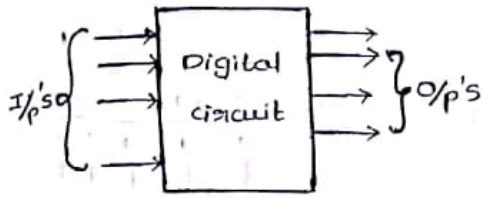
Digital circuits

Combinational circuits

→ the o/p is any instant of time it depends on same instant of i/p's.



→ any circuit construct with only gates, with out any memory element

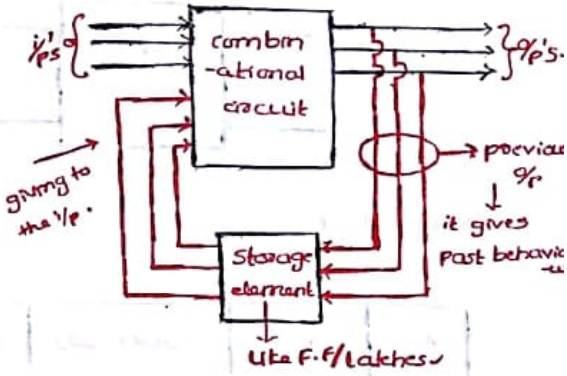
→ no feed back

Examples:

01. Half Adder, Half subtractor
Full Adder, Full subtractor
Quarto Adder, Quarter subtractor
02. Encoder, Decoder
03. MUX, Demux
04. ROM
05. Binary parallel Adder, etc.

Sequential circuits

→ the circuit o/p depends on present i/p and past o/p.



→ previous o/p gives to the i/p.
means past o/p

→ feed back require.

→ These are constructed by gates Flip Flops.

Examples:

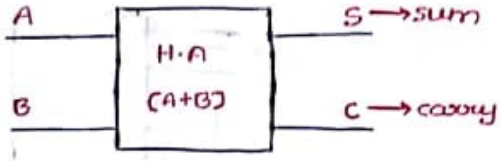
01. Serial Adder, Stakmachines
02. RAM (or Read & write memory)
03. Registers
04. counters
05. Sequential generators.

Combinational circuits.

I) (a) Half Adder:

→ A logic circuit for the addition of (2) one-bit numbers is referred to as an "HALF ADDER" (H.A)

→ It generates sum (S) and carry (C).



				(1)	(2)
A	0	0	1	1	
+B	+0	+1	+0	+1	
	0	1	1	(1) 0	
	↑	↑	↑	↑	↑
	S	S	S	C	C

Truth table:

A	B	sum (S)	carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

EX-OR

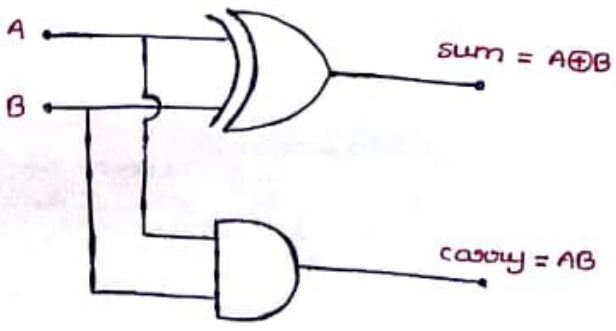
AND

Logical operation:

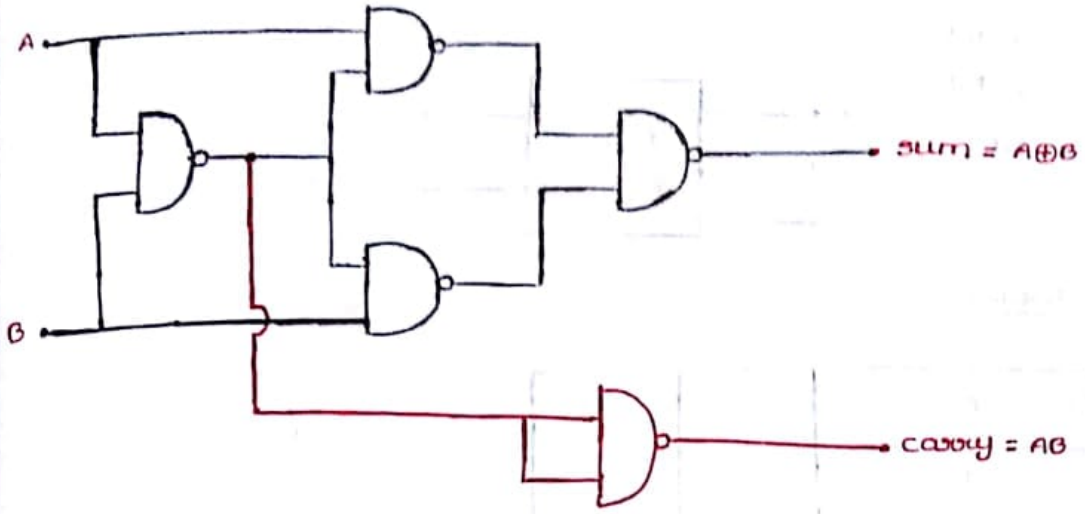
$S(A,B) = A\bar{B} + \bar{A}B = A \oplus B$ [EX-OR]

$C(A,B) = AB$ [AND]

Realization of H.A:



NAND implementation of an H.A ckt:

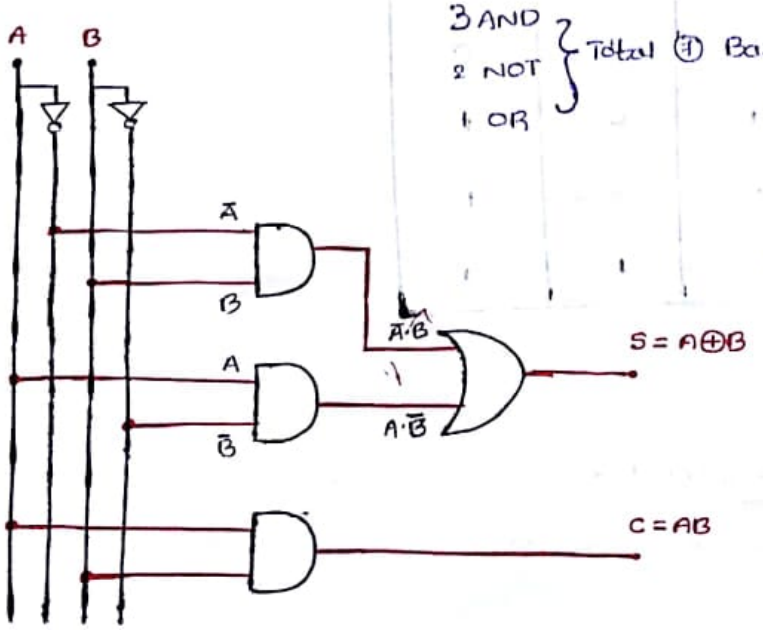


→ total number of NAND gates = ⑤

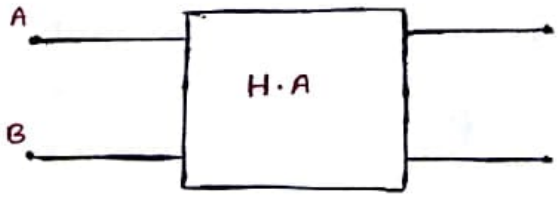
→ also total number of NOR gates = ⑤

by using And/OR gates:

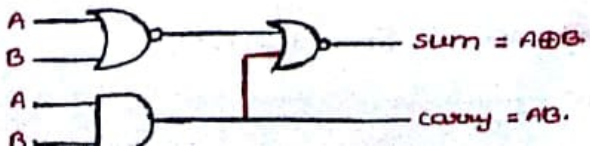
3 AND
2 NOT
1 OR
} Total ④ Basic gates required.



The ANSI/IEEE standard logic symbol:



* implement the H.A circuit by minimum number of logic gates if we have all gates except Ex-OR/Ex-NOR is "③"



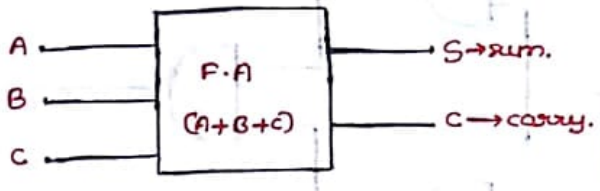
b) Full Adder:

→ It performs the arithmetic sum of the three i/p bits i.e

addend bit

augend bit

carry bit.



Truth Table:

A	B	C	S	C
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

Logical expression:

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

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

sum → $S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

$$S = A \oplus B \oplus C$$

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

$$C = AB + BC + CA$$

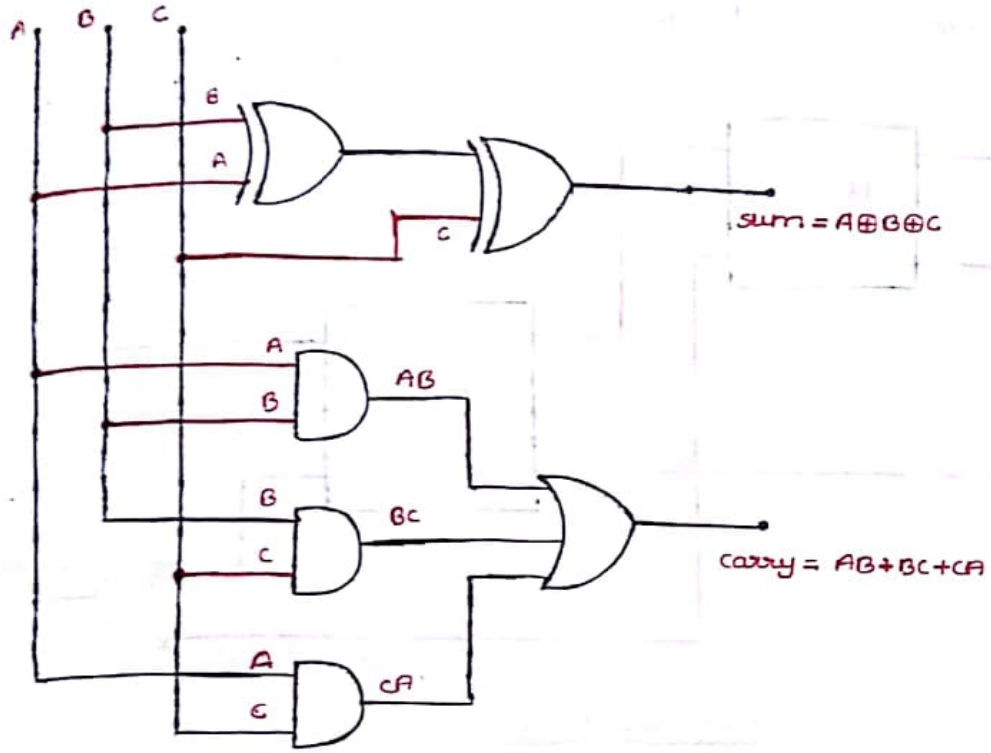
(Q) In F.A the carry has the logical expression same as that of the 3-bit majority logical expression function [NTPC-2009]

Sol:

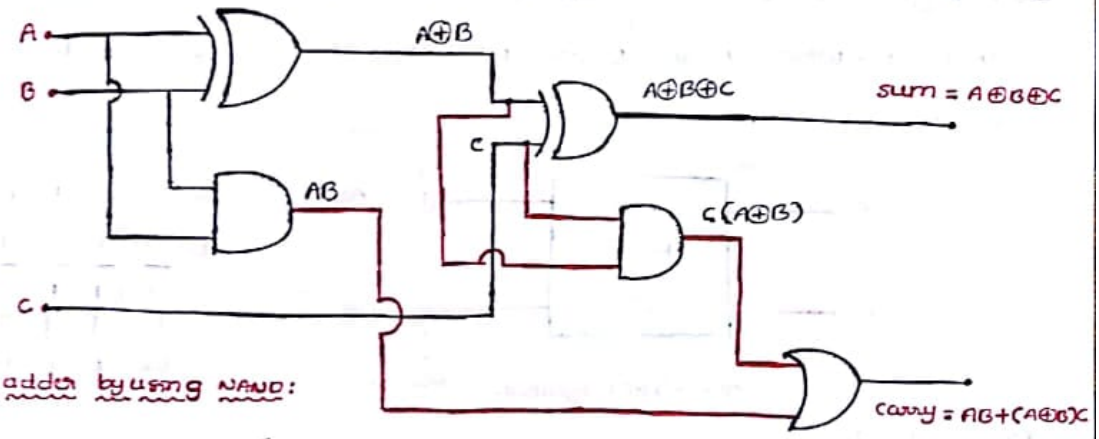
$C = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC = \Sigma m(3,5,6,7)$

$$C = C[A \oplus B] + AB$$ ✓

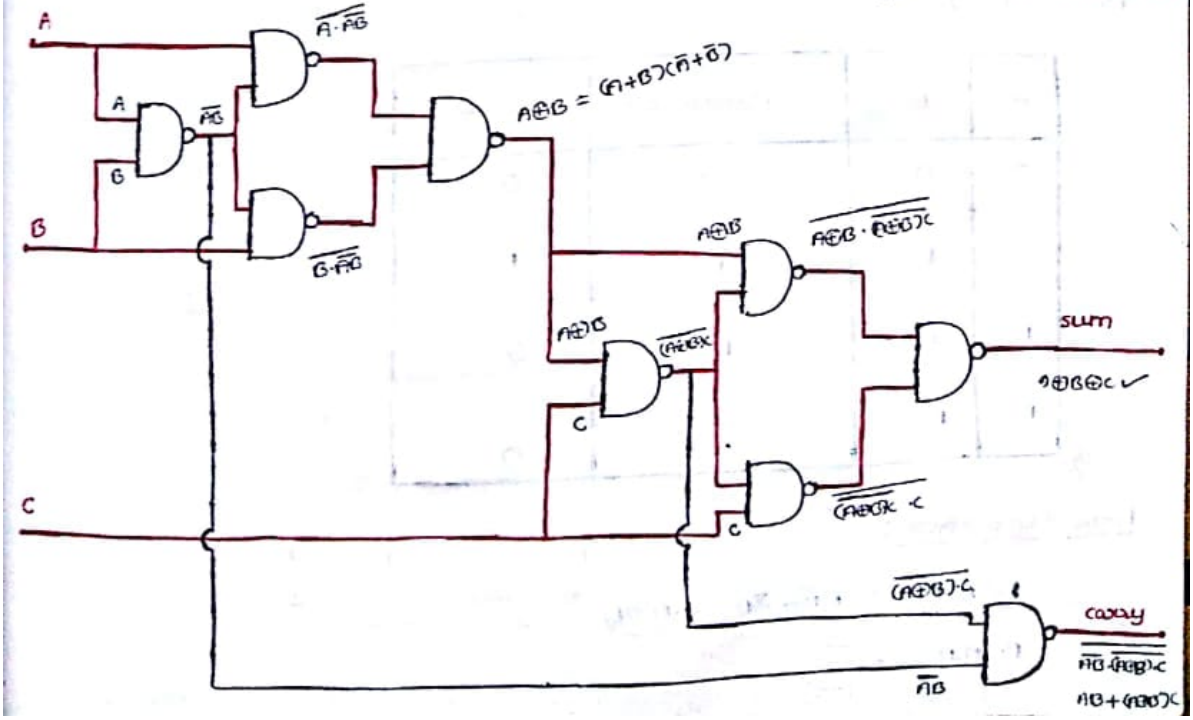
→ 2-level realization of F.A:



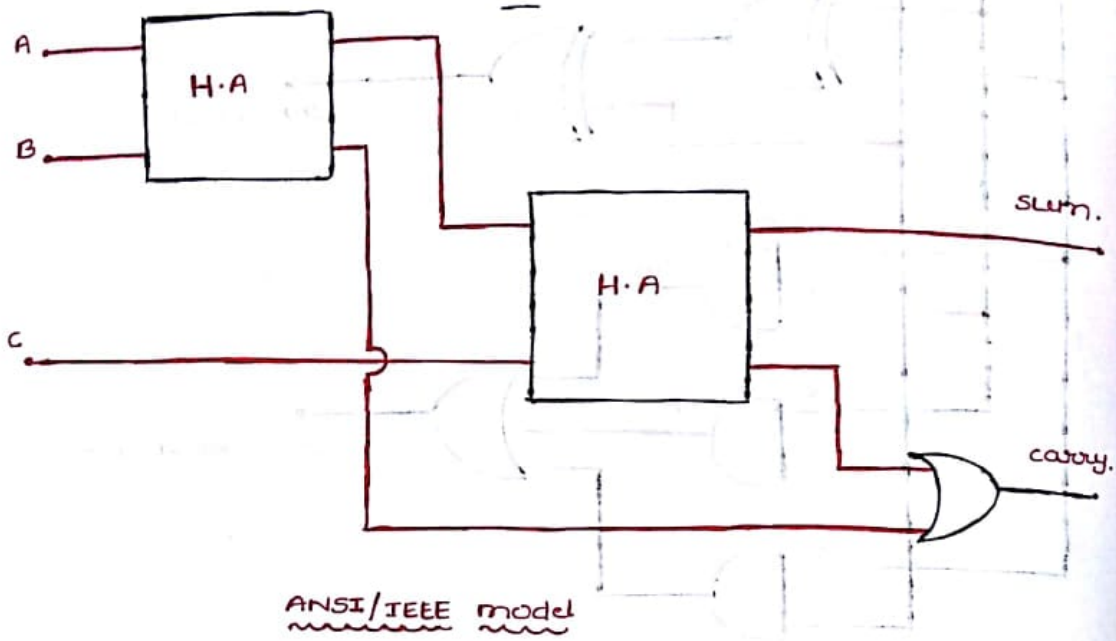
→ A F.A can be implemented by ② H.A and OR-gate:



→ full adder by using NAND:



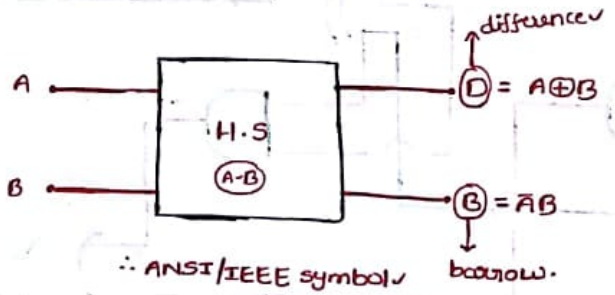
* Total number of NAND-gate/NOR-gate required to implement a full adder is equals to "9".



ANSI/IEEE model

c) Half subtractor:

→ A logic circuit for the subtraction of B (subtrahend) from A (minued). where A and B are 1-bit numbers is referred to as an H.S.



∴ ANSI/IEEE symbol

A	0	0	1	1
-B	-0	-1	-0	-1
	0	1	1	0
		B		
	D	D	D	D

Truth Table:

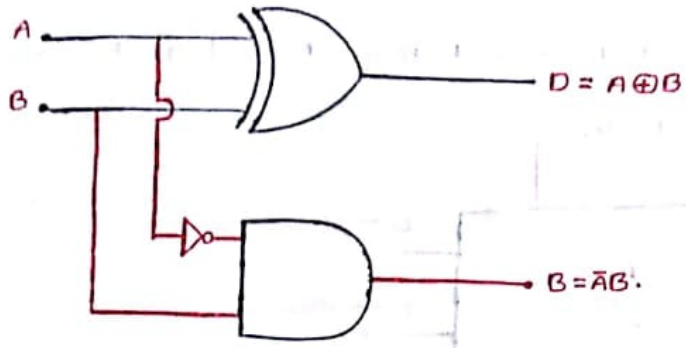
A	B	Difference D	Borrow B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Logic expression:

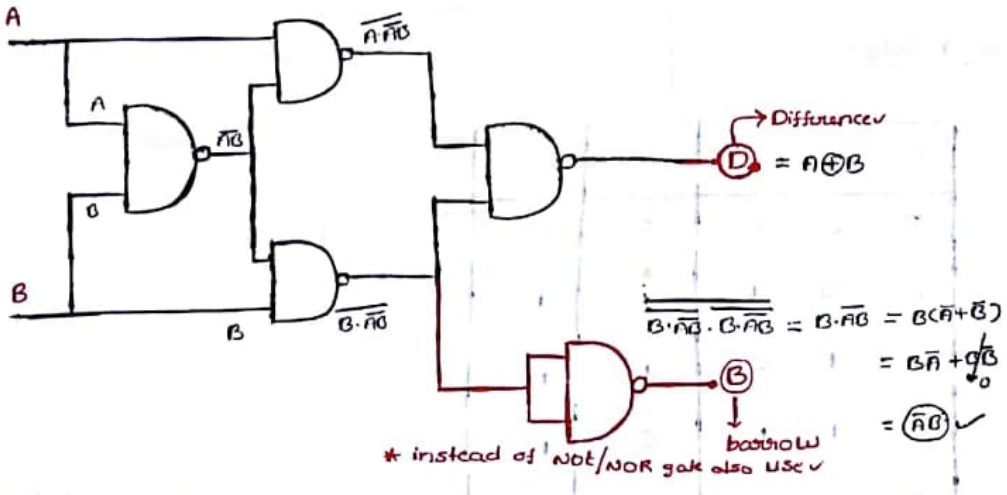
$D(A,B) = A\bar{B} + \bar{A}B = A \oplus B$

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

Realization of a H.S:

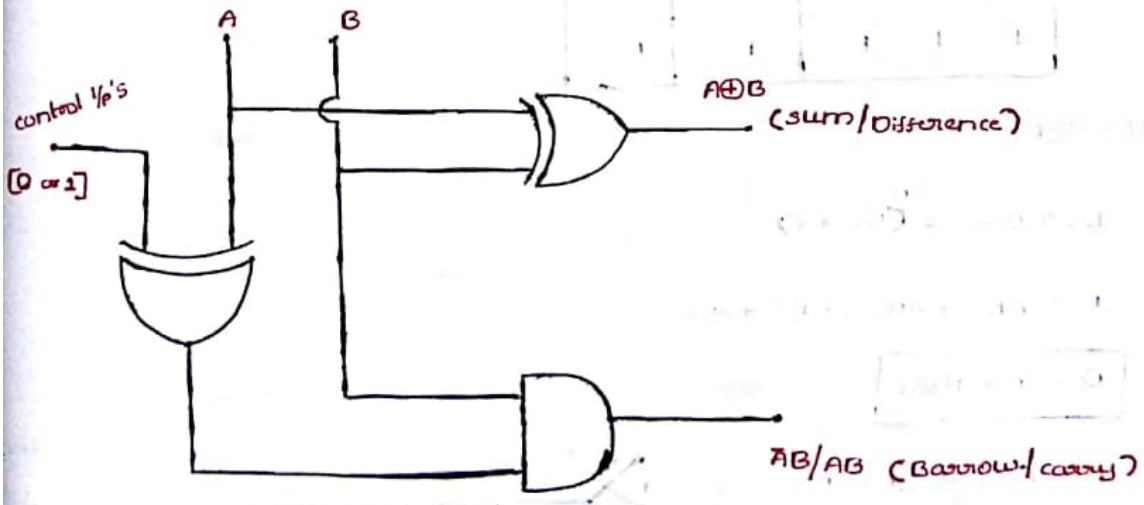


By using NAND gates:



→ Total number of NAND/NOR gates required to implement the Half subtractor is equals to ⑤.

* H.A/H.S circuit by using control action of EX-OR gate:



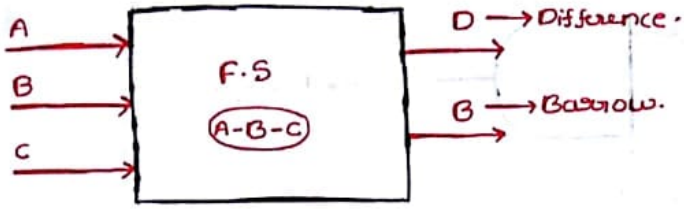
When control i/p = 0 ⇒ H.A operation.

When control i/p = 1 ⇒ H.S operation.

→ control ① means EX-OR gate acts like as an inverter and for '0' it is only a Buffer.

d) Full subtractor:

→ It is a circuit which performs a subtraction b/w ② bits taking into account that a '1' may have been borrowed by a lower significant stage.



∴ ANSI/IEEE symbol ✓

Truth Table:

A	B	C	D	B
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

Logic expression:

$$D(A,B,C) = \sum m(1,2,4,7)$$

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

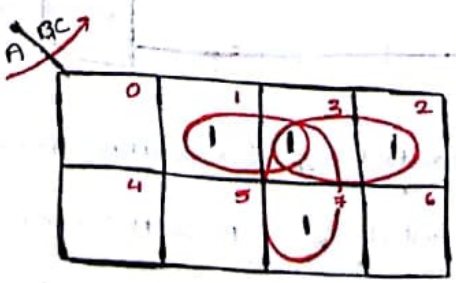
$$\boxed{D = A \oplus B \oplus C}$$

$$B(A,B,C) = \sum m(1,2,3,7)$$

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

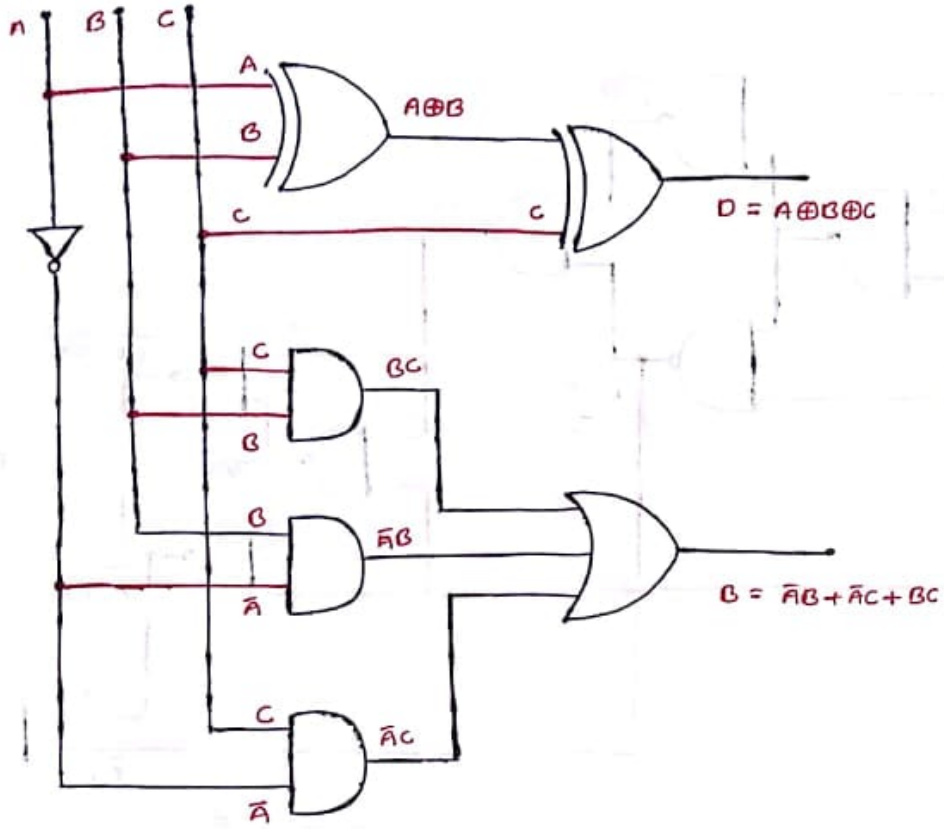
$$B = \bar{A}B(\bar{C}+C) + C(\bar{A}\bar{B}+AB)$$

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

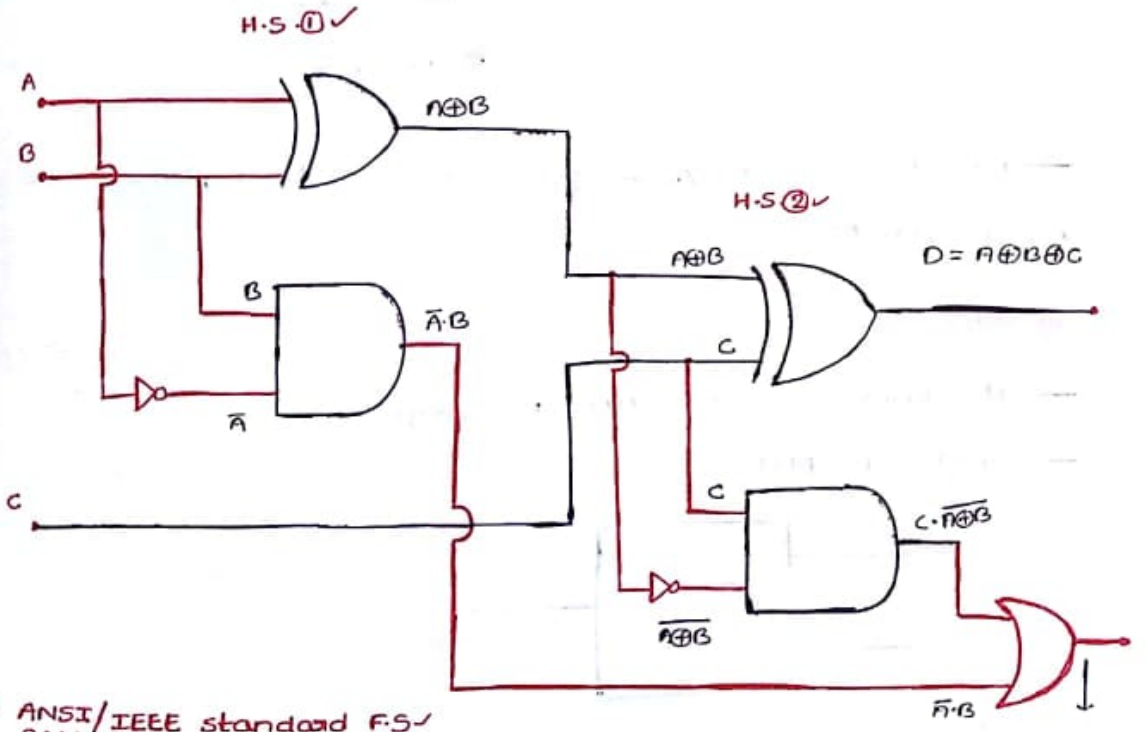


$B = \bar{A}B + \bar{A}C + BC$ ✓

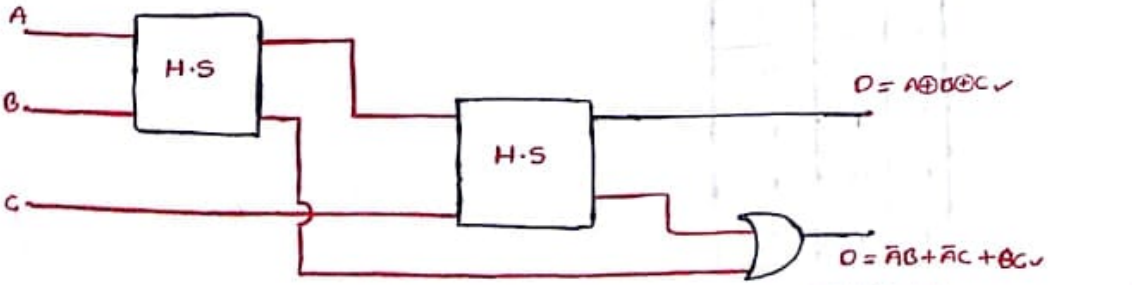
Realization of F.S Ckt:



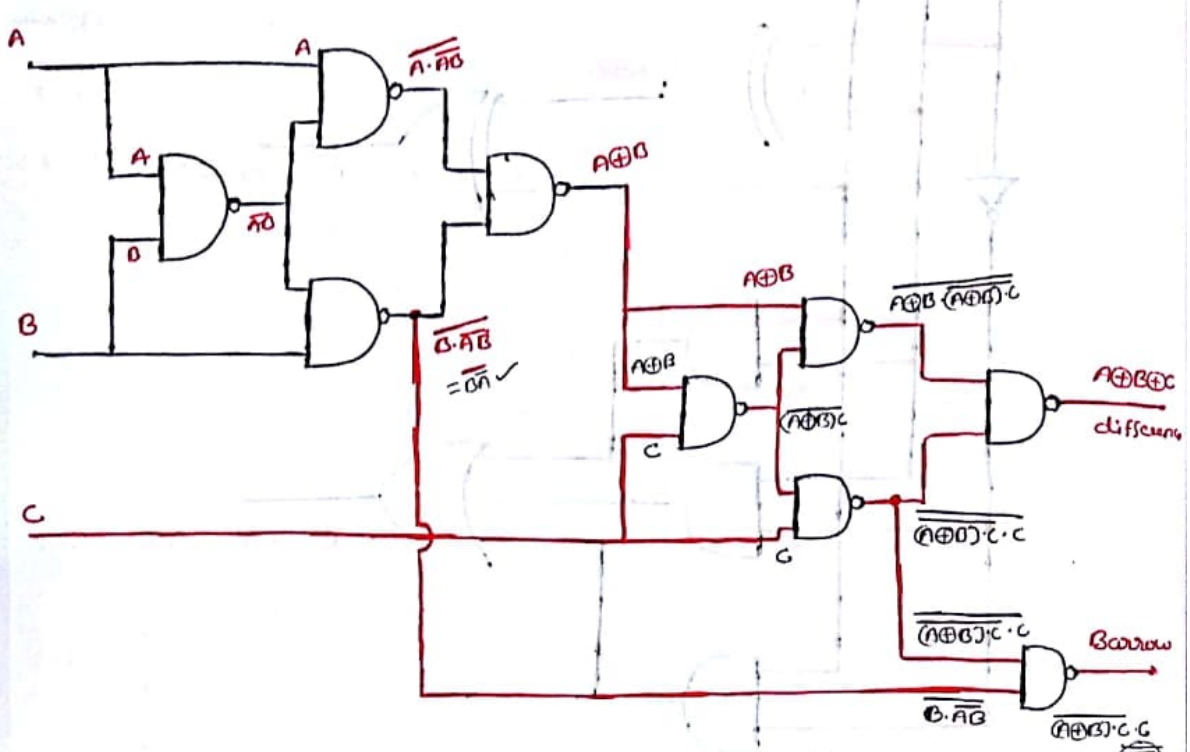
→ A F.S can be implemented with ②-H.S and one OR gate below shown ✓



→ ANSI/IEEE standard F.S ✓



by using NAND gates:



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

$$[\overline{(A⊕B)} + \bar{C}]C + \overline{B\bar{A}}$$

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

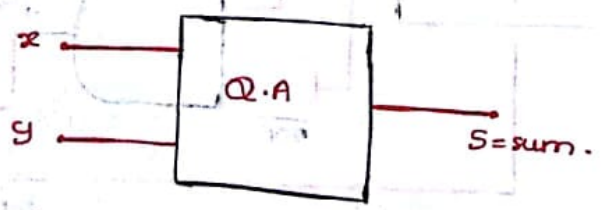
$$B = B\bar{A} + \overline{(A⊕B)}\cdot C$$ ✓

→ Number of NAND/NOR gates required to implement the F.S is equal to (9) ✓

e) Quarter Adder:

→ It generate only sum, no carry.

→ single out put.



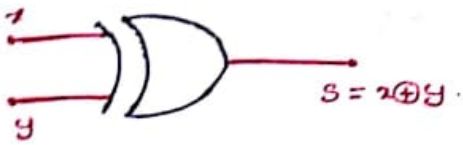
Truth Table:

x	y	S
0	0	0
0	1	1
1	0	1
1	1	0

Logic expression:

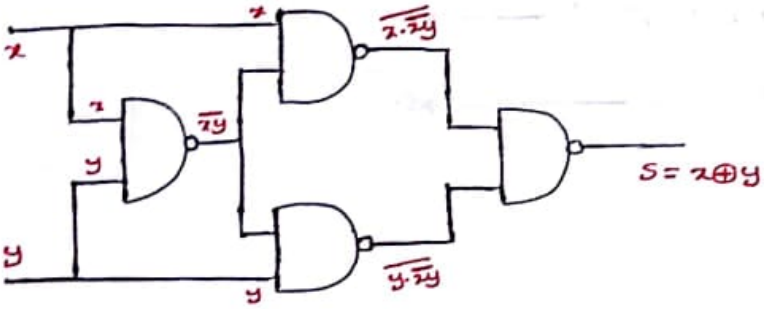
$S(x,y) = x\bar{y} + \bar{x}y = x \oplus y$ ✓

Realisation of Q.4:



EX-OR ✓

by using NAND gates:



5) Quarter subtractor:

→ It generates only difference, no barrow.

→ single o/p.

Truth Table:

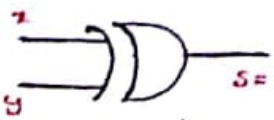
x	y	D
0	0	0
0	1	1
1	0	1
1	1	0

→ multi bit addition takes full adder

→ multi bit subtraction takes full subtractor ✓

Logic expression $S(x,y) = x\bar{y} + \bar{x}y = x \oplus y$

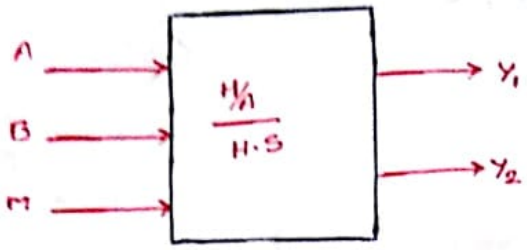
Realisation of Q.5:



→ Realised by NAND gate by construction Q.5 more accurate

Q) Design and realize controllable half adder/half subtractor circuit

Sol:



If M=0 : H·A ⟶ S = Y_1 ; C = Y_2.

M=1 : H·S ⟶ D = Y_1 ; B = Y_2.

$Y_1(m,A,B) = \Sigma m(1,2,5,6)$ ⟶ ①

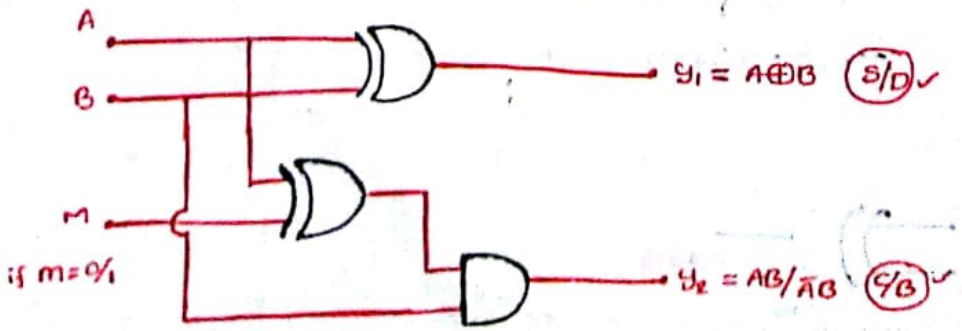
$Y_2(m,A,B) = \Sigma m(3,5)$ ⟶ ②

$Y_1 = A \oplus B$

$Y_2 = \bar{m}AB + m\bar{A}B$

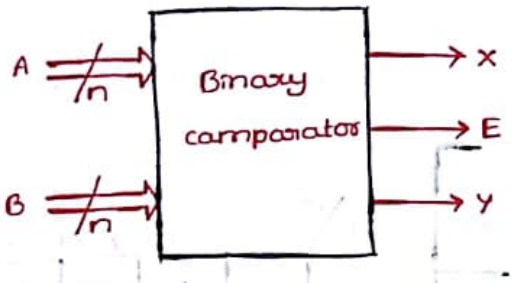
$Y_2 = B[m \oplus A]$

M	A	B	y_1	y_2	
0	0	0	0	0	H·A
0	0	1	1	0	
0	1	0	1	0	
0	1	1	0	1	
1	0	0	0	0	H·S
1	0	1	1	1	
1	1	0	1	0	
1	1	1	0	0	



Binary comparator:

→ It must compares two binary numbers. when it compares 2 bin -ary numbers bellow possibilities are available.



n-bit comparator.

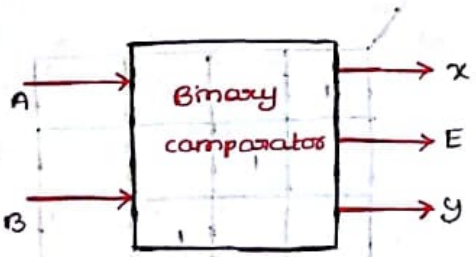
if $A > B \longrightarrow X = 1$

if $A = B \longrightarrow E = 1$

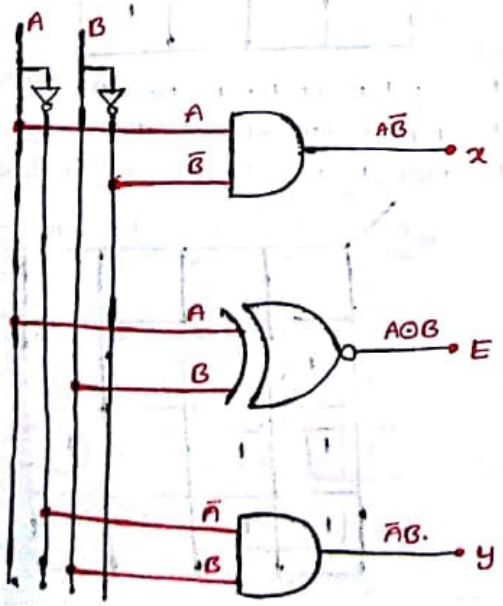
if $A < B \longrightarrow Y = 1$

→ It compares multibit numbers.

1-bit comparator: (n=1)



A	B	X	E	Y
0	0	0	1 (A=B)	0
0	1	0	0	1 (A<B)
1	0	1 (A>B)	0	0
1	1	0	1 (A=B)	0



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

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

(E) = $A \odot B$

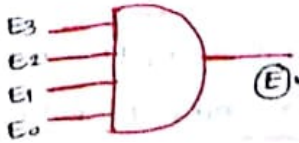
(Y) = $\bar{A}B$

→ for 4-bit comparator ✓ ⟸ Ex-NOR ✓

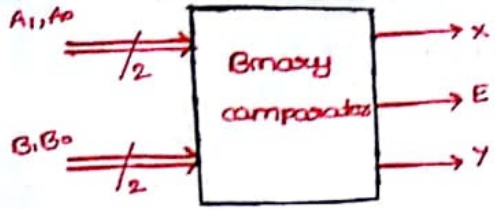
Ex:

	4	3	2	1
A =	A_3 1	A_2 1	A_1 0	A_0 1
B =	B_3 1	B_2 1	B_1 0	B_0 1
	0	(E3)	(E2)	(E1)	(E0)

$2^8 = 256$ ↓ combinations

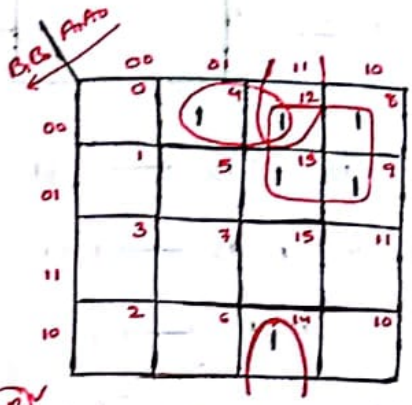


2-bit comparator (n=2)



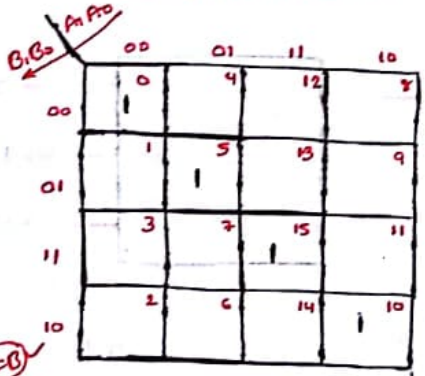
D.N	A1	A0	B1	B0	X	E	Y
0	0	0	0	0 ✓	0	(1)	0
1	0	0	< 0	1	0	0	1 ✓
2	0	0	< 1	0	0	0	1 ✓
3	0	0	< 1	1	0	0	1 ✓
4	0	1	> 0	0	1 ×	0	0
5	0	1	0	1 ✓	0	(1)	0
6	0	1	2	1	0	0	1 ✓
7	0	1	< 1	1	0	0	1 ✓
8	1	0	> 0	0	1 ×	0	0
9	1	0	> 0	1	1 ×	0	0
10	1	0	1	0 ✓	0	(1)	0
11	1	0	< 1	1	0	0	1 ✓
12	1	1	> 0	0	1 ×	0	0
13	1	1	> 0	1	1 ×	0	0
14	1	1	> 1	0	1 ×	0	0
15.	1	1	1	1 ✓	0	(1)	0

$X(A_1A_0B_1B_0) = \Sigma m(4,8,9,12,13,14)$



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

$E = \Sigma m(0,5,10,15)$

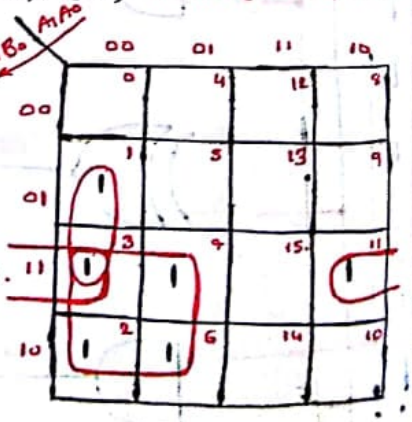


(A=B) ✓

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

$E = \bar{A}_1\bar{B}_1(A_0 \odot B_0) + A_1B_1(A_0 \odot B_0)$

$E = (A_0 \odot B_0)(A_1 \odot B_1)$ $Y = \Sigma m(1,2,3,6,7,11)$



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

→ 2 I/p Ex-NOR ⇒ Equality checking gates.

→ 2 I/p Ex-OR ⇒ Inequality checking gates.

* ⇒ i) if $A_1 = B_1$ then $E_1 = 1$

$A_0 = B_0$ then $E_0 = 1$

So (A=B) then $E = E_1E_0$ → ①

ii) if $A_1 > B_1$ & $A_1 = B_1$ but $A_0 > B_0$

(A>B) then $X = A_1\bar{B}_1 + EA_0\bar{B}_0$ → ②

iii) similarly

(A<B) ✓ $Y = \bar{A}_1B_1 + E_1\bar{A}_0B_0$ → ③

03/08/11

Binary Adders:

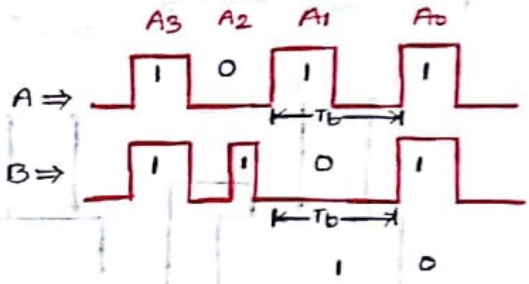
01. Serial Binary Adder
02. Parallel Binary Adder
03. Carry Look ahead Binary Adder ✓

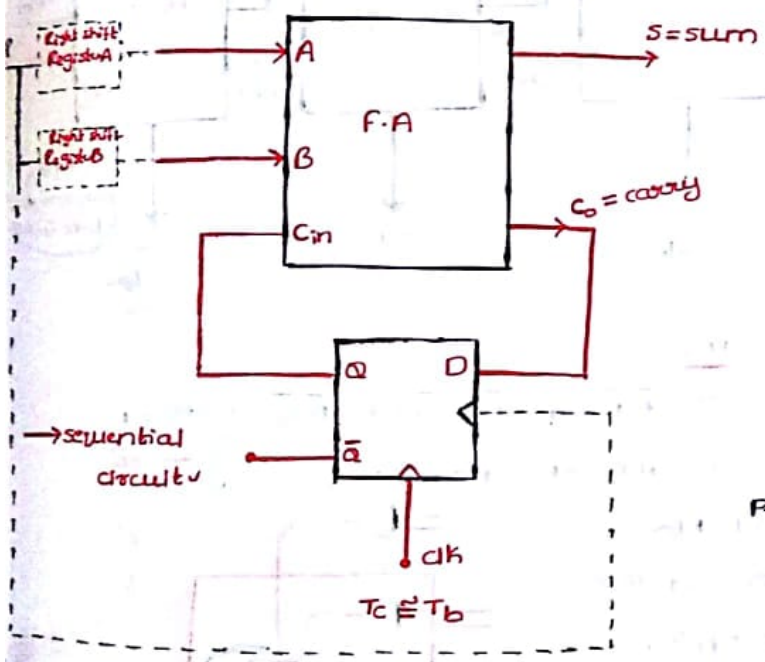
01. Serial Binary Adder:

$A \Rightarrow A_{n-1}\ A_{n-2} \cdots A_1\ A_0$

$B \Rightarrow B_{n-1}\ B_{n-2} \cdots B_1\ B_0$

(+)





clk	D	Qn+1
0	x	Qn
1	0	0
1	1	1

propagation delay (tpd) → ns, ps

$T_b \rightarrow$ μ·sec.

$T_b > t_{pd}$ ✓

* Serial Binary Adder → $(n-1)T_b$; Ex: $n=4$ → $3T_b$ ✓

Parallel Binary Adder → $(2n-1)t_{pd}$ → $7t_{pd}$ ✓

→ We can add numbers stored in the right shift registers A and B serially

→ The full adder is used to perform bit by bit addition

→ D flip flop is used to store the carry o/p generated after addition.

→ This carry is used as carry i/p for the next addition.

→ Initially, the D-flip flop is cleared and addition starts with the least significant bits of both Register.

→ After each clock pulse data with in the Right shift registers are shifted right 1-bit and we get bits from next digit and carry of previous addition as new i/p's for the full Adder.

02) Parallel Binary Adder (Ripple carry Binary Adder)

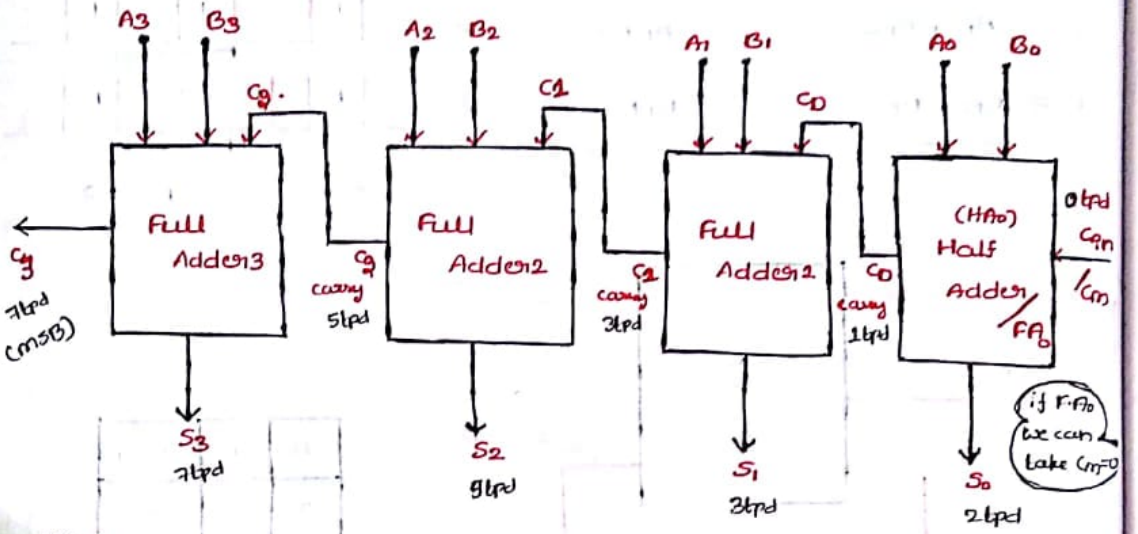
→ By using this we can add ② numbers.

A → A_3 A_2 A_1 A_0

(+) B → B_3 B_2 B_1 B_0

C_y S_3 S_2 S_1 S_0

final carry — sum ✓



HA_0:

$S_0 = A_0\bar{B_0} + \bar{A_0}B_0$

$C_0 = A_0B_0$ } → (A)

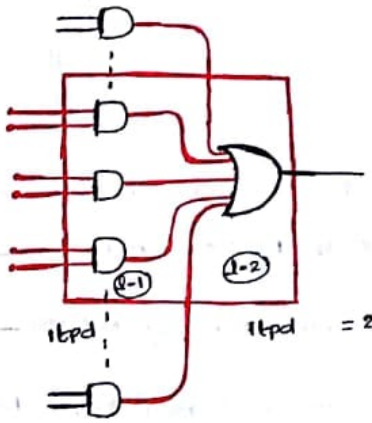
FA_i:

$S_i = \bar{A_i}\bar{B_i}C_{i-1} + \bar{A_i}B_i\bar{C_{i-1}} + A_i\bar{B_i}\bar{C_{i-1}} + A_iB_iC_{i-1}$

$C_i = A_iB_i + B_iC_{i-1} + A_iC_{i-1}$ } → (B)

→ In which the right most block labelled H/A is a half adder and the remaining blocks labelled F/A's.

→ Why because in first stage we have only ② i/p's i.e A_0, B_0 so we can take H/A. it produces S_0 & C_1 (i.e C_0)



→ suppose in 1st stage we can take full adder we make input carry (Cm) = 0 ✓

→ 2nd stage onwards F.A consists ③ i/p. i.e A_1, B_1, C_0 ✓

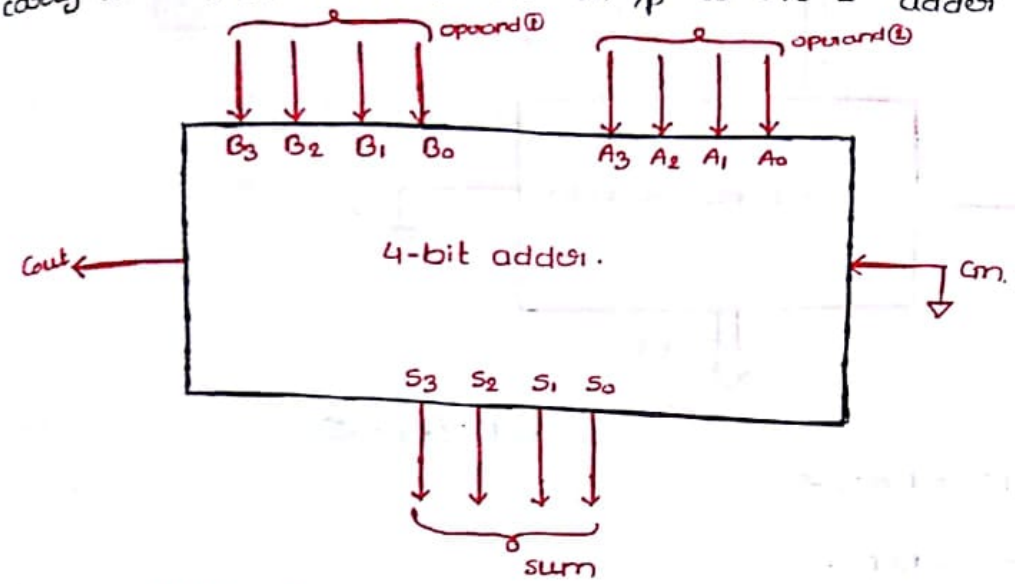
→ If we use H/A there is no i/p carry.

→ we can increase the capacity of the adder by adding full adders on the left side.

Ex: To add ⑧ bit number —need→ ⑦ F.A
① H.A

→ this type of binary adder known as the ripple adder because

carry from first adder forms an i/p to the 2nd adder and so on.

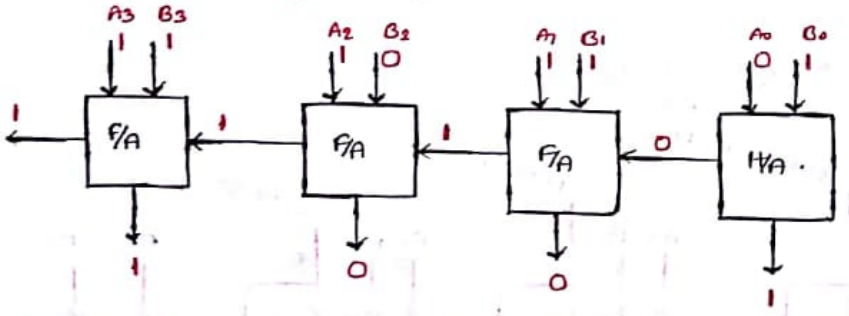


→ Logic diagram of 4-bit binary parallel adder

Ex①: A = 1110

B = 1011

? by using F.A ✓ ?



$C_3\ S_3\ S_2\ S_1\ S_0$ → 1 1 0 0 1

final carry ✓

1110 → 14

1011 → 11

1 1001 ✓ 25

Gate-03

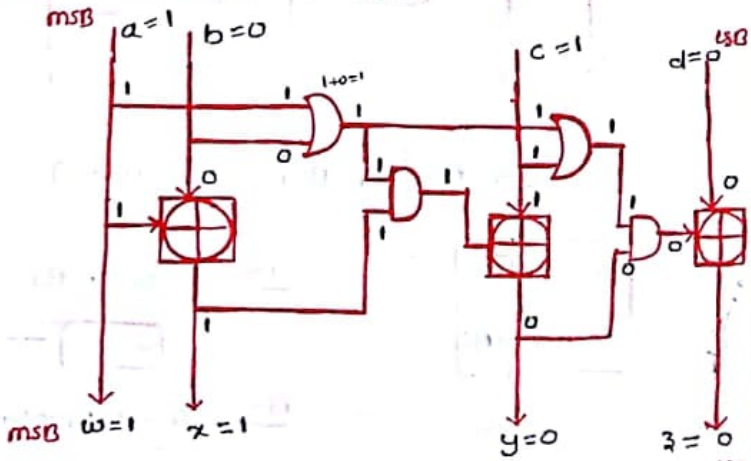
2Q) The circuit shown in the figure converts I/p's.

a) BCD to Binary code

b) Binary to excess-3 code

c) Excess ③ to Gray code

d) Gray to Binary code.



