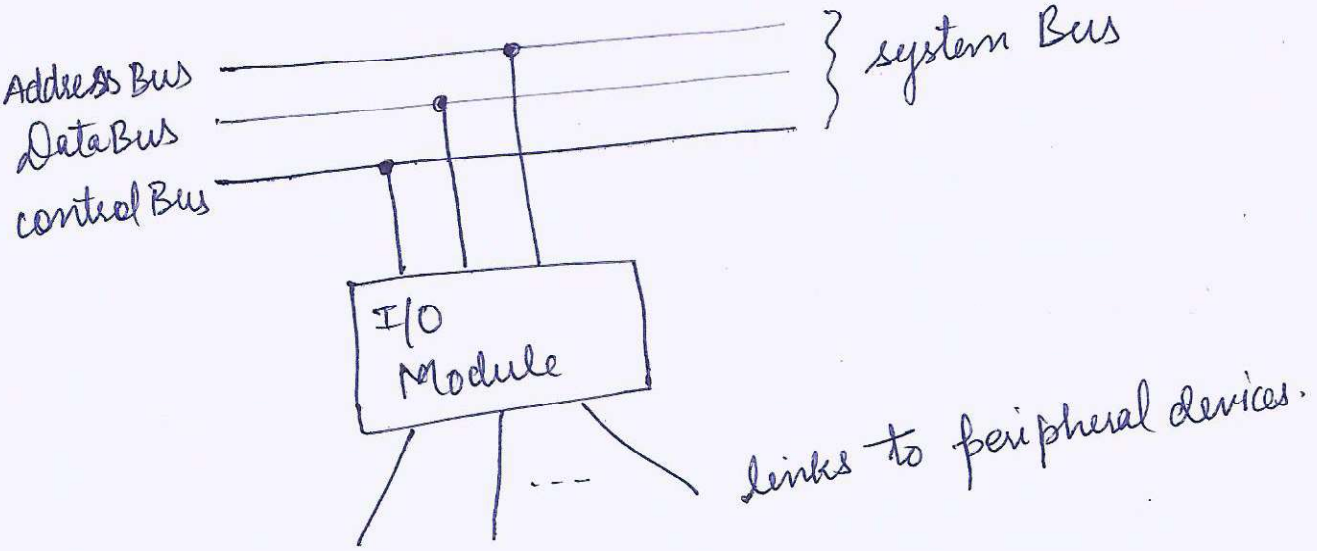
An external device connected to an I/O module is often referred to as a peripheral device or simply a peripheral.

3 categories of external devices:-

① Human Readable Device:
suitable for communicating with the computer user (e.g. VDT (Video Display ~~Tex~~ Terminal), Printer, Keyboard, etc).

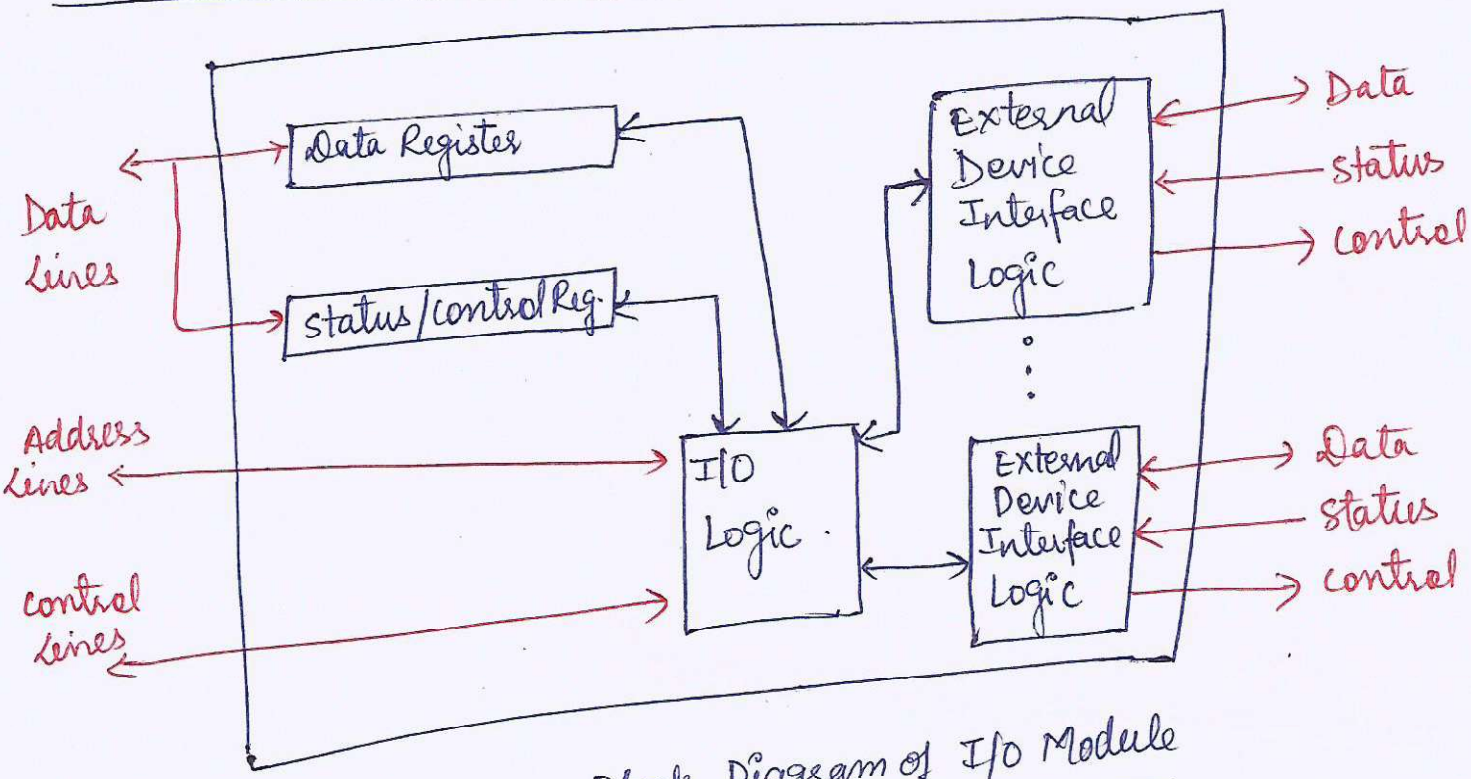
② Machine Readable Devices:-
suitable for communicating with equipment. eg. magnetic disk, sensor etc.

③ Communication Device:
suitable for communicating remote devices.
Eg- The remote device can be the human readable device such as terminal, a machine readable device or another computer.



Generic Model of an I/O Module

Block Diagram of I/O Module



Block Diagram of I/O Module

I/O Port: A connection Point that acts as an interface b/w the computer and external devices like mouse, keyboard, printer, modem etc.

Ports are of two types:-

Internal Port: It connects motherboard to internal device like Hard disk drive, internal modem etc.

External Port: It connects the motherboard to external device like, mouse, printer, and keyboard etc.

I/O Systems:

User programs never directly interact with I/O devices. All I/O operations are performed with the help of system calls.

There are 4 types of I/O commands:-

→ Control
→ Test Status
→ Read
→ Write

Modes of Data Transfer (I/O Techniques)

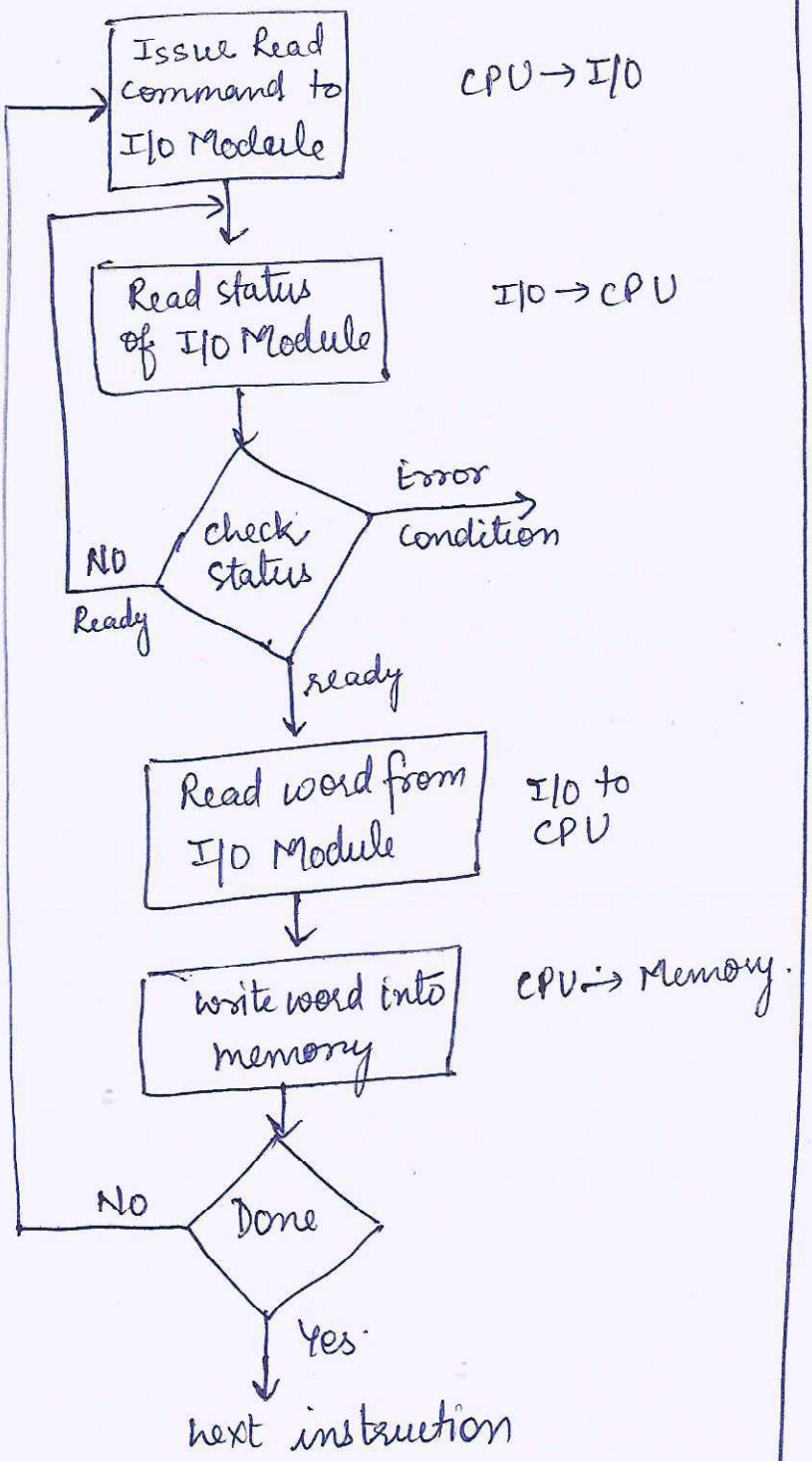
- Programmed I/O
- Interrupt Driven I/O
- Direct Memory Access

Programmed I/O

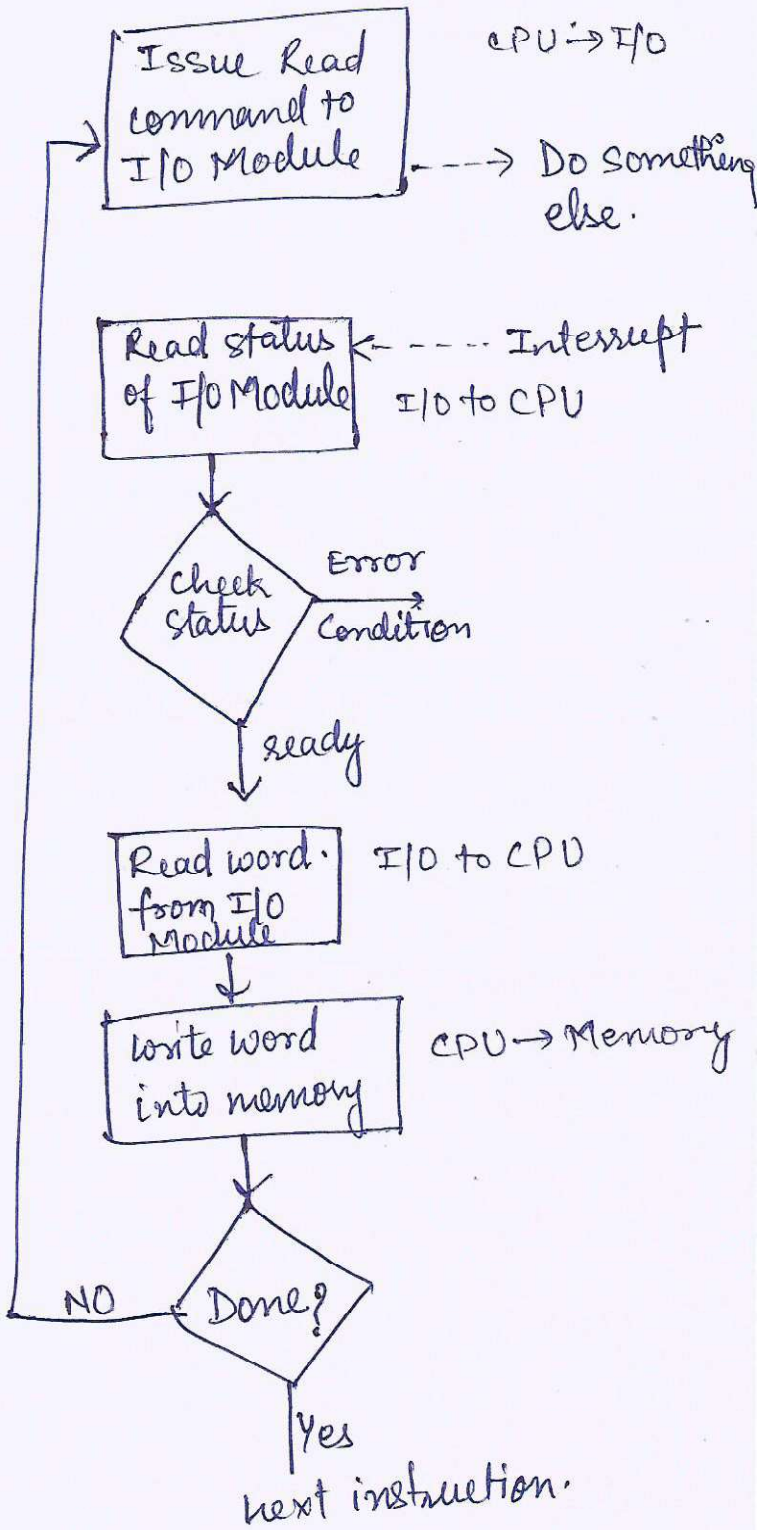
- Input-output instructions written in computer program.
- Each data item transfer is initiated by an instruction in the program
- Transferring data under program control requires a constant monitoring of perpherals by CPU. (once initiated when to transfer)

⇒ ~~Once initiated~~
⇒ In this method, the CPU stays in a program loop until I/O unit indicates that it is ready for the data transfer.

