03/08/15 SWITCHING THEORY AND LOGIC DESIGN.

Text books :- R.P. Jain
Morris Mano.

Reference book :- Sanjay sharma
P. Raja
Flyord
Salivahan.
Anand kumar (from Library)

04/08/15

INTRODUCTION :-

① **Signal** :- A signal is a fn representing a physical quantity which may be a fn of one or more independent variables like, time, position, distance temperature, force etc and contain information about the nature or behaviour of the same phenomenon.

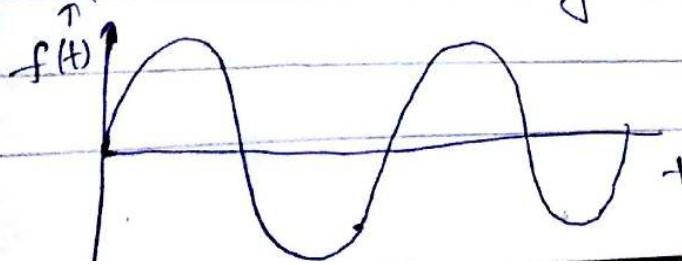
Classification of signals.

There are basically 2 types of signals.

① **Analog signal**

These are the signals which are continous in nature and can have any value in a limited range.

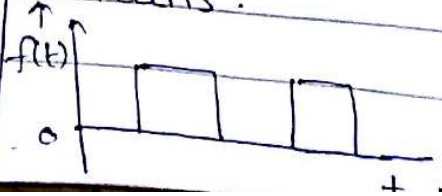
The electronic circuits used to process these signals are known as analog circuits.



② **Digital signals**

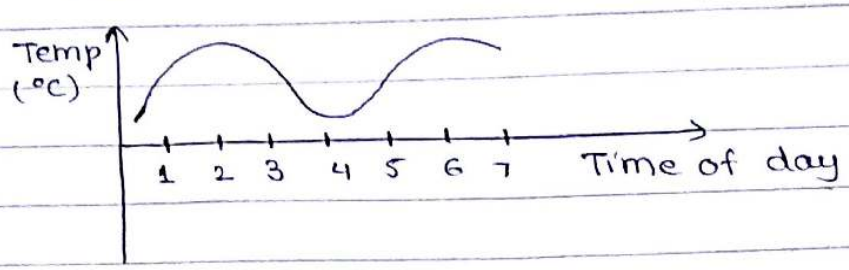
Digital signals are the signals which have only 2 levels namely low (LOW) & high (HIGH). False, 0 (OFF) 1, on, true.

The electronic ckt used to process these signals are known as digital circuits.

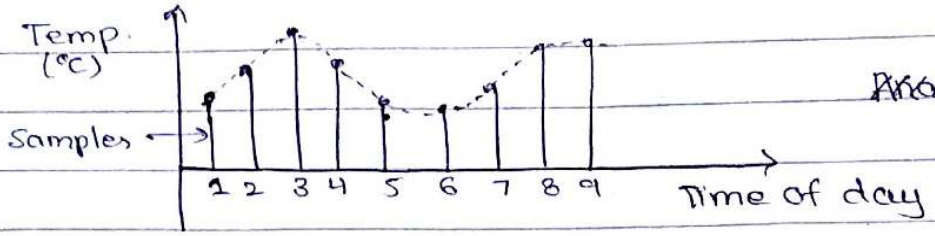


Digital Vmax = 5V, 0Hz.
Analog Vmax = 220, 50Hz.

Variation in temperature.



Analog



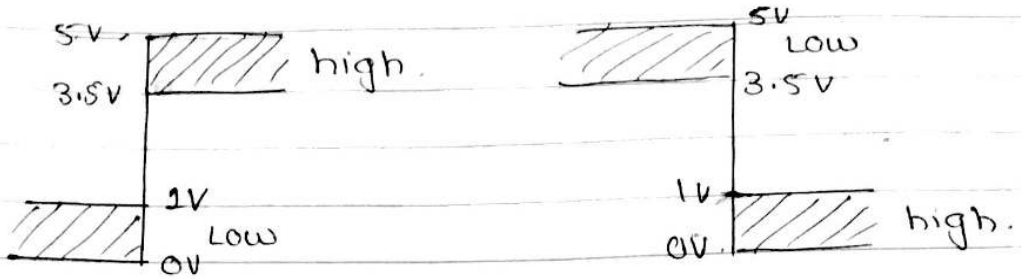
Analog Digital

LOGIC LEVELS :- There are two types of logic levels
1) positive logic level 2) Negative logic level

1) +ve logic level :- If the high voltage level is used to represent 1 and low voltage level is used to represent zero then the system is known as +ve logic level

2) -ve logic level :- If the high voltage level represent zero and low voltage level represent 1 then the system is known as -ve logic level.

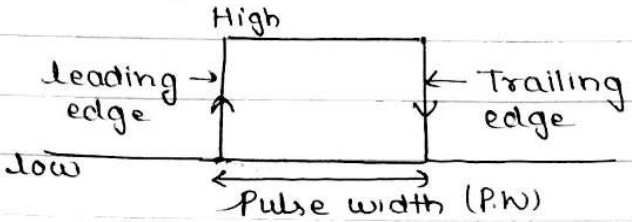
In general any voltage b/wn 0V and 0.8V represents the logic zero and any voltage b/w 2V & 5V represents logic 1. The range b/wn 0.8 V and 2V is known as intermediate range. & If the signal falls b/w this range then the response is not predictable.



+ve LL

-ve LL

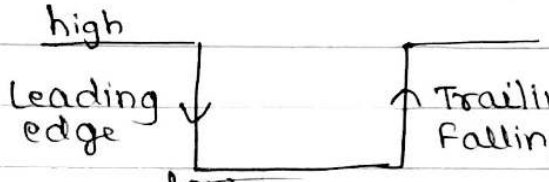
PULSE WAVE FORM :-



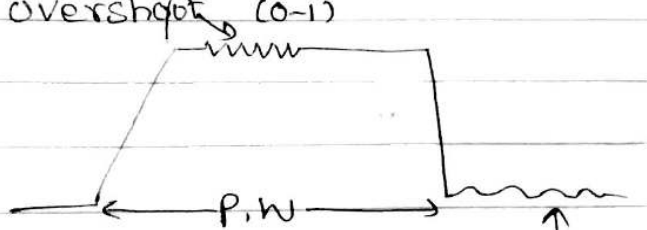
(+ve Pulse)

$t_r \rightarrow 0-1 \rightarrow$ Rise Time

$t_f \rightarrow 1-0 \rightarrow$ Fall time



(-ve pulse)



(Non-ideal pulse)

Undershoot (Removable) (1-0)
(using Rectifier)

05/08/15

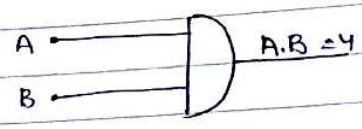
LOGIC GATES

Logic gates are the fundamental building blocks of the digital system. The name logic gate is derived from the ability of such a device to make decisions in the sense that it produce one output level when some combination of input level are present and a different output level when other combination of I/P levels are present.

(Def.)
TRUTH TABLE :- A table which list or shows all the possible combination of I/P variable and corresponding outputs is known as truth table. It shows how the logic circuit's output is ~~form~~ respond to various combination of logic levels at the I/P.

~~AND GATE~~ :- BASIC GATES / AOI LOGIC / ALL GATES

1) AND GATE :- (nothing gate)



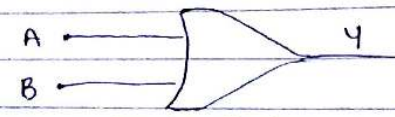
Logic design.

Truth table.

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

$[Y = A \cdot B]$
Logic expression

2) OR GATE :- (any gate)



Logic Design.

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

$[Y = A + B]$

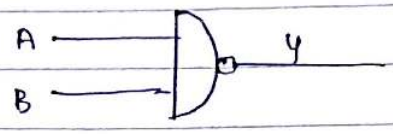
3) NOT GATE :-

A	Y
0	1
1	0

$Y = \bar{A}$

UNIVERSAL GATES :-

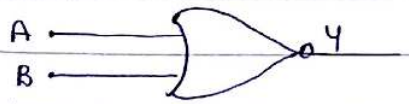
1) NAND GATE (NOT + AND) :-



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

$\overline{A.B} = Y$

2) NOR-GATE (NOT + OR) :-

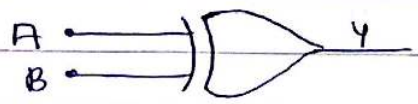


A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

$Y = \overline{A+B}$

EXCLUSIVE GATES :-

1) EX-OR GATE / Anticoincidence gate / Inequality detector



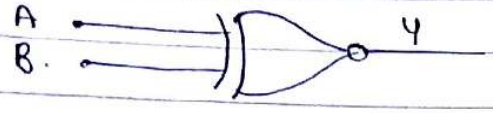
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

modulo sum

$Y = A \oplus B$

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

2) EX-NOR GATE | Coincidence gate | equality detector

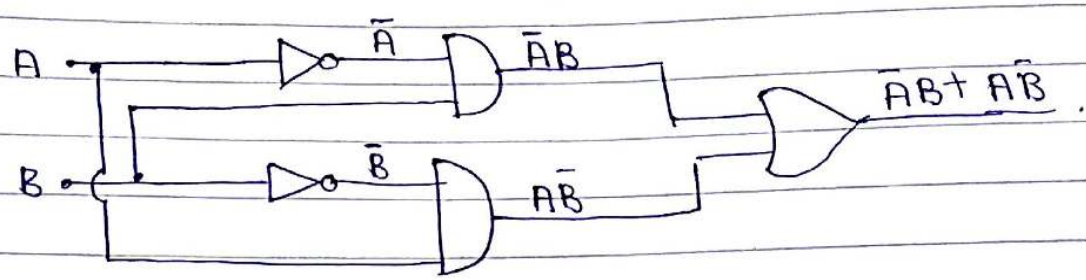


A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

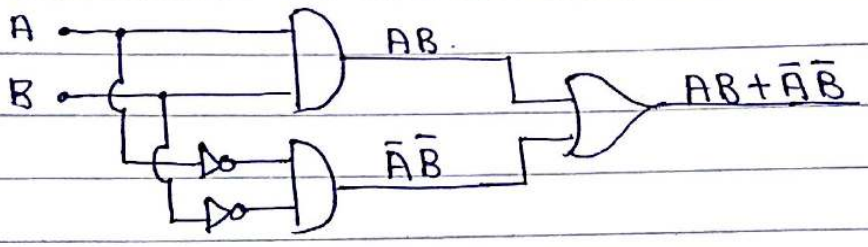
$Y = A \odot B$

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

Q⇒ $\bar{A}B + A\bar{B}$ draw using AOI logic

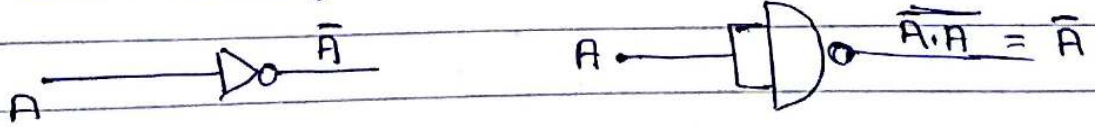


Q⇒ $AB + \bar{A}\bar{B}$

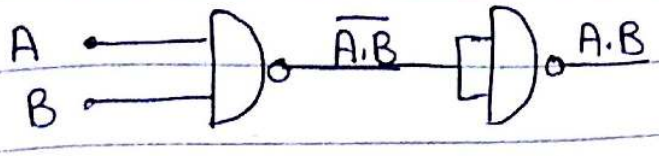


REALIZATIONS:-

1) NOT GATE from NAND

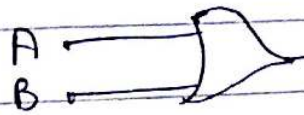


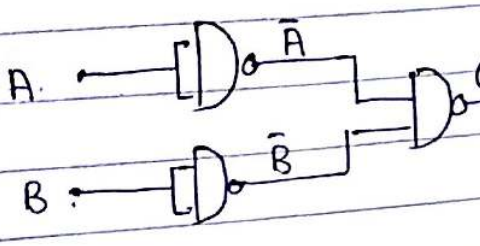
2) AND Using NAND



3) OR GATE using NAND

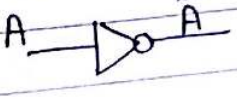
$\overline{\overline{A+B}} = (\overline{\bar{A}.\bar{B}}) = B$