Sol: Take an example i/p → 1010

w = a

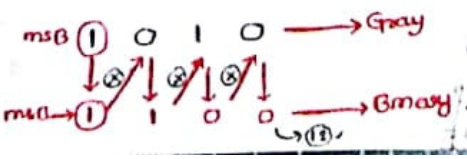
x = a⊕b

y = c⊕x(a+b)

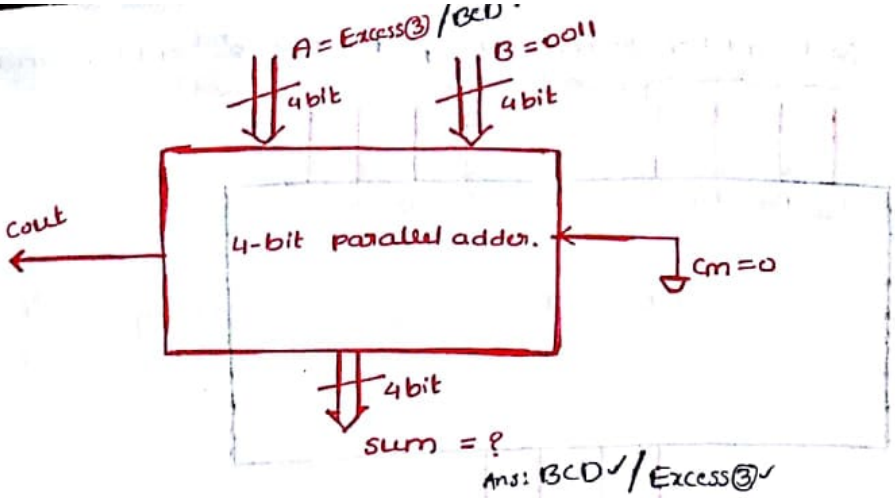
z = d⊕y(a+b+c)

1010 = i/p = ⑩ o/p's

→ By sub. given option in the Boolean equation of ② circuit, it shows Gray to binary code converter.



3Q)



Ans: BCD ✓ / Excess③ ✓

BCD + 3 = Excess-③

Excess③ - 3 = BCD.

A - B = sum ✓

A + 2's compliment of (B) = sum.

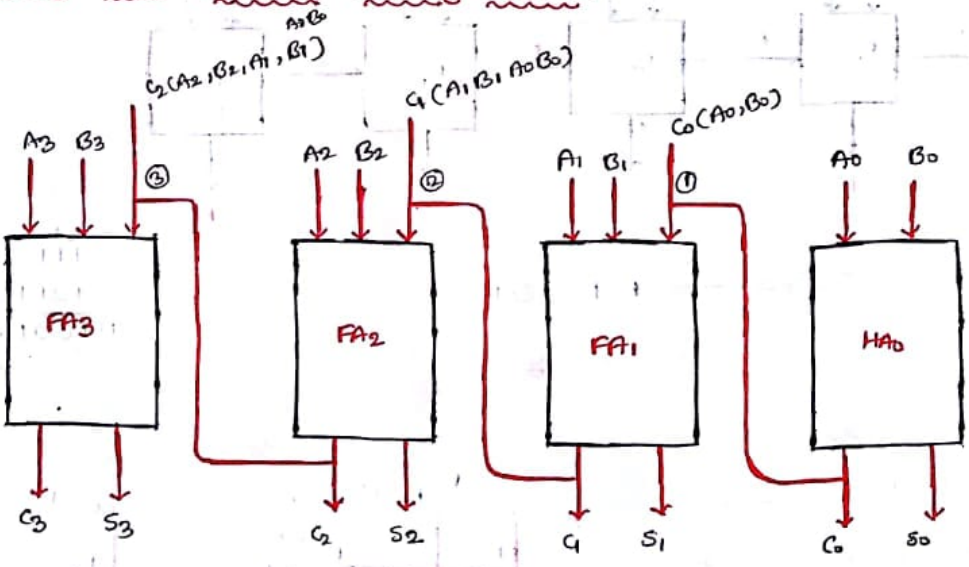
0011 = (B) ✓

1100 → 1's compliment

1101 → 2's compliment

Excess-3 + 1101 = BCD ✓

03) carry Look ahead Binary adder:



$C_i(A_i, B_i, A_{i-1}, B_{i-1}, \ldots\ldots A_0, B_0)$

$G_i = A_i B_i$ → 1 tpd

$P_i = A_i + B_i$

→ 2 tpd.

HA0: $C_0 = A_0 B_0$

$C_0 = G_0$ → (1)

FA1: $C_1 = A_1 B_1 + B_1 C_0 + A_1 C_0$

$C_1 = A_1 B_1 + C_0 (A_1 + B_1)$

$C_1 = G_1 + G_0 P_1$ → (2)

FA2: $C_2 = A_2 B_2 + B_2 C_1 + A_2 C_1$

$C_2 = G_2 + (G_1 + G_0 P_1) P_2$

$C_2 = G_2 + G_1P_2 + G_0P_2P_1$ ⟶ ③

In general $C_i = G_i + G_{i-1}P_i + G_{i-2}P_{i-1}P_i + \cdots + G_0P_iP_{i-1}\cdots P_1$ ⟶ ④

C_i ↓ 2 tpd.

1 tpd + 2 tpd. ⟶ 2 tpd + 2 tpd = 4 tpd ✓

3 tpd + 2 tpd = 5 tpd

⟶ for reduction of 1 tpd time delay, the cost of circuit increases more.

NOTE:

⟶ serial binary adder is an example of sequential circuit where as other two binary adders are the example of combinational circuit ✓

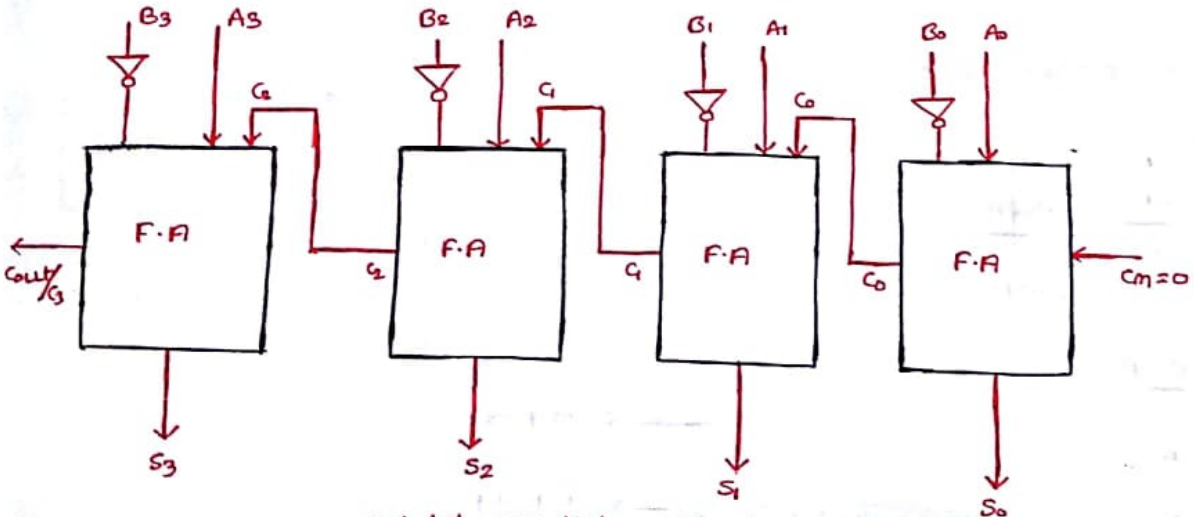
Binary subtractor:

⟶ the subtraction of binary numbers can be done most conveniently by means of complements.

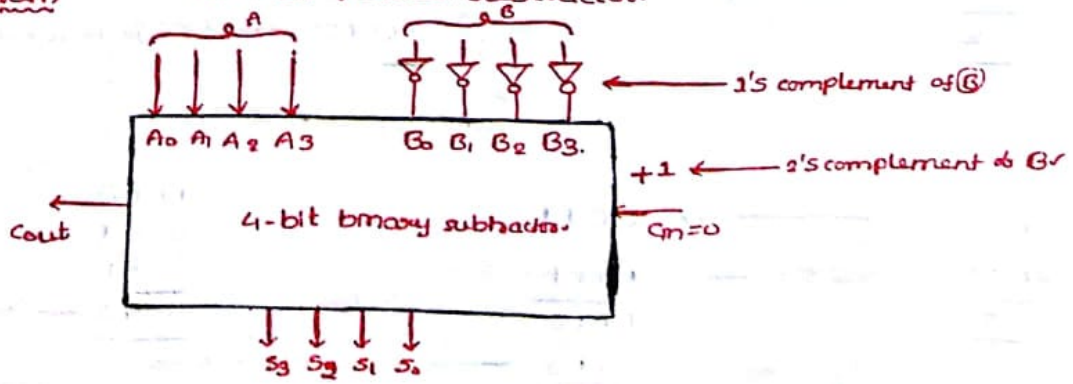
⟶ A-B

⟶ A + 2's complement of (B) ✓

⟶ 1's complement ⟶ inverters & '1' is added to the sum through the i/p carry.



Logic diagram ⟶ 4-bit parallel subtractor.



a) 1's

b) 2's

a) 1's complement:

A A

−B → + 1's complement of (B) (∵ A>B)

i.e

① A>B:

A = 5 ⟶ 0 1 0 1

−B = 3 ⟶ 1's of ③ is 1 1 0 0

2 ① 0 0 0 1

carry → +1 (Add end around carry)

0 0 1 0 ⟶ result +ve ✓

② A<B ✓

A = 3 ⟶ 0 0 1 1

−B = 5 1's of ⑤ ⟶ 1 0 1 0

−2 ① 1 0 1 ⟶ It is in 1's form

no carry → 0 0 1 0 ⟶ True magnitude

result in (−ve) ✓

b) 2's complement:

⟶ A A

−B 2's of B

i.e

i.e A>B ✓

i) A>B:

A = 5 ⟶ 0 1 0 1

−B = 3 2's complement of 3's is ⟶ (+) 1 1 0 1

2 ① 0 0 1 0 ⟶ True magnitude

carry (negleted)

ii) A<B:

A = 3 ⟶ 0 0 1 1

−B = 5 2's of '5' is ⟶ 1 0 1 1

−2 ① 1 1 0 ← 2's form

no carry → 0 0 1 0 ← True magnitude. result −ve

1 1 1 0

0 0 0 1

+1

0 0 1 0 → True form

Reasons to choose 2's procedure:

(a) How requirement for 2's complement procedure is less comparing to the 1's complement procedure.

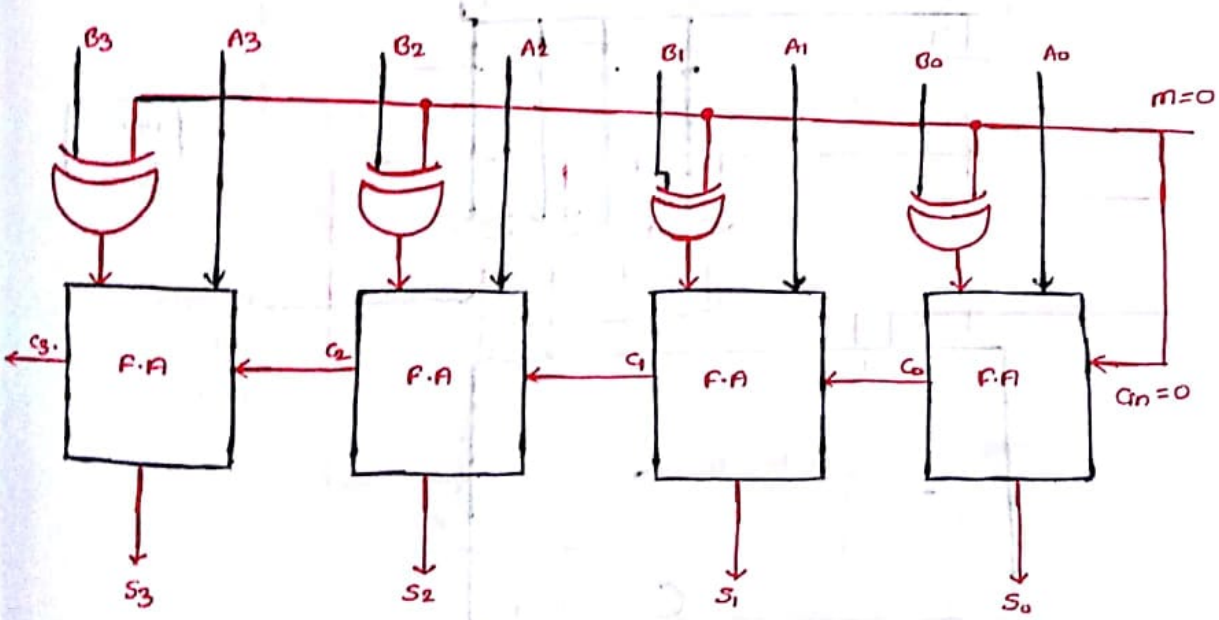
→ the advantage of signed 2's complement representation over the signed-1's compliment form is that it contains only one type of zero.

→ 2's complement form is usually choosen over 1's complement to avoid the occurance of a negative zero

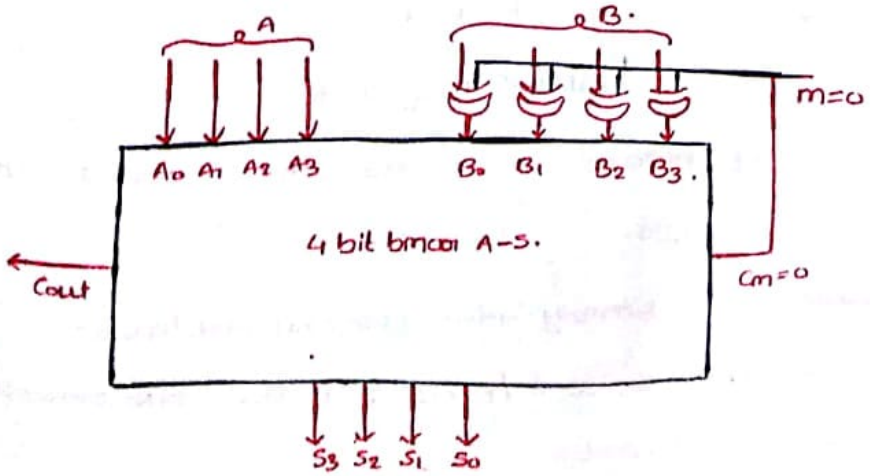
2's { 0 0000
0 0000 +0
1 1111 -0 } 1's form ✓

Binary Adder-subtractor:

→ the addition and subtraction can be combined into one circuit with one common binary adder. by using Ex-OR gate.



→ 4 bit adder-subtractor



Steps to be followed in design:

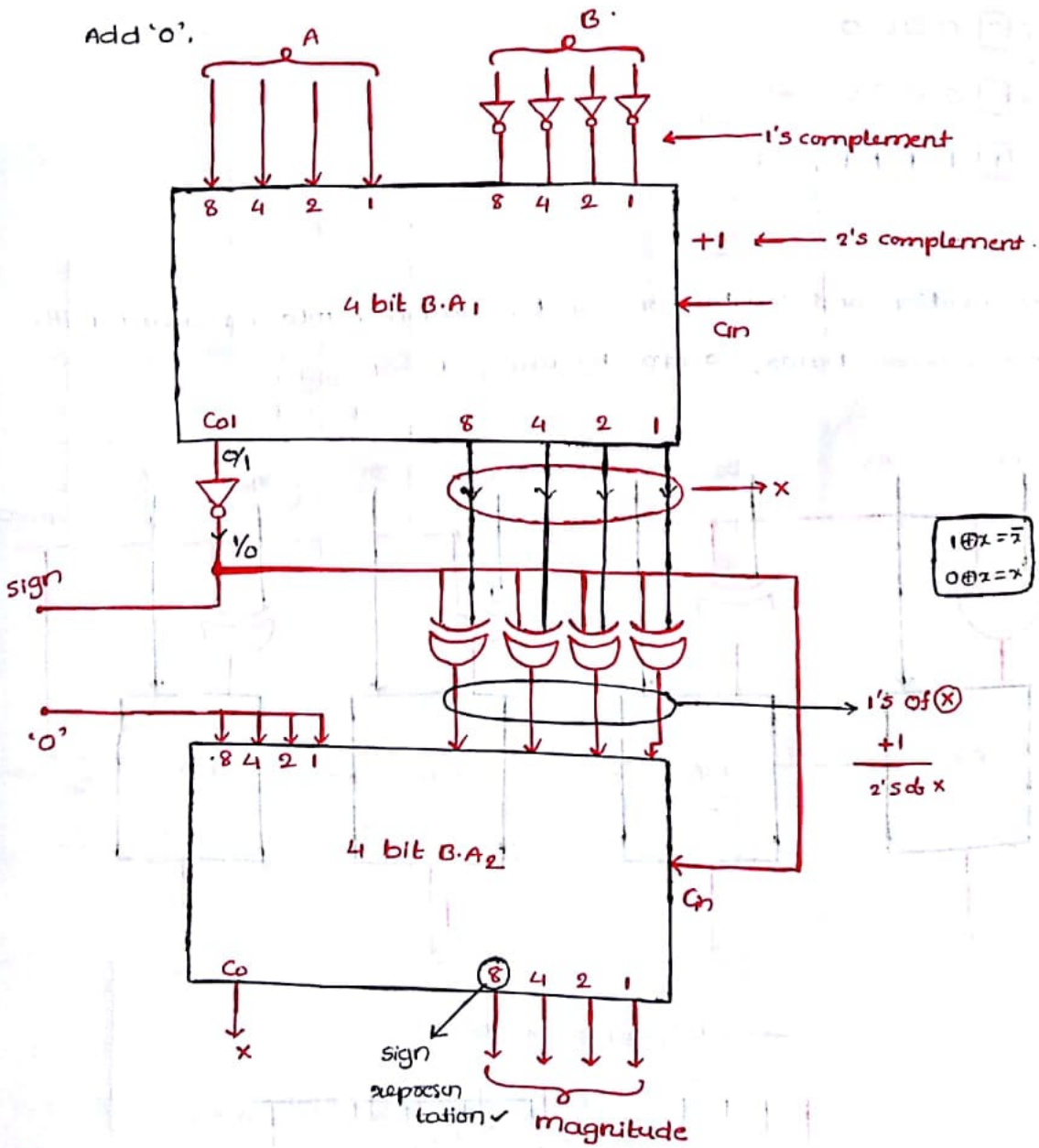
1) 2's complement circuit

2) 4 bit binary adder (i) (7483 → 4 bit binary parallel adder)

3) if $C_{o_1} = 0$ (i.e A<B)

2's compl. ckt → we need 4 bit B.A2

if $C_{o_1} = 1$ (i.e A>B)



10) By using 4 bit binary adder and other required minimum no. of logic gates realise.

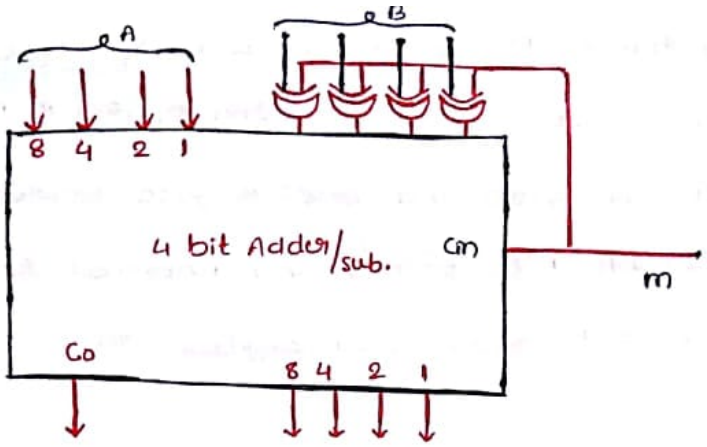
a) controllable 4 bit binary adder / binary subtractor.

b) controllable BCD to Excess-3 / Excess-3 to BCD code converter.

c) BCD to 9's code converter.

Sol

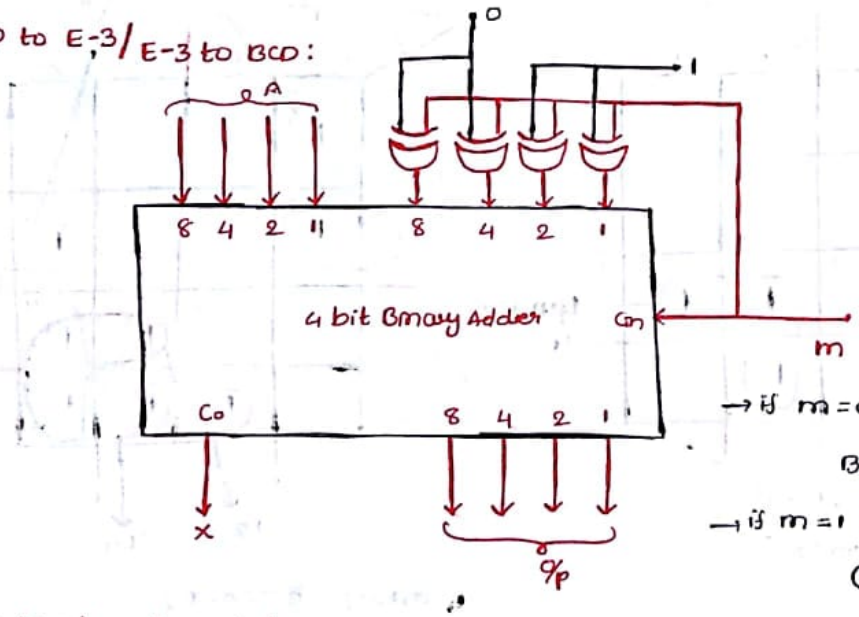
a)



→ if M=0 ; A+B. ⟶ Adder

→ if M=1 ; A+2's complement of B (A-B) ⟶ subtractor.

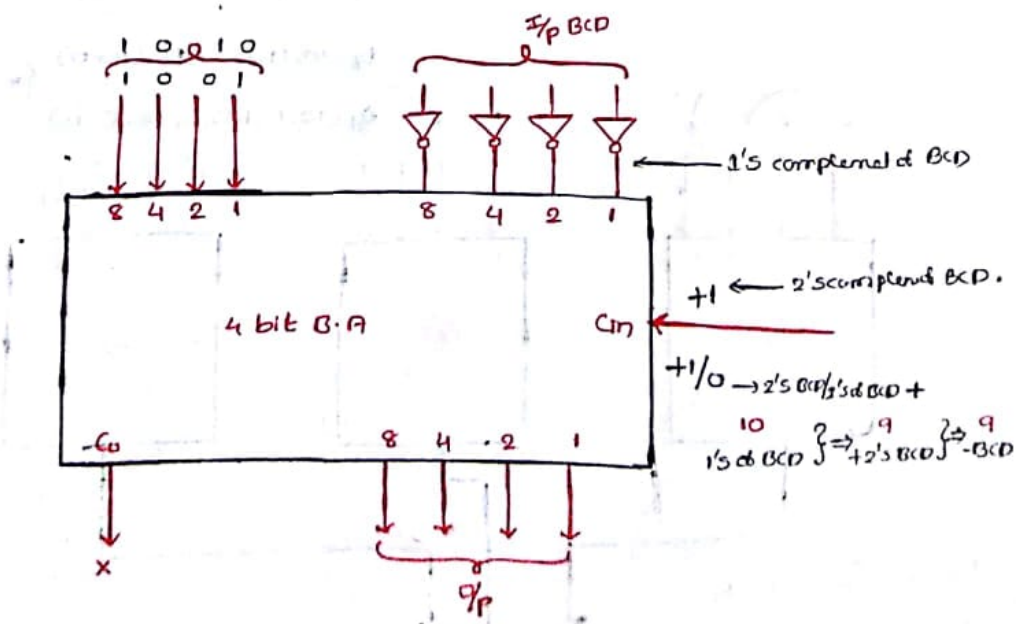
b) BCD to E-3/E-3 to BCD:



→ if m=0 i/p+3 :
BCD to E-3

→ if m=1 i/p + 2's comp. of 3
(I/p-3)
E-3 to BCD ✓

c) BCD to 9's code converter:



20) A LEMON GATE is having the o/p LEMON (w,x,y,z) = wy(x+z)

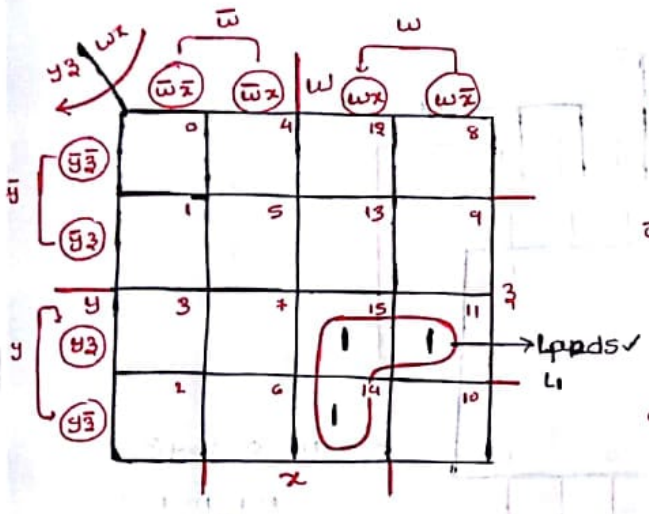
a) Realize the boolean function f(A,B,C,D) = Σm(0,1,6,9,10,11,14,15) by using three LEMON gates and one OR gate. Consider the variables are available both primed and unprimed form.

b) Identify is it possible to realize any digital circuit with LEMON/OR logic gates.

Sol:

LEMON (w,x,y,z) = wy(x+z)

a) f(A,B,C,D) = Σm(0,1,6,9,10,11,14,15)

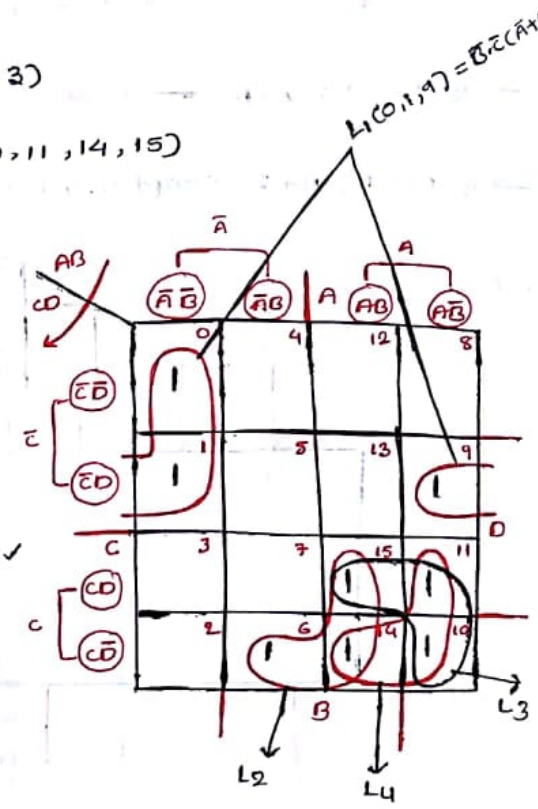


→ wxyz + wx̄yz + wxyz̄

→ wxy(z+z̄) + wȳyz

→ wy(x+z)

→ wy(x+z) = Σm(11,14,15)

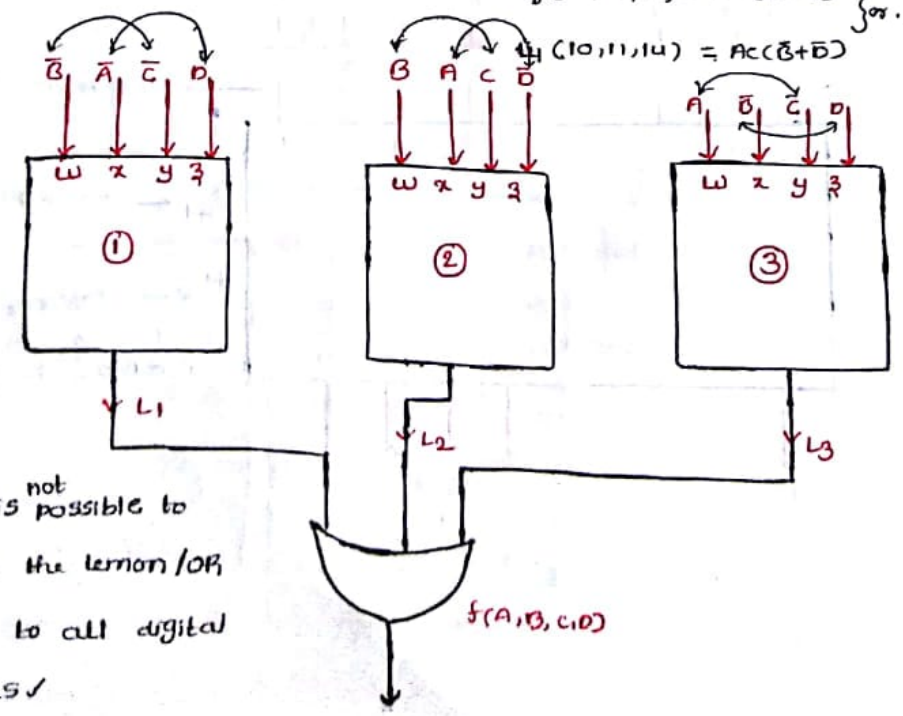


L4(0,1,9) = B̄·z̄(Ā+D)

L2(6,14,15) = BC(A+D̄)

L3(10,11,15) = AC(B̄+D) } or.

L4(10,11,14) = AC(B̄+D̄)



b) It is not possible to realize the lemon/OR logic to all digital circuits ✓

BCD Adder: It is a circuit that adds two BCD digits and produces a sum digit also in BCD

→ BCD numbers 0 to 9 (Total 10)

(0000 to 1001) — binary form.

→ 10 valid } in Decimal for BCD.
6 invalid }

Ex: 526

0101 0010 0110 → BCD representation

① A = 8 → 1000

+B = 5 → 0101

13 1101 ← Invalid BCD

correction factor is (6)

A = 1000

B = 0101

1101

1101

0110

0001 0011 → valid BCD.