⇒ Time consuming
⇒ ~~Less efficient method~~ Less efficiency.



Programmed I/O.



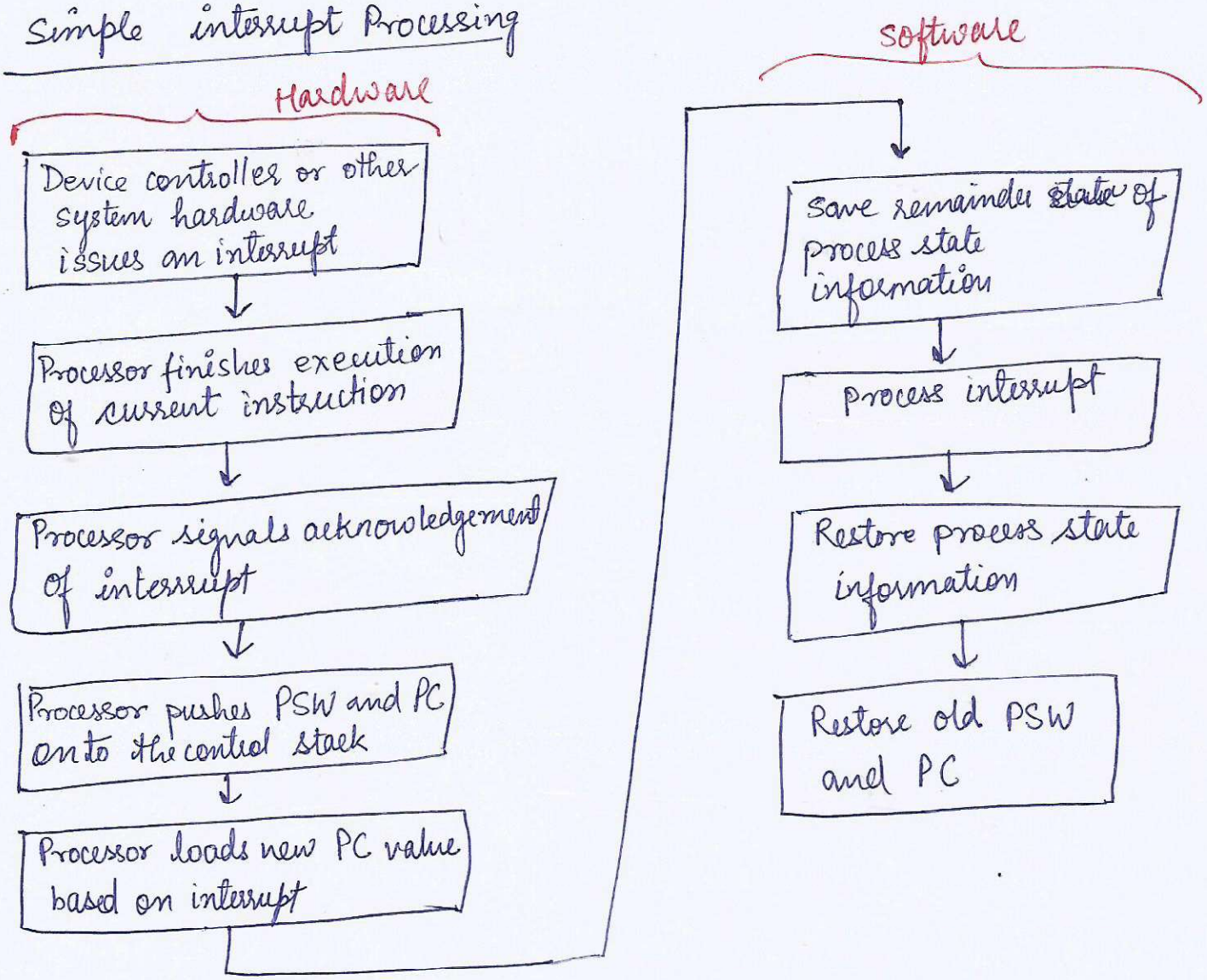
Interrupt Driven I/O

Interrupt Processing:

The basic method of interrupting the CPU is done by activating a control line that connects the interrupt source to the CPU.

CPU ⇒ * recognizes the presence of interrupt
* executes a specific interrupt handling program.
* determines the source of interrupt
* determines the address (branch address) of the interrupt handling program.

Simple interrupt Processing



Design issues:

Two design issues in implementing ~~I/O~~ Interrupt I/O.

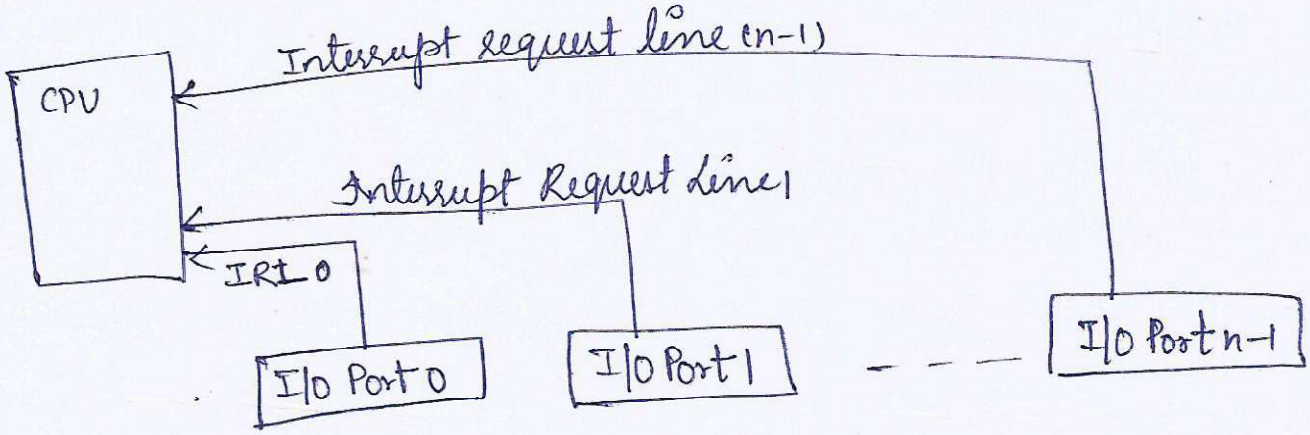
1. → To determine which device issued the interrupt from multiple devices.

2. if multiple ~~devices~~ interrupts have occurred, how does the the processor decide which one to process.

Issue: Device identification

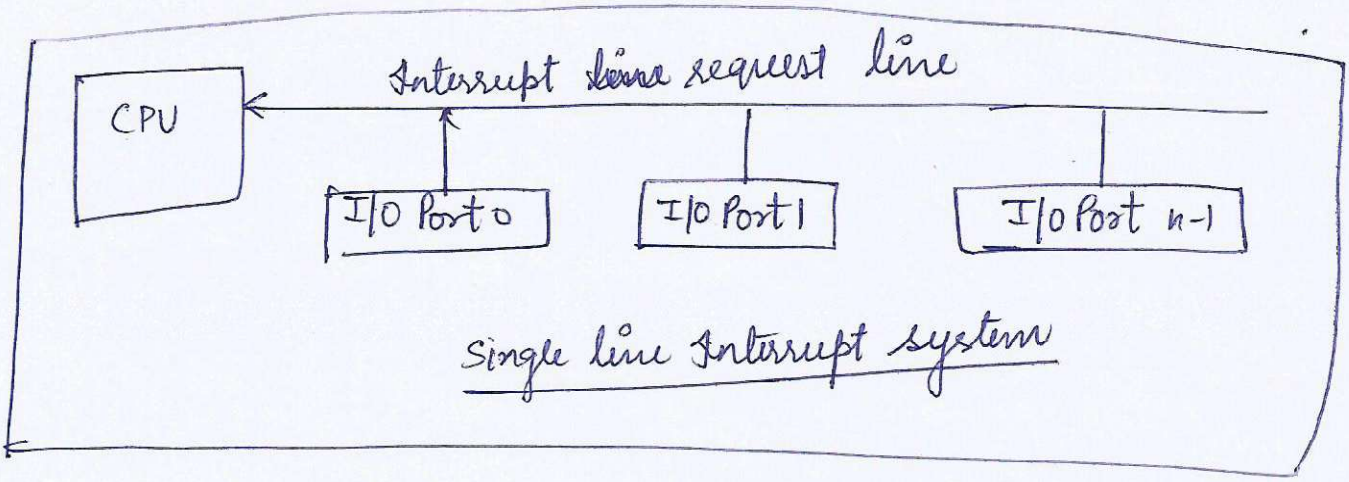
4 Techniques exist.
- → Multiple interrupt lines
- → Software Poll
- → Daisy chain (Hardware Poll, vectored)
- → Bus Arbitration (vectored)

Multiple Interrupt Lines:-



⇒ immediate recognition of device.
⇒ separate Interrupt request line
⇒ impractical approach.

multiple Interrupt line system



Single line Interrupt system

Software Poll:

⇒ when processor detects the interrupt, it branches to an interrupt ~~ser~~ service-routine whose job is to poll each I/O module to determine which module caused the interrupt.

⇒ Time consuming

⇒ Priority ~~is~~ can be implemented by defining polling sequence.

Vectored Interrupt using Daisy chaining: (Hardware Poll)

⇒ All I/O module shares common interrupt request line

→ The interrupt acknowledgement line is daisychained through the modules.

– when a processor senses the interrupt, it sends out an interrupt acknowledge propagating through a series of I/O module until it gets a requesting module.

– The requesting module ~~&~~ responds by placing a word on data lines. This word is ~~referred~~ referred to as vector.

Bus Arbitration Technique:- (vectored interrupts)

→ with bus arbitration technique, an I/O module first gain control of bus before it can raise the interrupt request line.

→ Thus only one module can raise the line at a time.

Types of Interrupts:

Program interrupts

These are generated by some condition that occurs as a result of an instruction execution such as:

- → Arithmetic Overflow
- → Division by zero
- → Execution of illegal machine instruction
- → Segment limit violation
- → Execution of privileged instruction

Timer Interrupts

These are generated within the processor. This allows operating system to perform certain operations on regular basis.

Input-output interrupts:

These are generated for initiation or completion of I/O operations. I/O failure or I/O error too can generate an interrupt.

Hardware Failure Interrupts

These are generated by a failure; such as power failure or memory parity error.

Interrupts v.s. Exceptions:

An interrupt is generated by a signal from hardware, and it may occur at random times during execution of a program.

An exception is generated from software, and it is provoked by the execution of an instruction.

Hardware and Software interrupts

→ when microprocessor receive interrupts signals through pins (hardware) of microprocessor. these are known as Hardware interrupts

→ Software interrupts are those which are inserted in the between the program which means these are mnemonics of the microprocessor.

Vectored and non vectored interrupts

- In vectored interrupts, the source (or device) that interrupts supplies the branch information to the computer.
- In a non-vectored interrupt, the branch address is assigned to a fixed location in the memory.

Maskable and non-maskable:

- Maskable interrupts are those which can be disabled or ignored by the microprocessor.
- Non-maskable: which can not be disabled or ignored by the microprocessor.

Types of interrupts:

4 categories

→ Program interrupts
→ Timer interrupts
→ Input/output interrupts
→ Hardware failure

Direct memory ACCESS :-

→ Both interrupt driven and programmed I/O require involvement of CPU for data transfer.

→ CPU can be better utilized for program execution.

→ I/O activities are slow and involvement of CPU will not have a desired effect on the system performance.

→ DMA increases the speed of I/O transfer by eliminating the role played by CPU in I/O operations.

→ when large amount of data is stored/transferred from CPU, a DMA module can be used.

DMA operates in the following ways:

when I/O is requested, the CPU instructs the DMA module about the operation, by providing information:

→ Source address

→ Target address

→ Read / write

→ Number of words to be read or written.