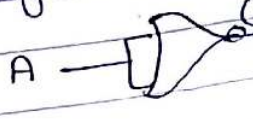




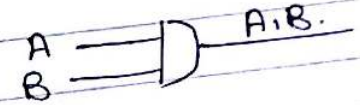
$= \bar{\bar{A}} + \bar{\bar{B}} = A + B.$

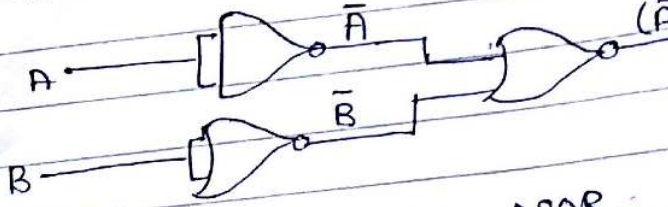
4) NOT Gate using NOR



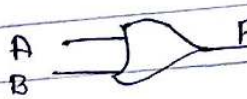


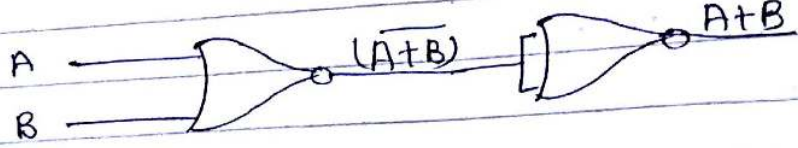
5) AND gate using NOR



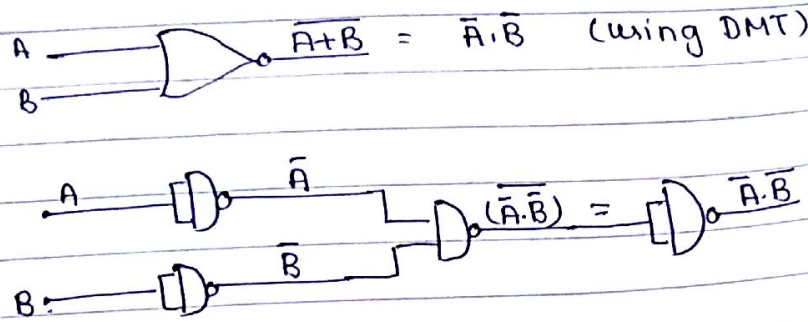


6) OR gate using NOR

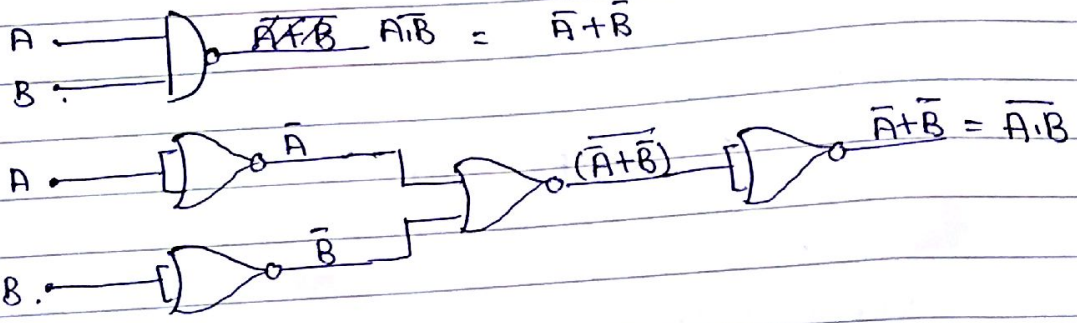




7) NOR gate using NAND.



8) NAND using NOR



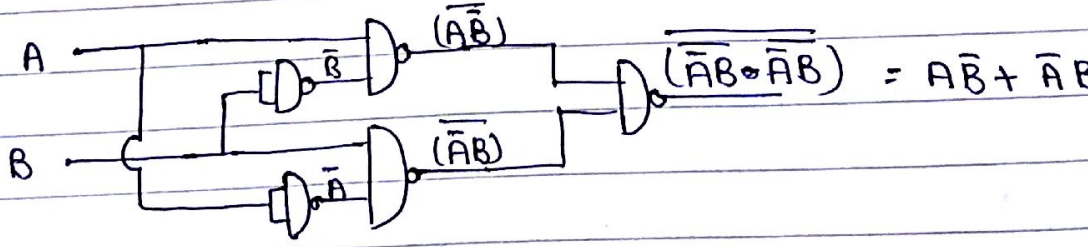
10/08/15

9) EX-OR using NAND GATE:-

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

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

$= \overline{A\bar{B}}.\overline{\bar{A}B}$ (DMT)

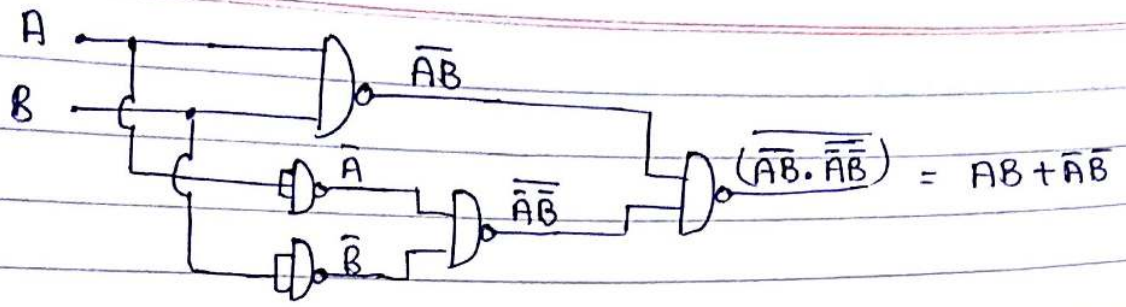


10) EX-NOR using NAND Gate:-

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

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

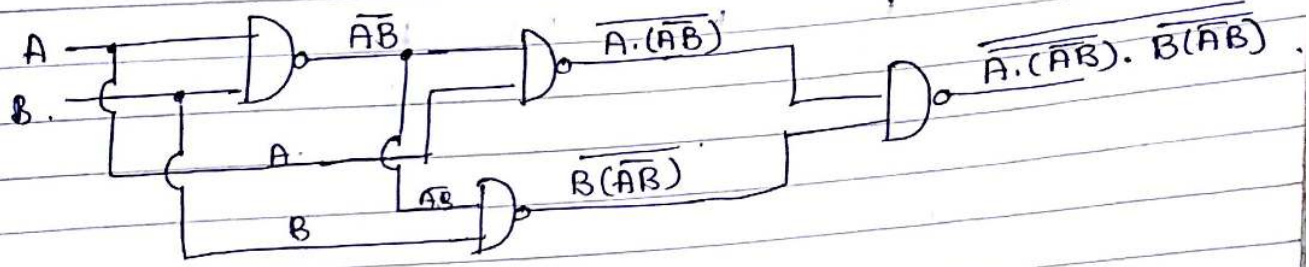
$Y = \overline{(\bar{A}B.\overline{A\bar{B}})}$



Q⇒ Realize Ex-OR gate using 4 NAND gate only.

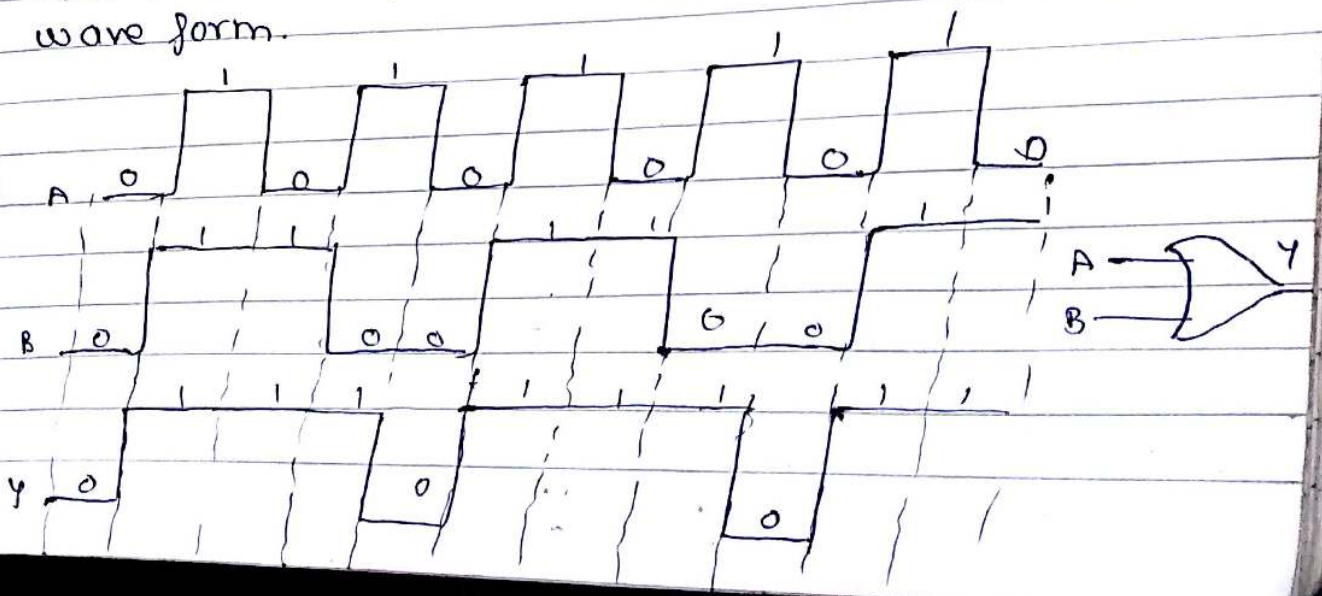
Imp

$$Y = A\bar{B} + \bar{A}B.$$
$$= A\bar{B} + A\bar{A} + \bar{A}B + B.\bar{B}$$
$$= A(\bar{A}+\bar{B}) + B(\bar{A}+B)$$
$$= A(\overline{AB}) + B(\overline{AB}) \quad \text{(DMT)}$$
$$= \overline{A(\overline{AB}) + B(\overline{AB})}$$
$$= \overline{A(\overline{AB})}.\ \overline{B(\overline{AB})} \quad \text{DMT}$$



ENABLE AND INHIBIT CIRCUIT :-

Given the two i/p or waveform represent the o/p wave form.



BOOLEAN ALGEBRA :-

1) Duality :- Duals

$+ \longrightarrow \cdot$ $\qquad$ $\cdot \longrightarrow +$

$0 \longrightarrow 1$ $\qquad$ $1 \longrightarrow 0$

eg :- $A + (B.C) + D = A.(B+C).D$

$1 + 1 = 1$

$\Rightarrow 0.0 = 0$

2) Prove DMT using truth table :-

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

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

a)

A	B	A+B	$\overline{(A+B)}$	A.B	$\bar{A}.\bar{B}$
0	0	0	1	0	1
0	1	1	0	0	0
1	0	1	0	0	0
1	1	1	0	1	0

from the (vi) and (iv) column the DMT is proved.