1 3

carry 10's

② A = 9 → 1001

(+) B = 8 → +1000

17 10001 → Invalid BCD

0010

0001 0111 → valid BCD

1 7

carry 10's

③ A = 6 → 0110

(+) B = 2 → (+) 0010

8 1000 → Valid BCD so no need to apply correction factor.

→ Whenever invalid then only, add (6) / and perform BCD operation.

→ Whenever invalid at that place carry must take place.

→ Based on these possibilities we can design BCD adder circuit.

Developed a algorithm:

01. 4-bit binary Adder (BA1) ⟶ purpose to add ⑥ four bit binary nos.

02. checking circuit ⟶ purpose is it valid/invalid

if $f = 1$ ⟶ invalid

$f = 0$ ⟶ valid

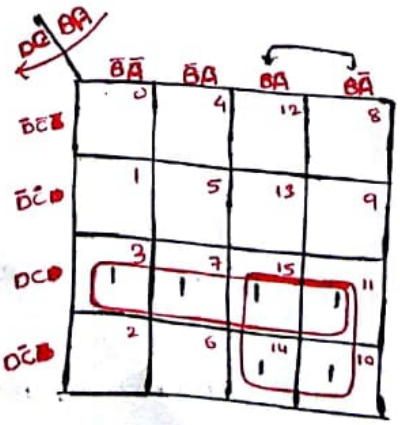
→ if $f = 1$ ⟶ operation BCD i.e Add ⑥

∴ so we need one more 4 bit binary Adder (BA_2)

→ if $f = 0$ ⟶ NO BCD operation

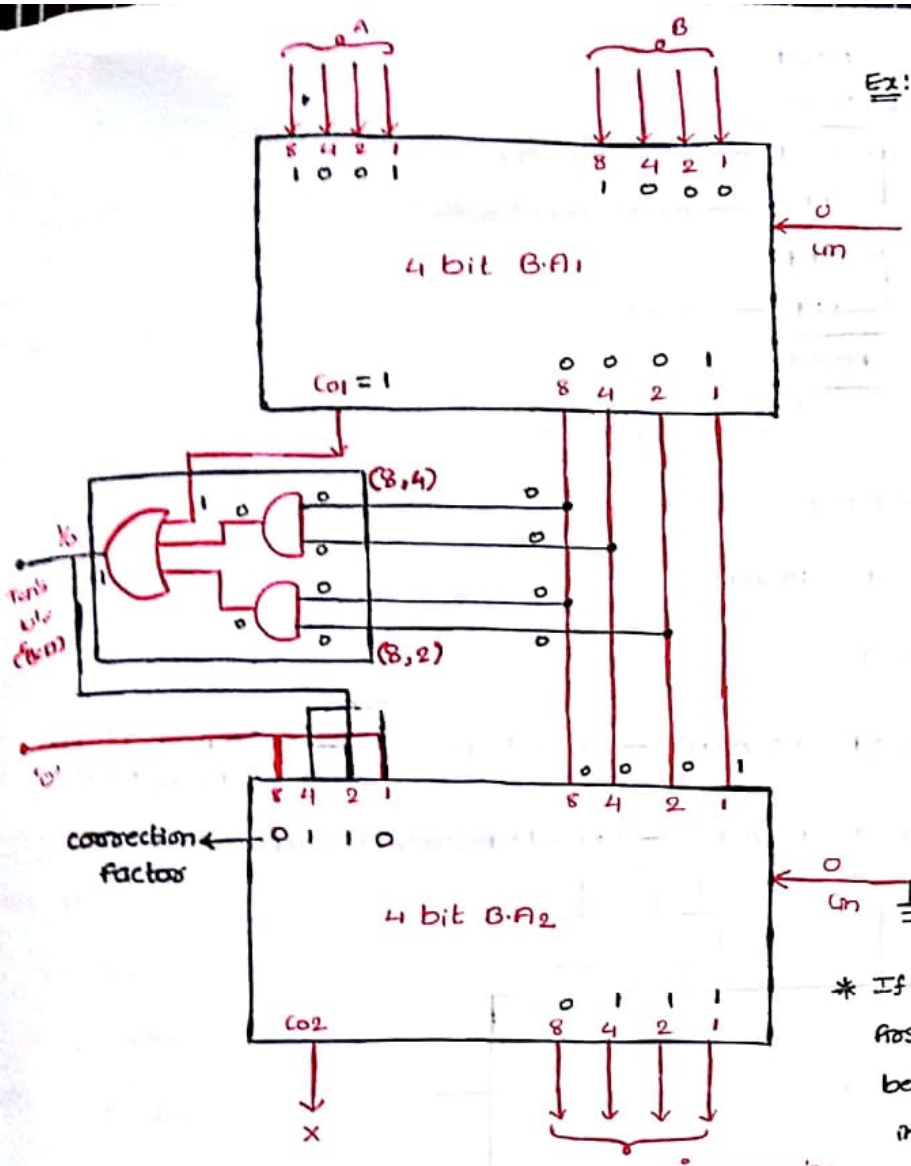
∴ So for reading operation only add BA_2 that means add zero operation.

D.N	I/p's				O/p
	D	C	B	A	f_1
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
⋮					⋮
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1



$f_1 = DC + DB$

$f_1 = D(B+C)$ ✓



Ex: A = 9
B = 8

* If carry generated in first addition that can be considered as it is in the 2nd addition also.

BCD subtractor:

i) **A>B:**

∴ A - B

∴ A + 9's complement of (B) ✓

A = 6 ⟶ 0110

-B = 4 ⟶ 0101 ⟶ 9's complement of (4)

2

1011 ⟶ Invalid BCD

0110 ⟶ correction factor. (6) ✓

(1) 0001

+1 ⟶ add end around carry (AEAC) ✓

0010 ⟶ Valid BCD

1001 - 9's comp
0100
0101 ✓

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

Ex:
13 ⟶ 9's complement (13) is 86 ✓

ii) **A<B:**

A = 4 ⟶ 0100

-B = 6 ⟶ 0011 ⟶ 9's complement of '6'

-2

0111 ⟶ Valid BCD
} - 9's complement of valid BCD.

1001
0111
1010

1010 ⟶ True magnitude

sign ✓

iii) A = 9 ⟶ 1001

−B = 1 → 9's complement ① 1000

8

① 0001 → Invalid BCD

0110 → correction factor.

① 0111

CBCD ✓ → +1 → AEAC

1000 ✓

→ True magnitude ✓

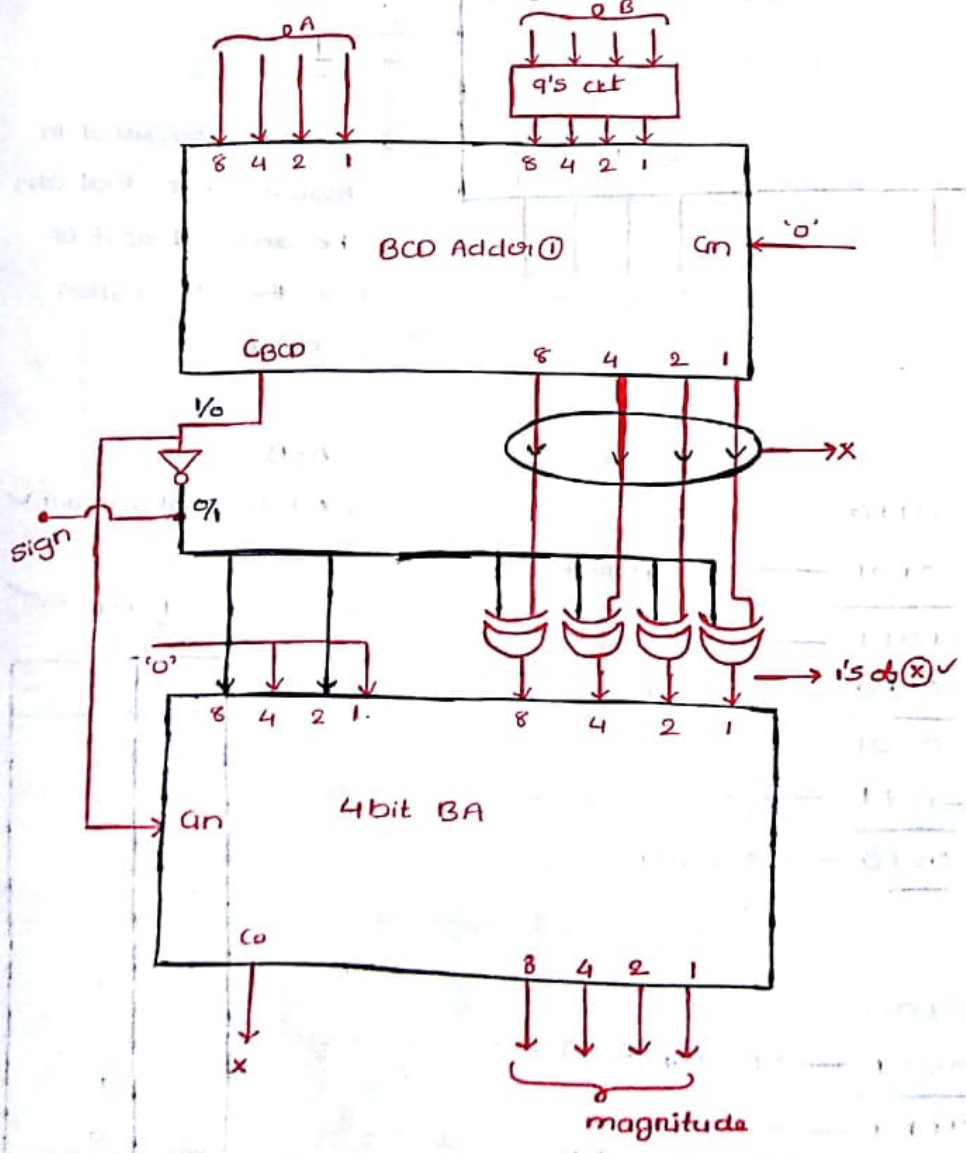
Developed a Algorithm:

01) 9's complement circuit

02) BCD Adder (BA)

03) If at all $C_{BCD} = 1$ (i.e A>B) ⟶ Add ① operation ⟶ so we need 4 bit binary Adder ✓

$C_{BCD} = 0$ (i.e A<B) ⟶ need to perform 9's complement form ✓



Comparision b/w Serial and Parallel Adder:

Serial Adder	Parallel Adder
01. Serial adder uses shift registers	01. Parallel adder uses registers with load capacity.
02. The serial adder requires only one full adder circuit.	02. The number of full-adder circuits in the parallel adder equal to the number of bits in the binary numbers.
03. The serial adder is sequential circuit	03. Excluding the registers, the parallel adder is a purely combinational ckt.
04. Time required for addition depends on number of bits.	04. Time required for addition does not depend on number of bits.
05. It is slower	05. It is Faster.

IES-2005.

10) 4-bit full adder takes 20ns to generate carry-out bit and 40ns for sum bit. what is the maximum rate of addition per second when four 1-bit full adders are cascade?

a) 10^7 ✓

b) 1.25×10^7

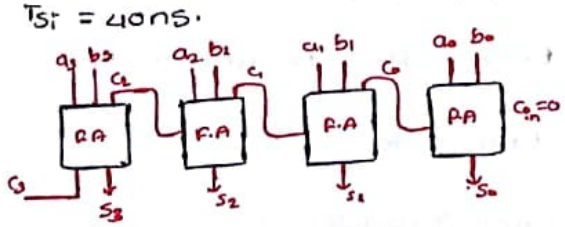
c) 6.25×10^6

d) 10^5.

Sol:

$T_{Ci} = 20ns$.

$T_{Si} = 40ns$.

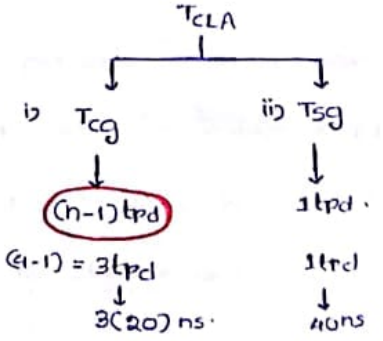


From figure. $T_{S_0} = 40ns$.

$T_{S_1} = (40+20)ns$

$T_{S_2} = (60+20)ns$

$T_{S_3} = (80+20)ns = 100ns$ ✓ final sum will take 100ns.



$\longrightarrow$ Rate of addition/sec $= \frac{1}{T_{S_3}} = \frac{1}{100ns}$

$R = 10^7$ ✓

Note: Given one is carry look a head adder. In this we have 2 stages. i) carry generation stage ii) sum generation stage. In this case sum will be performed at time that means sum required 1tpd and carry generation is (n-1) tpd.

Code converters:

01. BCD to Excess-3 code converter
02. Excess-3 to BCD code converter
03. 4 bit binary to Gray code converter
04. Gray to Binary code converter.
05. 5-4-2-1 BCD to Natural BCD code converter.
06. Natural BCD to 5-4-2-1 BCD code converter.

Q) What is the other name of Ex-3 code; why?

Ans) self complementary code.

Reason:

	BCD		Excess-3	
9's compliment of	5 → 0101	+3	1000	} 1's complement
	4 → 0100	-3	0111	
9's compliment of	7 → 0111	+3	1010	} 1's complement.
	2 → 0010	-3	0101	

→ The one's compliment of given BCD, Excess-3 code is 9's complement of the given BCD in the Excess-3 form. Because of this Excess-③ code is self complementary code.

Q) Where we are using these Excess-3 codes?

Ans) Ex-3 codes are used to design BCD subtractor circuits.

Q) Design and realise BCD to Excess-3 code converter?

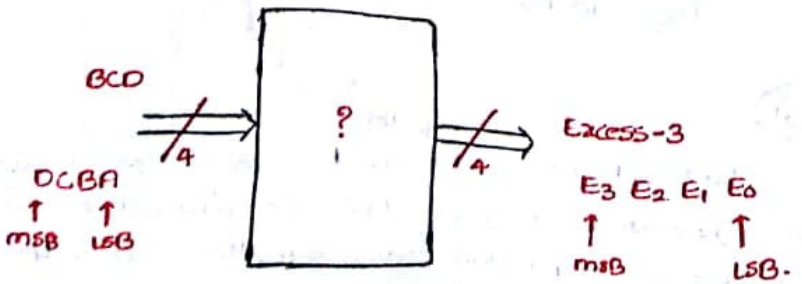
Sol:

01) BCD to Excess-3 code converter:

BCD + 3 = Excess-3

BCD ⇒ {0, 1, 2, ... 9}

E-3 ⇒ {3, 4, 5, ... 12}



Decimal	$B_3=D$	$B_2=C$	$B_1=B$	$B_0=A$	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	0
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

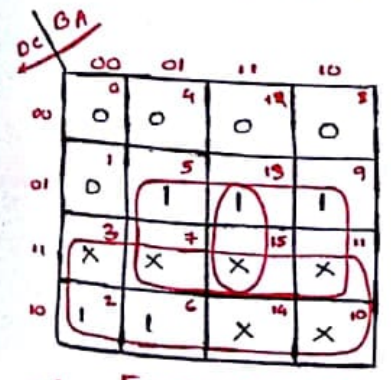
K-map simplification:

$E_3(D,C,B,A) = \Sigma m(5,6,7,8,9) + d(10,11,12,13,14,15)$ ⟶ ①

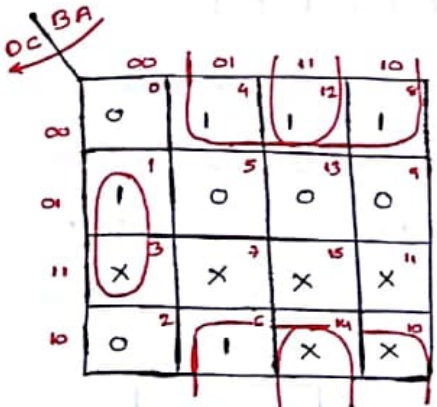
$E_2(D,C,B,A) = \Sigma m(1,2,3,4,9) + d(10,11,12,13,14,15)$ ⟶ ②

$E_1(D,C,B,A) = \Sigma m(0,3,4,7,8) + d(10,11,12,13,14,15)$ ⟶ ③

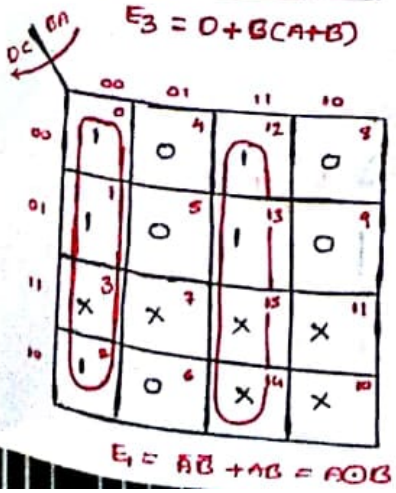
$E_0(D,C,B,A) = \Sigma m(0,2,4,6,8) + d(10,11,12,13,14,15)$ ⟶ ④



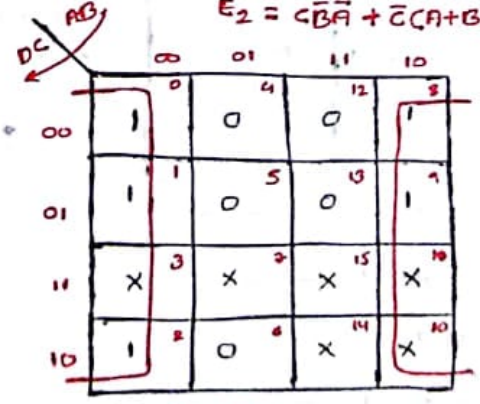
$E_3 = D + B(A+B)$



$E_2 = C\bar{B}\bar{A} + \bar{C}(A+B)$



$E_1 = A\bar{B} + \bar{A}B = A \oplus B$



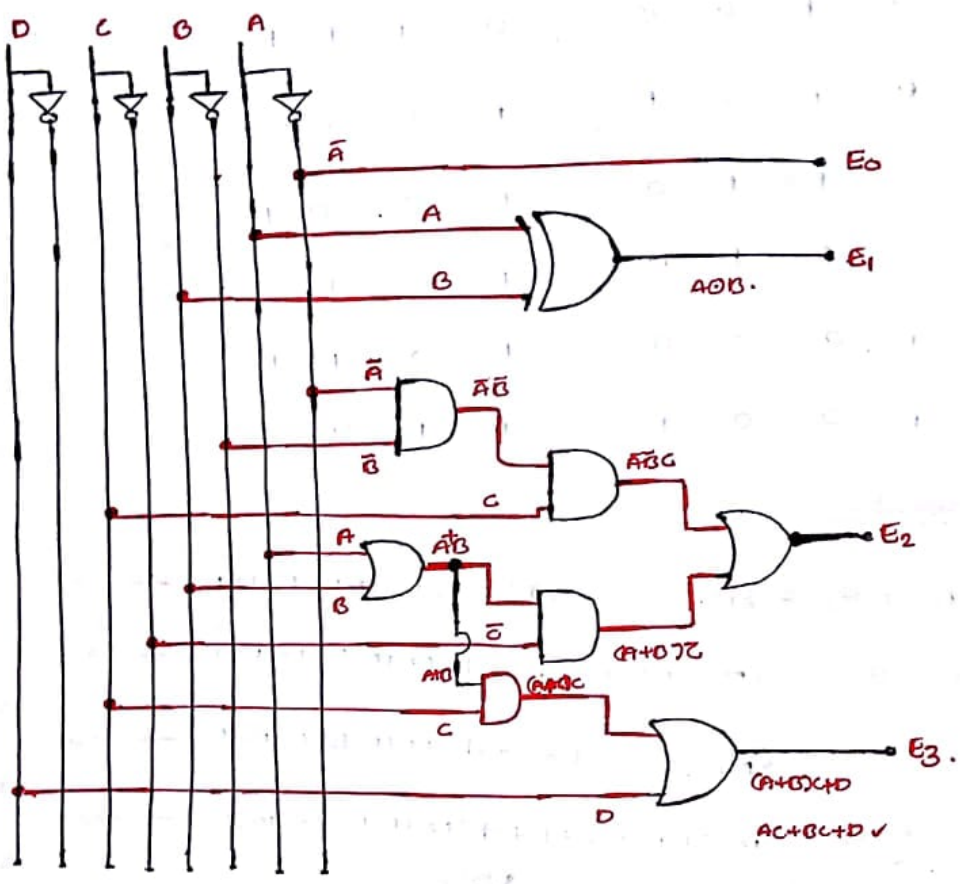
$E_2 = \bar{A}$

$E_0 = \bar{A}$ ⟶ (1')

$E_1 = A \odot B$ ⟶ (2')

$E_2 = C\bar{B}\bar{A} + \bar{C}A + \bar{C}B$ ⟶ (3')

$E_3 = D + CA + CB$ ⟶ (4')

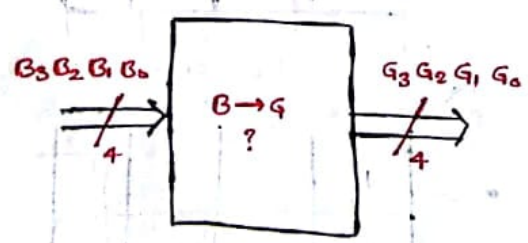


similar way Ex-3 to BCD ✓

3) 4 bit binary to Gray code convertor:

$B_{n-1}, B_{n-2}, \ldots\ldots, B_1, B_0$

$G_{n-1}, G_{n-2}, \ldots\ldots, G_1, G_0$.

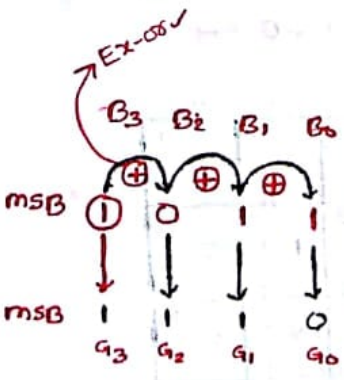


$G_3 = B_3$ ⟶ (1)

$G_2 = B_3 \oplus B_2$ ⟶ (2)

$G_1 = B_2 \oplus B_1$ ⟶ (3)

$G_0 = B_1 \oplus B_0$ ⟶ (4)



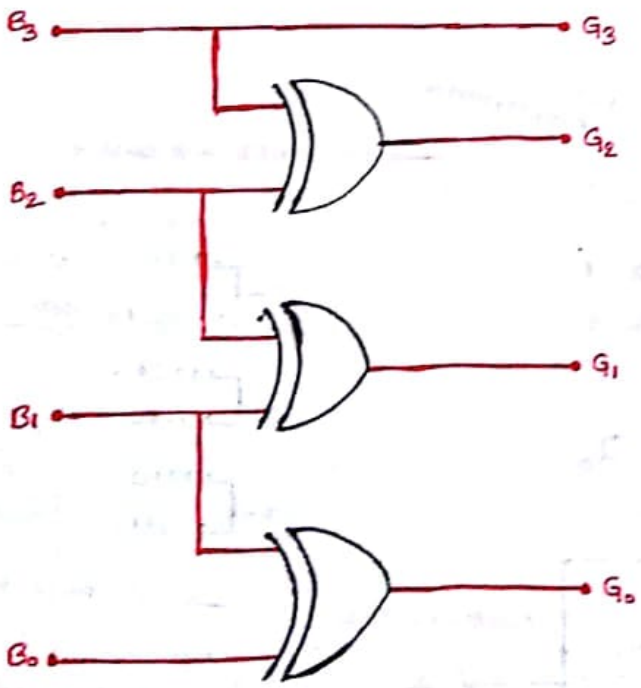
D.NO	B_2	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
⋮								
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0

→ They are differ only one bit

Note: Any two succeeding binary number equivalent Gray code are differ in only one bit, because of this Gray code is called "cyclic code".

Application of Gray code:

→ Gray codes are used in digital and data communication circuits for detection and correction purpose.



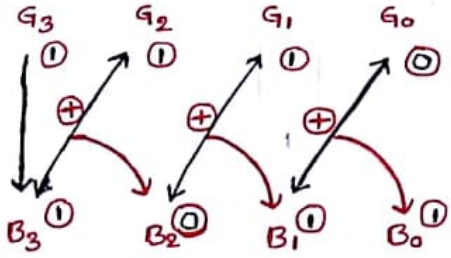
4 bit Gray to Binary code converter:

$$g_{n-1} = B_{n-1}$$

$$g_i = B_{i+1} \oplus B_i$$

$$i \neq n-1$$

4 bit Gray to Binary code converter:

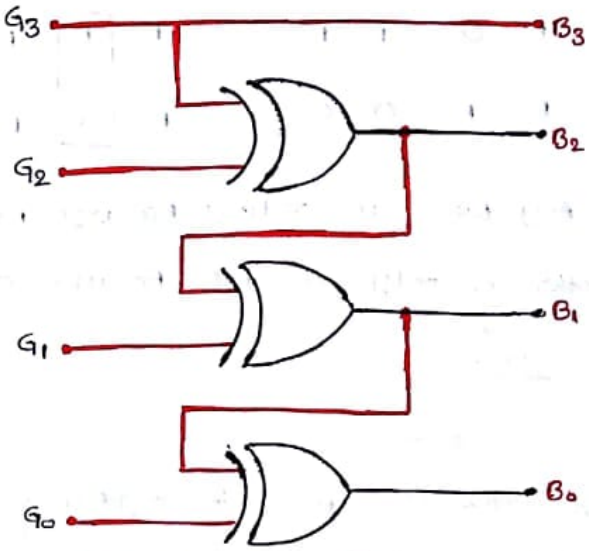


$B_3 = G_3$ ——→ ①

$B_2 = G_3 \oplus G_2$ } ——→ ②
$B_3 \oplus G_2$ }

$B_1 = G_3 \oplus G_2 \oplus G_1$ } ——→ ③
$B_2 \oplus G_1$ }

$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0$ } ——→ ④
$= B_1 \oplus G_0$ }



*
$B_{n-1} = G_{n-1}$

$B_i = B_i \oplus G_i$

$i = n-1$

→ Name it self 5421 BCD code ✓

5421 BCD to Natural BCD code:

→ 1996 EEE Gate question

5	4	2	1
X_3	X_2	X_1	X_0

8	4	2	1
D	C	B	A
2^3	2^2	2^1	2^0

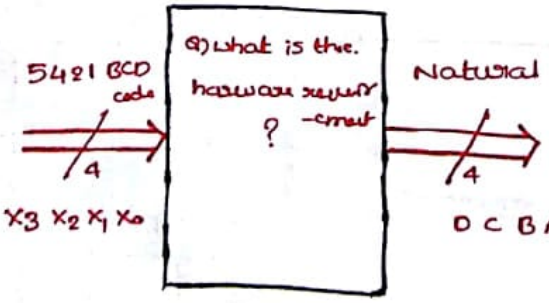
$5X_3 + 4X_2 + 2X_1 + 1X_0 = C\ B_0$

5421
$X_3 X_2 X_1 X_0$

5 → 1000 ✓ ①✓
0101 ✗ ②✗

6 → 1001 ✓
0110 ✗

7 → 1010 ✓
0111 ✗



→ with these three numbers.
→ no problem either ① or ② one is used

→ We must use first one why because 2nd one is related to BCD(8421) code, suppose we can consider 2nd one there is no difference to identify 5421 code.

Decimal no.		x_3	x_2	x_1	x_0	D	C	B	A
0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	0	1
2	2	0	0	1	0	0	0	1	0
3	3	0	0	1	1	0	0	1	1
4	4	0	1	0	0	0	1	0	0
5	5	0	1	0	1	0	1	0	1
6	6	0	1	1	0	0	1	1	0
7	7	0	1	1	1	0	1	1	1
8	8	1	0	0	0	1	0	0	0
9	9	1	0	0	1	1	0	0	1
10	7	1	0	1	0				
11.	8	1	0	1	1				
12.	9	1	1	0	0				

Ex: 2-4-2-1 BCD code

$x_3\ x_2\ x_1\ x_0$

(8) → 1 0 1 1 ✓ → (11) ✓

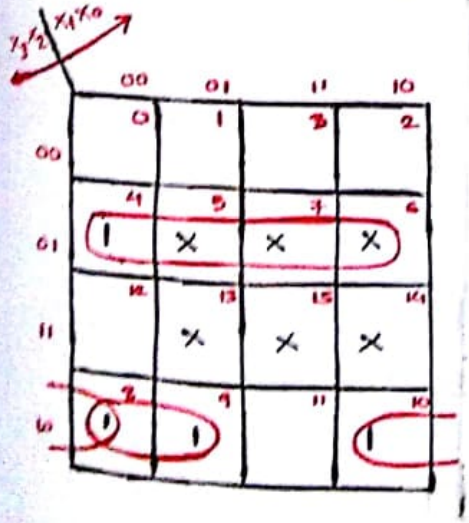
0 1 0 1 ×

$D(x_3, x_2, x_1, x_0) = \sum m(11,12) + d(5,6,7,13,14,15)$

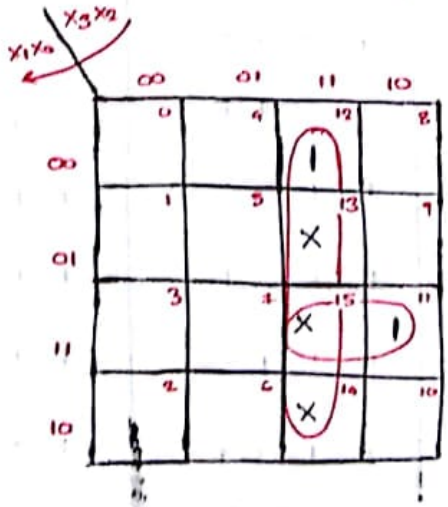
$C(x_3, x_2, x_1, x_0) = \sum m(4,8,9,10) + d(5,6,7,13,14,15)$

$B(x_3, x_2, x_1, x_0) = \sum m(2,3,9,10) + d(5,6,7,13,14,15)$

$A(x_3, x_2, x_1, x_0) = \sum m(1,3,6,10,12) + d(5,6,7,13,14,15)$



$C = \bar{x}_3 x_2 + x_3 \bar{x}_2 \bar{x}_1 + x_3 \bar{x}_2 \bar{x}_0$



$D = x_3x_2 + x_1x_0x_3$.