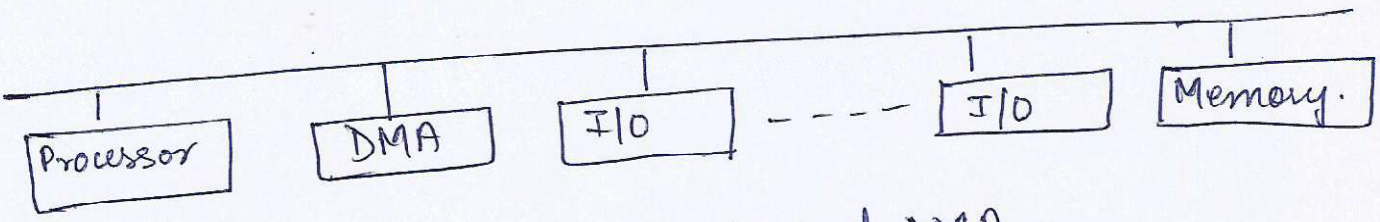
⇒ CPU loads the DMA registers 'Data count' and 'Address Register' with number of bytes to be transferred and starting addr memory location respectively.

when the DMA controller is ready to transmit or receive data, it activates the DMA request line to CPU. CPU wait for the next break point. CPU now leaves the control of system bus and activates DMA acknowledge.

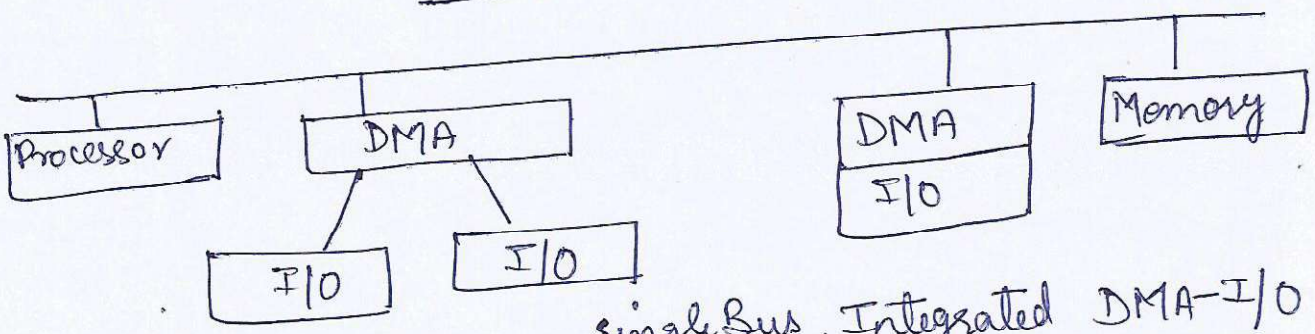
→ The DMA controller transfers the data directly from or to main memory. After, a word is transferred. Data count (or word count register) and Address register are updated.

→ If the data count register has not reached zero but the I/O device is not ready to send or receive the data, the DMA controller releases the system bus to CPU by deactivating DMA request line.

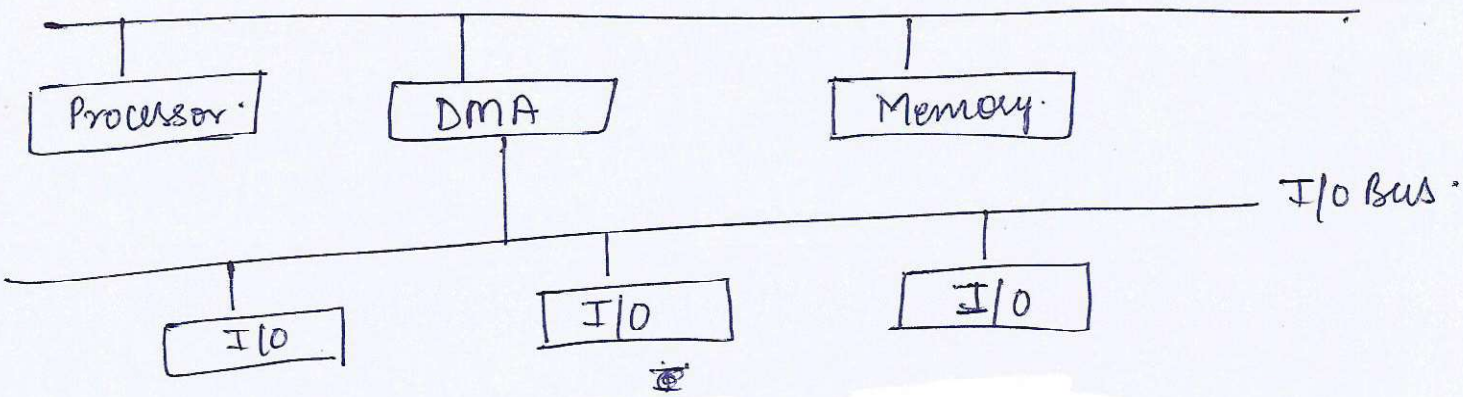
- if data count register is decremented to zero, DMA controller finally relinquishs the control of system bus. It may also send an interrupt signal to CPU to indicate the end of data transfer.



Single Bus Detached DMA

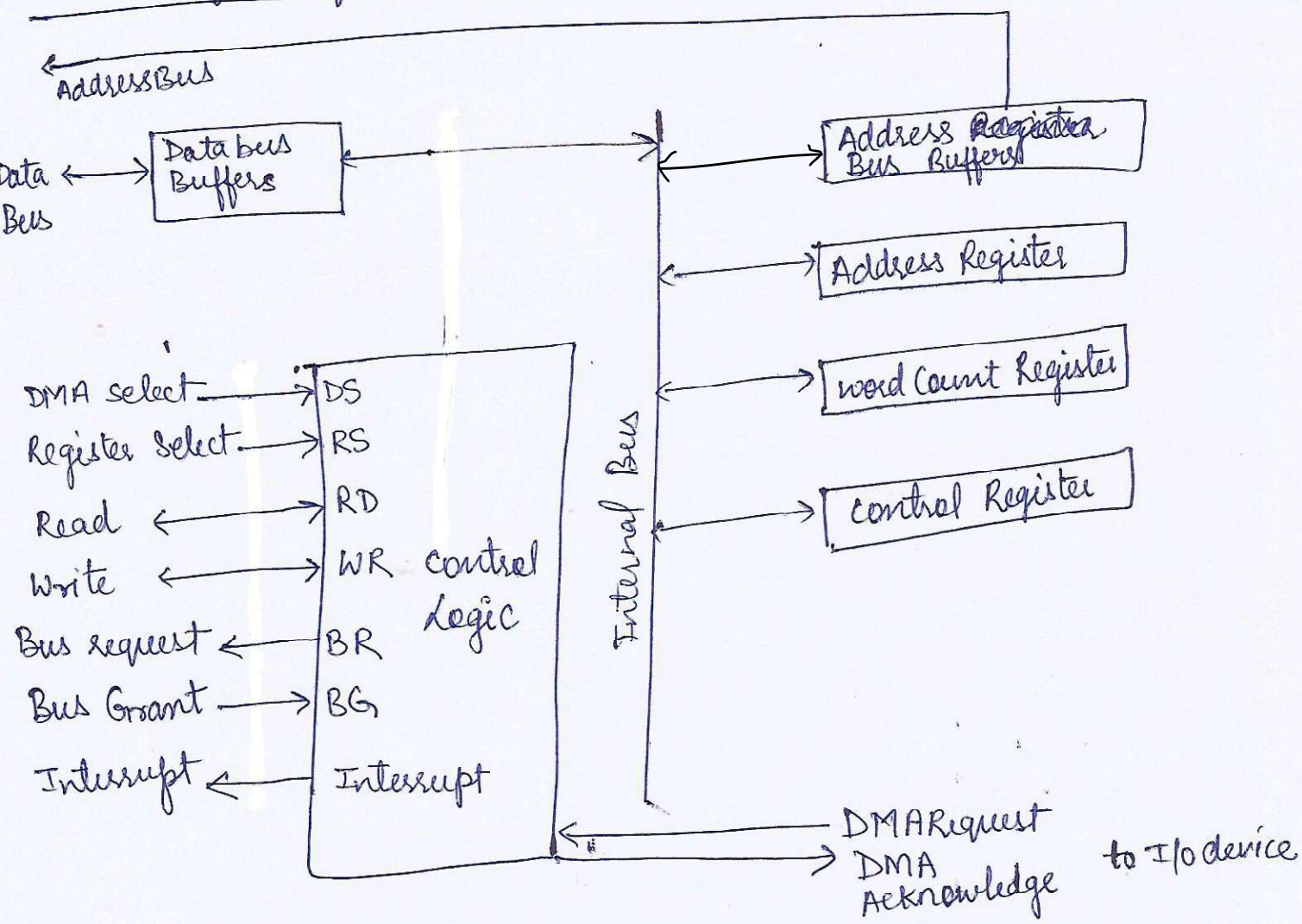


Single Bus, Integrated DMA-I/O



I/O Bus

Block Diagram of DMA



→ Bus Request (BR) input is used ~~to~~ by the DMA controller to request to CPU to leave the control of system bus.

→ The CPU activates the 'Bus Grant' (BG) output to inform the DMA controller.

→ when DMA terminates the transfer, it disable the bus request line. CPU disables the bus grant, takes control of buses and returns it to its normal operation.

DMA Data Transfer modes:

→ DMA block transfer

→ Cycle stealing mode

→ Transparent DMA.

(1) DMA ~~Bl~~ Block Transfer:-

→ In DMA block transfer technique, a block of data of arbitrary length can be transferred in a single burst.

→ This DMA mode is needed for secondary memories like disk drivers, where data transmission can not be stopped or ~~stowed~~ slowed without loss of data.

② cycle steading Mode:

In cycle stealing mode, the DMA controller is allowed to transfer one word at a time, after which it must return the control of the bus to the CPU.

* DMA module must use the bus only when the processor does not need it, or it must force the processor to suspend operation temporarily.

DMA module transfers a word and returns the control to the processor. Note that this is not an interrupt; the processor does not save a context and do something else. Rather, processor pauses for one bus cycle. The overall effect is to cause the processor to execute more slowly. Nevertheless, for a multiple-word I/O transfer, DMA is far more efficient than interrupt driven or programmed I/O.

③ Transparent DMA:

In transparent DMA, DMA is allowed to steal only those cycles when the CPU is not using the system bus.

CPU does not require to have control of system bus during Decode instruction or execute instruction phase.

* DMA transferring data during transparent DMA does not have any adverse effect on CPU performance.

Input - output channels:

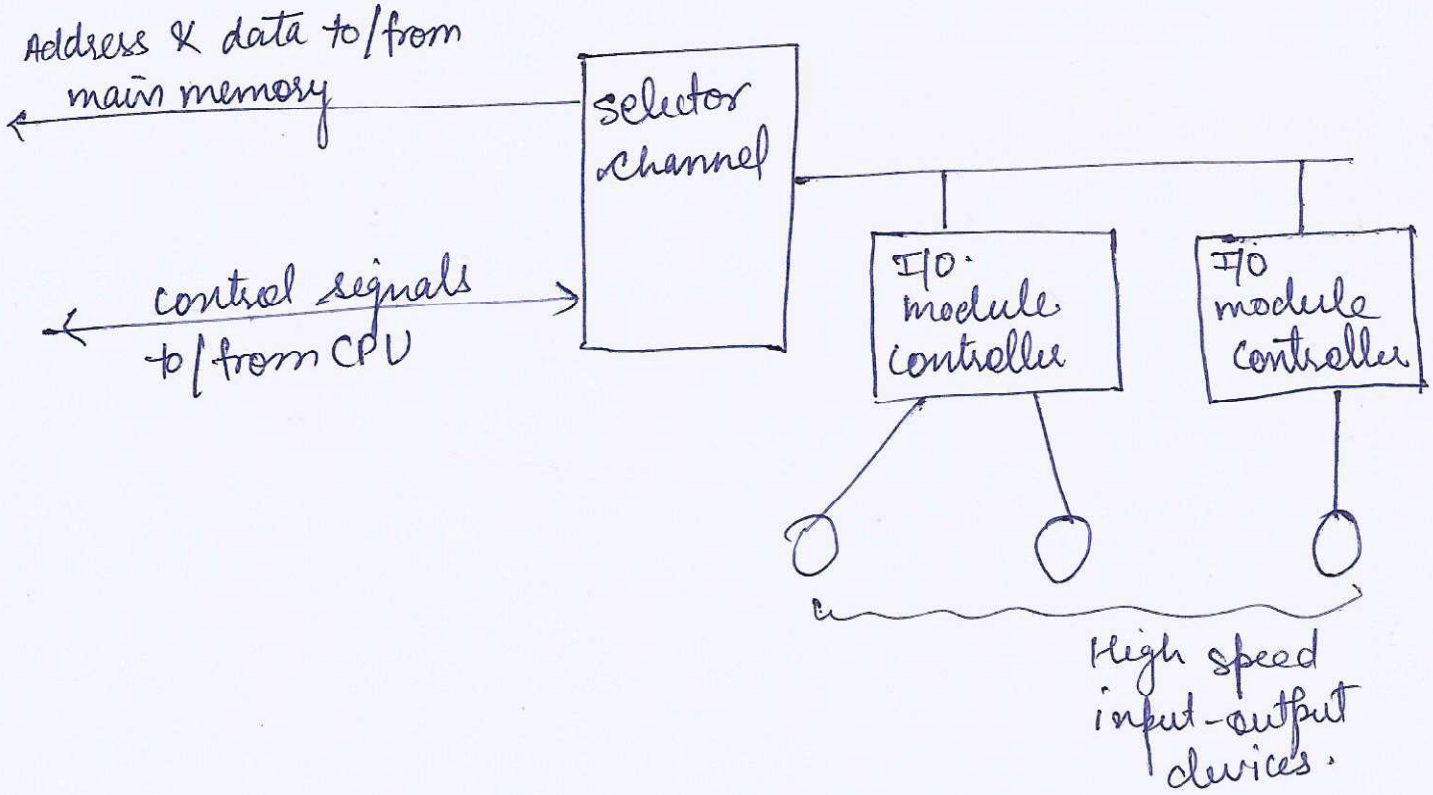
Input-output channels or input-output processors, virtually retrieves CPU from all I/O related activities. It has complete control over device. It can also communicate with CPU.