THEOREMS :-

1a) Consenses / Included factor theorem :-

It states that

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

L.H.S $\Rightarrow$ $AB + \bar{A}C + BC$

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

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

L.H.S → $AB + \bar{A}C + BC(A + \bar{A})$

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

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

$AB + \bar{A}C = R.H.S$

Hence proved

a) $1 + A = 1$

b) $1 + BC$, $1 + \bar{A} = 1$

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

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

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

} AXIOMS

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

L.H.S → $(A+B)(\bar{A}+C)(B+C)$

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

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

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

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

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

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

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

$=$

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

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

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

$= L.H.S$

Proved

2) Transposition theorem :-

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

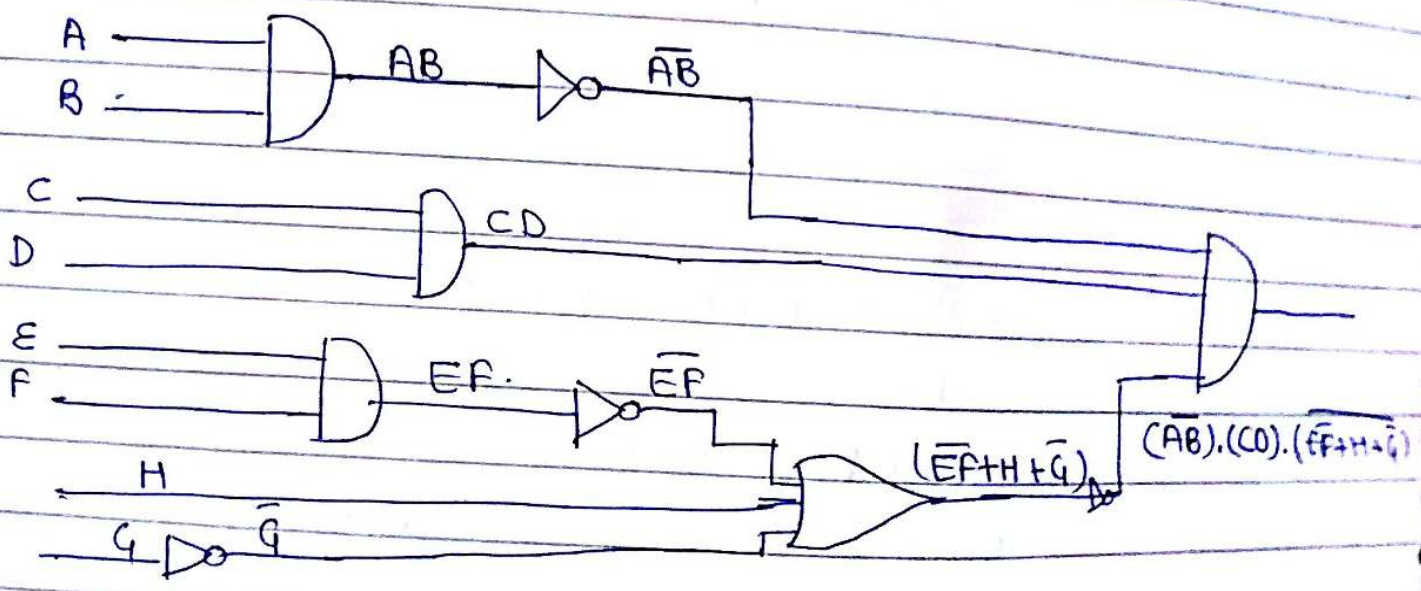
R.H.S = $(A+C)(\bar{A}+B)$

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

$= AB + \bar{A}C + BC$ = L.H.S

hence proved

Convert logic exp to Boolean exp :-



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

$= (\bar{AB})(CD).(\bar{\bar{E}\bar{F}}.\bar{H}.\bar{G})$

$= (\bar{AB})(CD).(EF.\bar{H}.\bar{G})$

$= (\bar{A}+\bar{B})(CD).(EF.\bar{H}.\bar{G})$

11/08/15

BINARY CODES :-

1) Binary coded Decimal (BCD code) :-

In BCD code each decimal ~~digit~~ digit is represented by a 4-bit binary no.. The positional weights associated to binary bits in BCD code are 8-4-2-1 which is equal to $2^3 - 2^2 - 2^1 - 2^0$

Decimal	Binary	BCD
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10 (A)	1010	00010000
11 (B)	1011	00010001
12 (C)	1100	00010010
13 (D)	1101	00010011
14 (E)	1110	00010100
15 (F)	1111	00010101

BCD 8 bi
-ry are sa

both
diff

2) Excess-3 / sey complementary code :-

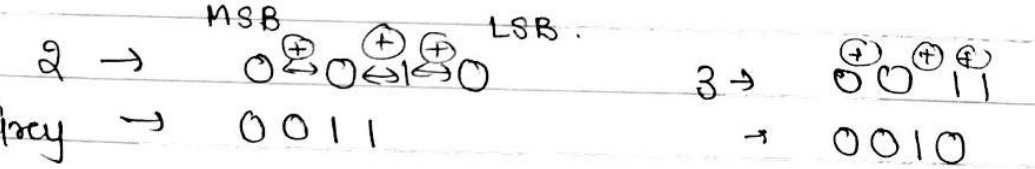
It is the modified form of BCD no. the excess-3 code can be derived from the natural BCD c by adding 3 (0011) to each code

Decimal	BCD	excess-3.
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

$174 \Rightarrow$ 001101010111

3) Gray-code :-

It is a special case of unit-distance code. In unit distance code bit patterns of two consecutive no. differs only in 1 bit position. These codes are also known as cyclic code or mirror reflected code.



Decimal	BCD Binary.	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101

7	0111	0100 ----
8	1000	to 1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Grey ⇒ 1 0 1 0 1

Binary ⇒ 1 1 0 0 1

14/08/15

ERROR CORRECTING AND DETECTING CODES.

10110 → 10110.

Transmitter reciever

10111 —even p. noise→ 10110

PARITY method (No. of 1's odd).

Even parity	Even.	Odd parity. odd.
10110	1	0
11100	1	0
11011	0	1
10011	1	0

NUMBER SYSTEM :-

- Binary (2) (0,1)
- Decimal (10) (0-9)
- Octal (8) (0-7)
- Hexadecimal (16) (0-9, A-f)

Conversions :-

Binary $(11011)_2$ ⇒ a) Decimal

1 1 0 1 1

- $1 \times 2^0 = 1$
- $1 \times 2^1 = 2$
- $0 \times 2^2 = 0$
- $1 \times 2^3 = 8$
- $1 \times 2^4 = 16$

$= (27)_{10}$

b) Octal ⇒ 011 011 (3 3) $= (33)_8$.

c) Hexadecimal :- 0001 1011 (1 B) ⇒ $(1B)_{16}$

eg :- (1) 1 1 1 1 0 . 1 1 0 0

$2^2 \times 1 + 2^3 + 2^5 + 2^6 + 2^7 + 2^8$

$= 4 + 8 + 32 + 64 + 128 + 256. = (492)_{10}$

b) Octal ⇒ 111 101 100 (7 5 4) $= (754)_8$.

) Hexa = 0001 1110 1100 (1 D C) $= (1DC)_{16}$

---

ecimal - $(21)_{10}$ a) Binary :-

2	21	
2	10	1
2	5	0
2	2	1
	1	0

$= (10101)_2$

b) Octal

8	21	
	2	5 ↑

$(25)_8$

c) Hexa

16	21	5 ↑
	1	

$(15)_{16}$.

eg:- $(98)_{10} \Rightarrow$

2	98	
2	49	0
2	24	1
2	12	0
2	6	0
2	3	0
2	1	1
	0	

$(01100010)_2$

8	98	
8	12	2
	1	4

$(142)_8$.

16	98
	6 2 ↑

$(62)_{16}$

3) Octal $(37)_8$.

a) Binary $(37)_8$ — 3 → 011, 7 → 111 $\Rightarrow (011111)_2$

b) Decimal $\Rightarrow$ 3 7
↳ $7 \times 8^0 = 7$
→ $3 \times 8^1 = 24$
$= (31)_{10}$

c) Hexa :- 0001 1111 $= (1F)_{16}$
1 F.

Oct → Bin → Hex

eg:- $(46)_8$.

a) $\Rightarrow$ Binary $\Rightarrow (100110)_2$

b) 4 6
↳ $6 \times 8^0 = 6$
→ $4 \times 8^1 = 32$
$(38)_{10}$.

c) 0010 0110 $= (26)_{16}$
2 6

1-0 = 1 | 0
1-1 = 0 | 0

4) Hexa $(2F)_{16}$.

a) Binary ⇒ 00101111 ⇒ $(101111)_2$

b) Decimal ⇒ 2 F
↳ $F \times 16^0 = 15$
↳ $2 \times 16^1 = 32$
⇒ $(47)_{10}$

c) Octal 00 101 111 ⇒ $(57)_8$.
(5 7)

eg :- $(1B)_{16}$

a) ⇒ 00011011 ⇒ $(11011)_2$

b) ⇒ 1 B
↳ $B \times 16^0 = 11$
↳ $1 \times 16^1 = 16$
$= (27)_{10}$

c) Octal ⇒ 011 011 $= (33)_8$.
(3 3)

x x x x x x

Addition

Binary	Decimal	Octal	Hexa
1101	23	47	2F
+1011	+13	+25	+2F
$(11000)_2$	$(36)_{10}$	6 12	4 30
		(14)$_8$	5 14
		8) 12 (1	$(5E)_{16}$
		8	16) 30 (1
		4 → R	16
			14

Subtraction

1000	43	3	2E (16+14)
-0001	-24	43 (8+3)	-1F (15)
$(1011)_2$	$(19)_{10}$	-24	$(0F)_{16}$
		$(17)_8$	

COMPLEMENTS :-

eg :- $(10111)_2$

1's ⇒ (00100) $(0 \to 1, 1 \to 0)$

+1

2's ⇒ 00101

9's ⇒ 99 − 71 ⇒ 28

10's ⇒ 28 + 1 = 29

17/08/15.

KARNAUGH MAP (K-MAP)

K-Map is a systematic method of simplifying the boolean expression. K-map is a chart or a graph composed of arrangement of adjacent cells in which each cell repres-nting a particular combination of variables either in sum or product form. for a n-variable f^n there are 2^n possible combinations where n → no. of variables for two variable f^n there are $2^2 = 4$ cells.

Ily 3 variable = $2^3 = 8$ cells.

Boolean expression can be exprssed in a standard/canonical/expanding sum of product (SOP) form or product of sum (POS) form

Standard SOP

It is one in which a no. of product terms, each one of which contains all the variables of f^n either in complemented or non-complemented form are summed together.

Standard POS

A standard POS form is one which a no. of sum terms each one of which contains all the variables of a f^n either in complemented or non complemented form are multiplied together

→ Every term in SOP form is known as min term.

→ In this
Non-comp variable = 1
comp-variable = 0

eg:- $A\bar{B}' + B\bar{C} + C\bar{A}$
$10 + 10 + 10$
Σm_2

→ $\Sigma m_0 + m_2 + m_3$

→ Decimal equivalent is SOP form is represented by lower case 'm' like $m_0, m_1, m_2, \ldots$

→ Each term in POS form is known as max term

→ In this
Non-comp variable = 0
Comp variable = 1.

eg:- $(A+\bar{B}).(B+C).C\bar{C}+A)$

→ The decimal equivalent in POS form is represented by Upper case 'M' i.e $M_0, M_1, M_2, \ldots$

PROCEDURE TO EXPAND THE (FUNCTION) EXPRESSION IN SOP FORM TO STANDARD SOP FORM:-

Step ① Write down all the terms given in expression
② If one or more variable are missing in any term, then expand that term by multiplying it with the addition of each one of the missing variable and its complement
③ Drop out the redundant terms.

Q⇒ Expand $\bar{A} + \bar{B}$ to min term and maxterm

Soln⇒ $\bar{A} + \bar{B}$
$= \bar{A}.(B+\bar{B}) + \bar{B}.(A+\bar{A})$
$= \bar{A}B + \bar{A}\bar{B} + A\bar{B} + \bar{A}\bar{B}$
$= \underset{01}{\bar{A}B} + \underset{00}{A\bar{B}} + \underset{10}{\bar{A}\bar{B}}$
$(m_1 + m_0 + m_2)$

$\Sigma m(1,0,2) \Rightarrow \Sigma m(0,1,2)$

interm

Max term ΣM_3. ΠM_3.

Procedure to expand the expression in POS to standard PS

① write down all the term

② If one or more variable are missing in any Sum ter then expand that term by adding the product of each of the missing term and its complement.

③ Drop out the redundant term.

Q⇒ Expand $A(\bar{B}+A)B$ to max term & min term.

Sol⇒ $A(A+\bar{B})*B$.

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

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

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

0 0 0 1 1 0

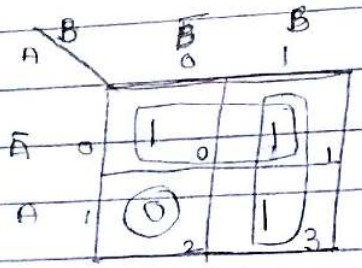
M_0 M_1 M_2.

Max → $\Pi M(0,1,2)$

Min → Σm_3.