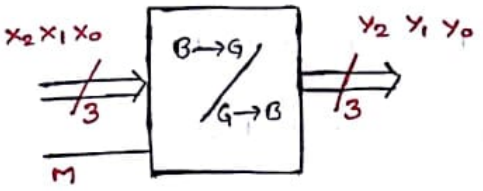
Similarly

$B = x_3\bar{x}_2x_1 + \bar{x}_2x_1\bar{x}_0 + x_3\bar{x}_1x_0$

$A = \bar{x}_3\bar{x}_2x_0 + x_3\bar{x}_1\bar{x}_0 + x_3\bar{x}_2\bar{x}_0$

Q) Design and realize controllable 3-bit binary to Gray code / Gray to Binary code converter?

Sol:



if $M=0 \Rightarrow B \rightarrow G$

if $M=1 \Rightarrow G \rightarrow B$.

$y_2(M, x_2, x_1, x_0)$

$y_1(M, x_2, x_1, x_0)$

$y_0(M, x_2, x_1, x_0)$

D.NO.	M x_2 x_1 x_0	y_2 y_1 y_0
0	0 0 0 0	0 0 0
1	0 0 0 1	0 0 1
2	0 0 1 0	0 1 1
3	0 0 1 1	0 1 0
4	0 1 0 0	1 1 0
5	0 1 0 1	1 1 1
6	0 1 1 0	1 0 1
7	0 1 1 1	1 0 0
8	1 0 0 0	0 0 0
9	1 0 0 1	0 0 1
10	1 0 1 0	0 1 1
11	1 0 1 1	0 1 0
12	1 1 0 0	1 1 1
13	1 1 0 1	1 1 0
14	1 1 1 0	1 0 0
15.	1 1 1 1	1 0 1

B→G

G→B

$y_2(M, x_2, x_1, x_0) = \Sigma m(4,5,6,7,12,13,14,15)$ ——→ ①

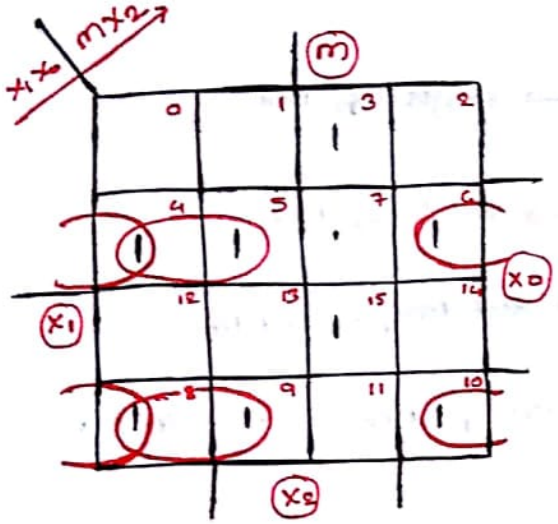
$y_1(M, x_2, x_1, x_0) = \Sigma m(2,3,4,5,10,11,12,13)$ ——→ ②

$y_0(M, x_2, x_1, x_0) = \Sigma m(1,2,5,6,9,10,12,15)$ ——→ ③

$Y_2 = X_2$ ⟶ (A)

$Y_1 = X_2 \oplus X_1$ ⟶ (B)

$Y_0 = mX_2 \oplus X_1 \oplus X_0$ ⟶ (C)



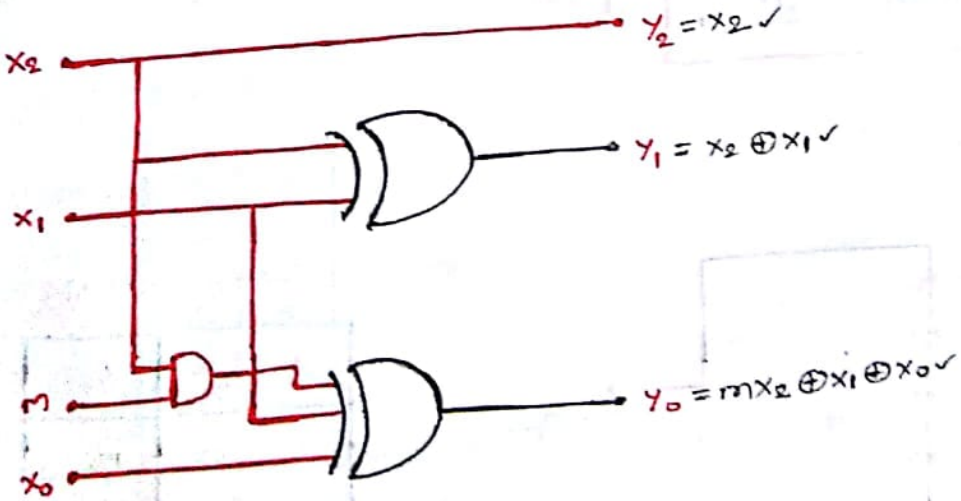
$Y_0 = \bar{m}\bar{X_1}X_0 + \bar{m}X_1\bar{X_0} + \bar{X_2}\bar{X_1}X_0 + \bar{X_2}X_1\bar{X_0} + mX_2\bar{X_1}\bar{X_0} + mX_2X_1X_0$

$= \bar{m}[X_1 \oplus X_0] + \bar{X_2}[X_1 \oplus X_0] + mX_2[\overline{X_1 \oplus X_0}]$

$= X_1 \oplus X_0[\bar{m}\bar{X_2}] + mX_2[\overline{X_1 \oplus X_0}]$ $[\because \bar{m} + \bar{X_2} = \overline{mX_2}$ - Demorgan's$]$

$= mX_2 \oplus (X_1 \oplus X_0)$ ⟶ Associated with EX-OR gate

$= mX_2 \oplus X_1 \oplus X_0$



MULTIPLEXER

→ The "Multiplexer" is a special versatile combinational circuit

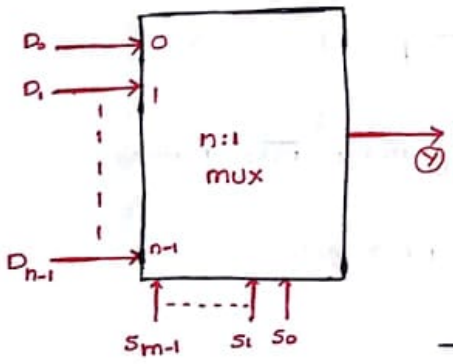
→ It is a digital combinational circuit that selects binary information from one of many inputs and directs it to a single o/p line.

i.e many i/p lines ⟶ single o/p line.

I/p lines (2^n) ⟶ single o/p line

where (n) ⟶ select line/ control line.

→ The selection of a particulars i/p line is controlled by a set of selection lines.



(n) (m)

2 to 1 mux 1

4 × 1 mux 2 $2^m = n$

8 : 1 mux 3

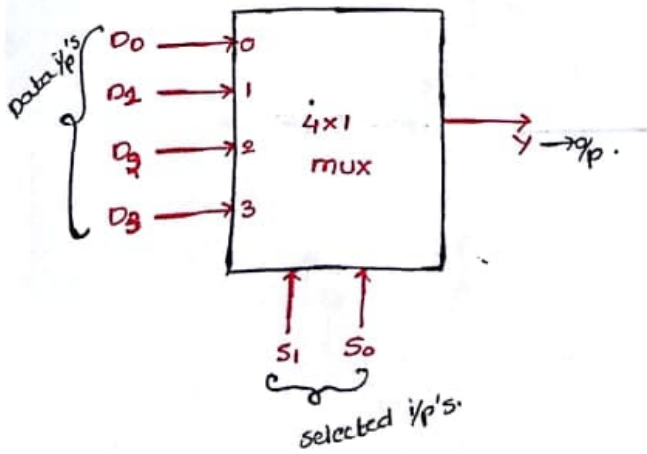
10 to 1 mux 4 ⟶ $2^m > n$ ✓

16 to 1 mux 4 ⟶ $2^m = n$ ✓

→ These are readily available in the IC' forms

condition: $2^m \geq n$ must satisfy

4x1 mux:

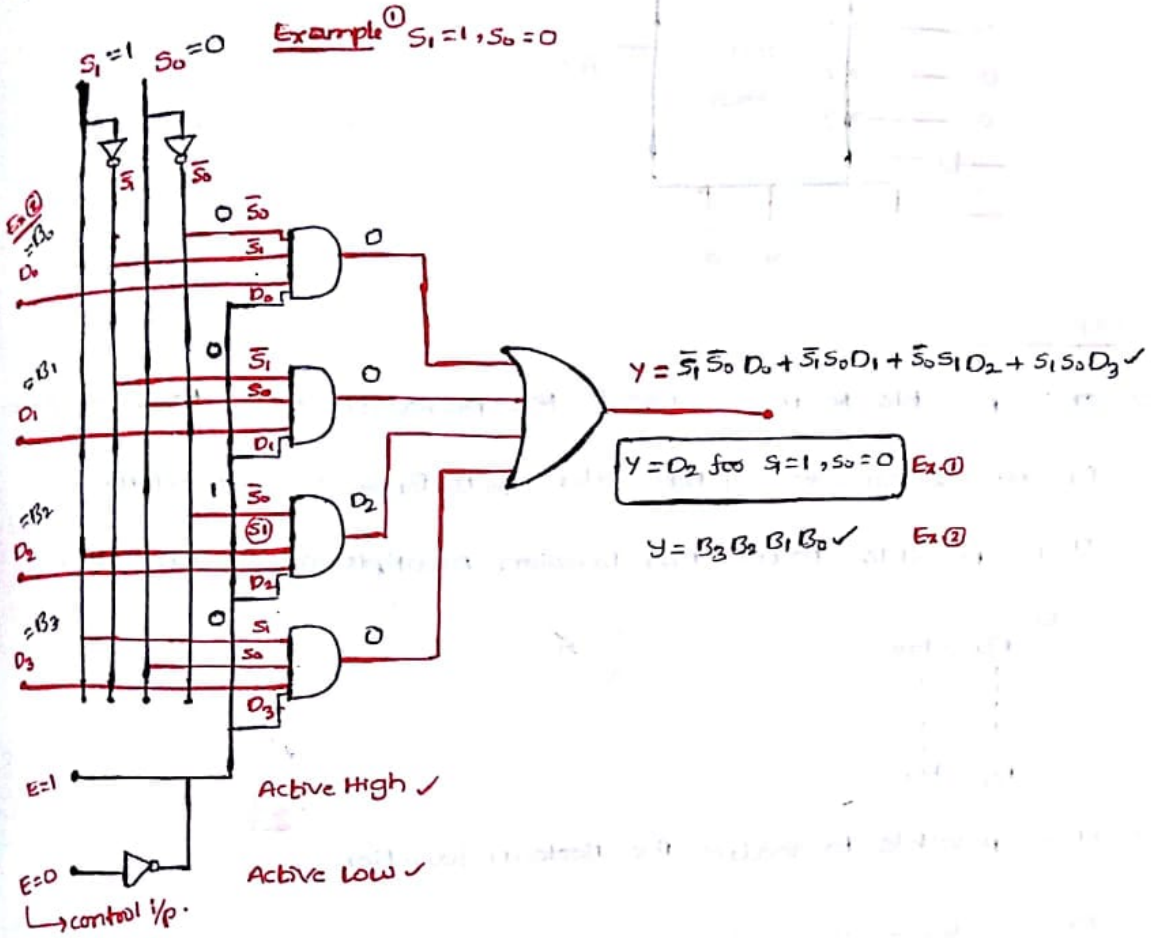


S_1	S_0	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$Y = \bar{S_1}\,\bar{S_0}\,D_0 + \bar{S_1}\,S_0\,D_1 + S_1\,\bar{S_0}\,D_2 + S_1\,S_0\,D_3$$

→ MUX contains AND gate followed by an OR-gate.

→ why because it is SOP (sum of product) so that we can implement the logic circuit by using AND-OR ✓ logic.



→ Multiplexer can also named as

01. Data selector
02. Many-to-one circuit
03. Universal logic gate ckt
04. Parallel to serial converter
05. Waveform generator.

→ the operation of the circuit is controlled by clock signal.

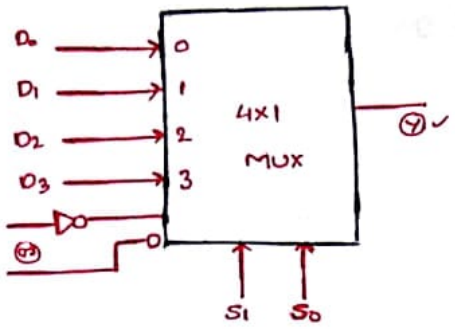
→ control i/p names,

01. Strobe
02. Inhibit ⟶ means not in a position accept any i/p
03. Disable
04. Enable

→ Whenever enable signal is selected ⟶ the ckt is activated.

Q) Enable signal '0' even though circuit is active how?

Ans) connecting the Not gate



Applications:

01. It is possible to use convert the parallel data to serial form

Ex: Let us consider 4 bit data $B_3 B_2 B_1 B_0$ in above 4x1 mux.

02. It is possible to use Time Division Multiplexing switch (TDM)

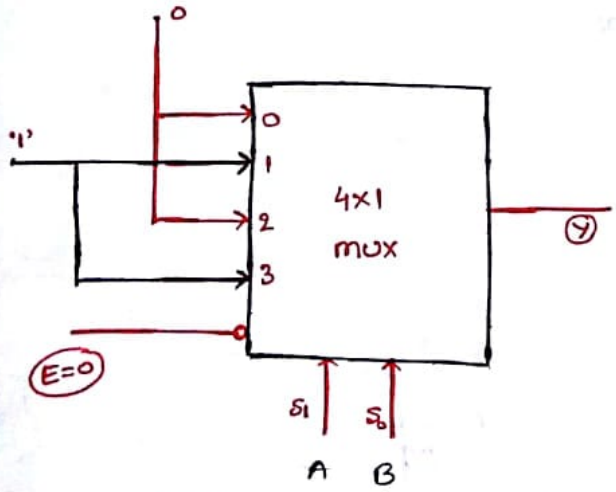
Ex:

$D_0 = P_{B0}$

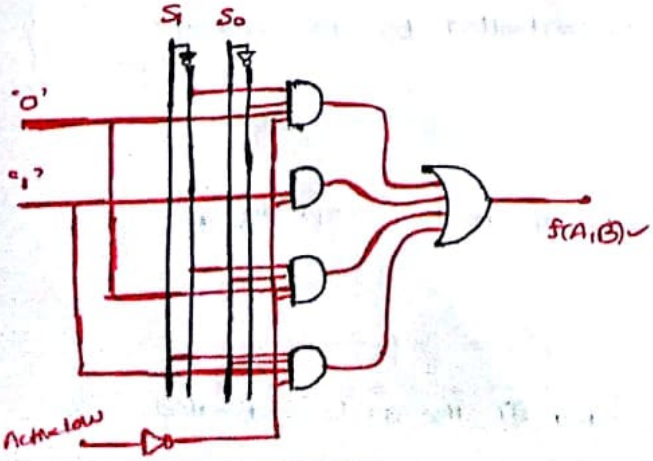
$D_3 = P_{B3}$

03. It is possible to realize the Boolean function.

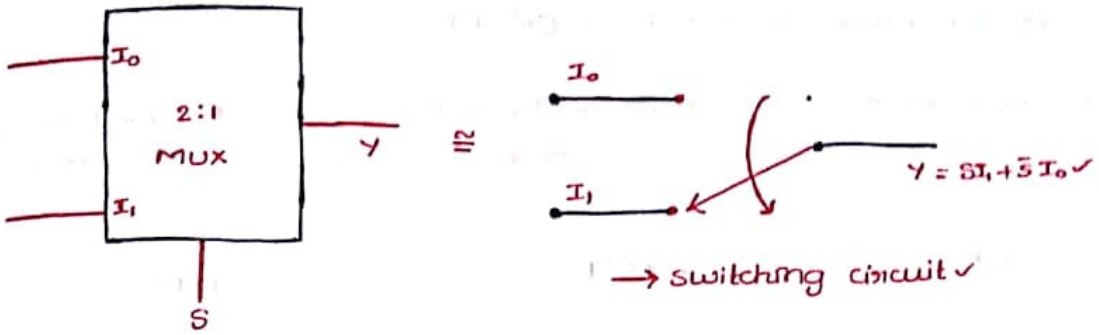
Ex: $f(A,B) = \Sigma m(1,3)$



S_1/A	S_0/B	f(A,B)	
0	0	0	→ D_0
0	1	1	→ D_1
1	0	0	→ D_2
1	1	1	→ D_3



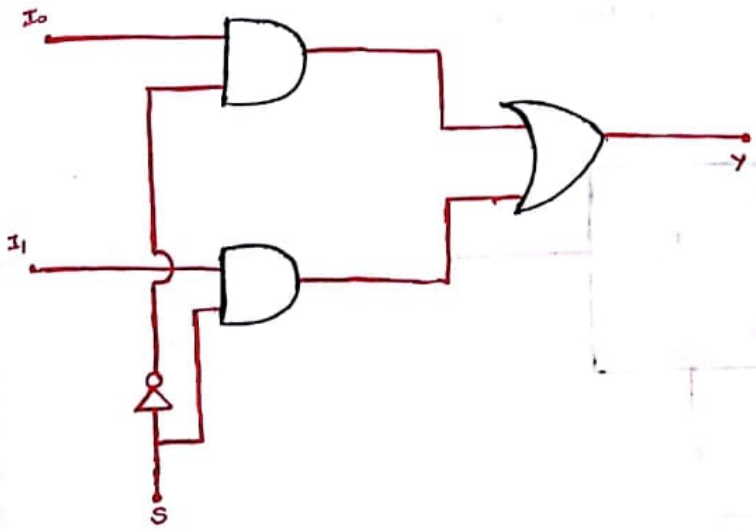
2:1 MUX:



→ switching circuit ✓

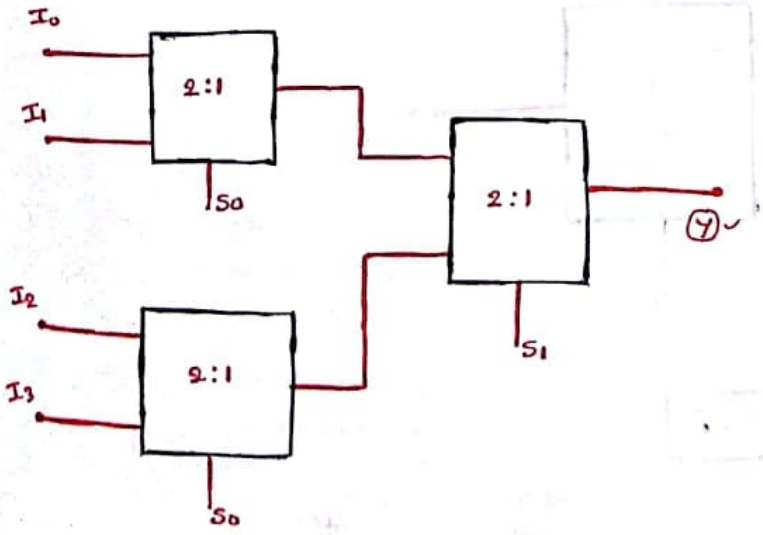
$$\therefore Y = \bar{S}I_0 + SI_1$$

logic diagram:



→ Implementation of higher order MUX using lower order MUX.

* 4:1 mux by 2:1 MUX.



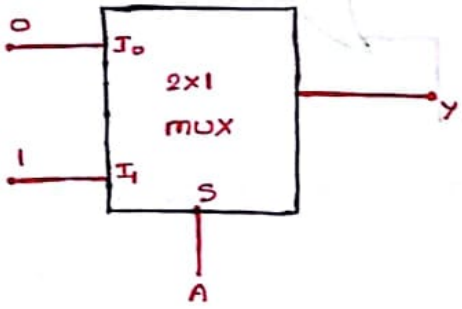
Total number of 2:1 MUX = (3) ✓

→ To implement $2^n:1$ mux by using 2:1 mux, the total number of 2:1 mux required is (2^n-1)

* Given MUX	To be implemented MUX	Required no. of MUX
4:1	16:1	4+1 = 5
4:1	64:1	16+4+1 = 21
8:1	64:1	8+1 = 9
8:1	256:1	32+4+1 = 37

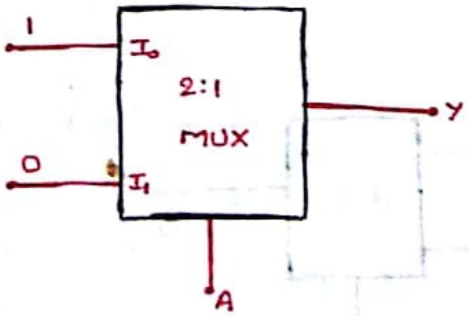
→ MUX as an Universal logic gate:

01. Buffer:



$Y = A$

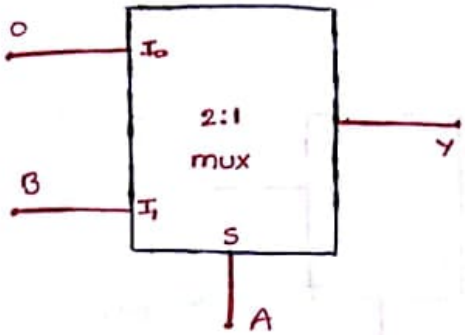
02. NOT-gate/Inverter:



$Y = \bar{A}$

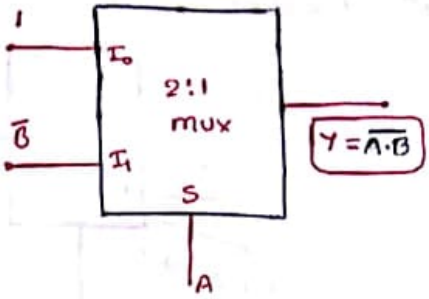
03. AND-gate:

$Y = AB$



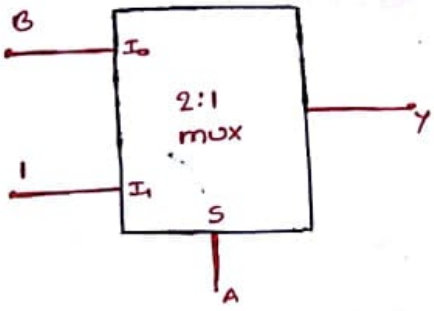
$Y = AB$

03) NAND gate:



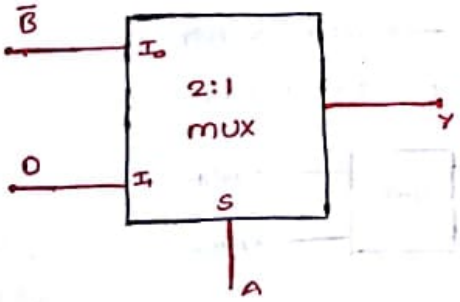
$Y = \overline{A \cdot B}$

04. OR-gate:



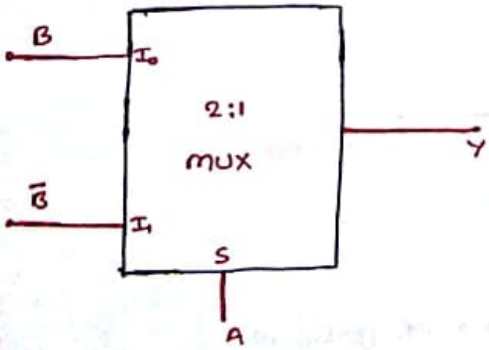
$Y = A + B$

05. NOR-gate:



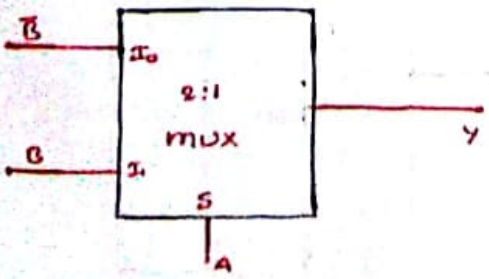
$Y = \overline{A+B}$

06) Ex-OR gate:



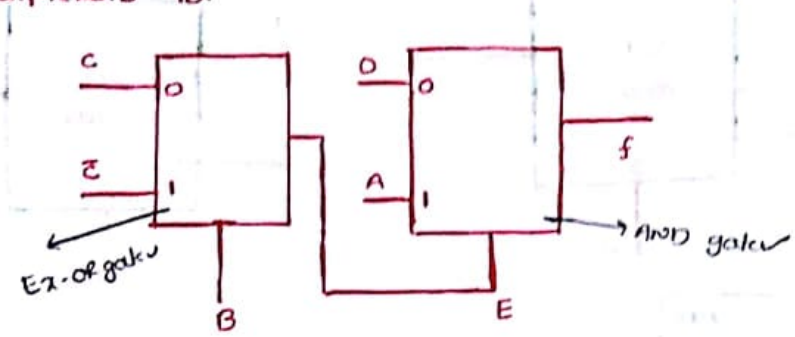
$Y = \bar{A}B + A\bar{B} = A \oplus B$

07. Ex-NOR gate:



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

Ex: The Boolean function (f) implemented in the figure using (2) 1/p GATE multiplexers is.



a) $A\bar{B}C + AB\bar{C}$

b) $ABC + AB\bar{C}$

c) $\bar{A}BC + \bar{A}\bar{B}\bar{C}$

d) $\bar{A}BC + \bar{A}B\bar{C}$

Sol: E = o/p of 1st mux

$E = B \oplus C$

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

② How many minimum number of 2:1 mux required for implem the Half Adder (H.A)

a) 2

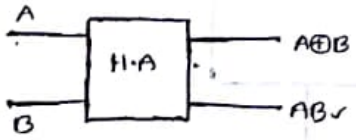
b) 3

c) 4

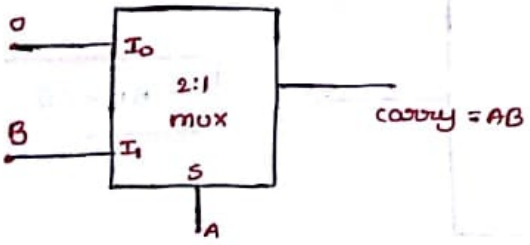
d) None of these.

Sol: H.A → sum = $A \oplus B$

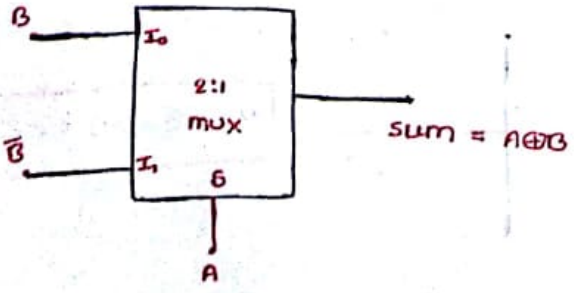
carry = AB



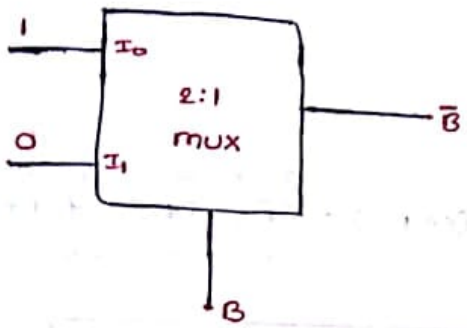
→ for carry, we required an AND-gate as.