* I/O processor is a processor like CPU with limited number of instructions. An I/O processor has the ability to execute the I/O instructions. It can also carry out the work of data conversion required by the input-output device.

There are two types of I/O channels:

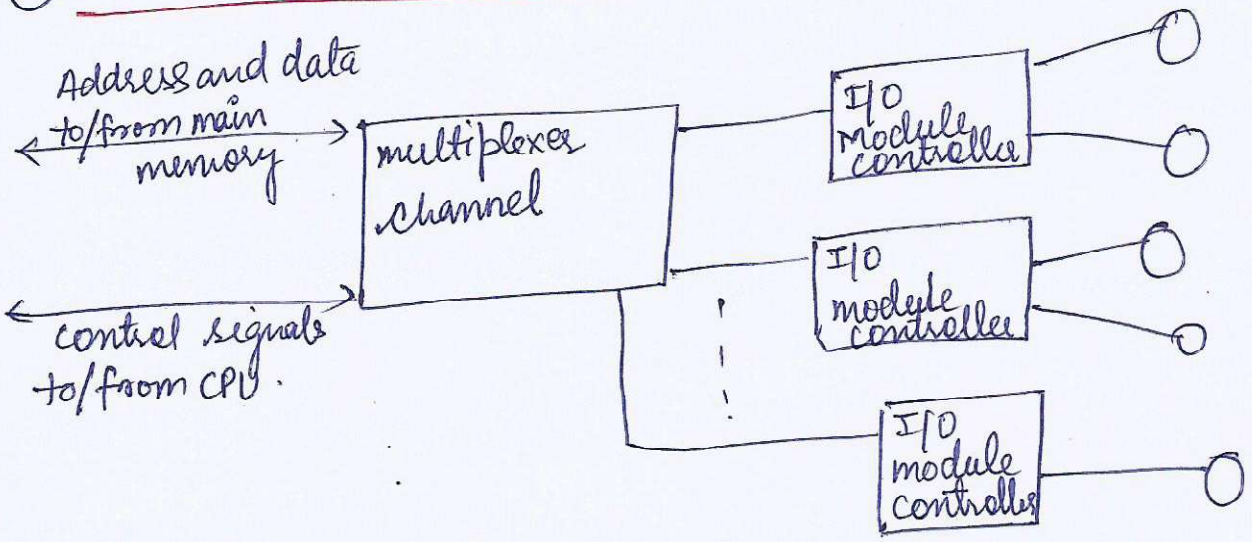
1. Selector channel
2. Multiplexer channel.

1. Selector channel:



- A selector channel can handle multiple high speed devices. Only one device is selected at a time for communication.
- A selector channel is useful in high speed transfer of data in burst mode.

2 A multiplexer channel:



A multiplexer channel can handle I/O with a number of devices at the same time. If the device is slow then byte, multiplexing is used.

<u>Example</u>. There are three devices which need to send individual bytes are

B1, B2, ----- B5 → data from 1st device.
X1, X2, ----- X5 → data from 2nd device.
Y1, Y2 ----- Y5 → data from 3rd device.

Then on a byte multiplexer channel may send the ~~data~~ bytes as B1 X1 Y1 B2 X2 Y2 B3 X3 Y3 B4 X4 Y4 -----

Data Transfer:

- Synchronous
- Asynchronous

Synchronous: If registers in the interface share a common clock with CPU registers, the transfer between two units is said to be synchronous.

Asynchronous: The internal timing in each unit is independent from the other in that each uses its own private clock for internal registers, in this case the two units are said to be asynchronous.

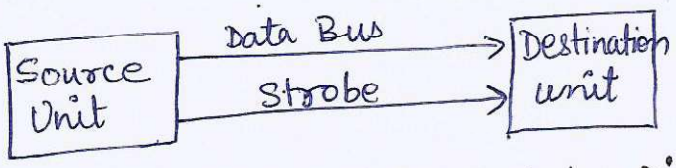
Asynchronous Data Transfer:

→ two independent units transmit control signals to indicate the time at which data is being transmitted.

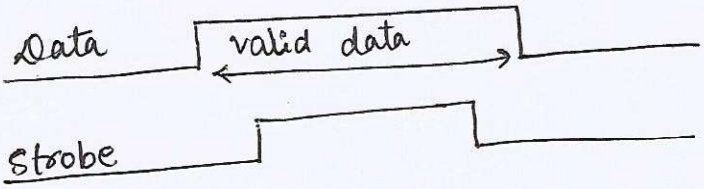
→ Two methods
- Strobe
- Handshaking

Strobed Method (one way) One of communicating device supplies all the control signals.

- Strobe can be activated by either source or destination



Source iniated Strobe Diagram



Timing Diagram

The strobe is a single line that informs the destination unit when a valid data word is available in the bus.

* The source unit first places the data on the databus. After a brief delay to ensure that the data settle a steady value, the source activates the strobe pulse.

* In destination initeated data transfer, destination unit activates the strobe pulse, informing the source to provide the data.

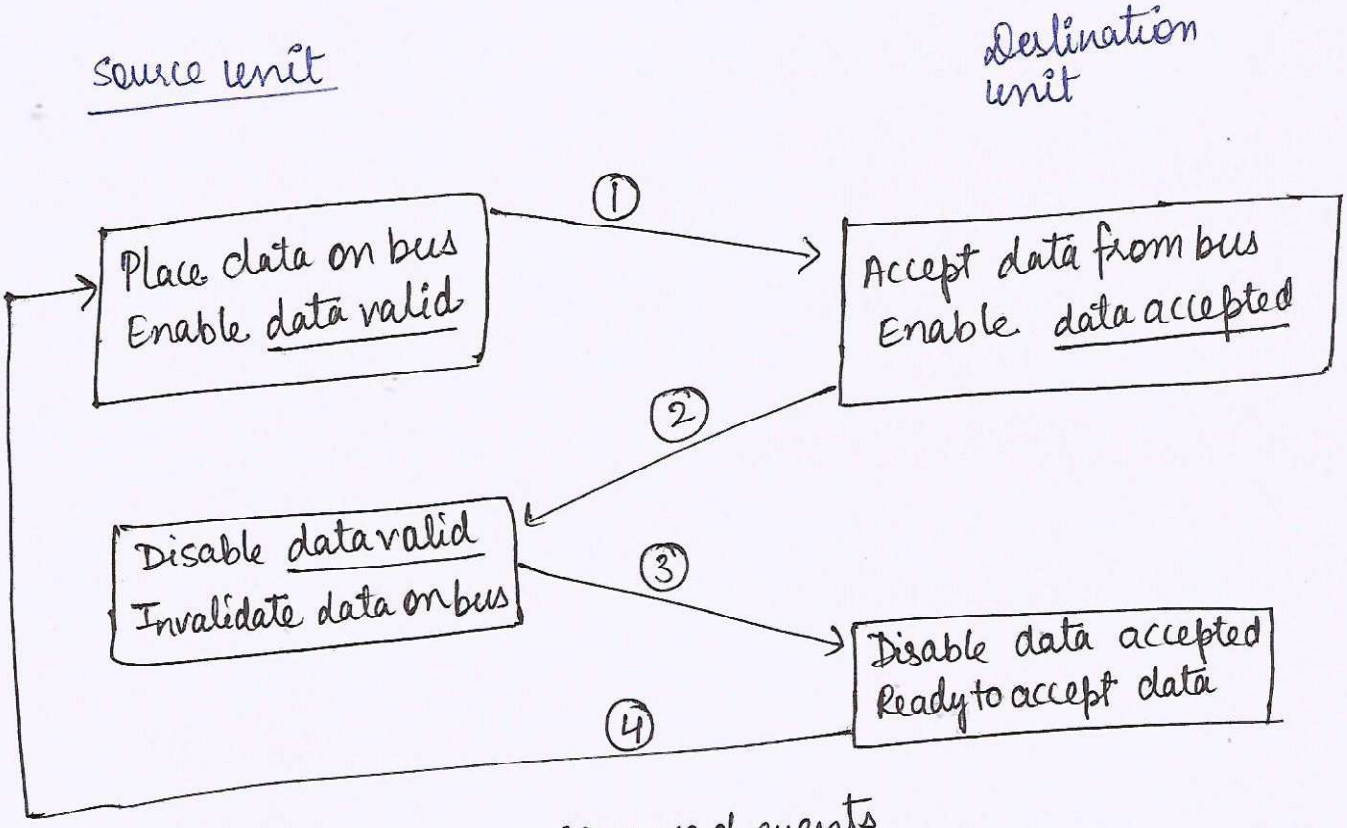
Disadvantage: In strobe method, the source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item.

Handshaking: (Two way)

* The disadvantage of oneway control is that it does not verify that the data transfer has been completed successfully. Data may be lost.

* Control of data transfer is based on the use of handshake between processor and the device selected for input-output.

Block Diagram



sequence of events

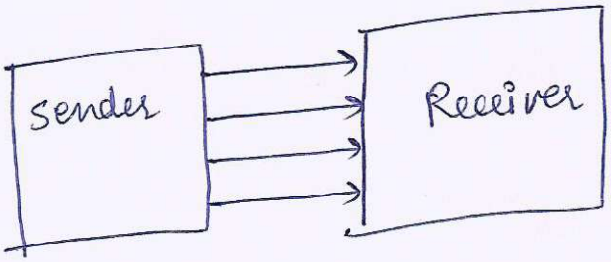
Serial communication
or
Serial Transmission

⇒ Serial data transmission occurs by transmitting data one bit at a time in sequential order over a bus.

Start bit Data bits Stop bits ⇒ To know the start and end of transmission.

Parallel Transmission

In parallel transmission several bits can be transmitted at the same time.



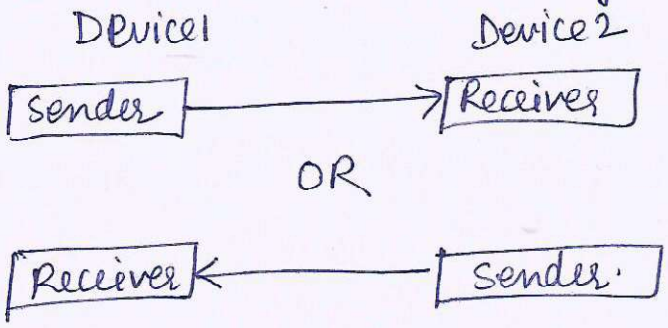
Serial Communication can be
- Simplex
- Half Duplex
- Full Duplex

Simplex Transmission: Signals travels in one direction only.
eg. Keyboard

Half Duplex:

→ capable of sending signals in both the directions, but in only one direction at a time

eg ~~Telephone~~ Walkie Talkie.



Full Duplex: (Bidirectional Transmission)

→ This method allows signal transmission in both the directions simultaneously.

eg Telephone IP service.

COA [redacted] Notes [redacted]

Computer Memory:

→ The memory unit is an essential component of any digital computer since it stores the program and data.