K-MAP ⇒ 2-variable

Q⇒ Reduce the expression $\Sigma m(0,1,3)$ using mapping



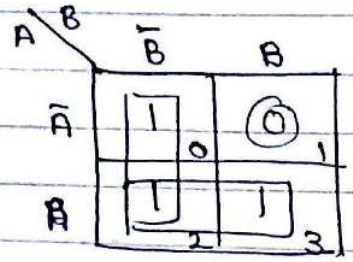
$\Pi M(2) \Rightarrow A\bar{B}$

0,1 → $\bar{A}\bar{B}+\bar{A}B$

1,3 → $\bar{A}B + AB$

$\Sigma m(0,1,3) = \bar{A}+B$

Q ⇒ $\Sigma m(0,2,3)$

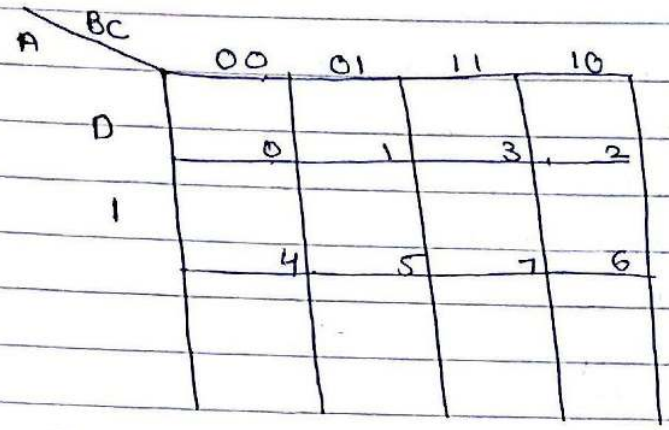


$\Sigma m(0,2,3) = A + \bar{B}$

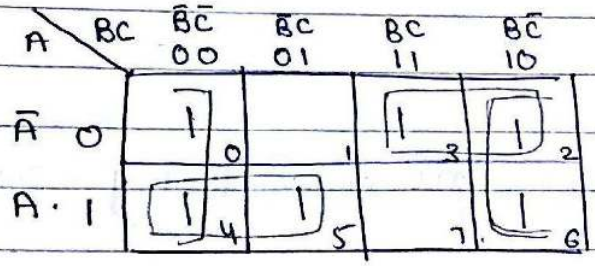
$\pi M(1) = A\bar{B}$

Boolean

2) Three Variable mapping $2^3 = 8$ cells.



Q ⇒ Reduce the expression $\Sigma m(0,2,3,4,5,6)$ using mapping and implement the result in AOI logic

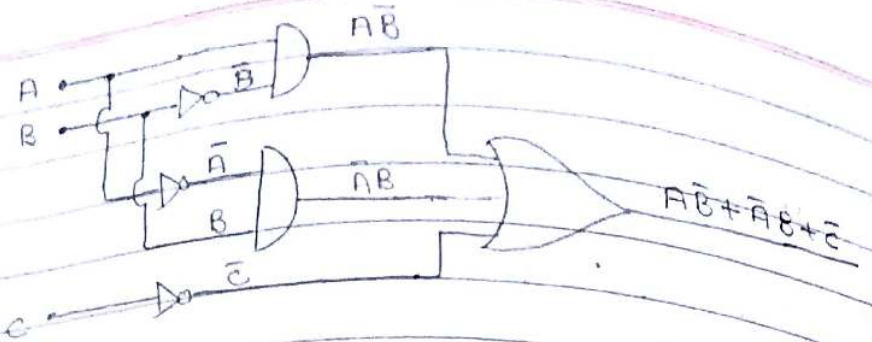


quad (0, 4, 2, 6) reduced to $\bar{C}$

pair (4, 5) " " $A\bar{B}$

pair (3, 2) " " $\bar{A}B$.

$\Sigma m(0,2,3,4,5,6) = \bar{C} + A\bar{B} + \bar{A}B.$



3) 4-variable mapping $\Rightarrow 2^4 = 16$ cells.

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

Q) Reduce $\Sigma m(2,3,6,7,8,10,11,13,14)$

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	1 3	1 2
$\bar{A}B$	4	5	1 7	1 6
AB	12	1 13	15	1 14
$A\bar{B}$	1 8	9	1 11	1 10

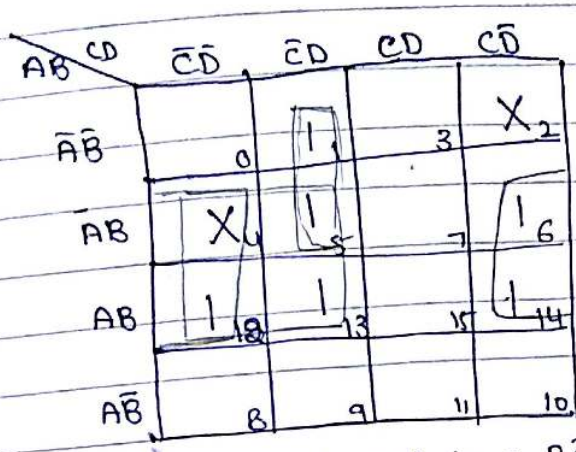
$\bar{A}C + C\bar{D} + \bar{B}C + A\bar{B}\bar{D} + AB\bar{C}D$

DONT CARE COMBINATIONS :-

The combination for which the values of expression are not specify are called dont care combinations and such expression stand incompletely specify. The output is don't care for these invalid combinations.

he don't care terms are denoted by (d, X, ϕ)

)⇉ Reduce the expression $\Sigma m(1,5,6,12,13,14) + d(2,4)$



quad (4,5,13,12) reduced to = $B\bar{C}$
quad (4,12,6,14) " " = $B\bar{D}$
pair (1,5) " " = $\bar{A}\bar{C}D$.

$\Sigma m(1,5,6,12,13,14) = B\bar{C} + B\bar{D} + \bar{A}\bar{C}D$

Tabular

Quine Mc Clusky (Q-M) method :-

$Y = \Sigma m(0,1,3,7,8,9,11,15)$

0000 0001 0011 0111 1000 1001 1011 1111

Group	Min term	Variables (ABCD)
0	0	0000 ✓
1	1	0001 ✓
	8	1000 ✓
2	3	0011 ✓
	9	1001 ✓
3	7	0111 ✓
	11	1011 ✓
4	15	1111 ✓

Group	Minterm	Variable
0	0,1	000_
	0,8	_000
1	1,3	00_1
	1,9	_001
	8,9	100_
2	3,7	0_11
	3,11	_011
	9,11	10_1
3	7,15	_111
	11,15	1_11

Prime Implicant table :-

Group	Min-term	Variable (A B C D)	
0	0,1,8,9	_00_	$\bar{B}\bar{C}$
	0,8,1,9		
1	1,3,9,11	_0_1	$\bar{B}D$
	1,9,3,11	_0_1	
2	3,7,11,15	__11	CD
3	3,11,7,15	__11	

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

Essential PI table :-

PI terms	Min-terms	Decimal no's (0,1,3,7,8,9,11,15)							
		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)

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

Q ⇒ Simplify the logic fn using Q-M method

$F(A,B,C,D) = \Sigma m(1,3,7,11,15) + d(0,2,5)$

0001 0011 0111 1011 1111 0000 0010 0101

Group	Min-term	Variables ABCD
0	0*	0000
1	1	0001
	2*	0010
2	3	0011
	5*	0101
3	7	0111
	11*	1011
4	15	1111

Group	Min-term	Variables
0*	0*,1	000–
	0,2	00–0
1	1,3	00–1
	1,5*	0–01
2	2*,3	001–
	3*,7	
2	3,7	0–11
	3,11	–011
3	7,15	–111
	11,15	1–11

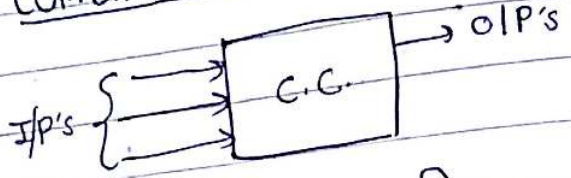
Group	minterm	variable	
0	0,2,1,3	00––	$\bar{A}\bar{B}$
1	1,5*,3,7	0––1	$\bar{A}D$
2	3,7,11,15	––11	CD
3	3,11,7,15	––11	

Group	min-term	Decimal
		1 3 7 11 15 0 2 5
$\bar{A}B$	$0^*, 2^*, 1, 3$	X X
$\bar{A}D$	$1, 5^*, 3, 7$	X X X
$C\bar{D}$	$3, 7, 11, 15$	X X (X) (X)

EPI term ⇒ CD

21/08/15

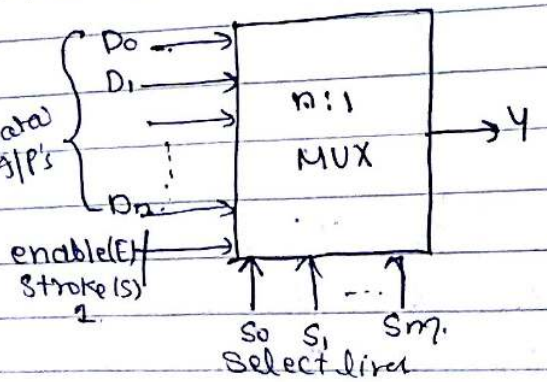
COMBINATIONAL CIRCUITS :-



- → Adders (H+F)
- → Subtractor (H+F)
- → Multiplexers (MUX)
- → Demultiplexers (DMUX)
- → Encoders.
- → Decoders.

} Combinational circuits.

① MULTIPLEXER (MUX) / Data Selectors :-



$2^m \rightarrow$ Select lines.
$\Rightarrow m$
$n \rightarrow$ I/P's.

A multiplexer of data selector is a logic ckt that accept several data I/P's and allow only only one of that at a time to go through the O/P.

The path of the desired data i/p to the o/p is controlled by select lines. To select n-inputs we require m select lines such that $2^m = n$

$2^1 = 2 \rightarrow$ I/P's. ($\rightarrow$ select line)

$2^2 = 4$

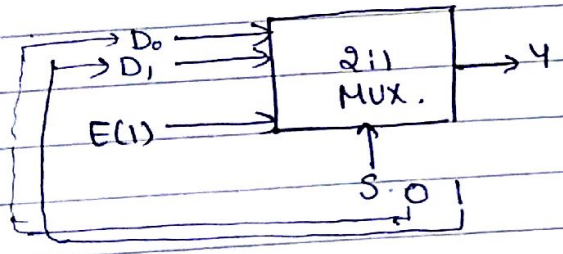
$2^3 = 8$

$2^4 = 16$

1) 2:1 MUX
2) 4:1 MUX
3) 8:1 MUX
4) 16:1 MUX.

} Types of MUX.