→ For SUM, we required an Ex-OR gate as,



→ But we don't available $\bar{B}$ as i/p so we require an inverter circuit also as;



∴ finally to implement a H.A. we required 3, 2:1 mux

for SUM ⟶ ②✓

CARRY ⟶ ①✓

③ (Q-09) What are the minimum no. of 2-to-1 multiplexers required to generate a ②-i/p AND gate and a 2-i/p Ex-OR gate ?

a) 1 and 2

b) 1 and 3.

c) 1 and 1

d) 2 and 2

Sol: AND gate ⟶ required ⟶ 2:1 mux ①

Ex-OR gate ⟶ required ⟶ ②

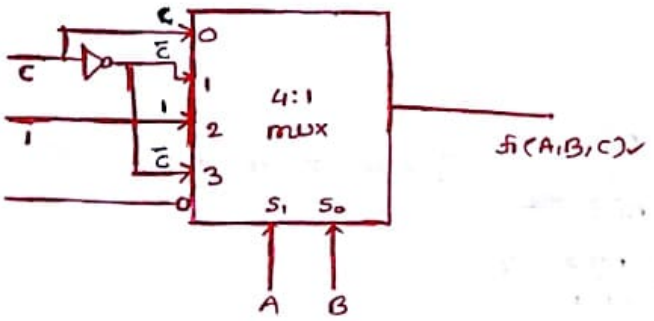
↓

inverting o/p is there

So we need inverter (mux)

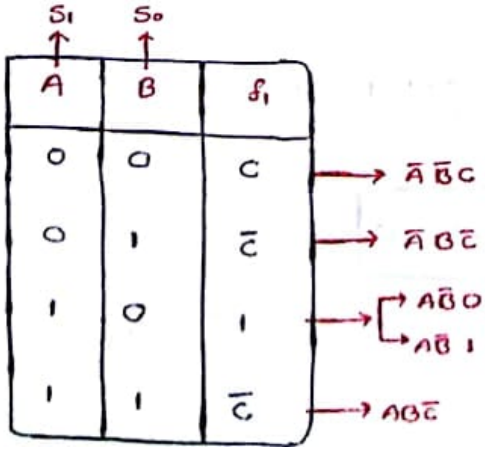
04. Identify the function $f_1(A,B,C)$ Realised by the given 4x1 mux.

model (i)



Sol:

Post method:



2nd method:

D.NO	A	B	C	f_1
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

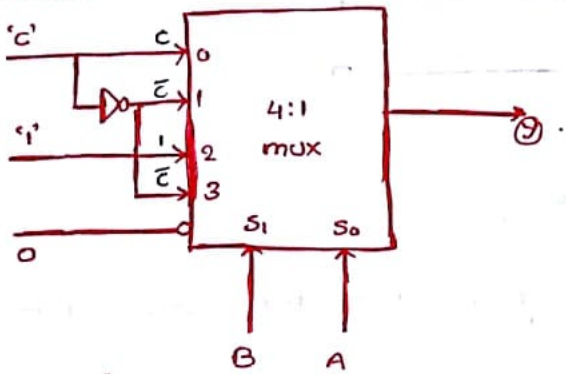
001 010 10-0 10-1 110

$f_1(A,B,C) = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\cdot 1 + AB\bar{C}$

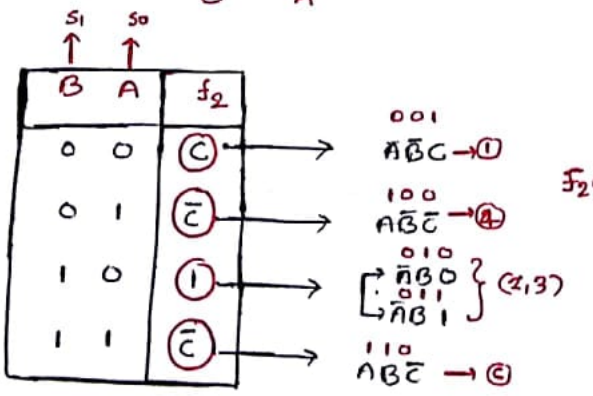
1 2 (4,5) 6

$f_1(A,B,C) = \Sigma m(1,2,4,5,6)$

5Q) Find the function $f_2(A,B,C)$ realised by given 4x1 multiplexer model(s)



Sol:



001 $\bar{A}\bar{B}C \rightarrow$ (1)

100 $A\bar{B}\bar{C} \rightarrow$ (4)

010 $\bar{A}B0$ } (2,3)

011 $\bar{A}B1$ }

110 $AB\bar{C} \rightarrow$ (6)

$f_2(ABC) = \Sigma m(1,2,3,4,6)$

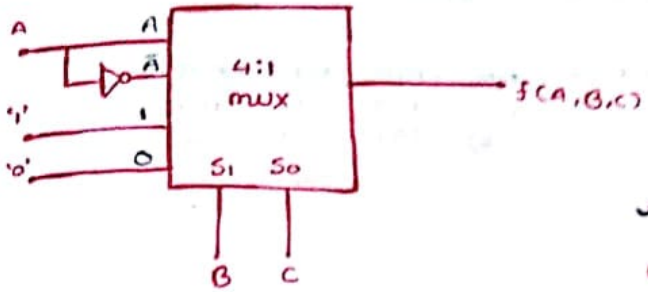
(or)

$f_2(A,B,C) = \bar{A}\bar{B}C + A\bar{B}C + \bar{A}B\cdot 1 + AB\bar{C}$

1 4 (2,3) 6

$f_2(ABC) = \Sigma m(1,2,3,4,6)$

6a) the o/p of given mux. equals to?



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

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

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

100 001 110 010

$f(A,B,C) = \Sigma m(1,2,4,6)$

a) $f(A,B,C) = \Sigma m(1,2,4,6)$

b) $f(A,B,C) = \Sigma m(1,2,6)$

c) $f(A,B,C) = \Sigma m(2,4,5,6)$

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

@

B	C	f
0	0	A ✓
0	1	Ā ✓
1	0	1 ✓
0	0	0

7a) Find $f_3(C,B,A)$ for '3a' diagram?

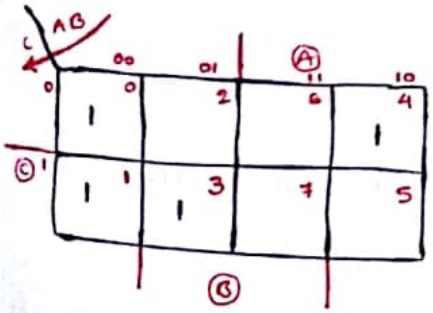
$f_3(C,B,A) = C\cdot\bar{B}\cdot\bar{A} + \bar{C}\cdot\bar{B}\cdot A + 1\cdot B\cdot\bar{A} + \bar{C}\cdot B\cdot A$

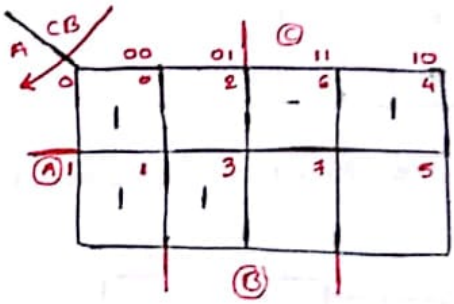
4 1 (2,6) 3✓

f_3	S_1	S_0
	B	A
C	0	0
$\bar{C}$	0	1
1	1	0 ✓
$\bar{C}$	1	1

$f_3(C,B,A) = \Sigma m(1,2,3,4,6)$

→





$f_2(A,B,C) = \bar{A}C + \bar{B}\bar{C}$

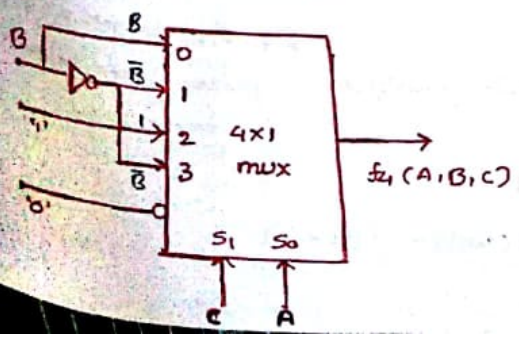
$= \Sigma m(0,1,3,4)$

$f_4(C,B,A) = A\bar{B} + \bar{A}\bar{B}$

$= \Sigma m(0,1,3,4)$

→ Looking the order of variable is different / changing → then minterms same but function change.

8a) Find the function $f_4(A,B,C)$ realized by given 4x1 multiplexer. model (iv).



S_1	S_0	f
0	0	B
0	1	$\bar{B}$
1	0	1
1	1	B

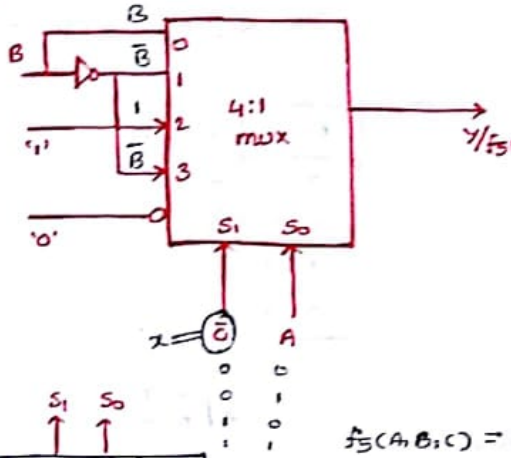
$f_4(A,B,C) = \Sigma m(1,2,3,4,5)$

$f_4(A,B,C) = \bar{A}B\bar{C} + A\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}C$

010 (2) 100 (4) 011 001 (3,1) 101 (5)

qq) find $f_5(A,B,C)$

Model ⓥ



$x \Rightarrow \bar{C}$

$\bar{x} \Rightarrow C$

S_1	S_0	f
ⓧ = $\bar{C}$	A	
1	0	B_0
1	1	D_1
0	0	D_2
0	1	D_3

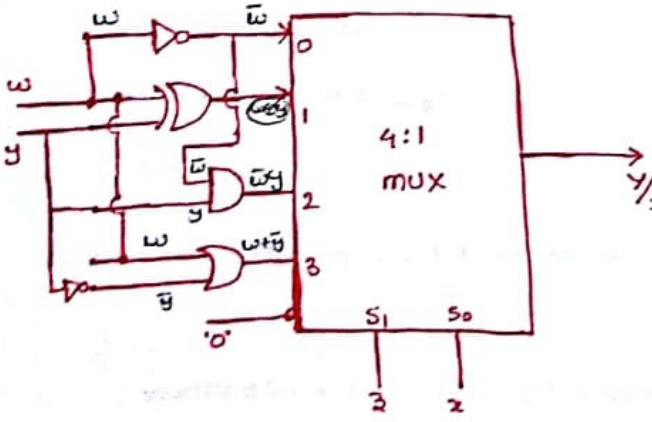
$f_5(A,B,C) = \bar{A}\bar{x}(B) + A\bar{x}(\bar{B}) + \bar{A}x(1) + Ax(\bar{B})$

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

3 5 (0,2) 4

$f = \Sigma m(0,2,3,4,5)$

10q) find the function $f_6(w,x,y,z)$ realized by the given 4x1 mux



ⓝ → no. of variables — 4

ⓜ — no. of selection lines — 2

n-m = 4-2 = 2 we can use 2 i/p's as 2 not gate + 2 i/p gate are used from below conclusions ✓

z	x	
0	0	$\rightarrow D_0 = \bar{w}$ ✓
0	1	$\rightarrow D_1 = w\cdot\bar{y} + \bar{w}y = w \oplus y$ ✓
1	0	$\rightarrow D_2 = \bar{w}y$ ✓
1	1	$\rightarrow D_3 = w + \bar{y}$ ✓

$f_6(w,x,y,z) = \bar{x}\bar{z}(\bar{w}) + x\bar{z}[w \oplus y] + \bar{x}z(\bar{w}y) + xz(w+\bar{y})$

$\Downarrow$

$w\cdot\bar{y} + \bar{w}y$

(y=0,1) 1 1 0 0 0 1 1 0 0 0 1 1 (y=0,1) (w=0,1)

$f = \bar{w}\bar{x}\bar{y} + w \cdot x \cdot \bar{y}\bar{z} + \bar{w}x y\bar{z} + \bar{w}\bar{x}yz + wx\bar{y} + x\bar{y}\bar{z}$

(0,2) 13 6 3 (13,15) (5,13)✓

$f_6(w,x,y,z) = \Sigma m(0,2,3,5,6,12,13,15)$ ✓

conclusions:

a) 16×1 mux — $m=4$ — $(n-m)=0$ — No logic gate

b) 8×1 mux — $m=3$ — $(n-m)=1$ — NOT Gate → ①

c) 4×1 mux — $m=2$ — $(n-m)=2$ — 2 I/p gates, NOT gates ⇒ few. (↓ max '2' no. of gates)

d) 2×1 mux — $m=1$ — $(n-m)=3$ — 2 Level ckt

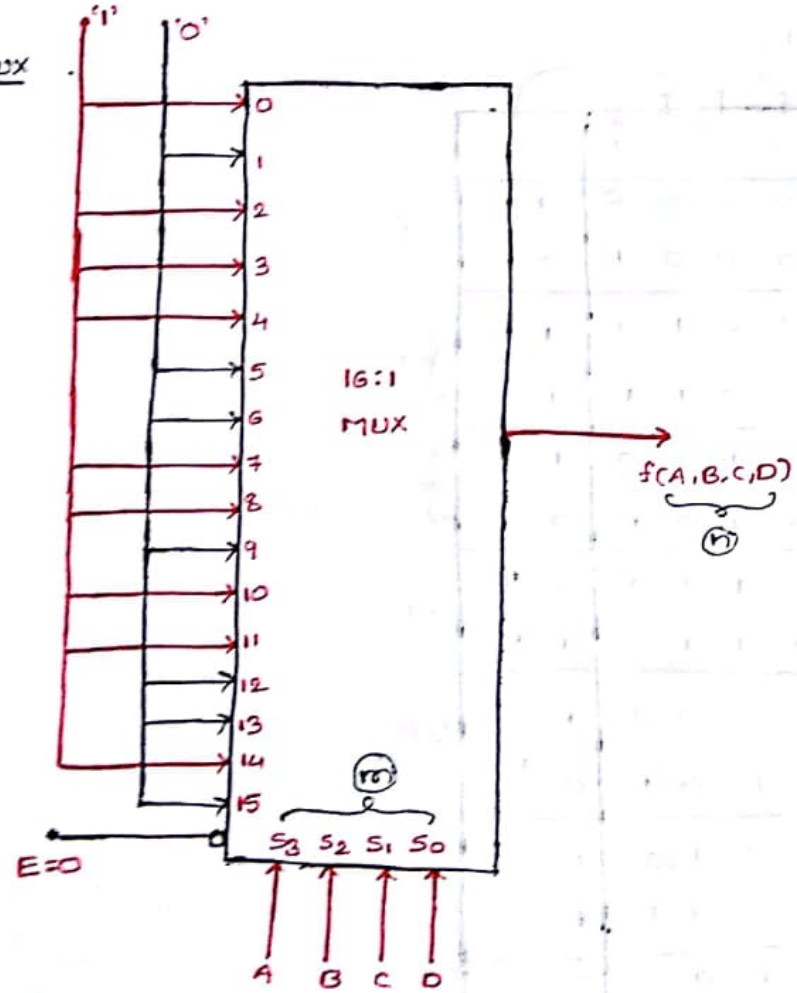
1) Realize the given boolean function $f(A,B,C,D) = \Sigma m(0,2,3,4,7,8,10,11,14)$ using a) 16×1 mux

b) 8×1 mux

c) 4×1 mux

Sol:

a) 16×1 mux



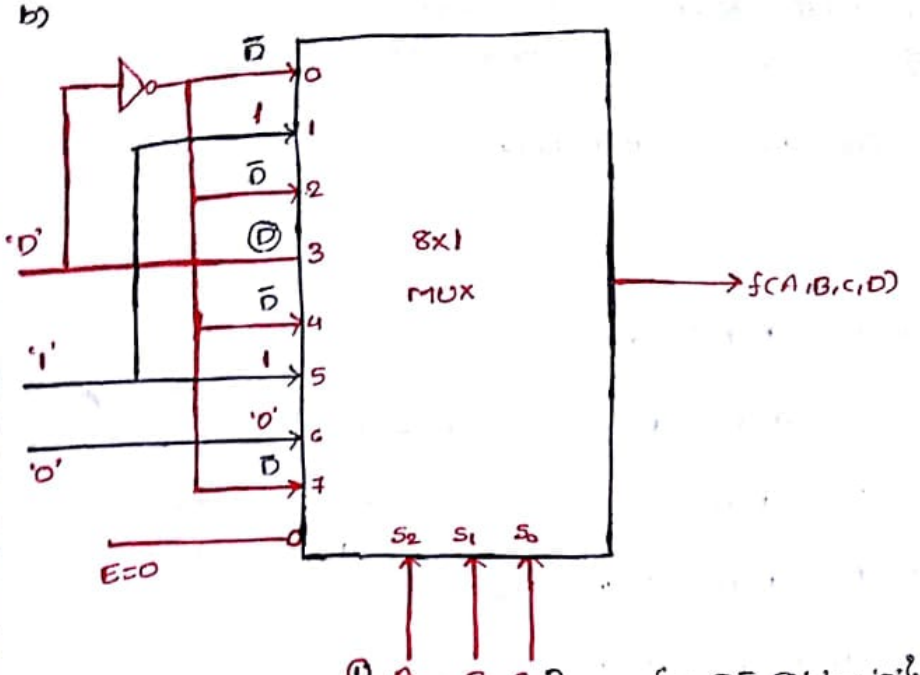
16×1 mux $m=4 \Rightarrow n-m=0$ so that No logic gates are used

$n=4$

where as (n) → no. of variables

(m) → no. of selected lines.

b)



① A B C } f ⇒ { D ⊕ $\bar{D}$ ⊕ '1' ⊕ '0' }
B A C

② C D A (f) ⟹ { B ⊕ $\bar{B}$ ⊕ '1' ⊕ '0' }

Method①:

→ Above diagram based on below table ✓

1) Truth table:

①

	DNO	A	B	C	D	f	
D_0	0	0	0	0	0	1	$\bar{D}$
	1	0	0	0	1	0	
D_1	2	0	0	1	0	1	'1'
	3	0	0	1	1	1	
D_2	4	0	1	0	0	1	$\bar{D}$
	5	0	1	0	1	0	
D_3	6	0	1	1	0	0	D
	7	0	1	1	1	1	
D_4	8	1	0	0	0	1	$\bar{D}$
	9	1	0	0	1	0	
D_5	10	1	0	1	0	1	'1'
	11	1	0	1	1	1	
D_6	12	1	1	0	0	0	'0'
	13	1	1	0	1	0	
D_7	14	1	1	1	0	1	$\bar{D}$
	15	1	1	1	1	0	

② If we want for

S_2 S_1 S_0
A C D

0 1 1 → D_3 → B=0 → 3, B=1 → 7

1 0 0 → D_4 → B=0 → 8, B=1 → 12

(f) 1 0 1 → D_5 → B=0 → 9, B=1 → 13

$D_3(3,7) \longrightarrow 1$

$D_4(8,12) \longrightarrow \bar{B}$

$D(9,13) \longrightarrow \bar{B}$

Method ②

S_2 S_1 S_0
↑ ↑ ↑
A B C

	I_0 000	I_1 001	I_2 010	I_3 011	I_4 100	I_5 101	I_6 110	I_7 111
$\bar{D} \rightarrow 0$	(0)	(2)	(4)	6	(8)	(10)	12	(14)
$D \rightarrow 1$	1	(3)	5	(7)	9	(11)	13	15
	$\bar{D}$	1	$\bar{D}$	D	$\bar{D}$	1	0	$\bar{D}$

→ Every intersection of Row and columns are i/p's. after that identify the minterms.

$$\bar{D} + D = 1$$

→ To read the function items of 4th variable.

EQ) Realize the given boolean function $f(A,B,C,D) = \Sigma m(0,2,3,4,7,8,10,11,14)$ using 8x1 mux, while connecting D, A and 'C' variables to the select lines S_2 S_1 S_0 of multiplier.

Sol: S_2 S_1 S_0
D A C

		D_0 000	D_1 001	D_2 010	D_3 011	D_4 100	D_5 101	D_6 110	D_7 111
$\bar{B}$	0	(0)	(2)	(8)	(10)	1	(3)	9	(11)
B	1	(4)	6	12	(14)	5	(7)	13	15
		1	$\bar{B}$	$\bar{B}$	1	0	1	0	$\bar{B}$

Ex: ✓ DAC – DS

101 ↓

ACD

011

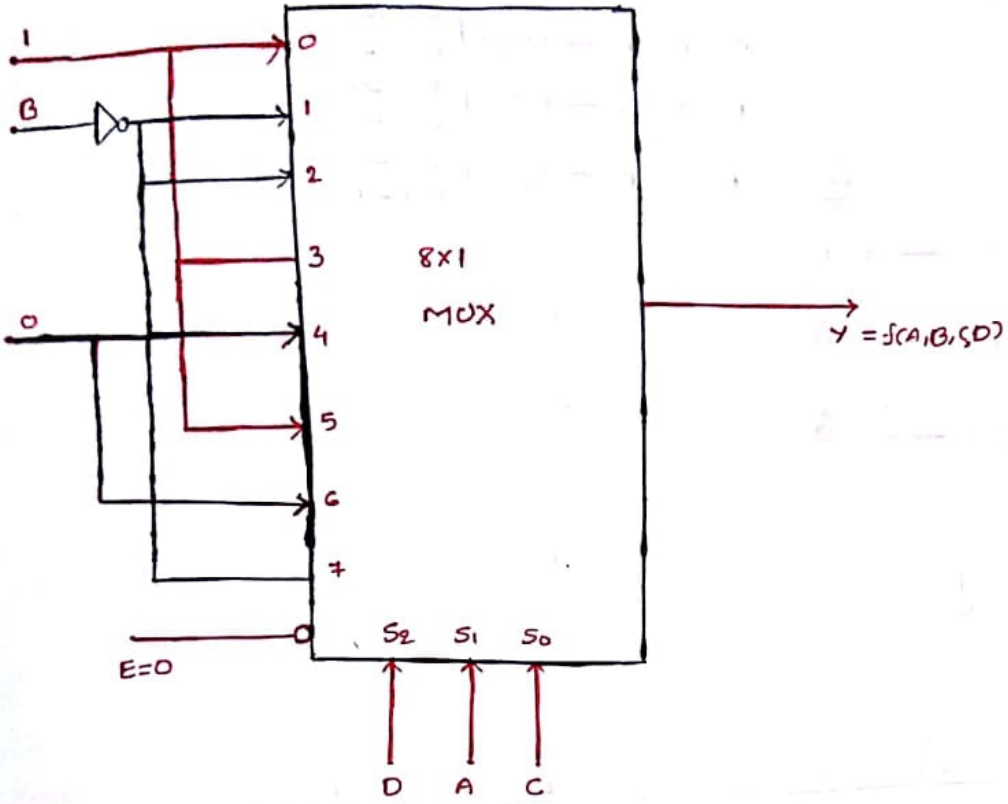
ABCD

↓↓

0 1

0011 – (3) ✓

0111 – (7) ✓

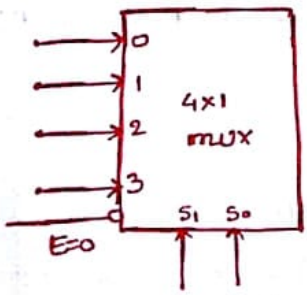


n=4
m=3

n-m = 1 ⟶ one not gate required. ✓

3@) By using 4×1 mux for above boolean function. selected lines given below in figure

$$f(A,B,C,D) = \Sigma m(0,2,3,4,7,8,10,11,14)$$



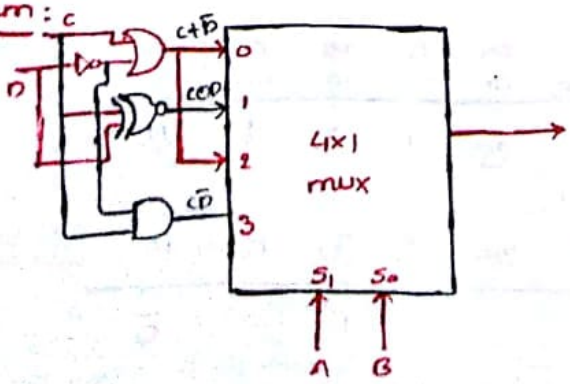
$\left.\begin{matrix} A\ B \\ B\ A \end{matrix}\right\} \longrightarrow f = \{C,D\}$

$C\ A \longrightarrow f = \{B,D\}$

} which must Read the sum of minterms.

Ex:

Logic diagram :



1st method

Truth Table:

I/p D/no	A	B	C	D	O/p f(A,B,C,D)	
0	0	0	0	0	1 ✓	
1	0	0	0	1	0 ×	
2	0	0	1	0	1 ✓	
3	0	0	1	1	1 ✓	
4	0	1	0	0	1 ✓	
5	0	1	0	1	0 ×	
6	0	1	1	0	0 ×	
7	0	1	1	1	1 ✓	
8	1	0	0	0	1 ✓	
9	1	0	0	1	0 ×	
10	1	0	1	0	1 ✓	
11	1	0	1	1	1 ✓	
12	1	1	0	0	0 ×	
13	1	1	0	1	0 ×	
14	1	1	1	0	1 ✓	
15	1	1	1	1	0 ×	

$\bar{C}\bar{D} + C\bar{D} + CD$ (0, 2, 3) ✓

$\bar{D}(\bar{C}+C) + CD$

$\bar{D} + C$ ✓

$\bar{C}\bar{D} + CD$ (4, 7) ✓

$C \odot D$ ✓

$\bar{C}\bar{D} + C\bar{D} + CD$ (8, 10, 11) ✓

$C + \bar{D}$ ✓

$C\bar{D}$ ✓ (14) ✓

2nd method:

S_1 S_0
↑ ↑
A B

	D_0 00	D_1 01	D_2 10	D_3 11
$\bar{C}\bar{D}$ → 00	0	4	8	12
$\bar{C}D$ → 01	1	5	9	13
$C\bar{D}$ → 10	2	6	10	14
CD → 11	3	7	11	15
	$\bar{D}+C$	$C \odot D$	$C+\bar{D}$	$C\bar{D}$

4Q) Realize the given boolean function $f(A,B,C,D)$ using 4x1 mux while connecting 'C' and 'A' variables to the select lines S_1 & S_0 resp.

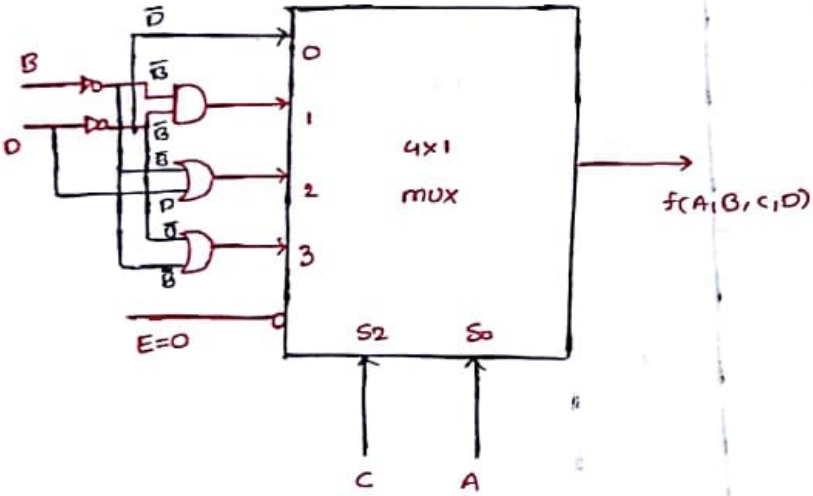
$$f(A,B,C,D) = \Sigma m(0,2,3,4,7,8,10,11,14)$$

Sol:

BD \ CA	I_0 00	I_1 01	I_2 10	I_3 11
$\bar{B}\bar{D}$ → 00	0	8	2	10
$\bar{B}D$ → 01	1	9	3	11
$B\bar{D}$ → 10	4	12	6	14
BD → 11	5	13	7	15
	$\bar{D}$	$\bar{B}\bar{D}$	$D+\bar{B}$	$\bar{B}+\bar{D}$

CA
00
01
10
11

A B C D	A C B D
0 0 0 0 ⑧	1 0 0 0 ⑧
0 0 0 1 ①	0 0 1 0 ②
0 0 1 0 ②	0 1 0 1 ⑩
1 0 0 1 ⑨	
0 0 1 1 ③	
1 0 1 1 ⑪	

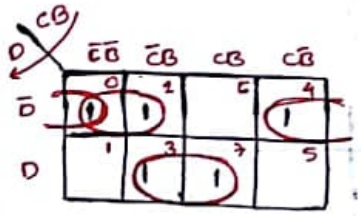


5Q) by using 2x1 mux $f = \Sigma m(0,2,3,4,7,8,10,11,14)$ (MSB)