Key characteristics of computer Memory systems:

1. **Location**
   - Processor
   - Internal (Main memory)
   - External (Secondary)

2. **capacity**
   - word size
   - Number of words

3. **Unit of Transfer**
   - Word
   - Block

4. **Access Method**
   - Sequential
   - Direct
   - Random
   - Associative

5. **Performance**
   - Access time
   - Cycle time
   - Transfer rate

6. **Physical Type**
   - Semiconductor
   - Magnetic
   - optical
   - Magneto-optical

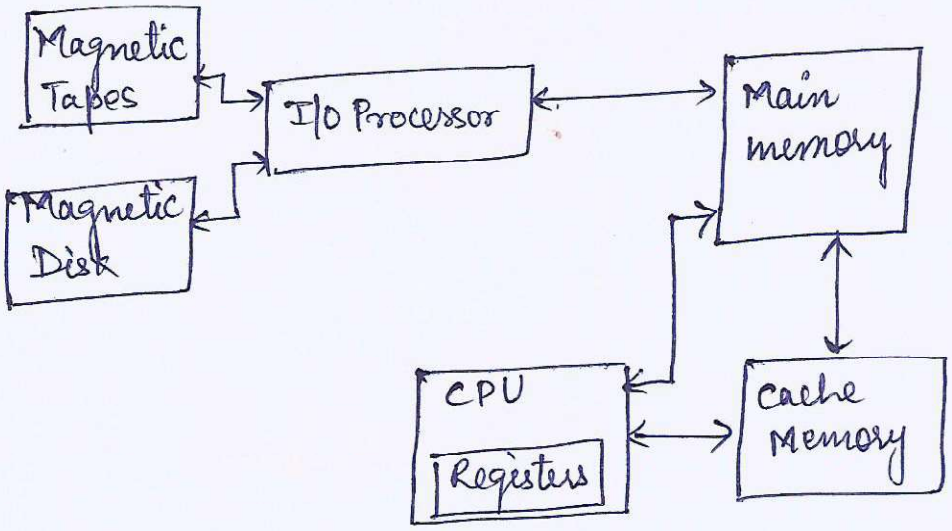
7. **Physical characteristics**
   - Volatile / non volatile
   - Erasable / non-erasable

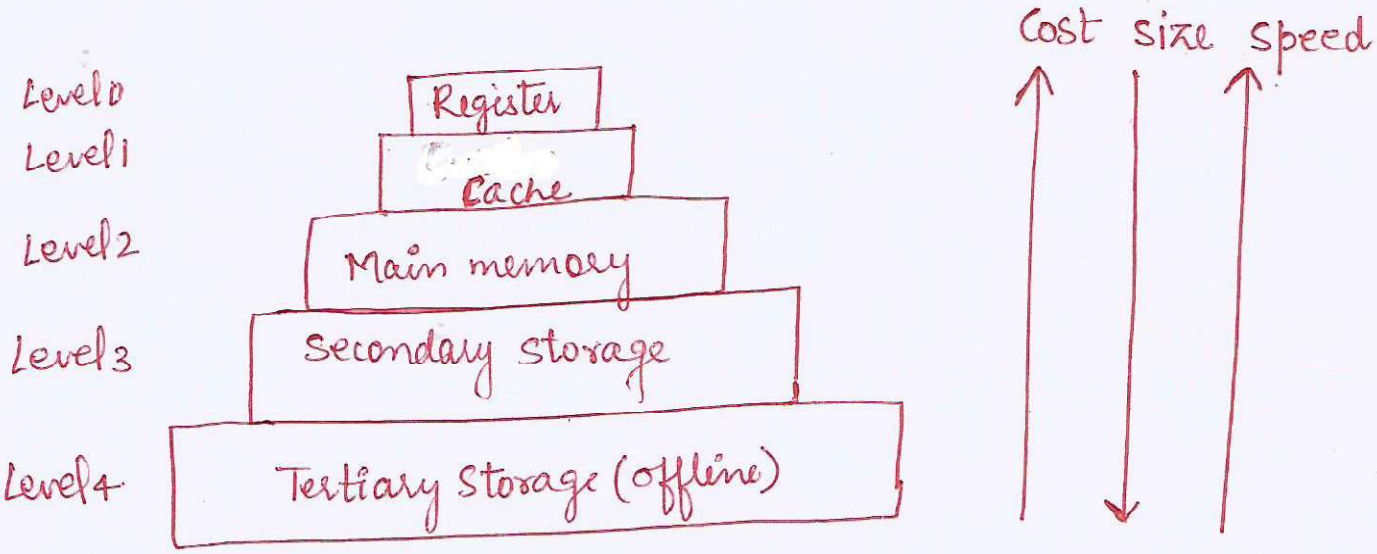
8. **Organization**

Memory Hierarchy:

→ The memories used in computer system are organized in a hierarchy.

Memory Hierarchy in computer System





Registers:

→ Registers are the part of CPU
→ fastest
→ smallest
→ costly
→ implemented using Flip Flops

Cache Memory:

- cache memory is placed between CPU and main memory.
- Faster than Main memory.
→ Most recently used information is stored in cache.
→ Cache memory is implemented using static RAM.

Main Memory:

- Main memory can directly communicate to the CPU.
- The information is first kept in the main memory to get accessed by the CPU.
- Main Memory is implemented using Dynamic RAM (DRAM)

Secondary Storage: (External Memory)

Level 3: Secondary storage like magnetic disk are present.

- Non-volatile type of memory.
- used as backup storage.
- cheaper and larger than main memory.

Level 4: (Removable Media or offline media)

- The data not being currently used can be kept in removable media. Eg. CD, Pendrive etc.
- Non-volatile
- cheaper
- Larger in size.

Main memory: (Primary Memory or simply memory)

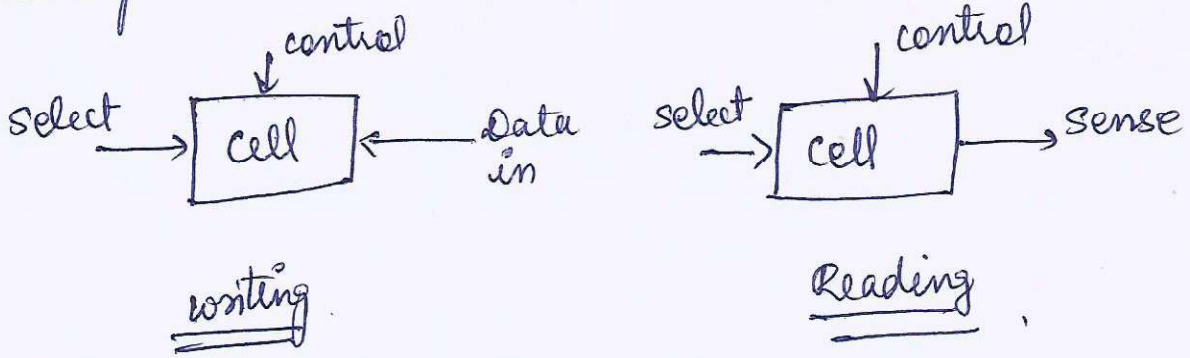
- Semiconductor memory and internal memory.
- can directly communicate to the processor and auxilliary memory with the help of I/O processor.

- Two types of main memory
  - → ROM (Read only Memory) ⇒ Non volatile
  - → RAM (Random Access Memory) ⇒ Volatile

- Data transfer b/w main memory and CPU takes place by.
  - MAR ⇒ Memory Address Register
  - MDR ⇒ Memory Data Register

Organization: The basic element of semiconductor memory is 'Memory cell.'

- Memory cell represents two states zero and one.
- Memory cell is capable of being written into (at least once), to set the state
- Memory cell is capable of being read to sense the state.



Types of RAM:

→ Static RAM
→ Dynamic RAM

Static RAM:

→ Faster than DRAM
→ Expensive
→ Used for cache memory.
→ single block of memory needs 6 transistors
→ complex uses transistors and latches.

DRAM:

→ Slower than SRAM
→ cheaper than SRAM.
→ used in Main memory
→ simple and uses capacitors and very few transistors
→ stores charges on capacitors
→ needs periodic charging (or referenshing)

ROM:

- ROM stands for Read only Memory. ~~The mem~~
- we can only read from this type of memory.
- information is stored in such memories during manufacturing
→ used as Bootstrap (a program to ~~to~~ start computer).

Types of ROM

→ PROM
→ EPROM
→ EEPROM
→ Flash Memory

PROM (Programmable ROM):

→ can be modified only once by a user.
→ it is not erasable.

EPROM: (Erasable and Programmable ROM)

→ can be altered or modified multiple times
→ updated each time from initial stage.
→ Data erased completely.
→ More expensive than PROM.

EEPROM: (Electrically Erasable Programmable ROM)

→ can be altered or modified multiple times
→ selectively programmed and erased
→ More flexible than EPROM.
→ More expensive than EPROM.

Flash Memory:

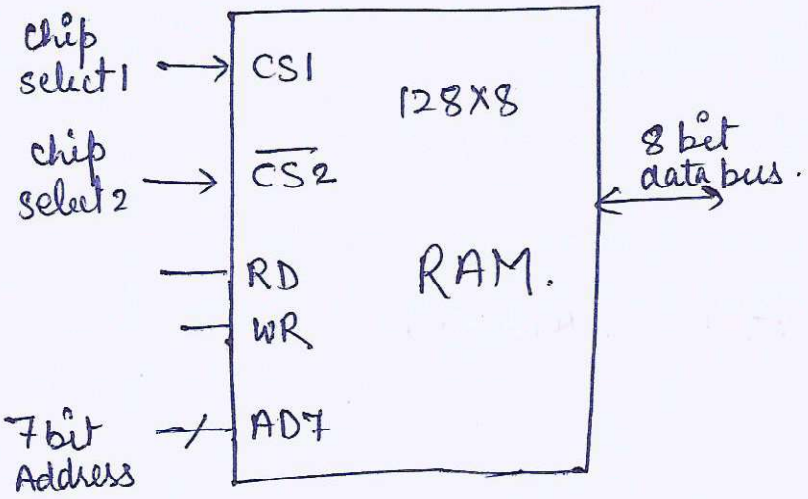
→ is a type of EEPROM.
→ is programmed and erased in large blocks
→ may also be a standalone memory storage device ~~dek~~ such as a USB drive.
→ can be put between EPROM and EEPROM in both the cost and functionality.

EPROM → entire chip erasable.

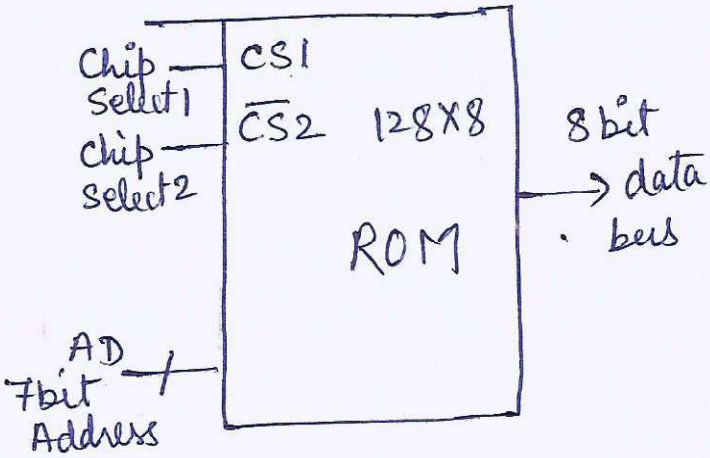
Flash → block erasable

EEPROM → byte erasable

RAM chip:



ROM chip



RAM size ⇒ m X n ← ~~word~~ size of each word.

↑ denotes number of words in RAM.

Ques: if RAM chip size is 128X8. then how many RAM chip will be used to make 2048 bytes of RAM.

RAM chip size = 128X8

RAM size required = 2048X8 because 2048 bytes

Number of chips required $= \frac{2048}{128} = \frac{2^{11}}{2^{7}} \Rightarrow 2^{4} = 16$ chips

The decoder to select 16 chips

4X16 ⇒ size of decoder required to select one of the 16 chips.

Ques:

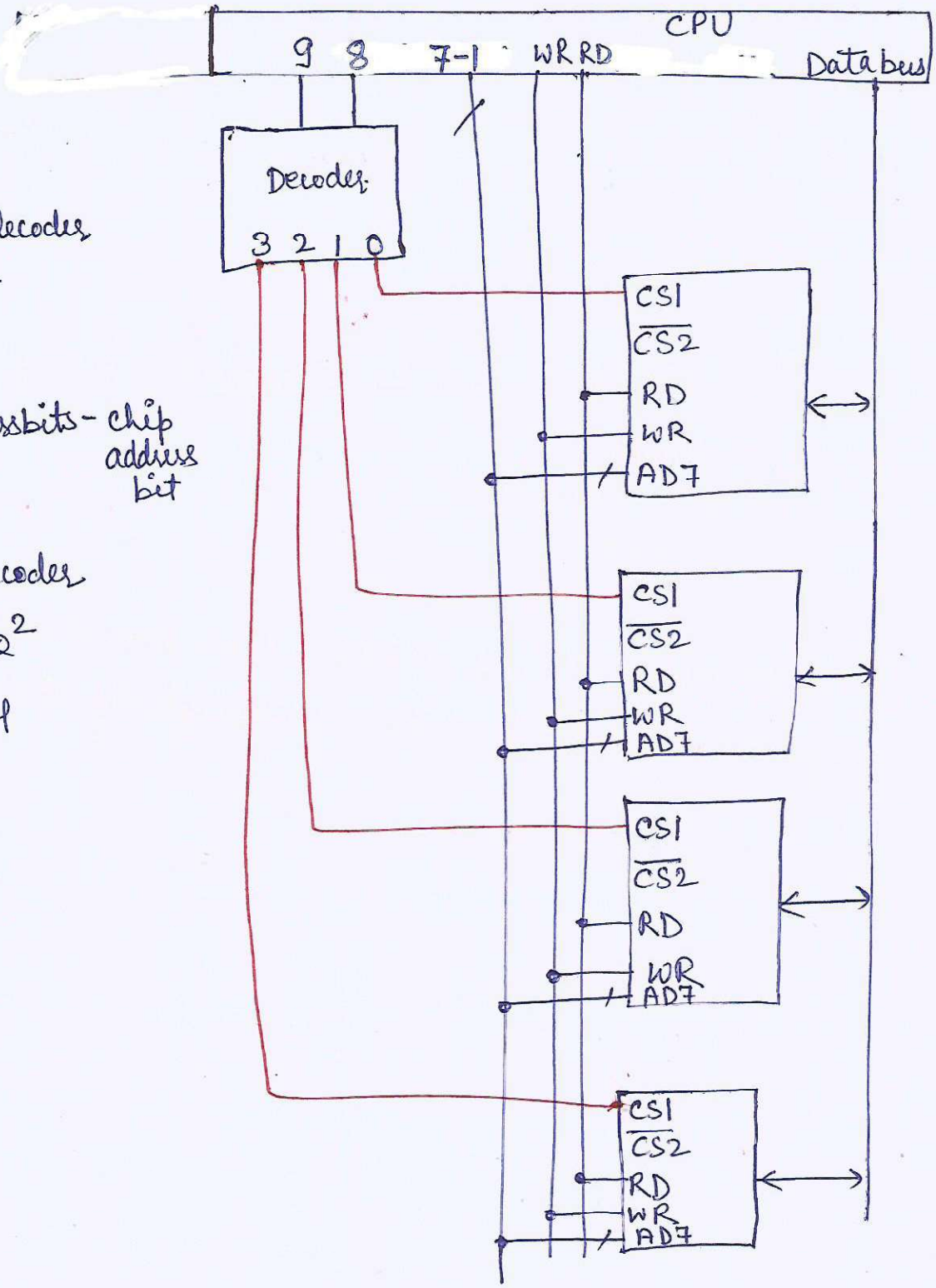
RAM chip size = 128X8

Memory size required 512X8

Number of chips required ⇒ $\frac{512}{128}$ ⇒ 4

Address lines in RAM chip ⇒ 7

Address bits for memory ⇒ 9



inputs of decoder

= 9-7

= 2

⇒ Addressbits - chip address bit

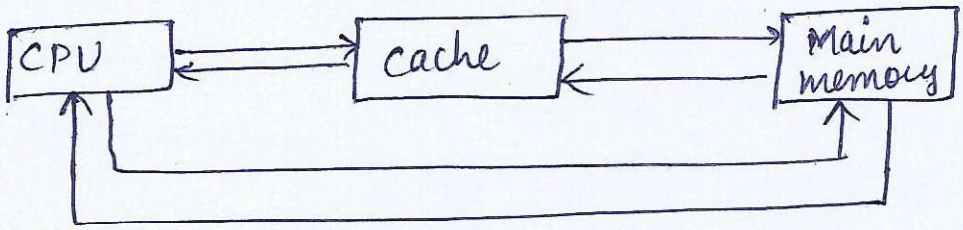
Size of decoder

= 2×2^2

= 2X4

Cache Memory:

- cache memory is an internal memory placed between CPU and main memory.
- Most recently used information is kept in cache memory.