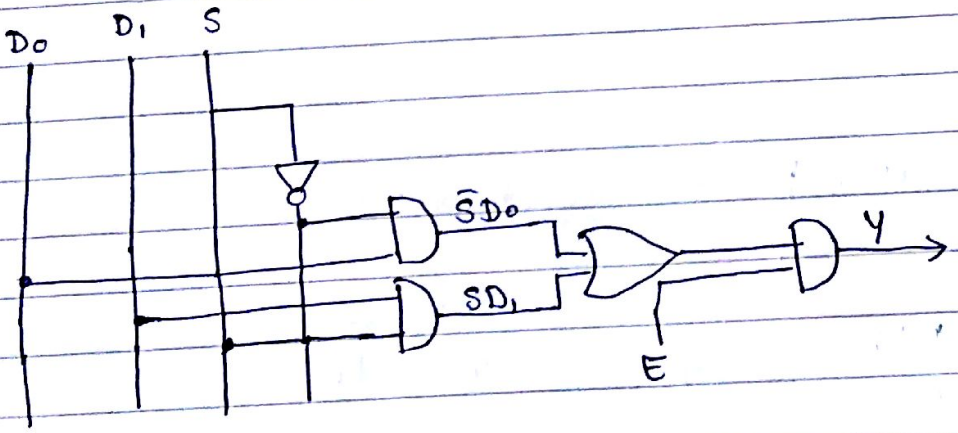
1) 2:1 MUX :-



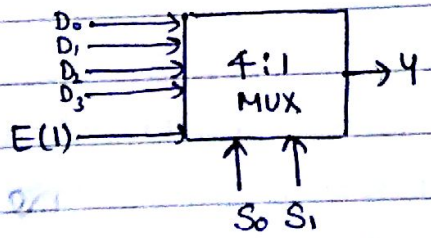
E	S	O/P (Y)
0	X	X
1	0	D_0
1	1	D_1

expression :- $Y = E\bar{S}D_0 + ESD_1$

$Y = E(\bar{S}D_0 + SD_1)$

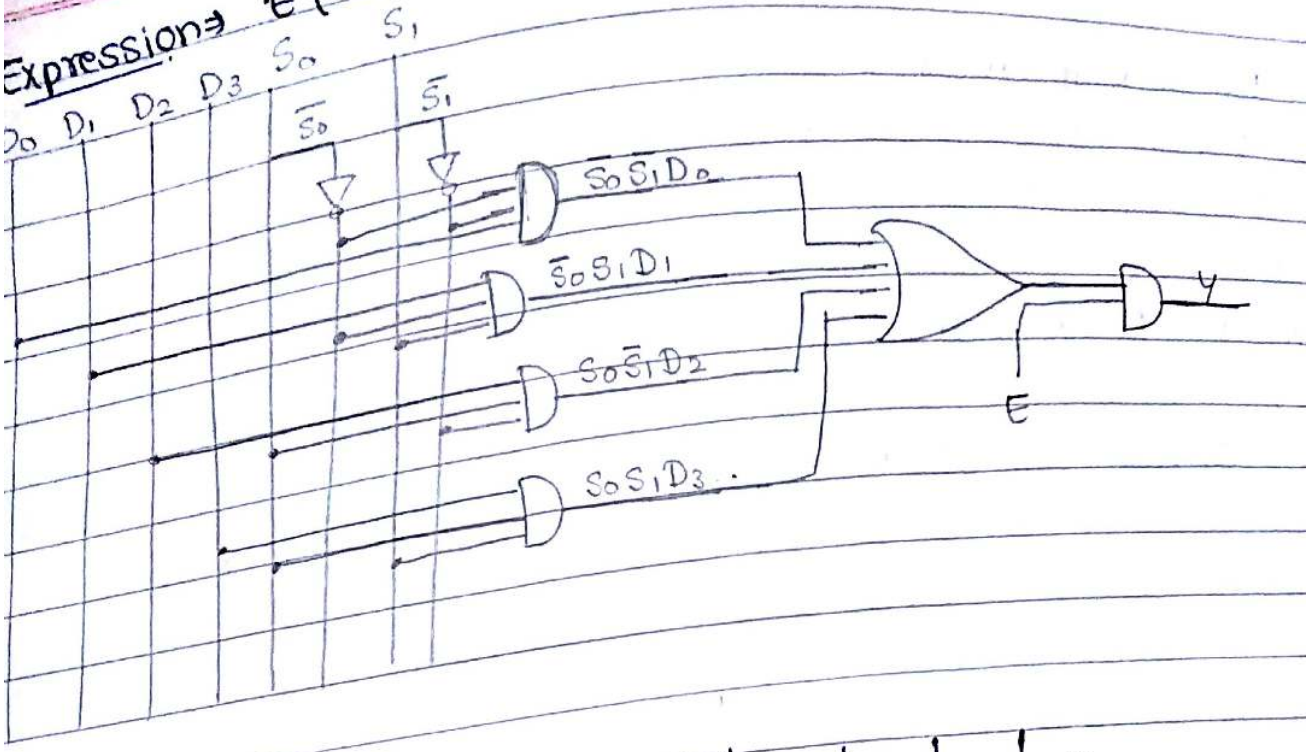


2) 4:1 :-

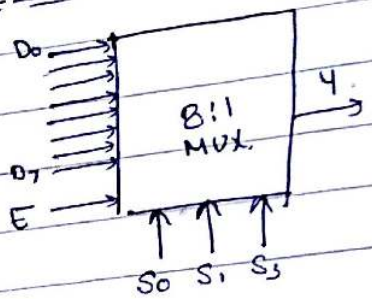


E	S_0	S_1	(Y) O/P
0	X	X	X
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3.

Expression $= E(\bar{S}_0\bar{S}_1D_0 + \bar{S}_0S_1D_1 + S_0\bar{S}_1D_2 + S_0S_1D_3)$.

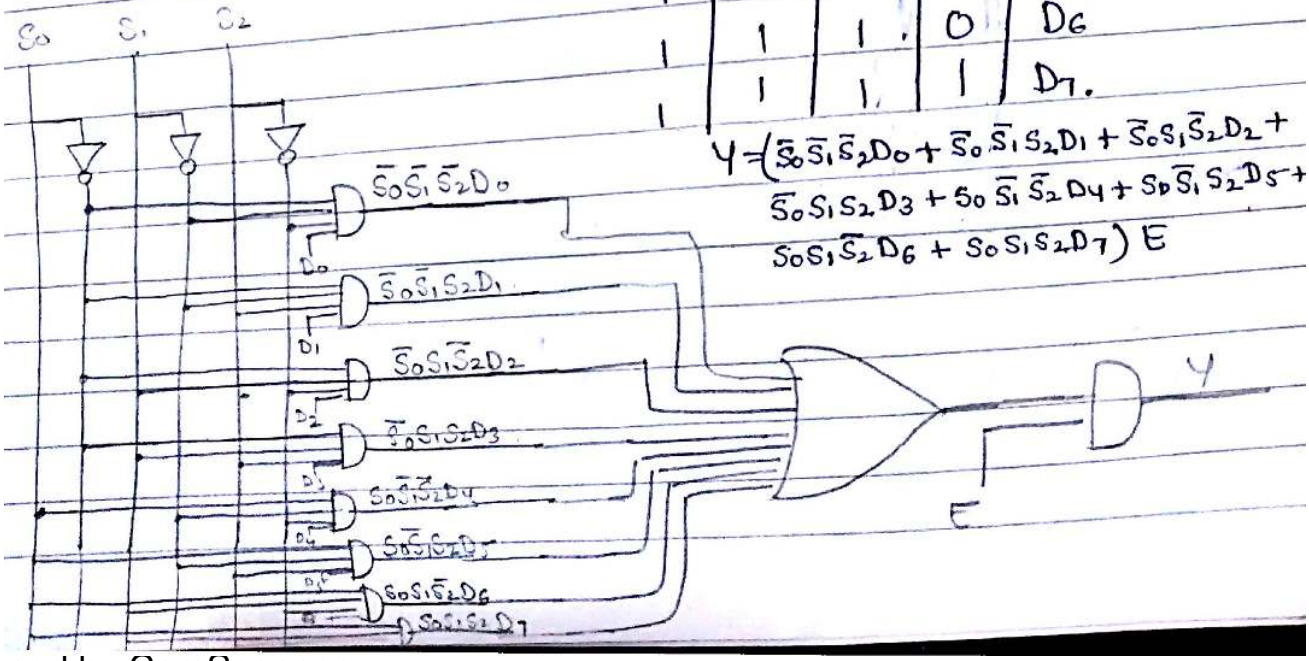


8:1 MUX :-



E	S0	S1	S2	Y
0	X	X	X	X
1	0	0	0	D0
1	0	0	1	D1
1	0	1	0	D2
1	0	1	1	D3
1	1	0	0	D4
1	1	0	1	D5
1	1	1	0	D6
1	1	1	1	D7

$Y = (\bar{S}_0\bar{S}_1\bar{S}_2D_0 + \bar{S}_0\bar{S}_1S_2D_1 + \bar{S}_0S_1\bar{S}_2D_2 + \bar{S}_0S_1S_2D_3 + S_0\bar{S}_1\bar{S}_2D_4 + S_0\bar{S}_1S_2D_5 + S_0S_1\bar{S}_2D_6 + S_0S_1S_2D_7)E$

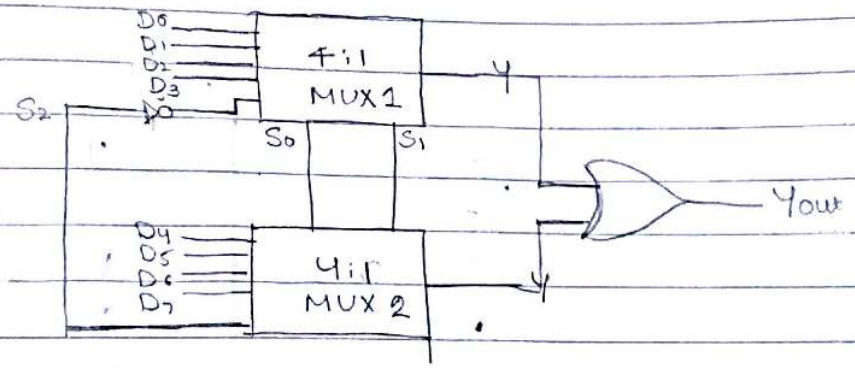


Designing of multiplexers (MUX)

a) By multiplexer tree
b) By combinational ckts
c) By Design tables.

a) MUX TREE:-

Q→ Obtain a 8:1 MUX using 2 (4:1) mux.

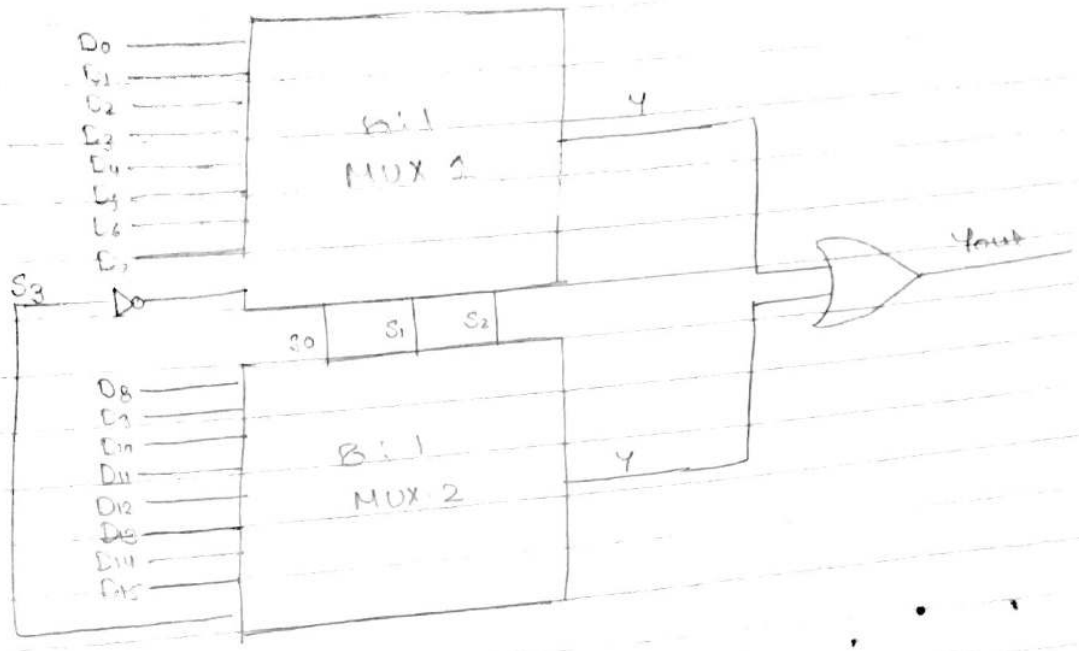


Select lines.			O/P. Y
S_2/E	S_1	S_0	
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

MUX 1 enabled.

Mux-2 enabled.

Q⇒ Obtain a 16:1 mux using 2 (8:1) mux.