S/P	A	B	C	D	f o/p
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0

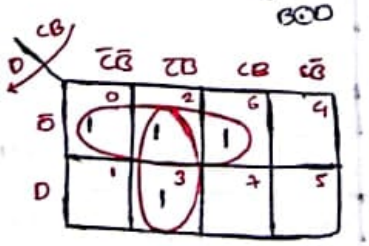
I_0 → SOP①

I_1 → SOP②

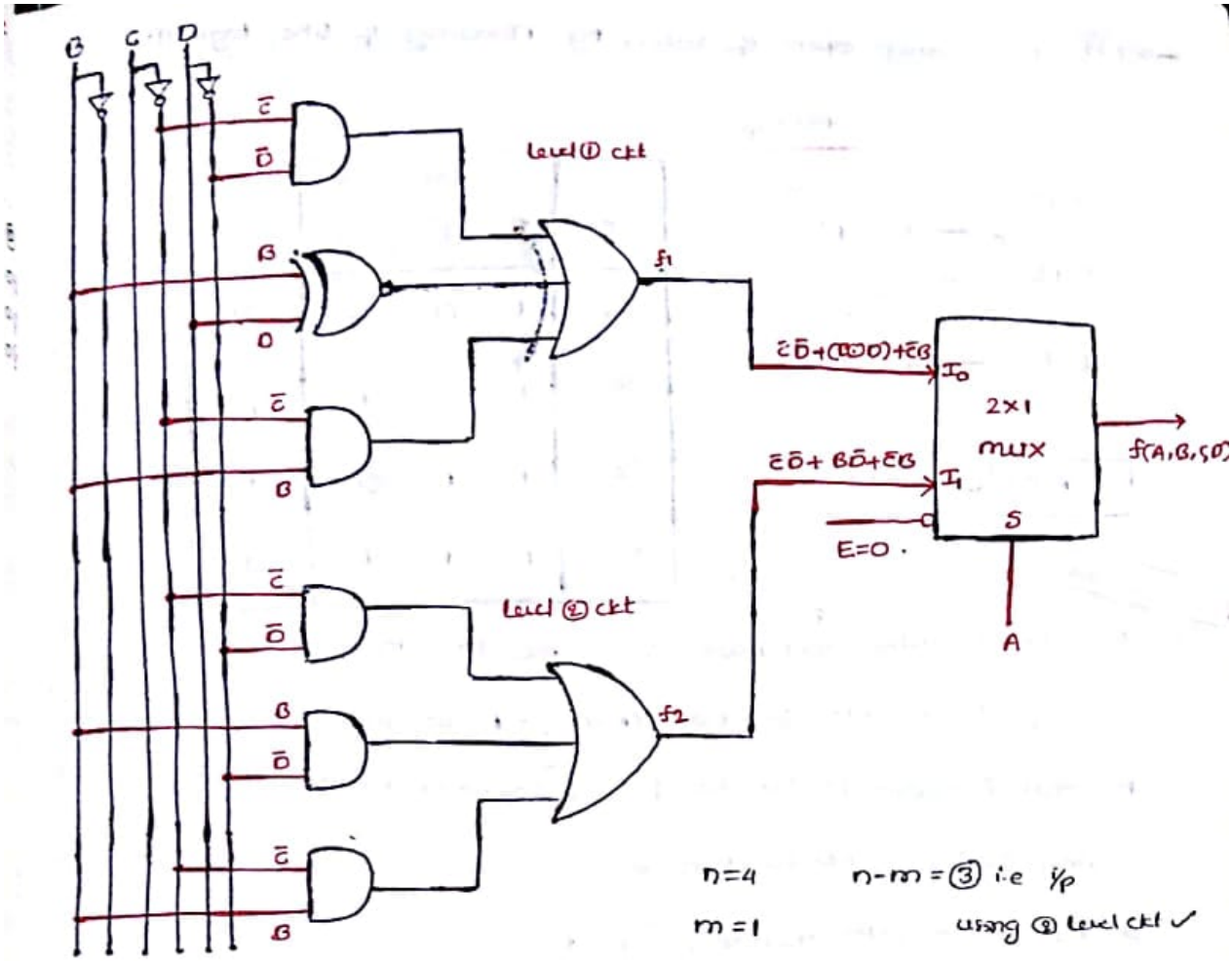


$f_1 = \bar{C}\bar{D} + \bar{B}\bar{D} + B\bar{D} + \bar{C}B$

$B\bar{D}$



$f_2 = \bar{C}\bar{D} + B\bar{D} + \bar{C}B$



⇒ i) $n-m=0$ ⟶ no need of any logic gate

ii) $n-m=1$ ⟶ at maximum ① NOT gate

iii) $n-m=2$ ⟶ need ② i/p logic gates and also need ② NOT gates.

iv) $n-m=3$ ⟶ need ② level circuits

complexity increase ↓

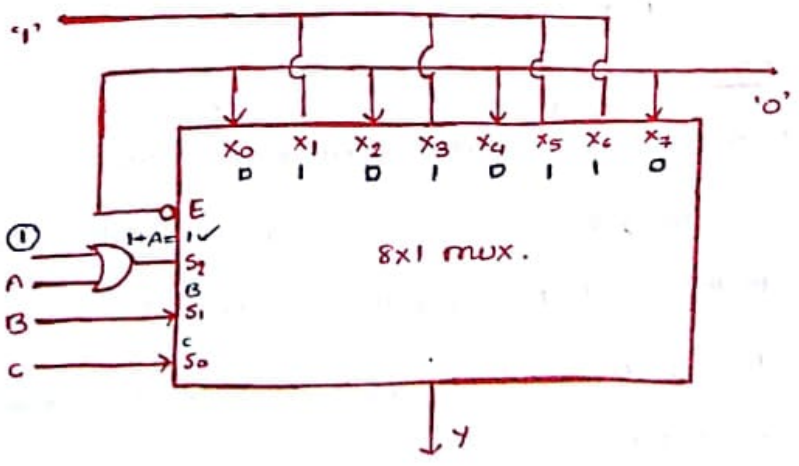
GQ) The given logic circuit is realised by using TTL logic family IC's. where $x_0, x_1, \ldots x_7$ are data i/p lines of 8×1 mux and S_2, S_1, S_0 are "select" i/p lines of mux. The function 'Y' implemented by the mux is.

a) Indeterminant

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

c) $\bar{A}[(B\oplus C)]\oplus A(\overline{B\oplus C})$

d) $B\oplus C$



→ TTL IC's any open i/p taken by floating i/p like logic ①

open i/p

TTL, OTL → '1'

ECL → '0'

$f = B \oplus C$ ✓

i/p	S_2	S_1	S_0	Y = B⊕C ✓
	①	⑧	ⓒ	
X_4	1	0	0	0
X_5	1	0	1	1
X_6	1	1	0	1
X_7	1	1	1	0

↑ selected lines

2003-ECE-Gak

70) By using only 4×1 mux, it is possible to realize

a) Any ③ variable function; $(n-m) = 3-2 = 1$ ↑ NOT gate need

b) only ② variable function; $2-2 = 0$ ✕ no need of gates

c) Any ②&③ variable function; ✕

d) Few ③ variable function; $(3-2) = 1$

Ex: $f(A,B,C) = \Sigma m(1,2,4,5,7)$ (ii) (②) (①)

S_1 S_0 / A B	00	01	10	11
$\bar{C}$ 0	0	②	④	6
C 1	①	3	⑤	⑦
	C (i) 0/$\bar{C}$ (ii)		1	C

Note: All three variable boolean equation can be implemented by using 8×1 mux. without using any extra gate. At the same time some but not all 3-variable boolean equations can be implemented by using 4×1 mux with out using any extra gate.

conventional question in IES

Boolean function implementation by using MUX:

* → For implementing any Boolean function of n-variables with a $2^{n-1}:1$ mux;

procedure:

fun. inits

01. Express the sum of minterms form.

02. In the ordered sequence of ⓝ variables, connect the (n-1) variables to the select line and the single highest order position variable to the i/p line with complemented & uncomplemented form including ⓒ and ①.

03. List the i/p's of mux (all the minterms) in (2) rows. The 1st row lists all those minterms where single variable is complemented and 2nd row with uncomplemented form.

04. Circle all the minterms of the function and inspect each of colum speacldy.

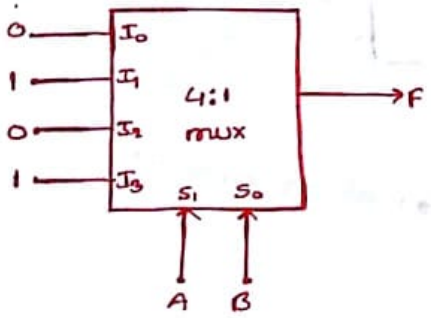
05. If (2) minterms in a colowmn are not circled, apply '0' to the corresponding mux i/p.

06. If (2) minterms are circled, apply '1' to the corresponding mux i/p.

07. If the one minterm is circled (either upper row (or) lower row) then its front value is the corresponding mux i/p.

Q) Find o/p of mux given below.

IE-08



a) A

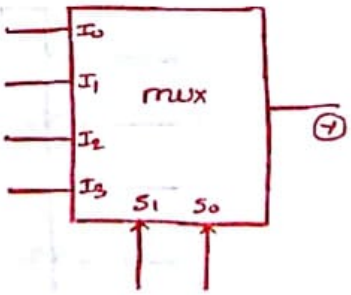
b) $\bar{A}B$

c) B

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

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

QQ) The Boolean function $F(A,B,C) = \Pi(0,2,4,7)$ is to be implemented using 4x1 multiplexer shown in figure. which one of the following choices of i/p's to multiplexer will realite the Boolean function? JTO-09



a) $(I_0, I_1, I_2, I_3, S_1, S_0) = (1, 0, \bar{A}, A, C, B)$

b) $(I_0, I_1, I_2, I_3, S_1, S_0) = (1, 0, \bar{A}, A, B, C)$

c) $(I_0, I_1, I_2, I_3, S_1, S_0) = (0, 1, \bar{A}, A, C, B)$

d) $(I_0, I_1, I_2, I_3, S_1, S_0) = (0, 1, A, \bar{A}, B, C)$

Sol:

$f(A,B,C) = \Pi(0,2,4,7)$

n=3 → required muxes is → $2^{n-1} : 1$

→ $2^{3-1} : 1$

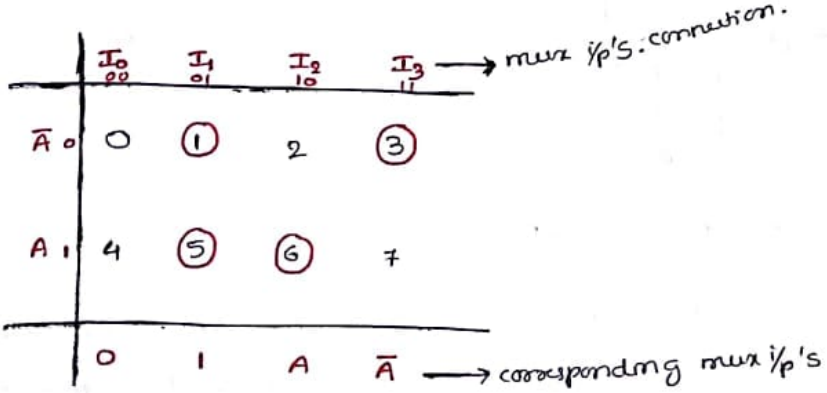
→ 4:1 mux

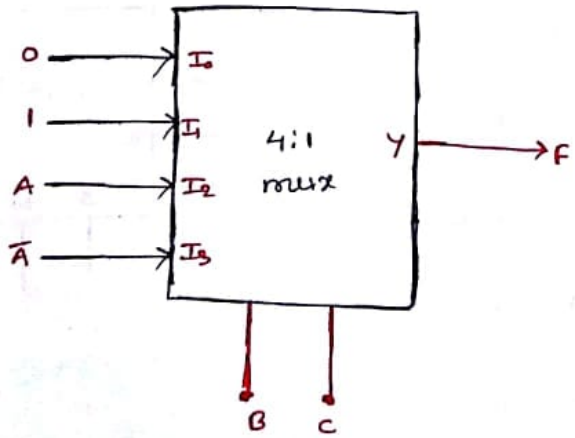
$F(A,B,C) = \pi(0,2,4,7)$

$F = \sum m(1,3,5,6)$

Truth Table:

	minterm	A	B	C	F	
I_0, I_1	0	0	0	0	0	$\} 0 \rightarrow (0,4)$
	1	0	0	1	1 ✓	
	2	0	1	0	0	$\} 1 \rightarrow (1,5)$
	3	0	1	1	1 ✓	
I_2, I_3	4	1	0	0	0	$\} A \rightarrow (2,6)$
	5	1	0	1	1 ✓	
	6	1	1	0	1 ✓	$\} \bar{A} \rightarrow (3,7)$
	7	1	1	1	0	





So, $(I_0, I_1, I_2, I_3, S_1, S_0) = (0, 1, A, \bar{A}, B, C)$

DEMULTIPLEXER: (DE-MUX)

→ A Demultiplexer is a combinational circuit that receives information on a single line and transmits this on one of (2^n) possible o/p lines.

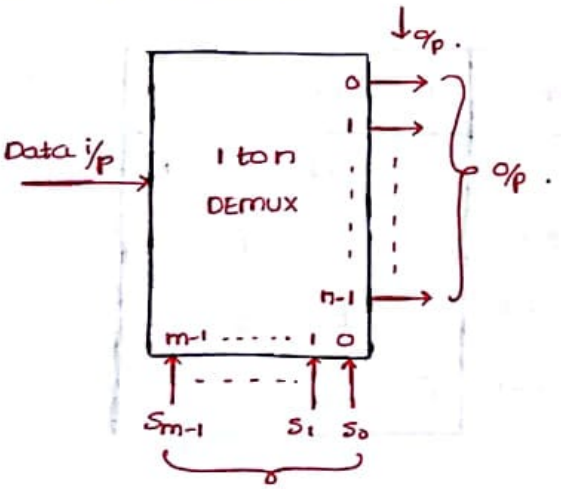
→ The selection of a specific o/p line is controlled by the bit values of "n" selection lines.

→ A "Decoder" with an enable i/p can function as a demultiplexer.

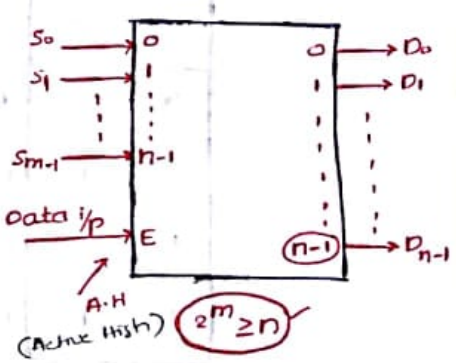
→ It is also known as "Data Distributor"

→ It is used as to peform the reverse operation of mux

single i/p — many o/p



(or):



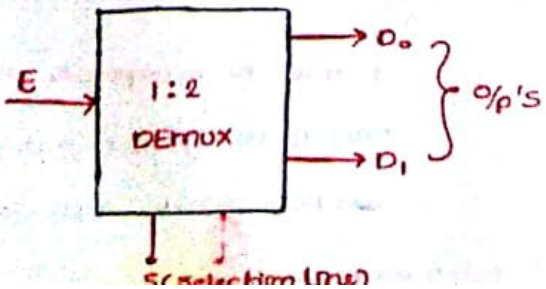
selection lines/controlled lines.

→ The purpose of selection line is selected the o/p line ✓

Types

a) 1 to 2 demux
b) 1 to 4 demux
c) 1x10 demux
d) 1x8 demux
e) 1x16 demux

→ 1:2 demux:



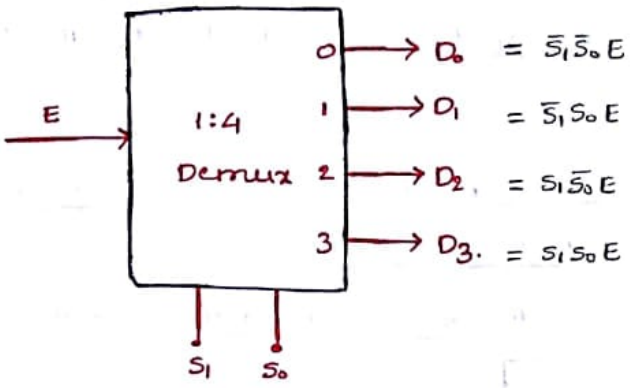
$D_0 = \bar{S}E$

$D_1 = S\bar{E}$

S	D_1	D_0
0	0	E
1	E	0

ii) 1x4 Demux:

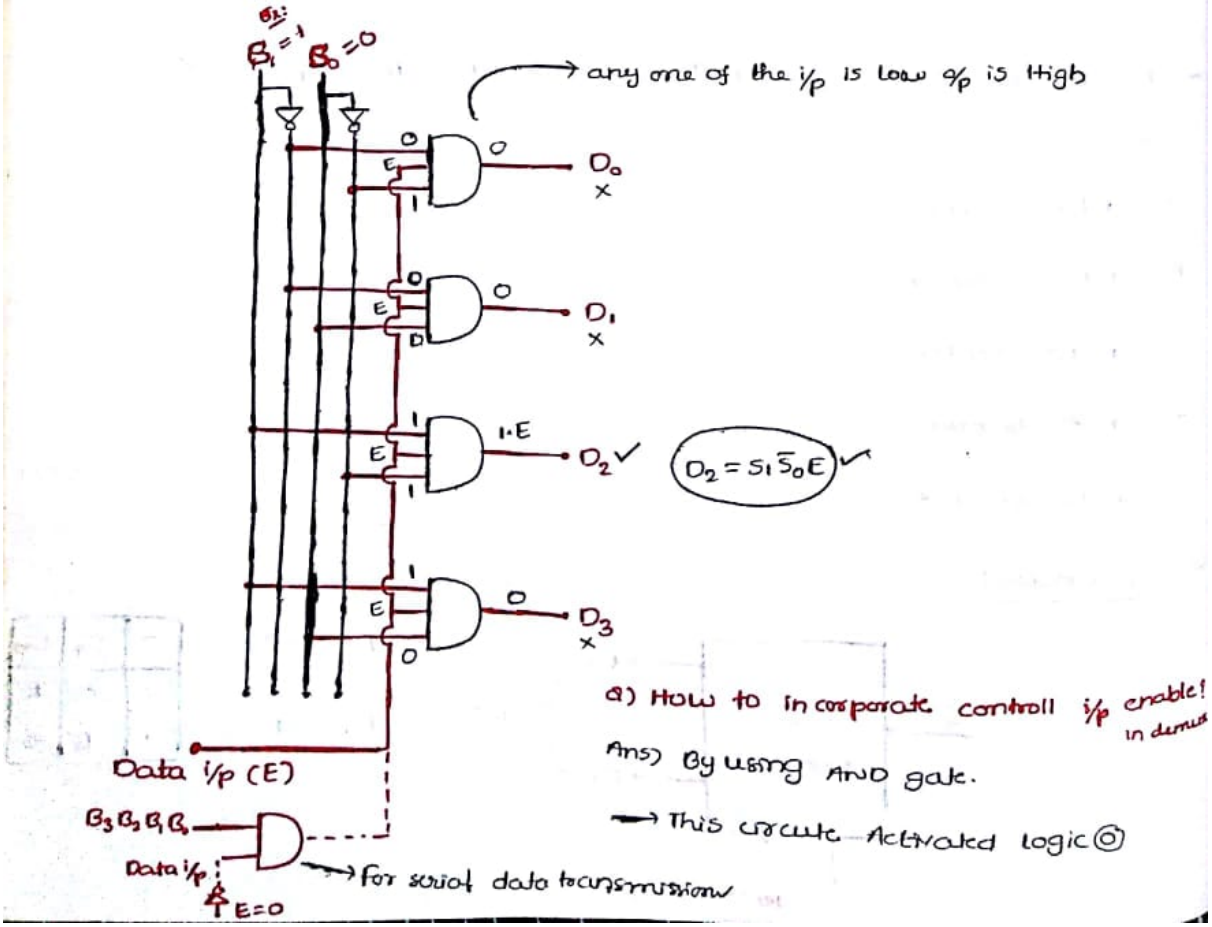
a) Circuit diagram:



b) Truth table:

S_1	S_0	O/P	D_3	D_2	D_1	D_0
0	0	$D_0 = \bar{S}_1 \bar{S}_0 E$	0	0	0	(E)
0	1	$D_1 = \bar{S}_1 S_0 E$	0	0	(E)	0
1	0	$D_2 = S_1 \bar{S}_0 E$	0	(E)	0	0
1	1	$D_3 = S_1 S_0 E$	(E)	0	0	0

c) Logic diagram:



a) How to incorporate controll i/p enable! in demux

Ans) By using AND gate.

→ This circuit Activated logic ◎

Given demux	To be implemented demux	Required no. of demux
1:2	1:4	1+2 = 3
1:2	1:8	1+2+4 = ①
1:2	1:16	1+2+4+8 = 15
1:2	1:64	1+2+4+8+16+32 = 63.
1:4	1:16	1+4 = ⑤ ✓

IE-06

10) What is the number of select lines required in a single i/p and 'n' o/p DEMUX.

Sol: let no. of select lines required = (m) ✓

$n = 2^m$

$\log_2^n = \log_2^{2^m}$

$* (m = \log_2^n)$ ✓

a) 2

b) n

✓ $\log_2^n$

d) 3

Decoders:

Decoder is a combinational ckt, it will convert the code form one form to another form.

→ many i/p's — many o/p lines ✓

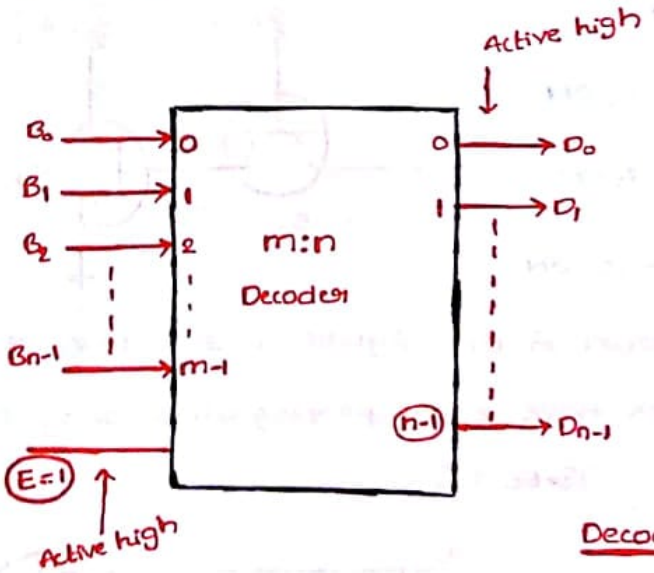
(n) — (2^n)

→ If the n-bit decoded information has unused @ don't care combinations, the decoder o/p will have less than 2^n o/p's

i.e if (m) = total number of i/p lines &

(n) = total number of o/p lines.

then $n \leq 2^m$



→ $2^m \geq n$

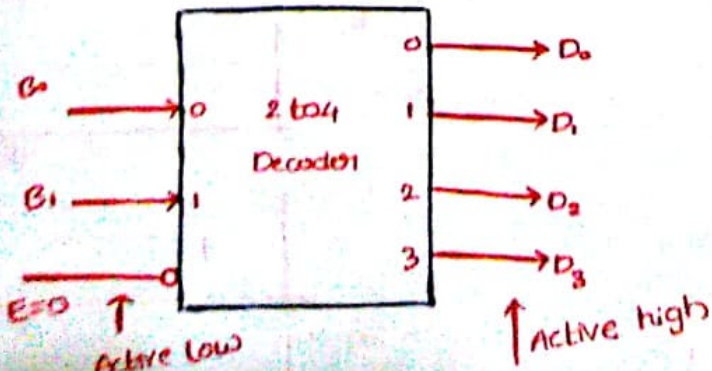
Decoders

$2^m = n$
01. 1 to 2 line decoder
02. 2×4 line decoder / Binary to Nibble decoder
03. 3:8 line decoder / Binary to octal decoder

$2^m > n$
04. 4×10 line decoder / BCD to Decimal

$2^m = n$ ✓
05. 4×16 line decoder / Binary to Hexadecimal

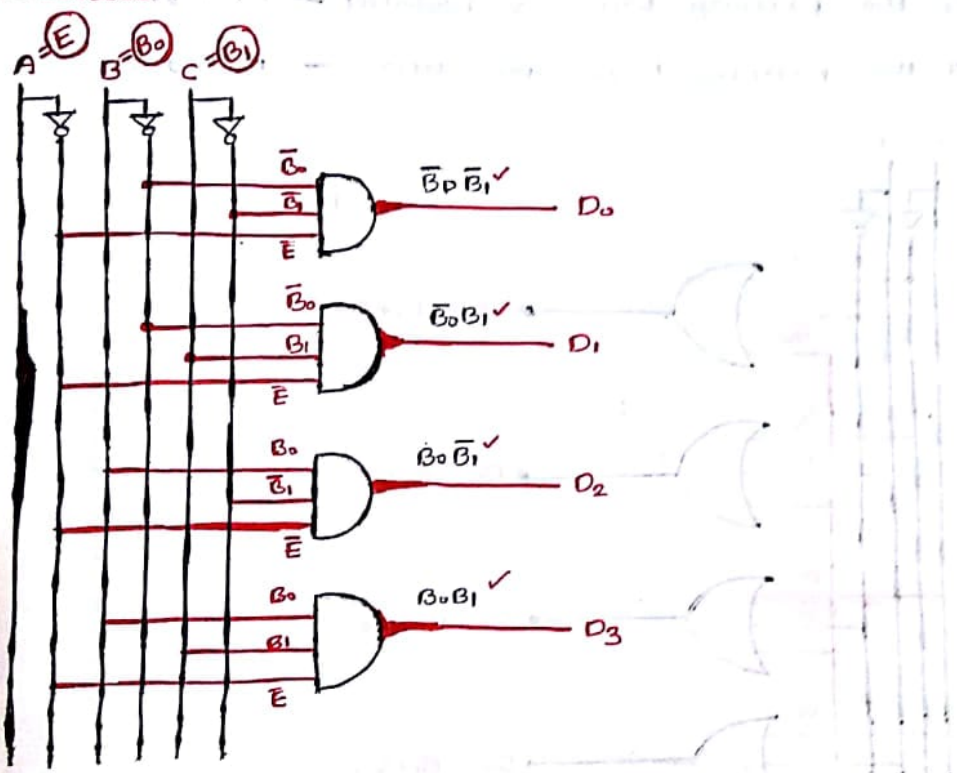
1) 2 to 4 Decoder:



Truth Table:

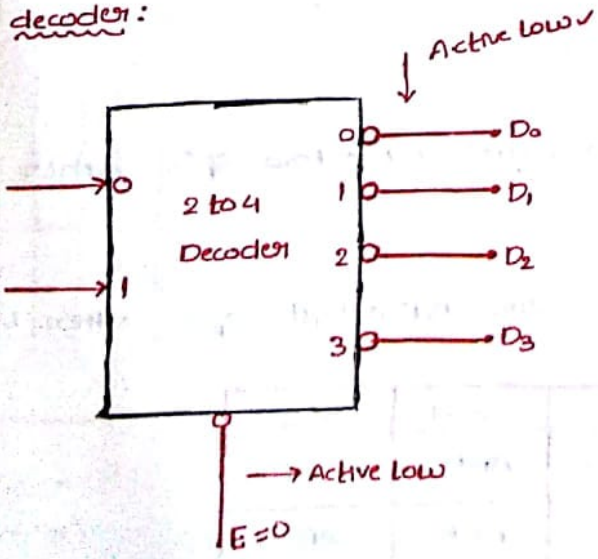
E	B_0	B_1	o/p D_0	D_1	D_2	D_3	
1	x	x	1	1	1	1	
0	0	0	0	1	1	1	$D_0 = \bar{B}_1 \bar{B}_0$
0	0	1	1	0	1	1	$D_1 = \bar{B}_1 B_0$
0	1	0	1	1	0	1	$D_2 = B_1 \bar{B}_0$
0	1	1	1	1	1	0	$D_3 = B_1 B_0$

Logic diagram:



Another Representation:

2 to 4 decoder:



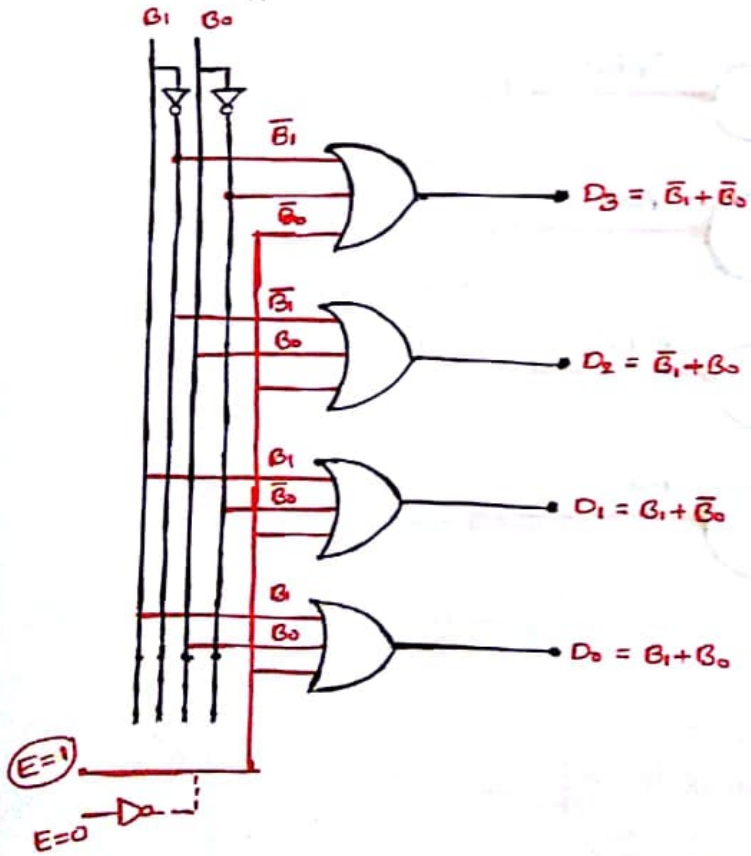
→ If we want realise the decoder using OR gates, we need to read the o/p in the reverse order

Truth table:

B_1	B_0	D	
0	0	$D_0 = (B_1 + B_0)$	= 1
0	1	$D_1 = (B_1 + \bar{B}_0)$	= 1
1	0	$D_2 = (\bar{B}_1 + B_0)$	= 0
1	1	$D_3 = (\bar{B}_1 + \bar{B}_0)$	= 1

→ Whenever the primary terms are product — AND, NAND

→ Whenever the primary terms are sum — NOR, OR

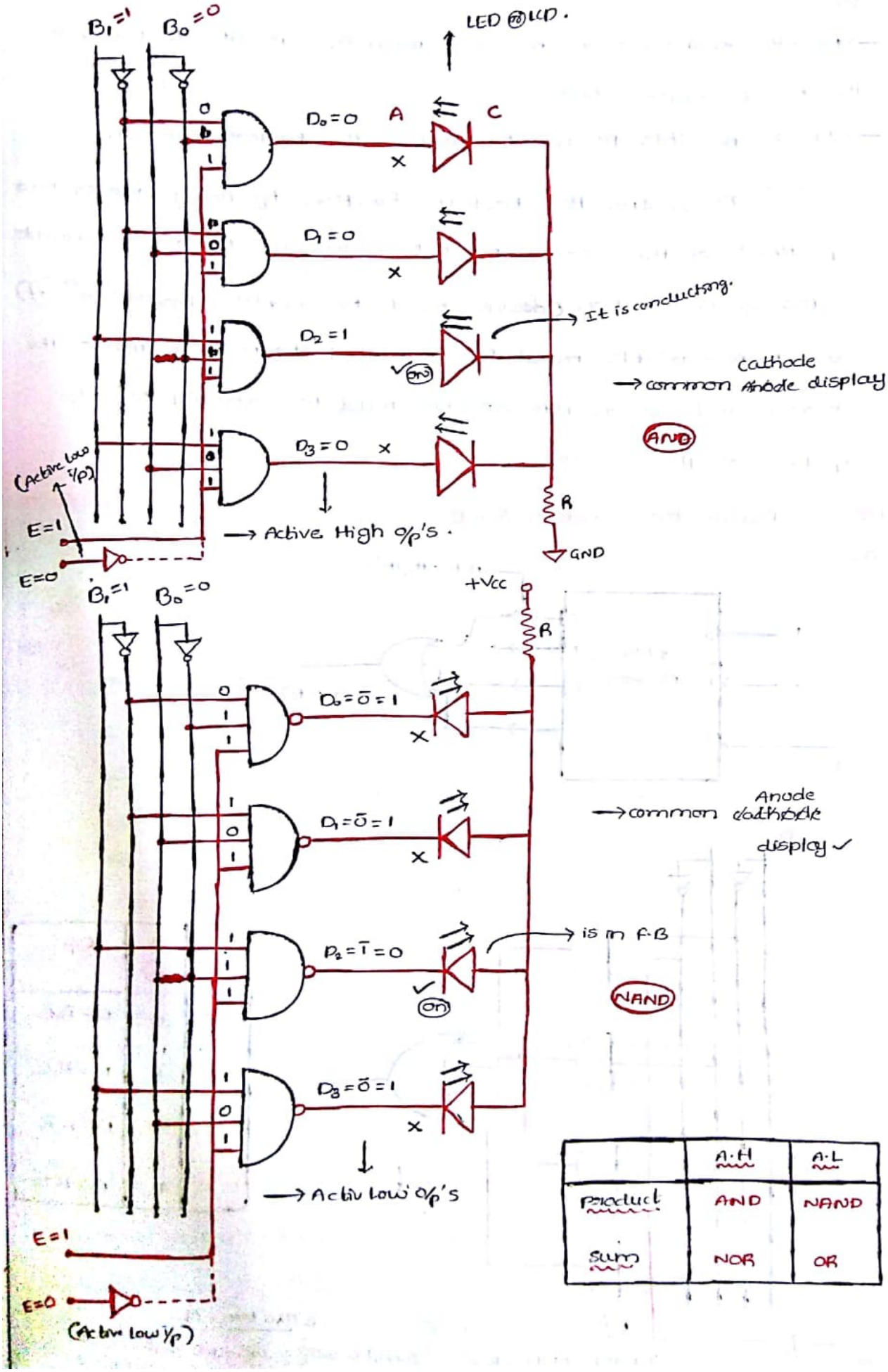


→ Decoder is possible to realize Active Low o/p's either using NAND @ OR gate.