* The cache contains the copy of some portions of main memory when the CPU tries to read a word from main memory, cache checked is performed, if that word is present in cache or not.

  if yes, the word is delivered to CPU

  if no, a block of memory consisting of that word is read into the cache and is delivered to the CPU.

Imp

* Also, cache memory satisfies 'Locality of Reference'. That is the block of data which is fetched from the main memory is likely that there will be more future reference for that block of word.

→ Temporal Locality of Reference

→ Spatial Locality of Reference.

Cache Fundamentals :

Cache Hit : when data is found in cache.

Cache Miss : when data is not found in cache.

Hit time : Time taken to access the cache. (CPU ← cache)

Miss Penalty : Time taken to move data from main memory to cache and then to CPU. when ~~word is~~ data is not found in cache
(CPU ← cache ← main memory)

Hit Ratio : The hit ratio is the fraction of accesses which are a hit

Miss Ratio : The miss ratio is the fraction of accesses which are a miss.

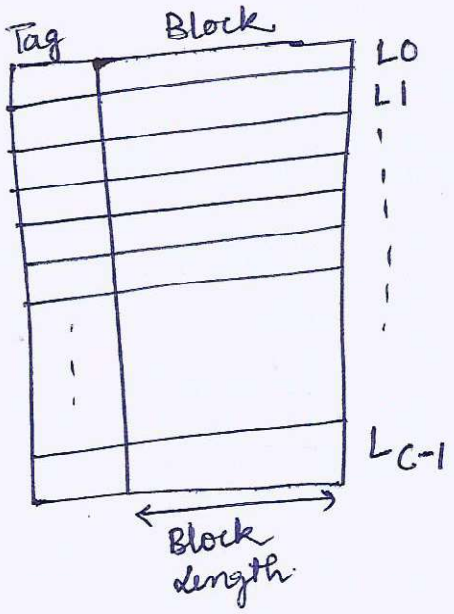
Structure of Cache / Main memory :

The main memory has 2^n addressable words with n-bit addresses each
For mapping purpose, main memory is devided into fixed length blocks with X-words each

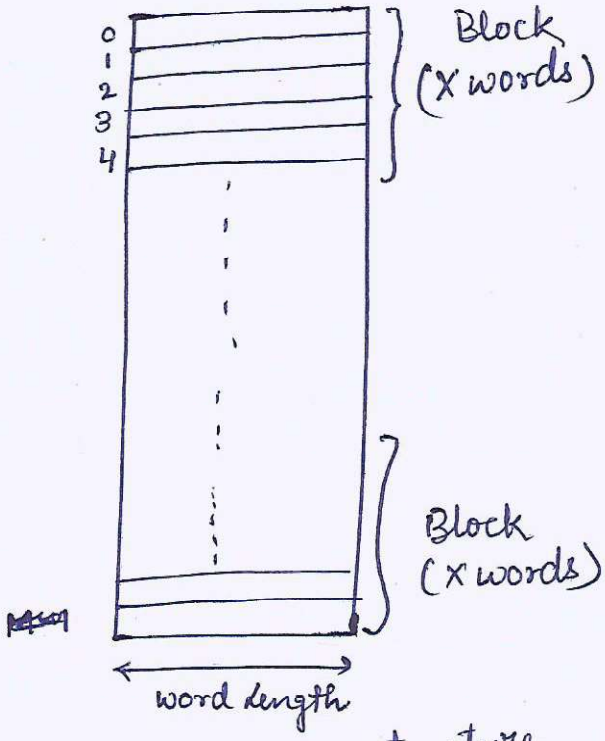
$$\text{Number of blocks } M = \frac{2^n}{X} \text{ blocks} = \frac{2^n \text{ words in main Memory}}{X \text{ number of words in each block.}}$$

- cache consists of C lines of X words each
- Each line consists of X words.

$$\boxed{C << M}$$



C ⇒ Number of lines in cache.
M ⇒ Number of blocks in MM



Main memory structure

When a block of memory is read, it is transferred to one ~~one~~ of the lines of cache. Because there are more blocks than lines, one line can not be assigned permanently to one particular block. So each cache line has 'tag' that identifies which particular block is being stored.

Cache Design issues:

Element of Cache Design

- Cache parameters
  - → Block size
  - → Line Size
  - → Cache Size
- Mapping function
  - → Direct
  - → Associative
  - → Set Associative
- Replacement Algo
  - FIFO
  - LRU
  - optimal
- Write Policy
  - Write Through
  - Write Back
  - Write Once

Cache Parameters:

- The size of the cache should be small enough, so that the cost per bit is less but the size should be large enough.
- The larger block size reduce the number of blocks that fits into the cache, so block size should not be large.

Mapping Function:

There are fewer cache lines than memory blocks, an algorithm is required ~~for~~ for mapping of memory blocks into cache lines.

* Mapping function: maps the memory block into a cache line

Types of mapping function: cache mapping techniques

→ Direct
→ Associative
→ Set Associative.

Address Mapping or Cache Mapping

- The size of memory is smaller than main memory. Therefore, There should be some mechanism to fill the data in the cache. Here the mapping techniques can be used.
- Address mapping is mapping of main memory blocks into cache lines
- Three techniques:
  - → Direct Mapping
  - → Associative mapping
  - → Set-Associative Mapping

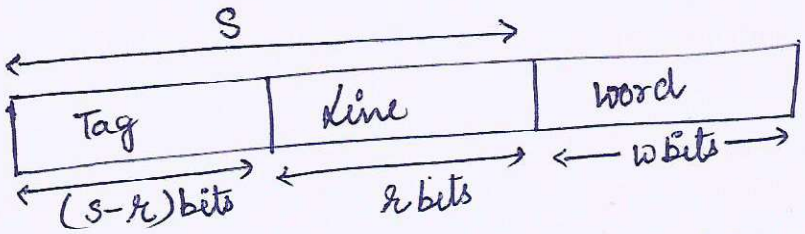
Direct Mapping:

- In this mapping, each block of main memory maps into only one possible cache line.
- It can be expressed as

$$\boxed{i = j \bmod m}$$

where $i \rightarrow$ cache line number
$j \Rightarrow$ Main Memory block number
$m \Rightarrow$ number of cache lines.

- Each main memory address can be divided into three parts



Tag ⇒ tells which block is currently being stored in cache
Line ⇒ line bits identifies one of the cache lines ($m = 2^r$)
$m \Rightarrow$ number of lines $\Rightarrow 2^r$

word bit or Block offset bit identifies which word (or byte in case of bytes ~~in blocks~~) in the block.

~~S is address length.~~

Address Length = $(S+W)$ bits

Number of words or addressable unit = 2^{S+W} words or bytes

Block size = Line size = 2^{W} words or bytes

Number of blocks in main memory = $\frac{2^{S+W}}{2^{W}} \Rightarrow \frac{\text{Total number of words in main memory}}{\text{Block size}}$

$= 2^{S}$

Number of bits to identify the blocks in main memory = S bits = Block index

Number of lines in cache = $\frac{\text{Total number of ~~lines~~ words in cache}}{\text{Line size}} = 2^{r}$

Number of bits to identify lines = r = cache index

Tag bits = Block index − cache index

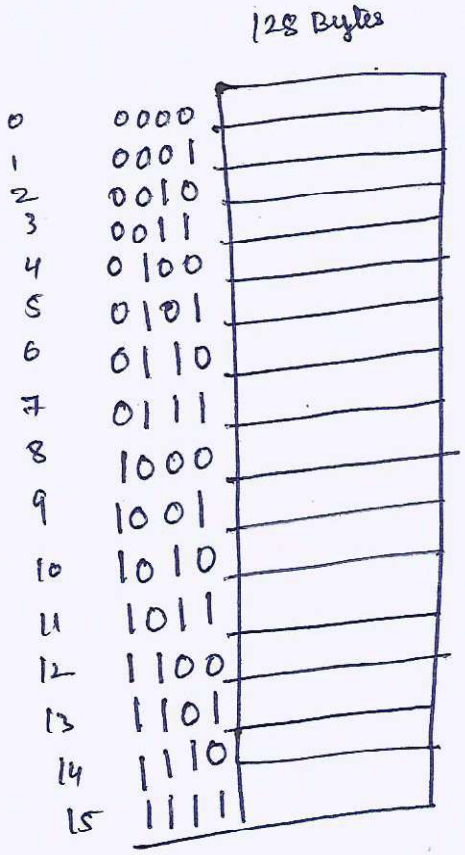
$= (S-r)$ bits

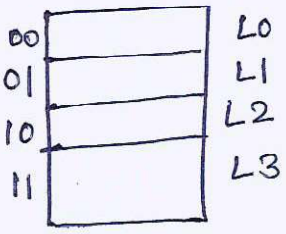
<u>Example</u> <u>Direct Mapping</u>

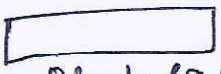
Main memory size = 128 bytes

Cache size = 32 bytes

Block size = 8 bytes







Block (8 bytes)

(BI) Block index ⇒ 4 bits

(CI) cache index = 2 bits

Formula ⇒ $\boxed{i = j \bmod m}$

$i \rightarrow$ line number.

$j \rightarrow$ block number.

$m \rightarrow$ number of lines in cache.

$m = 4$

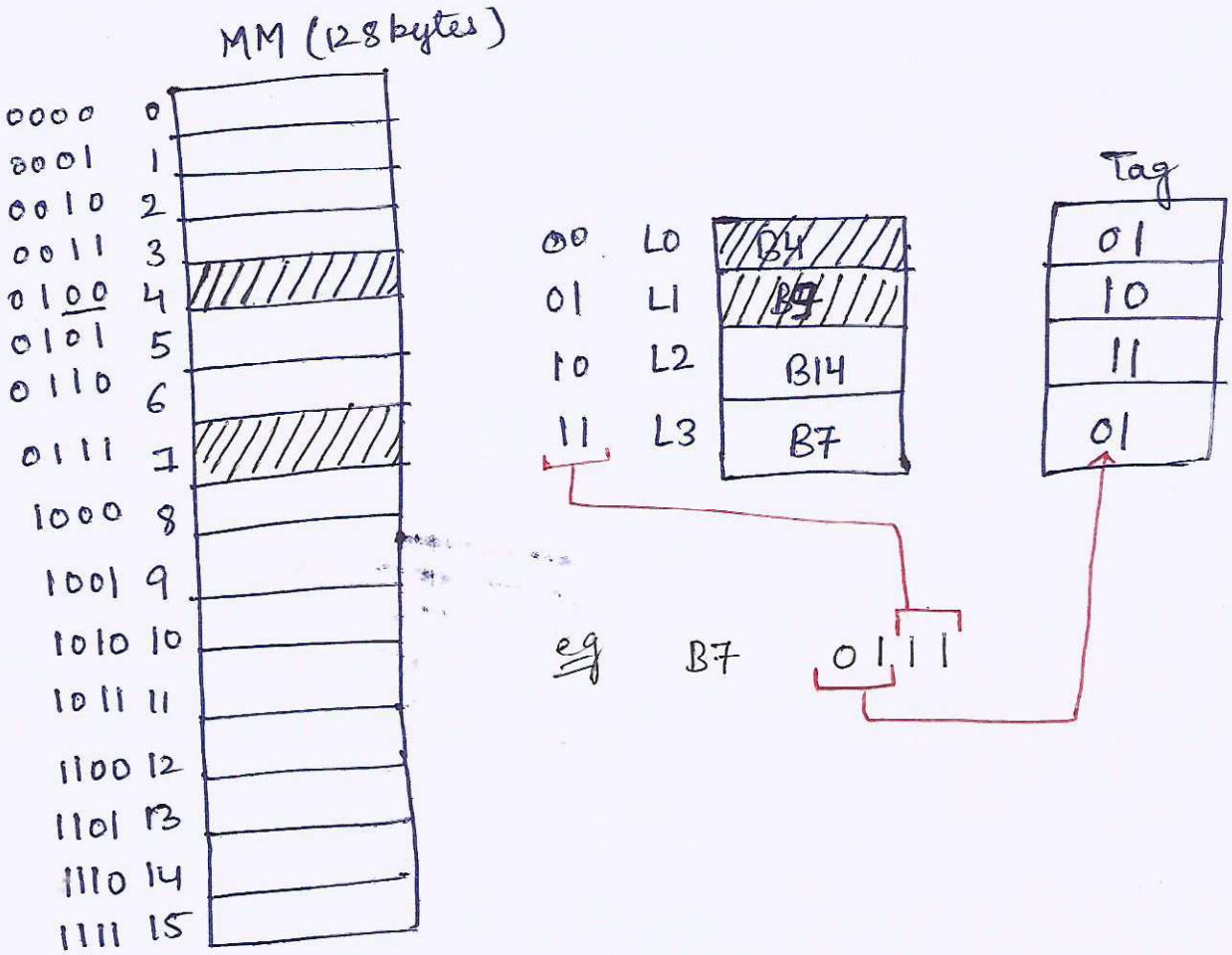
Line number	Blocks can be placed
L0	B0, B4, B8, B12
L1	B1 B5 B9 B13
L2	B2 B6 B10 B14
L3	B3 B7 B11 B15

* The least significant Block index bits are matched with cache index bits

So line 0 can hold B0, B4, B8 and B12. So CPU to understand which block in the cache, it maintains some extra information bit which is called tag bits.