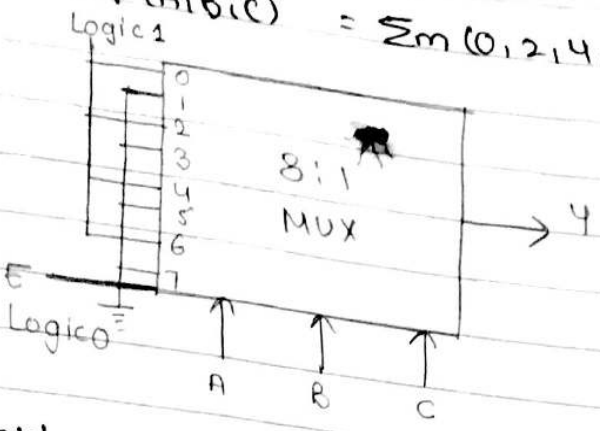


Select lines				O/P (Y)	
$S_3/\bar{E}$	S_2	S_1	S_0		
0	0	0	0	D_0	MUX-1 enabled
0	0	0	1	D_1	
0	0	1	0	D_2	
0	0	1	1	D_3	
0	1	0	0	D_4	
0	1	0	1	D_5	
0	1	1	0	D_6	
0	1	1	1	D_7	
1	0	0	0	D_8	MUX-2 enabled
1	0	0	1	D_9	
1	0	1	0	D_{10}	
1	0	1	1	D_{11}	
1	1	0	0	D_{12}	
1	1	0	1	D_{13}	
1	1	1	0	D_{14}	
1	1	1	1	D_{15}	

2) USING COMBINATIONAL CIRCUIT:-

Q→ Implement the following expression using a suitable mux.

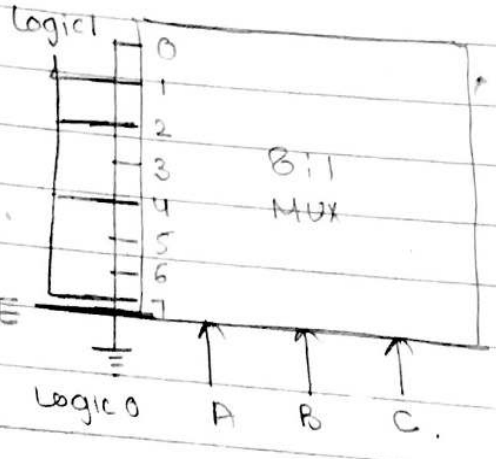
$F(A,B,C) = \Sigma m(0,2,4,6)$



Q→ Implement the following expression using a suitable MUX

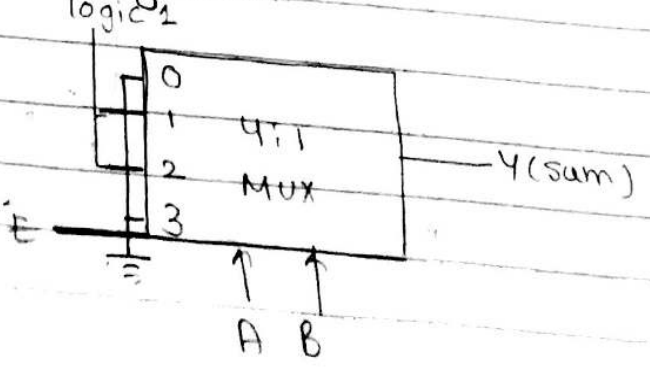
$F = A \oplus B \oplus C$

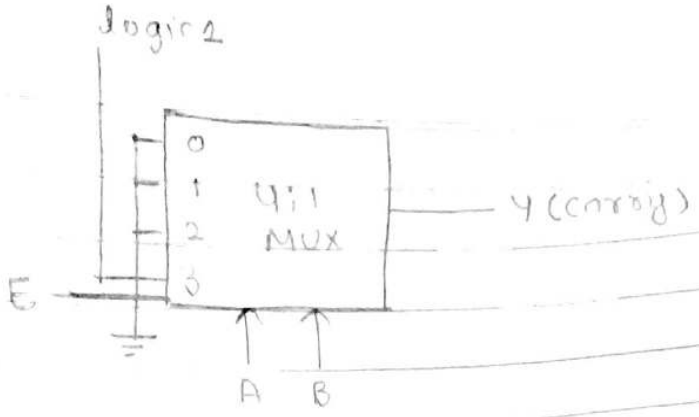
A	B	C	$f = A \oplus B \oplus C$	
0	0	0	0	0
0	0	1	1	1
0	1	0	1	2
0	1	1	0	3
1	0	0	1	4
1	0	1	0	5
1	1	0	0	6
1	1	1	1	7



→ Implement half adder using MUX.

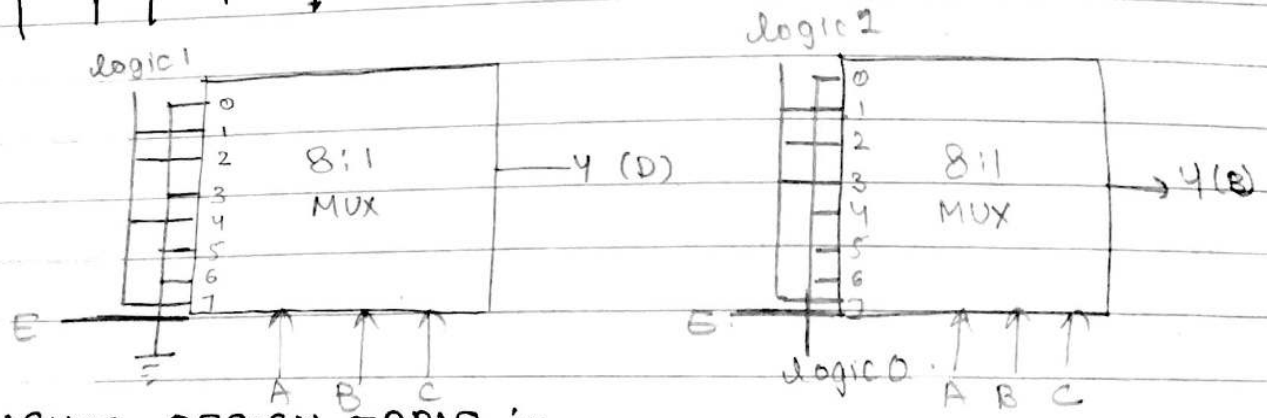
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1





Q) Implement full subtractor using suitable MUX

A	B	C	D	B	
0	0	0	0	0	0
0	0	1	1	1	1
0	1	0	1	1	2
0	1	1	0	1	3
1	0	0	1	0	4
1	0	1	0	0	5
1	1	0	0	0	6
1	1	1	1	1	7

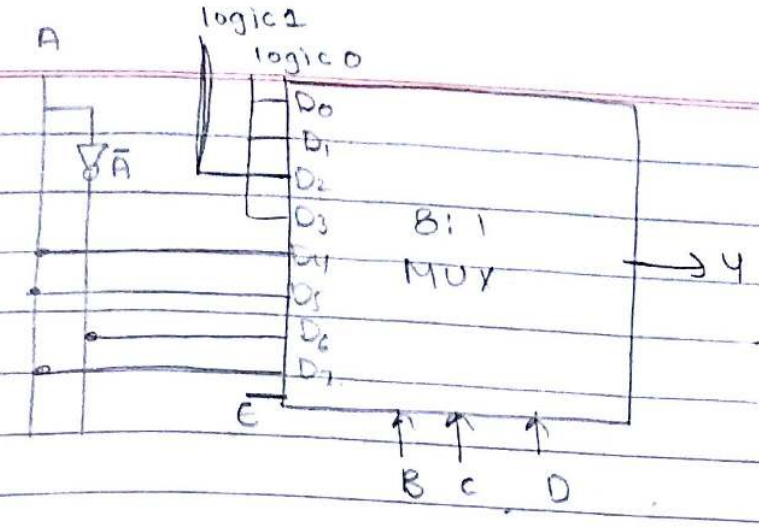


3) USING DESIGN TABLE :-

Q) Implement the following fn using 8:1 MUX.

$F(A,B,C,D) = \Sigma m(2,4,5,7,10,14)$

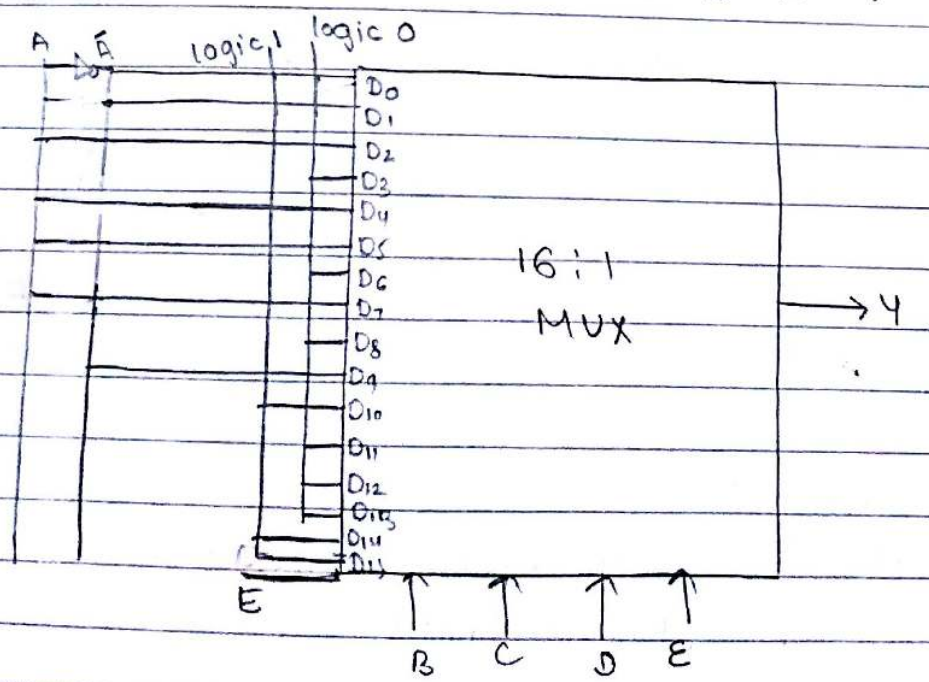
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
A	0	1	(2)	3	(4)	(5)	6	(7)
↑	8	9	(10)	11	12	13	(14)	15
	logic 0	logic 0	logic 1	0	A	A	$\bar{A}$	A.



Q→ Implement the following fn using 16:1 mux.

$f(A,B,C,D,E) = \Sigma m(2,4,5,7,10,14,15,16,17,25,26,30,31)$

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}
A	0	1	(2)	3	(4)	(5)	6	(7)	8	9	(10)	11	12	13	(14)	(15)
$\bar{A}$	(16)	(17)	18	19	20	21	22	23	24	(25)	(26)	27	28	29	(30)	(31)
	$\bar{A}$	$\bar{A}$	A	0	A	A	0	A	0	$\bar{A}$	1	0	0	0	1	1

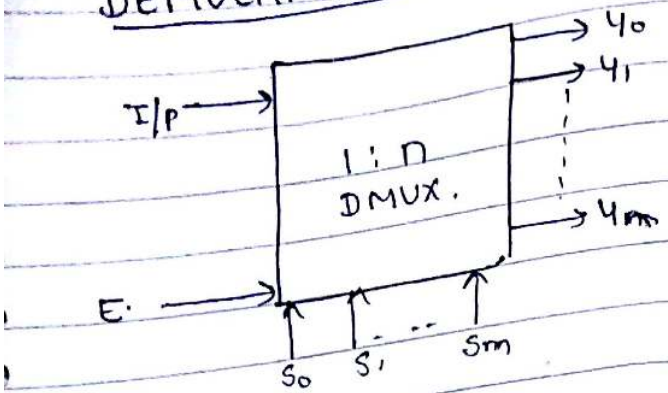


advantages of MUX:-

1) Simplification of logic expression is not required.
2) It minimises the IC package count
3) Logic design is simplified.

3/5/08/15

DEMULTIPLEXERS :-

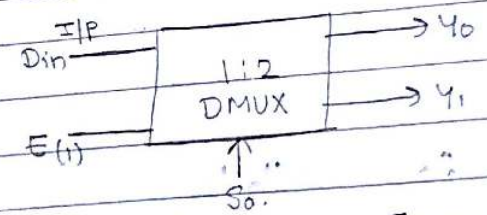


It has 1 I/P, n-outputs & m select lines. A DMUX performs the reverse operation of a MUX i.e. it recieves 1 I/P and distributes that I/P over several O/P lines. at a time only 1 O/P line is selected, by the select lines and the I/P is transmitted to the select O/P line.