→ So for each line CPU holds tag bits

* In direct mapping Tag holds the (BI-CI) ~~bits~~ MSB bits of the block index BI.
  - Tag holds the Most significant bits of the block.



Disadvantage of Direct mapping:

→ Direct mapping imposes restriction on the placement of a particular memory block into a particular cache line.

→ Poor hit ratio.

Advantage: Simple to implement.

- Number of blocks in main memory $= \dfrac{\text{Main memory size}}{\text{Block size}}$

$$= \frac{128 \text{ bytes}}{8 \text{ bytes}} = 16 \text{ blocks.}$$

- Number of lines in cache $= \dfrac{\text{cache memory size}}{\text{Block size}}$

$$= \frac{32 \text{ bytes}}{8 \text{ bytes}} = 4$$

- Block index bits $= \log_2(\text{Number of blocks})$

$$= \log_2 16 = \log_2 2^4 \Rightarrow 4$$

Block index = 4 bits

- Cache index bits $= \log_2(\text{Number of lines})$

$$= \log_2 4 = \log_2 2^2$$

$$= 2\log_2 2$$

$$= 2$$

- Tag bits = Blockindex − cache index

$$= 4 - 2$$

$$= 2 \text{ bits}$$

- Address bits $= \log_2(\text{Main memory words})$

$$= \log_2(128)$$

$$= 7 \text{ bits}$$

Associative Mapping: or Fully Associative Mapping:

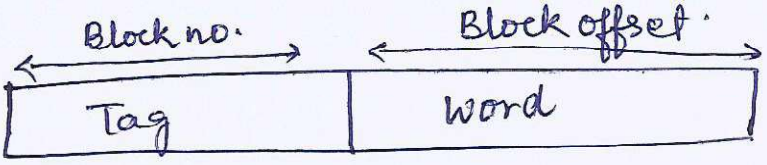
→ Over comes the problem of direct mapping method.

→ In associative mapping any mainmemory block can be placed in any cache line.

→ whenever a main memory block is copied into cache line, the complete block index is placed in the tag bits.

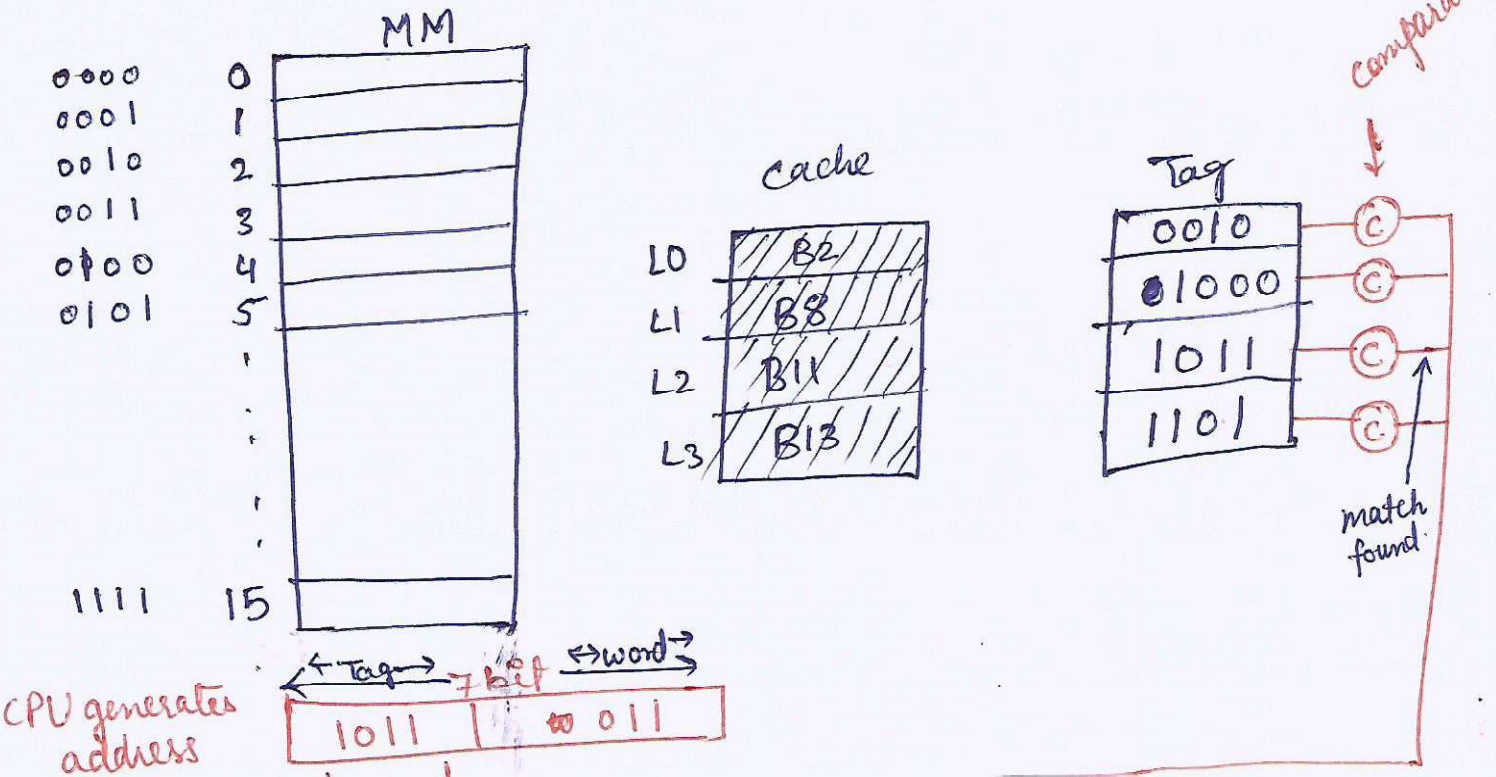
Tagbits = Block index bits

→ Address bits can be divided into two parts



Example Main memory size = 128 bytes
Cache size = 32 bytes
Block size = 8 bytes.

⇒ Number of blocks = 16 BI = 4
⇒ Number of lines = 4 CI = 2



Advantages:

→ Very High Hit ratio

→ better block placement strategy which ~~reduced~~ reduced contention.

Disadvantages:

→ Associative memory is costly.

→ Significant increase in tag length

→ complex :

→ Searching is an overhead.

Set Associative ~~Memory~~ Mapping

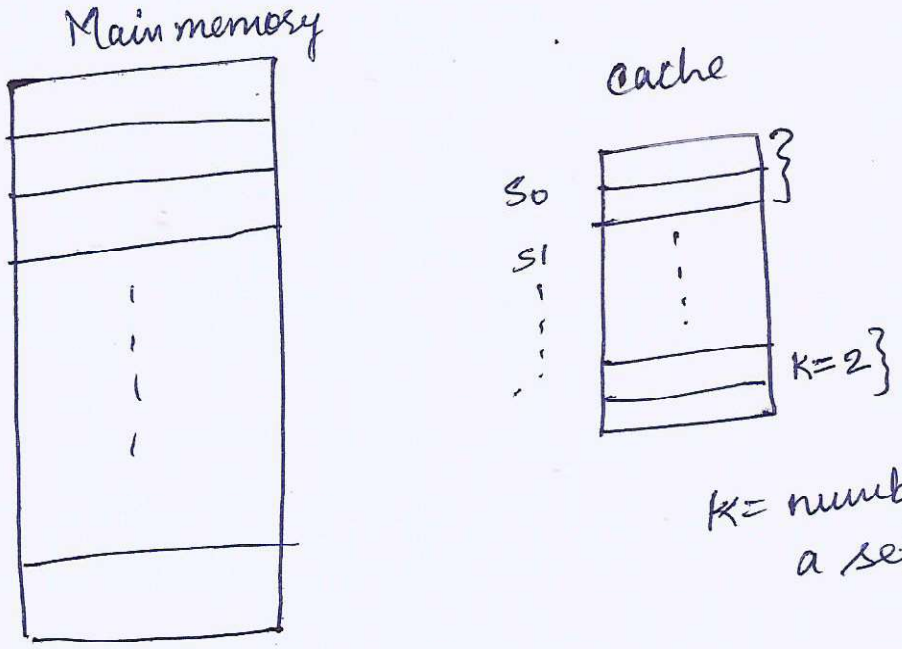
In set associative mapping, ~~the~~ cache lines are grouped in sets. Grouping is done in K-way where K is the number of lines in a set.

> Set associative mapping technique is the combination of direct mapping and associative mapping.

In this mapping the memory block are connected to a set using the logic of direct mapping ~~an~~ (this mapping is from main memory to set). and inside the set, the mapping is associative mapping.

Advantage: The advantage is, the associative mapping is only for a set. The number of lines in a set is only K. Therefore the number of comparisons will be less. So the search time will be reduced.

So the set associative is better than full associative mapping.



Grouping K-way ~~Hasek~~

e.g ⇒ 2-way set

K= number of lines in a set.

Virtual Memory.

→ Virtual memory is a technique that lets the execution of different processes which are not totally in main memory

→ One main benifit of this method is that programs can be larger than the physical memory (main memory).

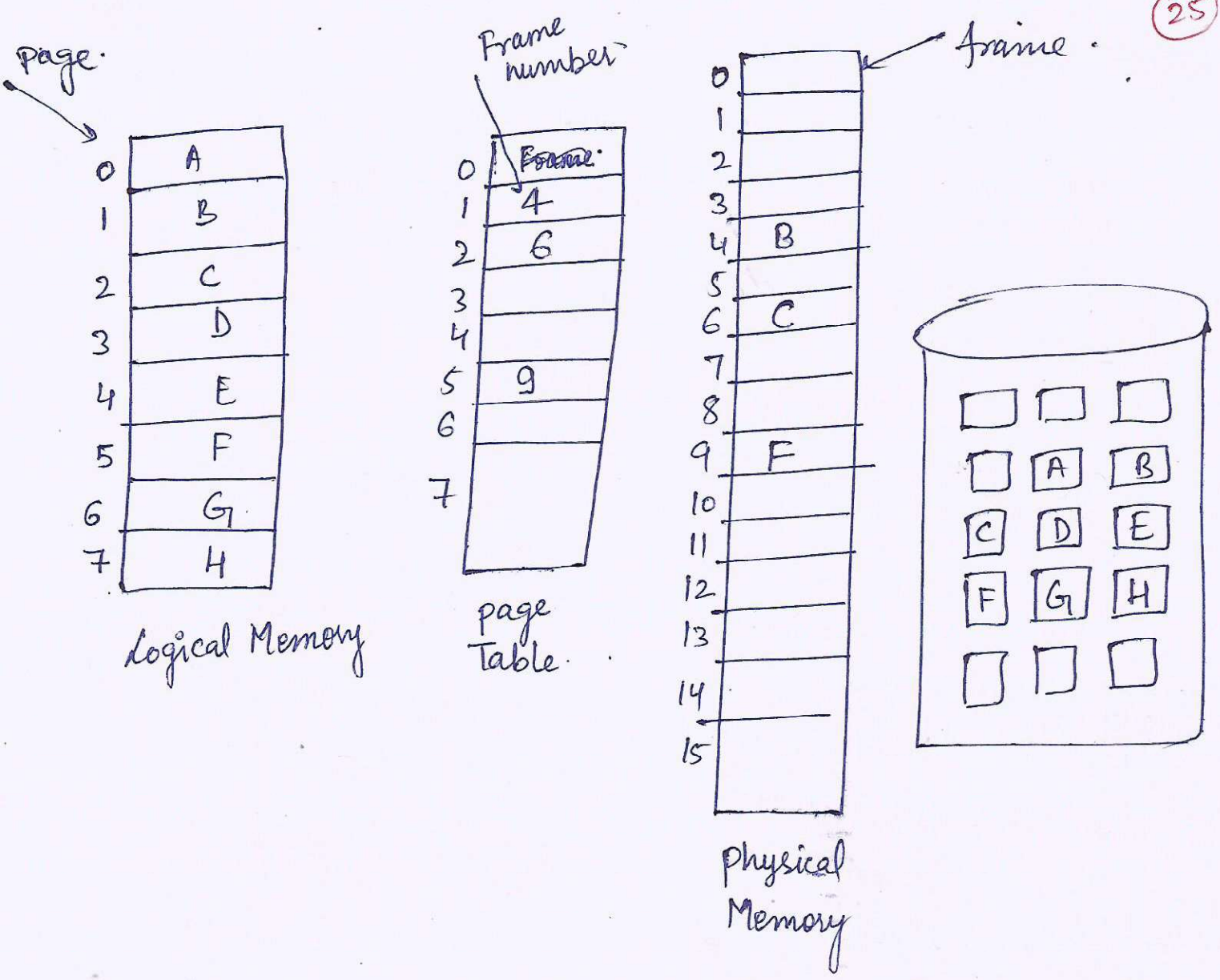
→ The virtual address space of any process is defined as the logical (or virtual) view of how any process gets stored in memory. Normally this view is where a process begins at a certain logical address (say address location 0) and then exists in continuous memory.
Although, the fact is physical memory (main memory might be structured in the form of page frames. and physical page frames are assigned to a process may not be adjacent.