2^m → select lines. = n O/P's

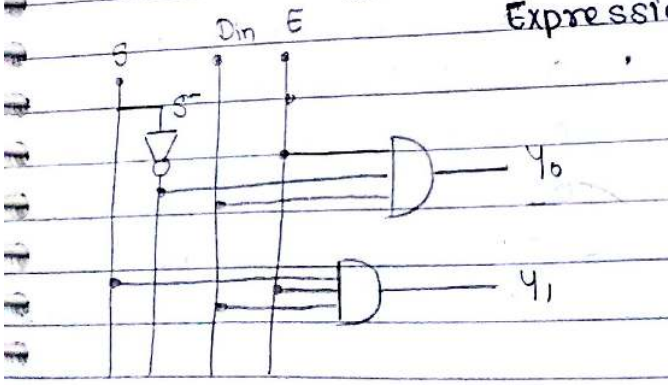
a) 1:2 DMUX
b) 1:4 DMUX
c) 1:8 DMUX
d) 1:16 DMUX.
e) 1:32 DMUX.

a) 1:2 DMUX :-

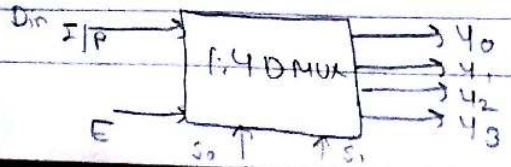


E	S	O/P Y0	O/P Y1
0	X	X	X
1	0	Din	0
1	1	0	Din

Expression ⇒ $E\bar{S}Din + ESDin$.

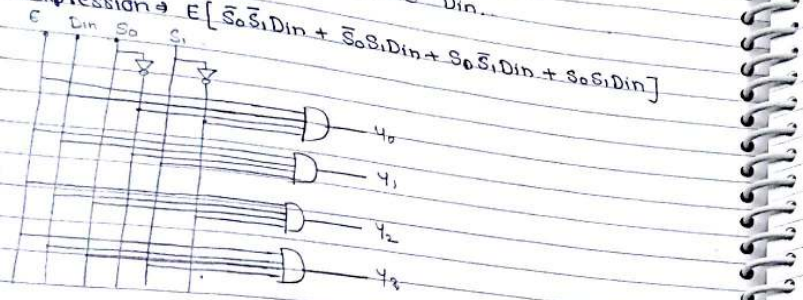


b) 1:4 DMUX :-

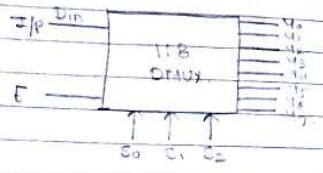


E	S.L S_0	S_1	O/P Y_0	Y_1	Y_2	Y_3
0	x	x	x	x	x	x
1	0	0	Din	0	0	0
1	0	1	0	Din	0	0
1	1	0	0	0	Din	0
1	1	1	0	0	0	Din

Expression $E[\bar{S}_0\bar{S}_1Din + \bar{S}_0S_1Din + S_0\bar{S}_1Din + S_0S_1Din]$

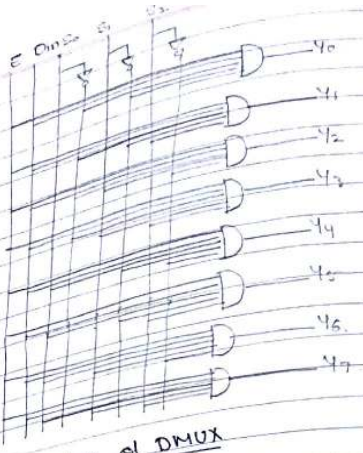


) 1:8 DMUX



E	S_0	S_1	S_2	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	x	x	x	x	x	x	x	x	x	x	x
1	0	0	0	Din	0	0	0	0	0	0	0
	0	0	1	0	Din	0	0	0	0	0	0
	0	1	0	0	0	Din	0	0	0	0	0
	0	1	1	0	0	0	Din	0	0	0	0
	1	0	0	0	0	0	0	Din	0	0	0
	1	0	1	0	0	0	0	0	Din	0	0
	1	1	0	0	0	0	0	0	0	Din	0
	1	1	1	0	0	0	0	0	0	0	Din

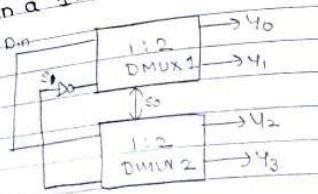
pression: $EDin(\bar{S}_0\bar{S}_1\bar{S}_2 + \bar{S}_0\bar{S}_1S_2 + \bar{S}_0S_1\bar{S}_2 + \bar{S}_0S_1S_2 + S_0\bar{S}_1\bar{S}_2 + S_0\bar{S}_1S_2 + S_0S_1\bar{S}_2 + S_0S_1S_2)$



Designing of DMUX

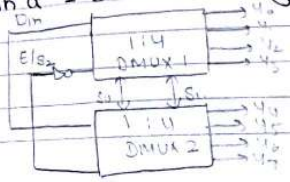
a) DMUX TREES

Obtain a 1:4 DMUX using 2 (1:2) DMUX.



E	S_0	S_1	Y_0	Y_1	Y_2	Y_3
	0	0	Din	0	0	0
	0	1	0	Din	0	0
	1	0	0	0	Din	0
	1	1	0	0	0	Din

b) Obtain a 1:8 DMUX using 1:4 DMUX.

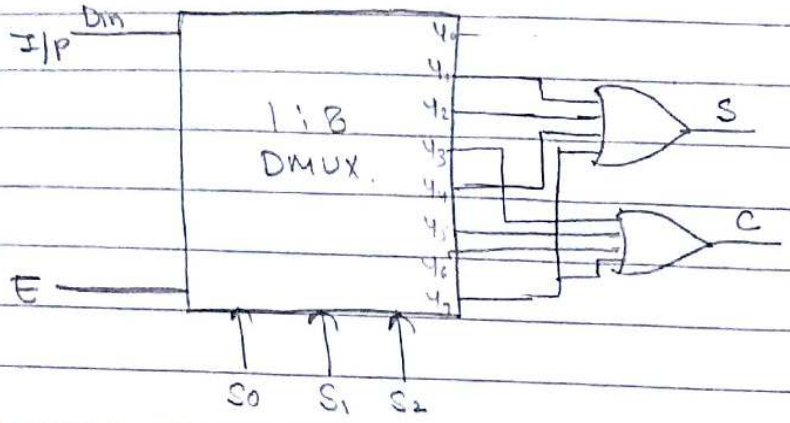


b) Using combination circuits

Q ⇒ Design full adder using DMUX.

$S = \Sigma m(1,2,4,7)$

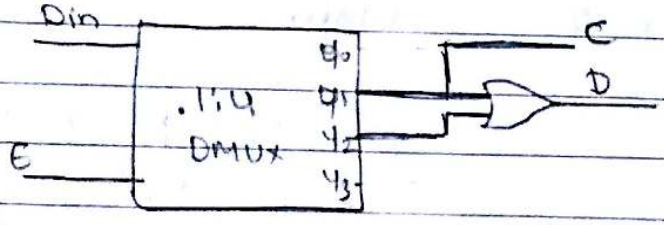
$C = \Sigma m(3,5,6,7)$



Q ⇒ Half Subtractor

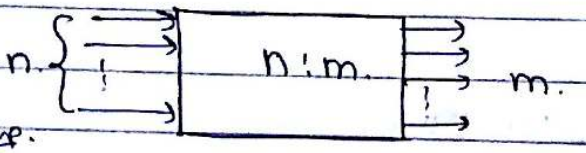
$D = \Sigma m(1,2)$

$B = \Sigma m(1)$

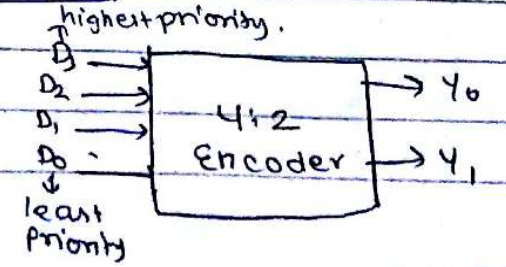


26/08/15

ENCODER'S (n:m)



Priority encoder :-

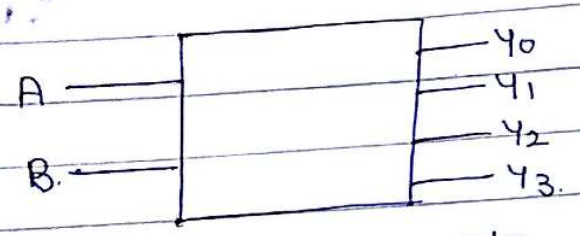


I/P				O/P	
D_3	D_2	D_1	D_0	Y_0	Y_1
1	X	X	X	1	1
0	1	X	X	1	0
0	0	1	X	0	1
0	0	0	1	0	0

DECODERS :-

It converts the n-bit binary information into a maximum of 2^n output lines.

1) 2 to 4 line decoder.



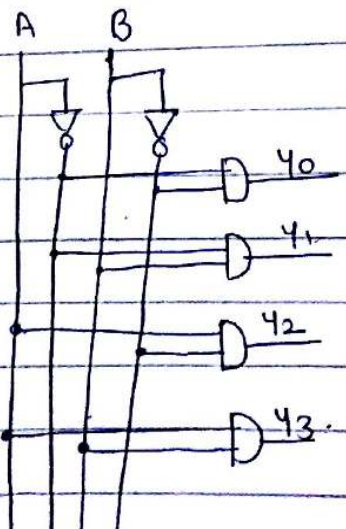
I/P		O/P			
A	B	Y_0	Y_1	Y_2	Y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$Y_0 = \bar{A}\bar{B}$

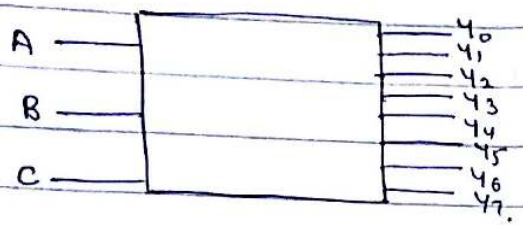
$Y_1 = \bar{A}B$

$Y_2 = A\bar{B}$

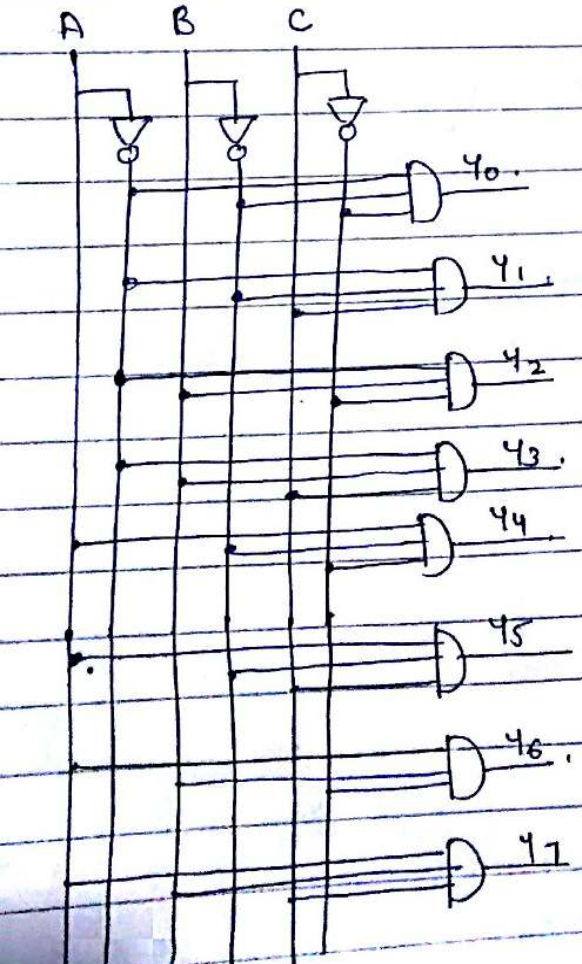
$Y_3 = AB$.