→ Memory Management Unit (MMU) maps the logical pages to physical page frames in memory.

page = logical memory divided into fixed sized block called pages.

frame ⇒ physical memory divided into fixed sized blocks called frames

page size = frame size.



when a page is referenced. if it is found in main memory, then page hit occurs. If not then a page miss occurs, and is called a page fault.

⇒ A computer can address more memory than the amount physically installed on the system. This extra memory actually called a virtual memory and it is the section of a hard disk that's set up to emulate the computers RAM.

Replacement Algorithm:

→ when a page fault (page being accessed is not in memory), the operating system has to choose a page to remove from memory to make space for the page that has to be brought in.

→ The criteria for selecting a page for replacement constitute a replacement policy.

→ The main goal is choosing a replacement policy to maximize the hit ratio.

→ Replacement Algorithms are:

→ FIFO

→ LRU

→ optimal

First In First Out Page Replacement Algorithm:

- FIFO select a page for replacement which has been least recently loaded.

- Easy to implement

→ It does not give high hit ratio

Ques Assume that memory consists of 3 frames and following pages are referenced

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1,

Show the behaviour of page replacement using FIFO.

Solution Number of frames = 3, initially all frames are empty

Reference string

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0
			1	1		1	0	0	0	3	3			3	2	
		0	0	0		3	3	3	2	2	2			1	1	
	7	7	7	2		2	2	4	4	4	0			0	0	
	F	F	F	F	H	F	F	F	F	F	F	H	H	F	F	H

F denotes page fault occurred. page 0 ~~was~~ was not available in frames

H denotes page Hit as page 3 found in frames.

	1	7	0	1
		2	2	1
		1	0	0
		7	7	7
	H	F	F	F

Total number of page fault = sum of all page faults.

= 15

Least Recently Used :

→ LRU policy replaces the page in memory that has not been referenced for longer time.

→ This algo replaces a page that has been unused for the longest time

Reference string Given :

{Number of frames = 3 , all frames initially empty}

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
7	7	7	2		2		4	4	4	0			1		1		1		
	0	0	0		0		0	0	3	3			3		0		0		
		1	1		3		3	2	2	2			2		2		7		
F	F	F	F	H	F	H	F	F	F	F	H	H	F	H	F	H	F	H	H

Here 2 replaces 7 because Most recently used 1, then page 0 and least recently used is page 7

Number of page fault = 12

Optimal Page Replacement Algorithm :

- In optimal page replacement policy, that page is replaced which will not be used for the longest period of time.
- It has lowest page fault.

→ This technique is not implemented because it looks into future references.

Number of frames = 3

Reference string:

At this point, page 7 will be replaced

So page 3 will be replaced

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
	7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
		0	0	0	0	0	0	4	4	0	0	0	0	0	0	0	0	0	0	0
			1	1		3	3	3		3	3			1				1	1	
	F	F	F	F	H	F	H	F	H	H	F	H	H	F	H	H	H	F	H	H

Number of page fault = 9

Ques 2: Consider a reference string 1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3. The number of frames in memory are 4. Find out the number of page faults respective to: optimal page replacement, FIFO and LRU page replacement algorithm.

Solution: Number of frames = 4

FIFO Algo.

Reference String:

	1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
	1	1	1			5			5					5				5	4	4
		2	2			3			3					3				2	2	2
			3			2			2					1				1	1	1
						1			6					6				6	6	3
	F	F	F	H	H	F	H	H	F	H	H	H	H	F	H	H	H	F	F	F

Number of page fault = 9

LRU:-

Number of frames = 4

Reference string:

page 1, 2, 5 will not be replaced

	1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
	1	1	1			1			1				3	3				3	4	4
		2	2			2			2				2	1				1	1	1
			3			3			6				6	6				6	6	3
						5			5				5	5				2	2	2
	F	F	F	H	H	F	H	H	F	H	H	H	F	F	H	H	H	F	F	F

at this point page 6, 5, 2 will not be replaced

Number of page fault = 10

Optimal Page Replacement

Number of frames = 4

Reference string:

at this point page 1 will be replaced

	1	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
	1	1	1			1			1					1					1	
		2	2			2			3					3					3	
			3			3			2					2					4	
						5			6					6					6	
	F	F	F	H	H	F	H	H	F	H	H	H	H	F	H	H	H	H	F	H

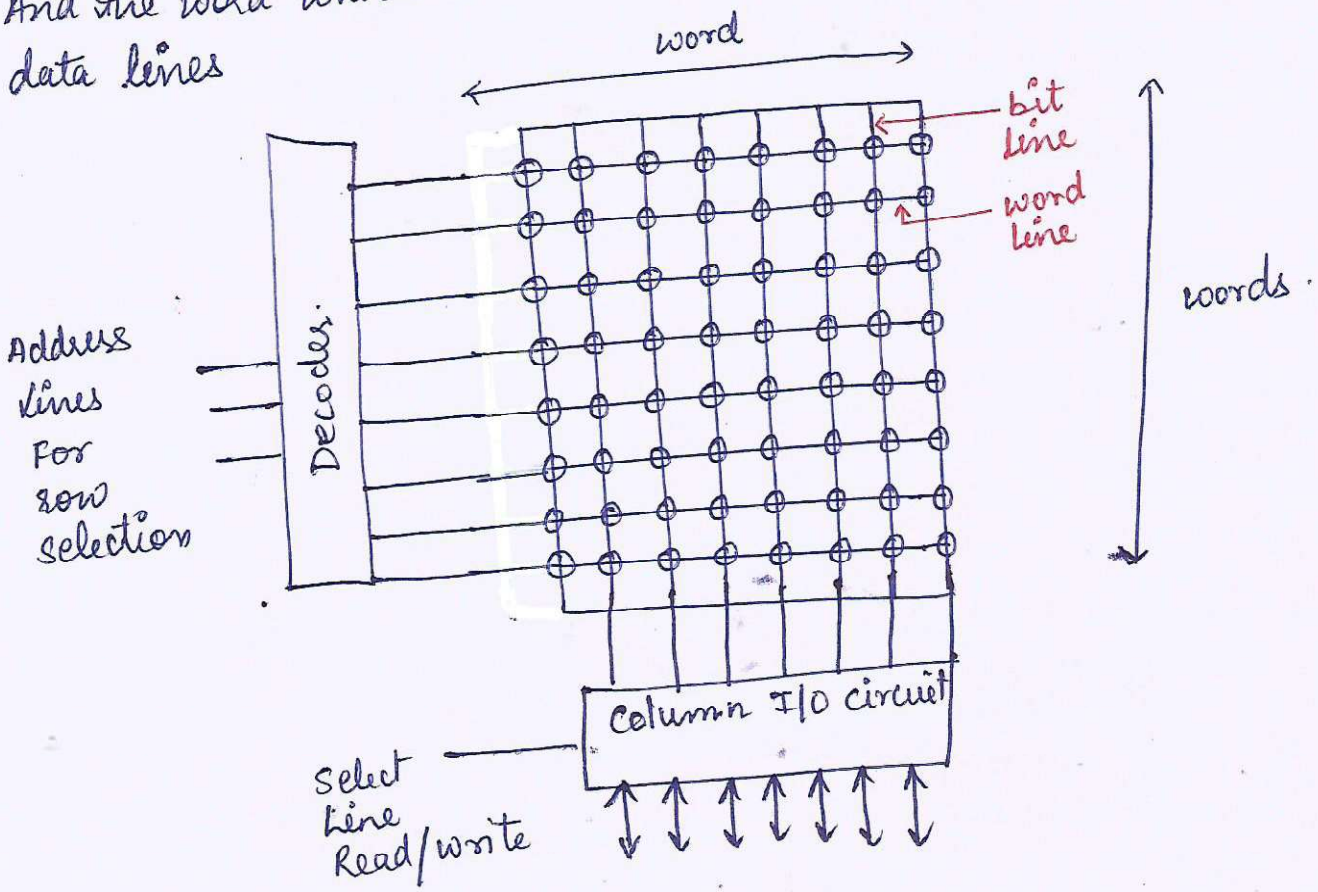
you can replace any page except page 3 or may use FIFO concept

Number of page fault = 7

2D Memory Organization :

In 2D organization, memory is divided in the form of rows and columns. Each row contains a word. A decoder is used to select the row which address is contained in MAR (Memory Address Register).

And the word which is selected will be read or written through the data lines



memory size in this example ⇒ 8×8 → each word size.

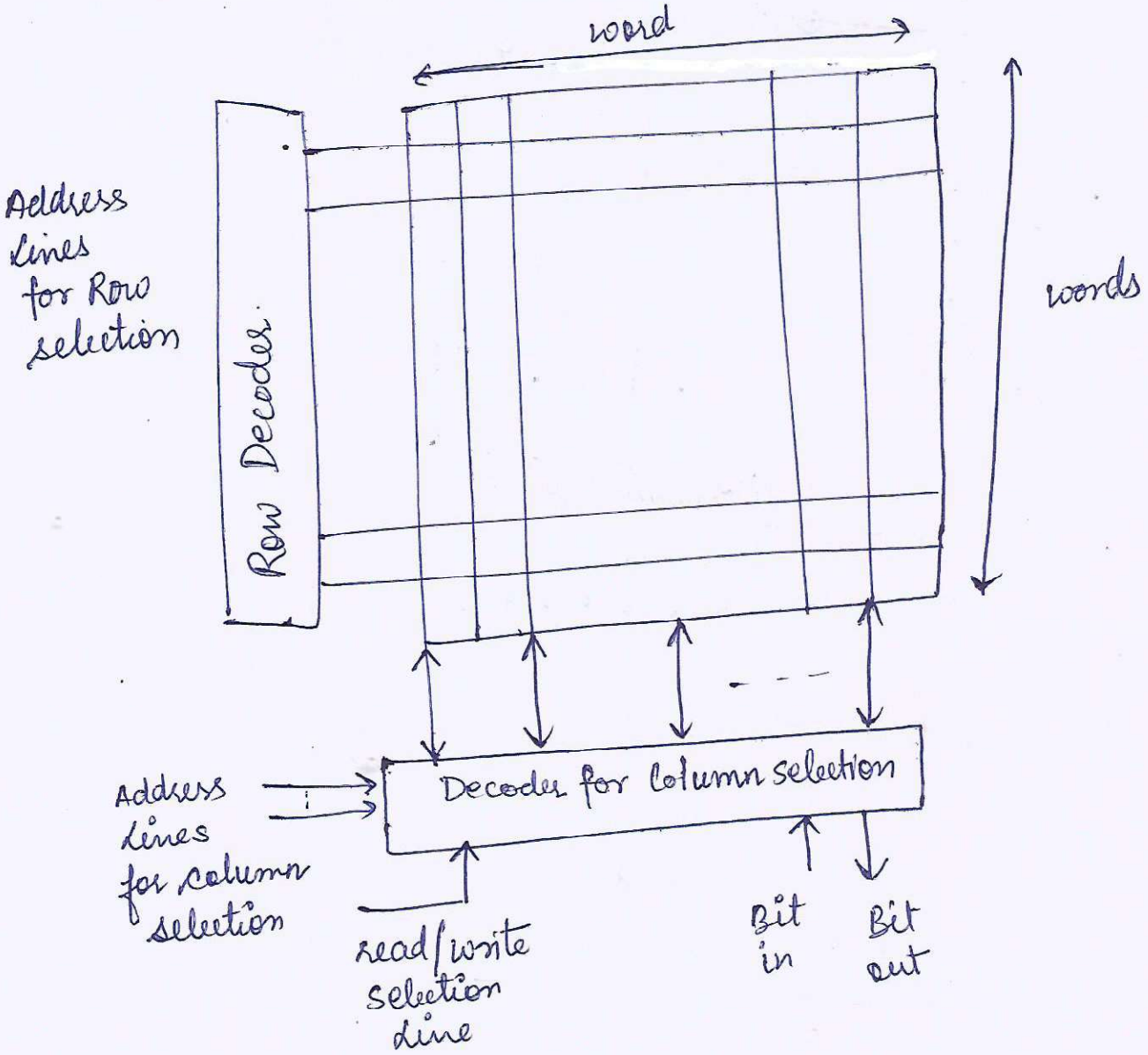
↓

8 number of words

2D Memory organization

$2\frac{1}{2}$ D Memory Organization :

→ This organization, two decoders are used.

→ One decoder to select the row (or words) and the column decoder selects the column.

→ The address from MAR will go in row decoder inputs. The decoders will select the respective cell. Through the bit outline, the data from that location will be read or through bit in line data will be written at that memory location.