2) 3 to 8 line decoder



A	B	C	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



3) 4 line to 10 line / BCD to decimal decoder

B_3	B_2	B_1	B_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

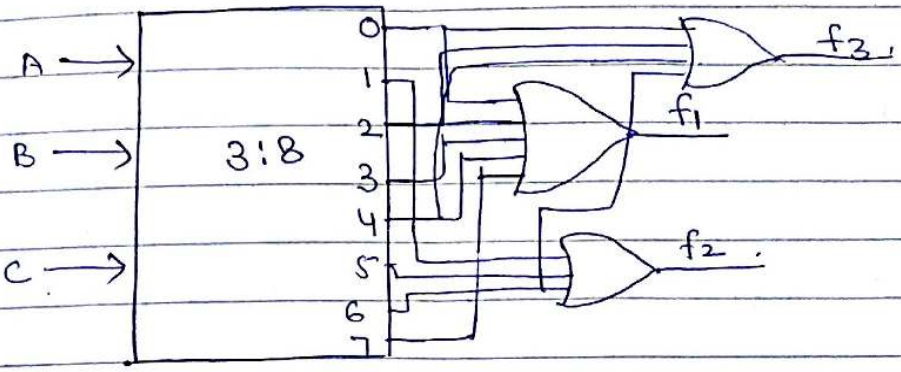
$D_0 = \bar{B}_0 \bar{B}_1 \bar{B}_2 \bar{B}_3$

Q ⇒ $f_1(A,B,C) = \Sigma m(0,4,7) + d(2,3)$.

$f_2(A,B,C) = \Sigma m(1,5,6)$

$f_3(A,B,C) = \Sigma m(0,2,4,6)$

Implement the above fn. using suitable decoders.

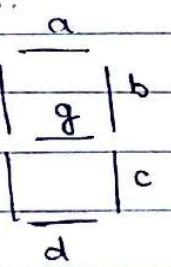


BCD to 7 Segment display :-

BCD - 0000 – 1001

0 – 9.

7-Segment

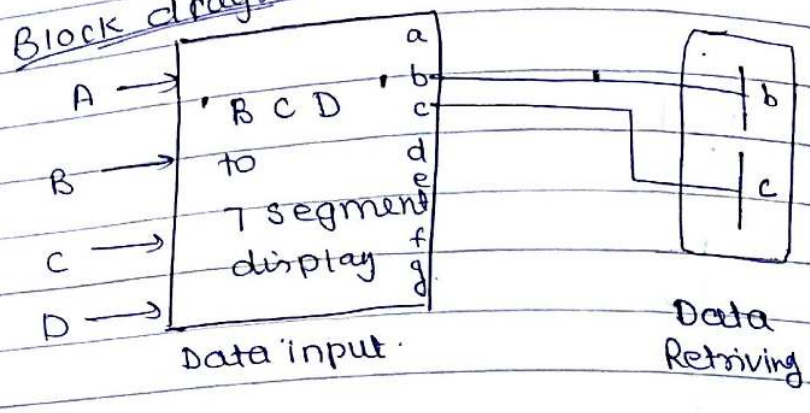


B_3	B_2	B_1	B_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	1	0	0							
1	1	0	1							

for 'a'

B_1B_0 \ B_3B_2	00	01	11	10
00	1			
01				
11				
10	1			

Block diagram



31/08/15

COMPARATOR :- (combinational circuit)

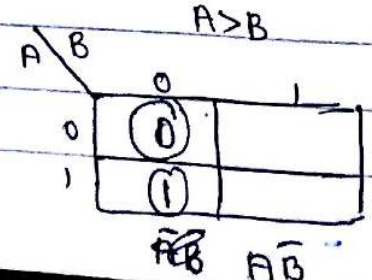
1-bit magnitude comp (A>B, A=B, A<B)

2-bit magnitude comp.

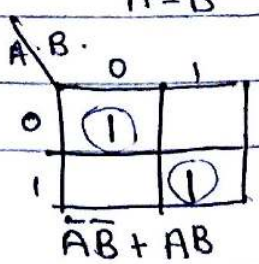
a) 1-bit magnitude comparator :-

A	B	A>B	A=B	A<B.
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

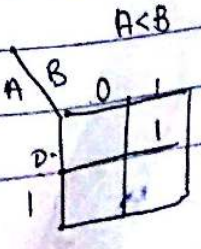
A>B

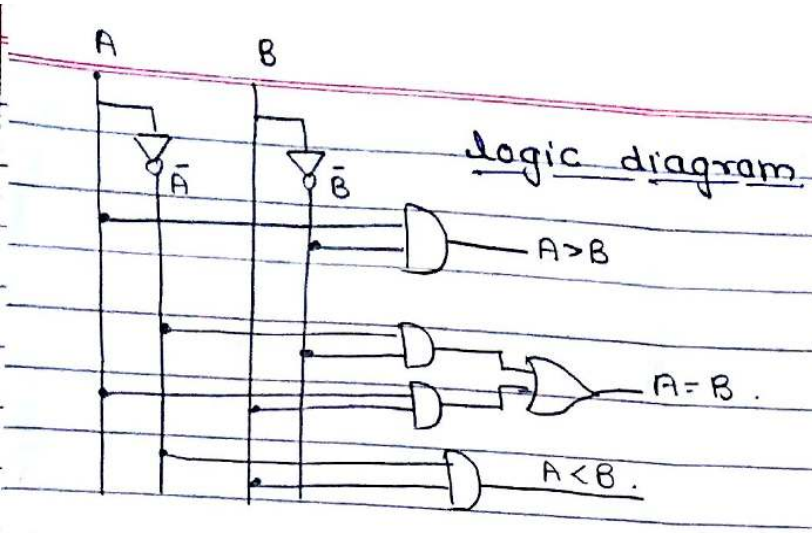


A=B



A<B





Logic diagram

b) 2-bit comparator :-

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

A>B

A_1A_0 \ B_1B_0	00	01	11	10
00				
01	1			1
11	1	1		
10	1	1		

ALU :- (Arithmetic logic unit)

It is an integrated circuit within a CPU that perform arithmetic and logic operation. Arithmatic instruct include addition, subtraction and shifting operation while logic instruction include boolean instruction a comparison such as AND, OR, XOR & NOT operation. ALU's are designed to perform integer calculations al They do not perform division operations or non-in calculations.

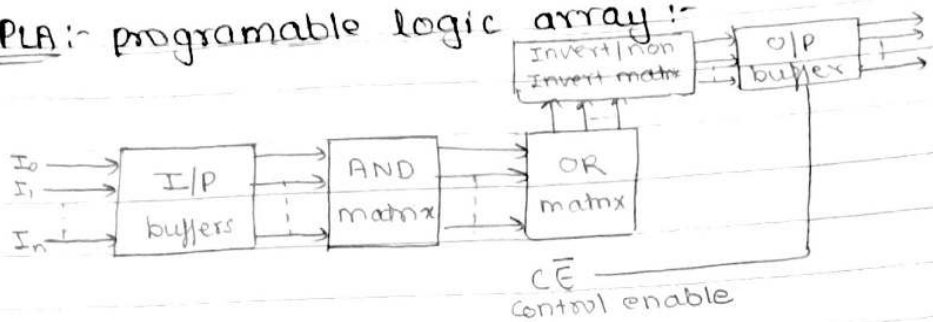
PLD :- (programable logic devices)

PLD are used to design large scale digital circuits. always perform their operation in matrix form. The gat wed to define design there matrix are the basic g ie AND, OR, NOT gate. A PLD generally consist of progr -mable array of logic gates. PLD's are classified in two catagories.

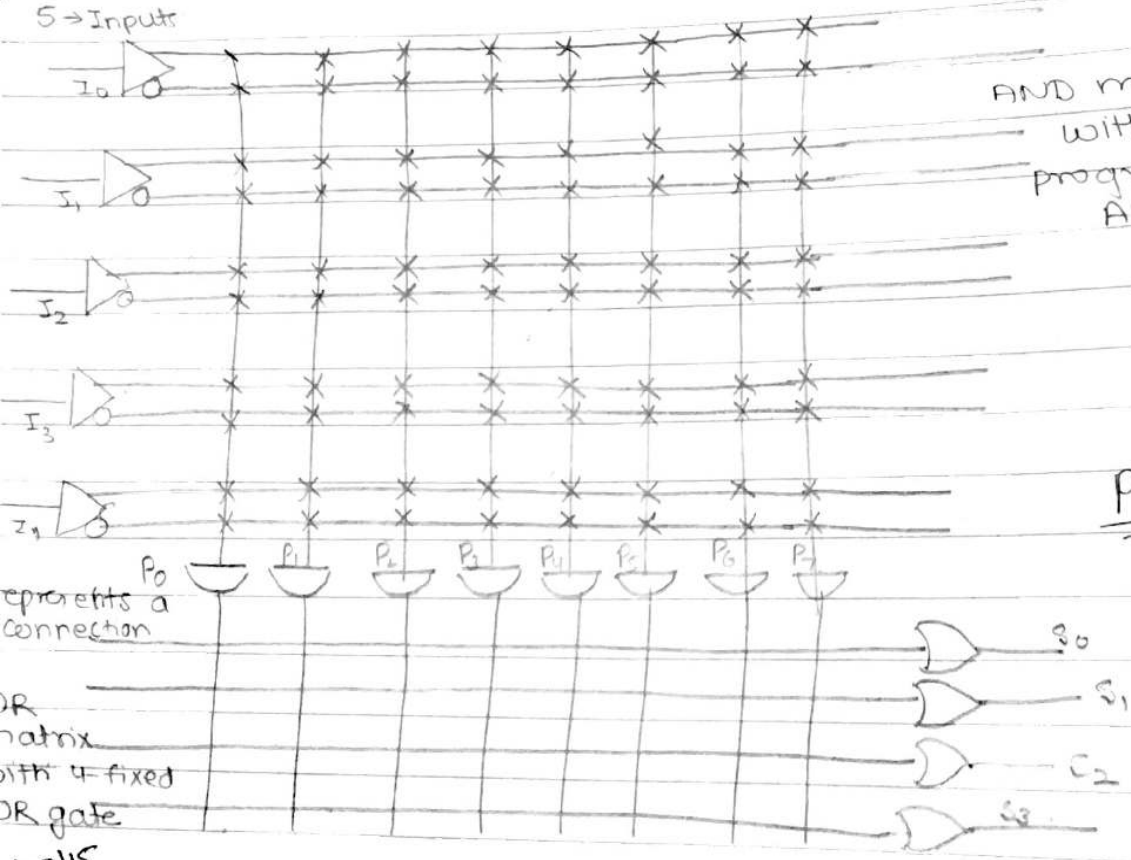
1) PAL → Programmable array logic
2) PLA → Programable logic array

buffer- control the loading

① PLA :- programable logic array :-



PLA



AND matrix with 8 programmable AND gates.

PAL

ogic

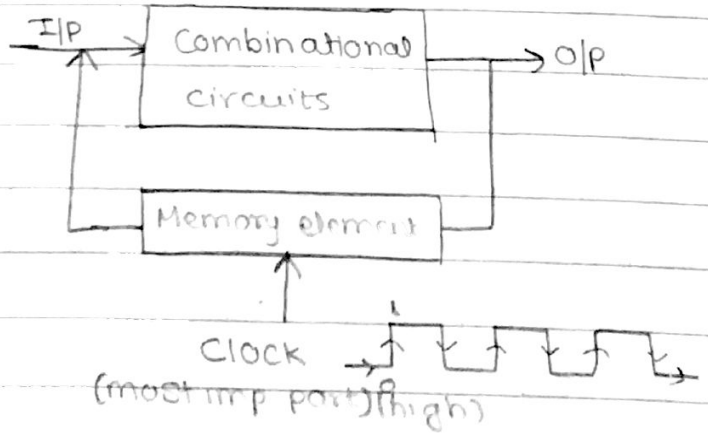
Difference between PROM | PLA | PAL

PROM	PLA	PAL
t is cheap & easy to se	It is expensive than PAL and PROM and complicated to use.	Moderately expensive and moderately complicated too.

PROM	PLA	PAL
② There are fixed 'AND' array's and programable OR arrays.	'AND and 'OR' both array's are programable	Only the AND array is programable while the OR array is fixed.
③ SOP fn in the standard form can only be implimented in PROM.	Any SOP fn can be implimented.	Any SOP fn can be implimented
④ It is possible to Decode any min-term in PROM.	We can get any desired min-term by programming the AND and OR matrix	We can get any desired min term by programing the AND matrix only.

Imp

SEQUENTIAL CIRCUITS :-



Types of Sequential circuits :-

① Flip Flops (FFs)
② Registers.
③ Counters.

① FLIP FLOP'S :-

It can store only one bit of binary information.