→ Decoder is possible to realize Active High o/p's either using NOR @ AND gate.

	A·H	A·L
product	AND	NAND
sum	NOR	OR

→ In logic diagram the AND/NAND o/p's are connected to the leds, it is represented below.



	A·H	A·L
Product	AND	NAND
Sum	NOR	OR

Applications of Decoder/Demux:

→ It is possible to use to convert serial data to parallel form.

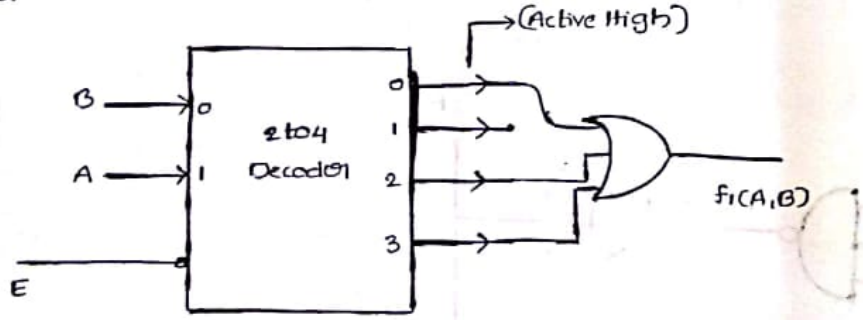
→ Decoder is used in memory mapping circuits to minimize the no. of address lines.

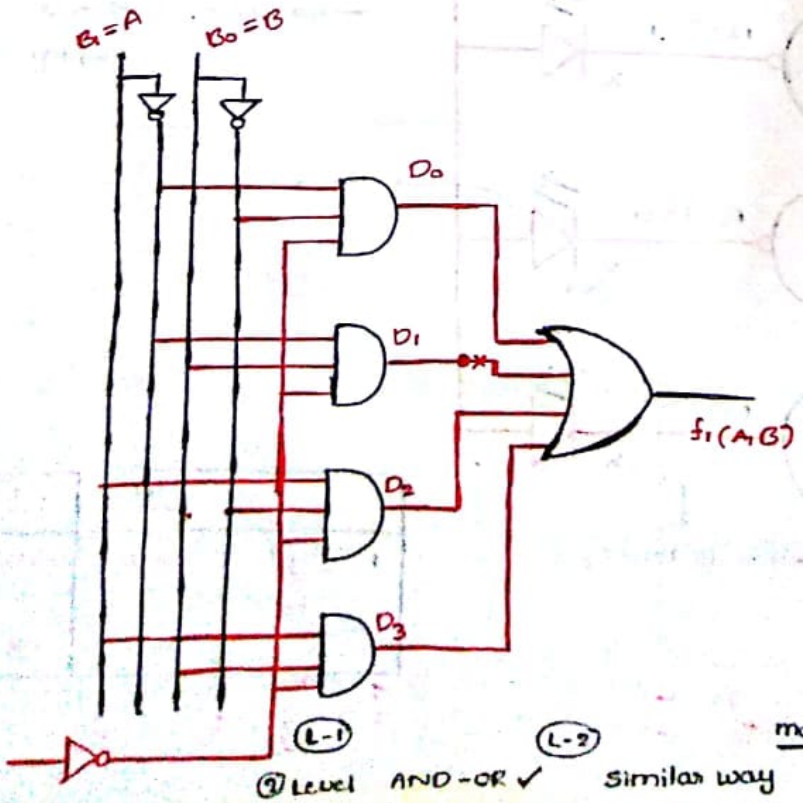
→ It is possible to use to realize the boolean functions.

∴ To realize the boolean function by using decoder no. of i/p lines of the encoder must be same as the no. of variables in the given function (decoder must be complete decoder i.e $2^m = n$) and MSB variable must be connected to MSB i/p line of the decoder and so on LSB variable must be connected to LSB i/p line of the decoder.

1Q) To Realize the function $f_1(A,B) = \Sigma m(0,2,3)$

Ans:





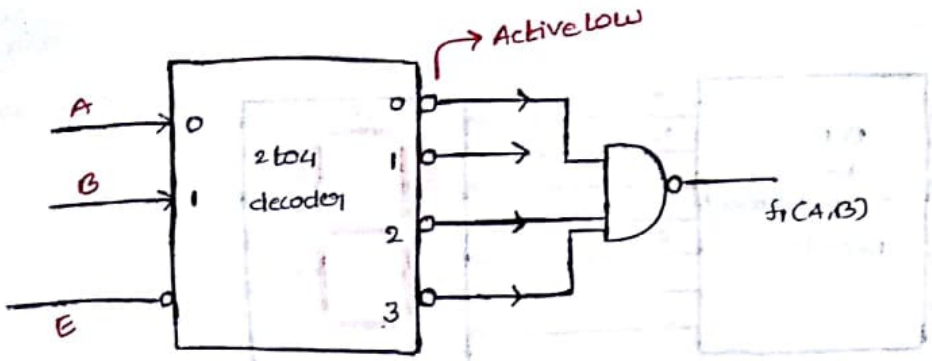
A (B_1)	B (B_0)	D/p
0	0	$D_0 = \bar{B_1}\bar{B_0}$
0	1	$D_1 = \bar{B_1}B_0$
1	0	$D_2 = B_1\bar{B_0}$
1	1	$D_3 = B_1B_0$

(L-1) (L-2)

max term - AL

② Level AND-OR ✓ Similar way OR-AND realization

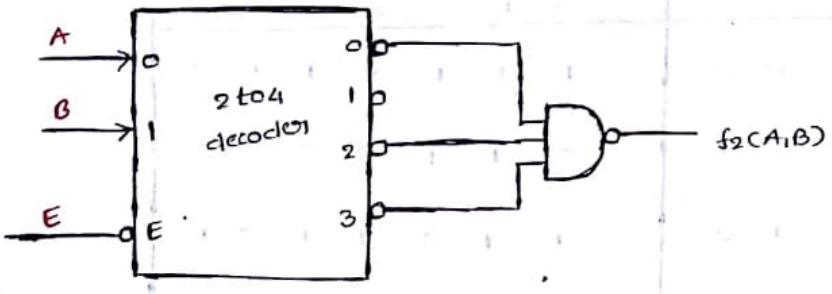
$f(A,B) = 0$



→ 2 level NAND - NAND - representation ✓

2) $f_2(A,B) = \sum m(1,3)$ (2)

By way NOR-NOR for makem (AH) $f(A,B) = 0$.



3) By using decoder it is possible to realize

a) more than one function

b) multi o/p circuit

Sol:

a) $f_1(A,B,C)$ → 3 to 8 decoder

$f_2(x,y,3)$ ✗ does not satisfy ✓

(or)

$f_2(C,B,A)$ ✗

(or)

$f_2(A,B,C,D)$ ✗

b) full adder, subtractor i.e it gives multi o/p we can realise these ckts with decoder ✓

4) Design and Realise BCD to 7-segment decoder to drive a bit

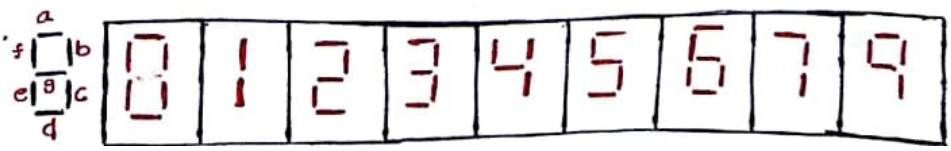
a) common cathode 7-segment display

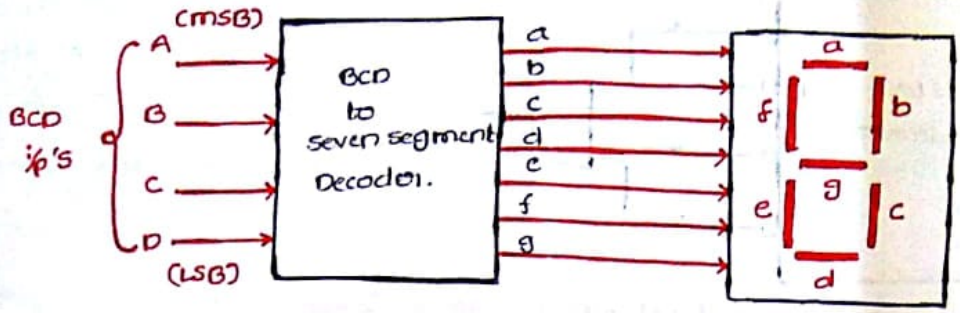
b) common anode 7-segment display

Sol:

BCD-To-7 segment Decoder/Drivers:

→ It is used to take a 4-bit BCD i/p and provide the o/p's that will pass current through the appropriate segments to display the decimal digit.





Truth Table:

Decimal digit	I/P A	B	C	D	O/P 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	0	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	0	0	1	1

common cathode:

$a(ABCD) = \Sigma m(0,2,3,5,6,7,8,9) + d(10,15) \longrightarrow (1)$

$b(A,BCD) = \Sigma m(0,2,3,4,7,8,9) + d(\ldots 10,15) \longrightarrow (2)$

common Anode:

$a(ABCD) = \Sigma m(1,4) + d(10,11,12,13,14,15) \longrightarrow (1')$

$b(ABCD) = \Sigma m(5,6) + d(10,11,12,13,14,15) \longrightarrow (2')$

→ After using k-maps the simplified minimum boolean expression are given by.

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

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

$c = B + \bar{C} + D = \overline{\bar{B}C\bar{D}}$

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

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

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

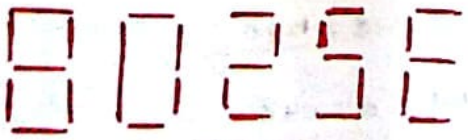
$g = A + B\bar{C} + \bar{B}C + C\bar{D}$

→ The o/p's of display has AND-OR logic, so it can be implemented by "24" NAND-gates.

G-09 statement and linked answer Question:

Two products are sold from a vending machine, which has two push buttons (P1) and (P2). When a button is pressed, the price of the corresponding product is displayed in a 7-segment display.

If no buttons are pressed, '0' is displayed, signifying 'Rs.0' If only (P1) is pressed, '2' is displayed, signifying 'Rs.2' If only P_2 is pressed, '5' is displayed, signifying 'Rs.5'. If both (P1) and (P2) are pressed 'E' is displayed, signifying 'Error'. The names of the segments in the 7-segment display, and the glow of the display for '0', '2', '5' & 'E' are shown below?



Consider

i) Push button pressed/not pressed is equivalent to logic 1/0 respectively

ii) a segment glowing/not glowing in the display is equalent to

logic 1/0 respectively.

A) If segments ⓐ to ⓖ are considered as functions of P_1 and P_2 then which of the following is correct?

a) $g = \bar{P}_1 + P_2$, $d = c + e$

b) $g = P_1 + P_2$, $d = c + e$

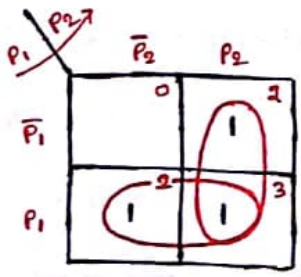
c) $g = \bar{P}_1 + P_2$, $e = b + c$

d) $g = P_1 + P_2$, $e = b + c$

Sol: We have ② i/p variables here as P_1 & P_2. Firstly we draw a truth table according to question.

I/p's		o/p's							
P_1	P_2	a	b	c	d	e	f	g	
0	0	1	1	1	1	1	1	0	→ display '0'
0	1	1	0	1	1	0	1	1	→ display '5'
1	0	1	1	0	1	1	0	1	→ display '2'
1	1	1	0	0	1	1	1	1	→ display 'E'

K-map for ⓖ:



$$g = P_1 + P_2$$

also, we have

→ a = 1

→ b = $\bar{P}_2$ ⊕

→ c = $\bar{P}_1$ ⊕ ≠

→ d = 1

→ e = $P_1 + \bar{P}_2$

→ f = $\bar{P}_1 + P_2$

∴ since $e \neq b + c$

but $d = c + e$

$d = P_1 + \bar{P}_2 + \bar{P}_1$

$d = 1 + \bar{P}_2$

$d = 1$

$g = P_1 + P_2$ & $d = c + e$

B) what are the minimum numbers of NOT gates and 2-i/p OR gates required to design the logic of the driver for this 7-segment display ?

a) 3 NOT & 4 OR

b) 2 NOT & 4 OR

c) 1 NOT & 3 OR

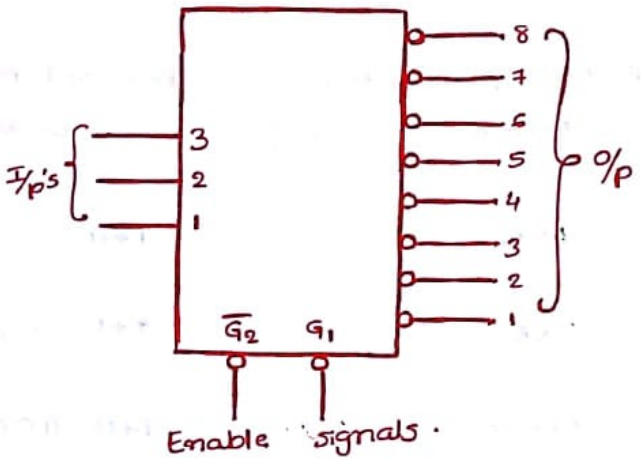
d) 2 NOT & 3 OR

Sol: minimum no. of NOT-gates required = ②

minimum no. of OR-gates required = ③

IE-02(EE)

* A 3 to 8 decoder is shown below.



All the o/p lines of the chip will be high, when all the i/p's 1, 2 & 3.

a) are high; and G_1, G_2 are low

b) are high; and G_1 is low, G_2 is High

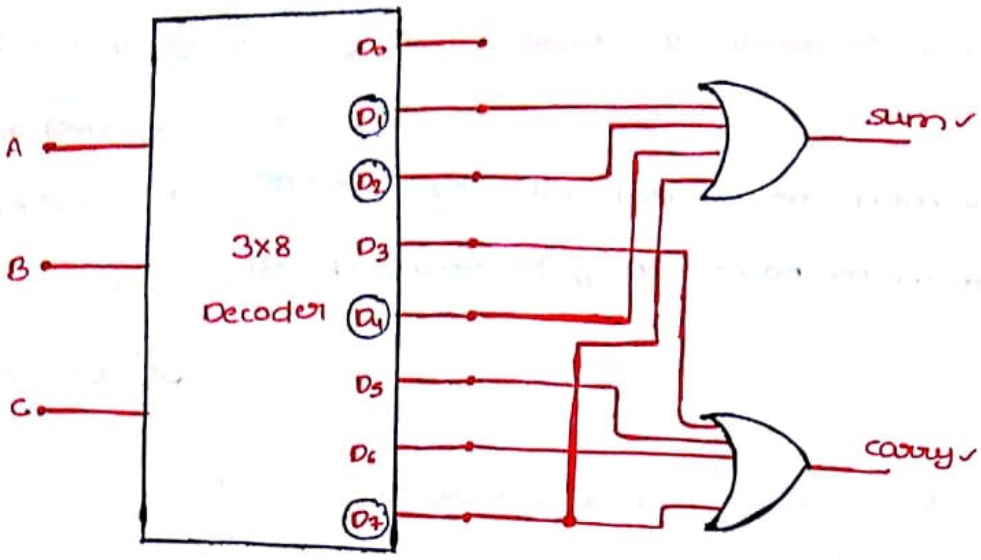
c) are high: and G_1, G_2 are high

d) are high; and G_1 is high, G_2 is low.

→ My 2×4 / 3×8 line decoder acts like as 1:4 / 1:8 Demux and vice versa.

→ To implement a F.A/F.S, we required a 3×8 / 2×4 line decoder and 2-OR-gate. / 1-OR

since, we know a 3-i/p (A,B,C) F.A has sum = $\Sigma m(1,2,4,7)$ and carry = $\Sigma m(3,5,6,7)$. so circuit is below represents.



*

Given Decoder	To be implemented Decoder	Required no. of decoder.
01. 2×4	4×16	1+4 = ⑤ ✓
02. 2×4	3×8	2+1 NOT gate
03. 4×16	8×256	1+16 = ⑰ ✓

3Q) Relize 16×1 mux using

a) 8×1 mux

b) 4×1 mux

$m=4$ A B C D

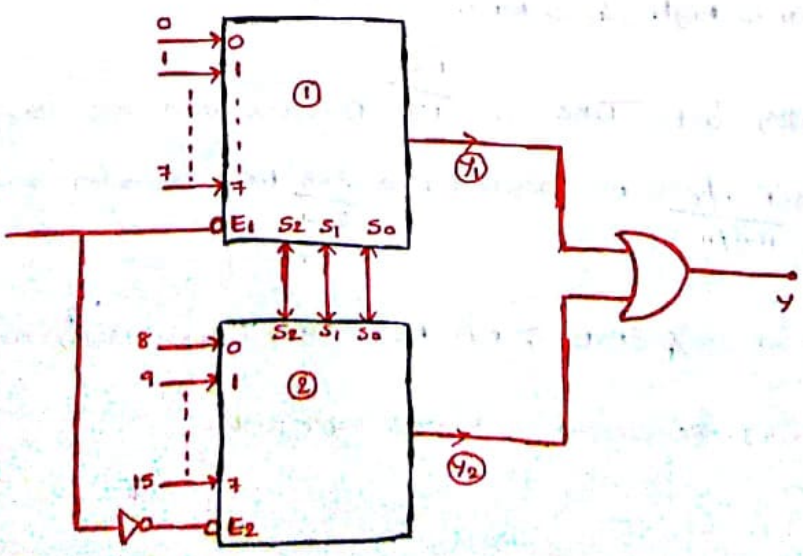
A ↑ MSB, D ↑ LSB

Sol:

No. of multiplexers required = n_1/n_2

[At level ①]

a) No. of mux required = $16/8$ = ② ✓



A	Y
0	Y_1
1	Y_2

Enabled Disabled

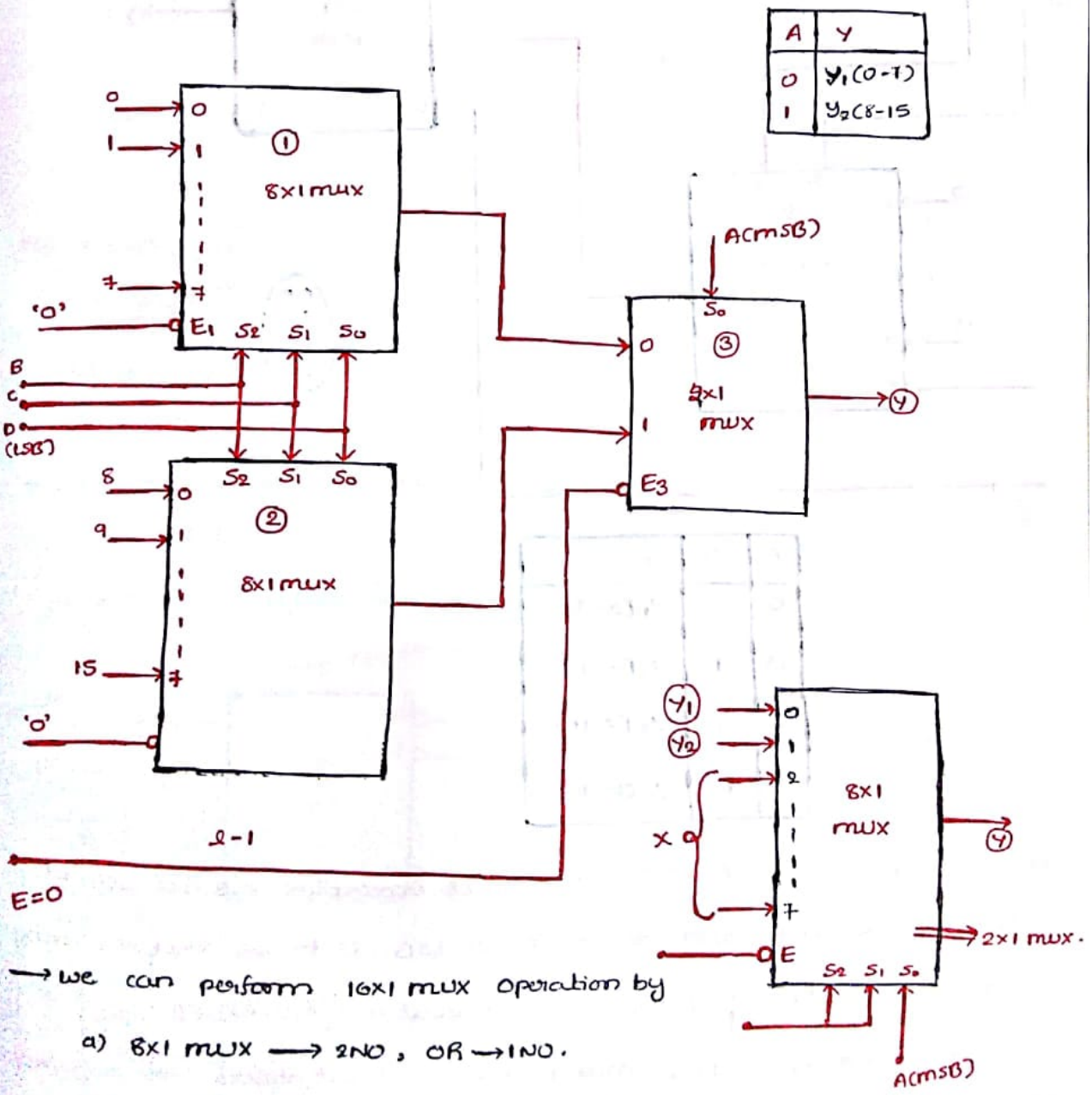
mux1 ✓ mux2 ✗

mux 1 ✗ mux 2 ✓

A	B	C	D	
0/1	0	0	0	0/8 ✓
0/1	1	1	1	7/15 ✓

→ To have control on overall circuit, we are adding one 2x1 mux at the o/p side.

A	Y
0	Y_1 (0-7)
1	Y_2 (8-15



→ we can perform 16x1 mux operation by

a) 8x1 mux ⟶ 2NO, OR ⟶ 1NO.

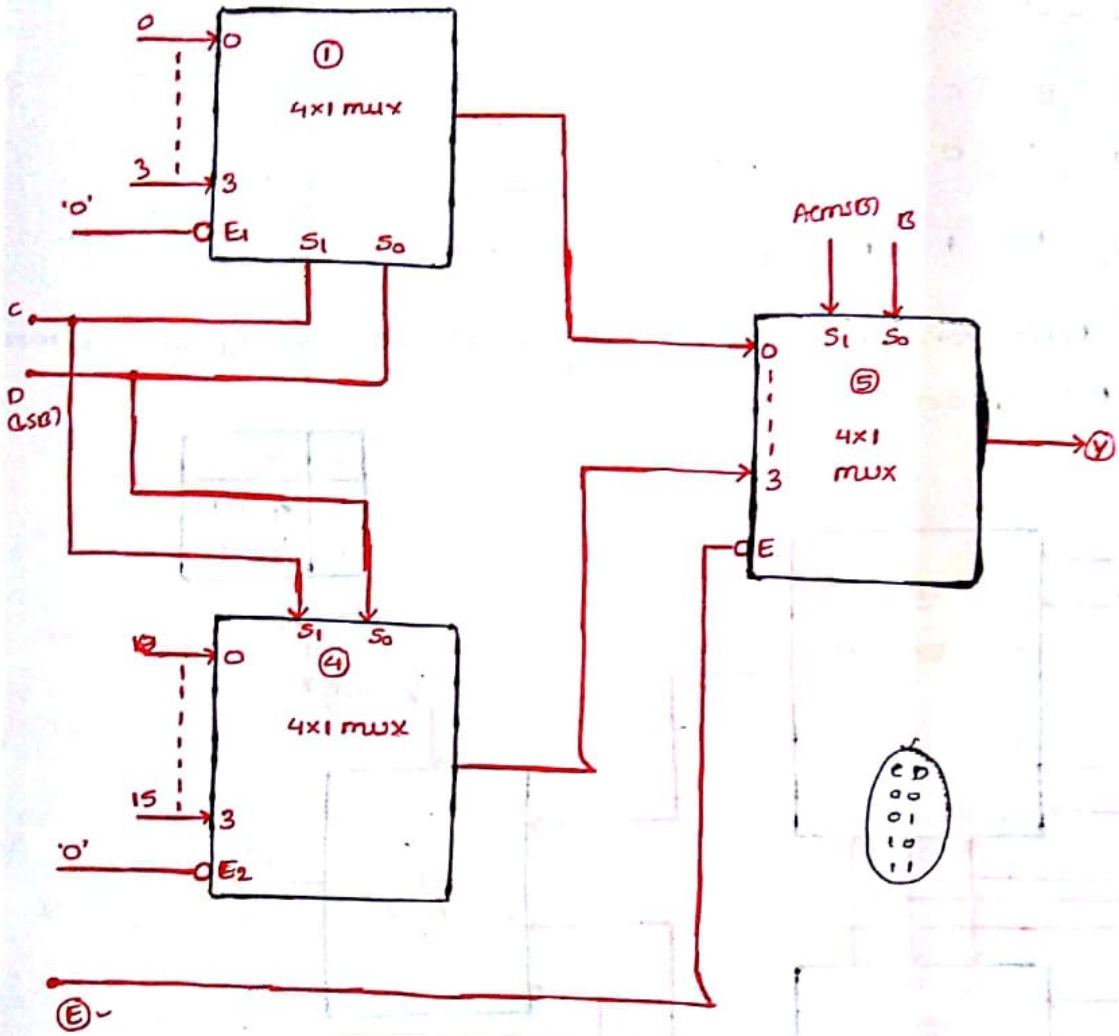
b) 8x1 mux ⟶ 2NO, 2x1 mux ⟶ 1NO.

c) 8x1 mux ⟶ 3NO.

b) No. of multiplexer required = $16/4 = 4$.

(At level 1)

4×1 mux ⇒ 4 No.

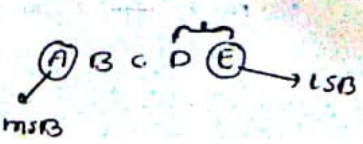


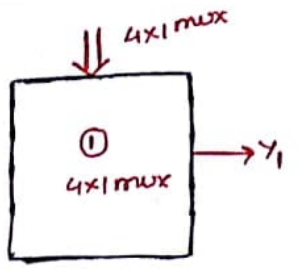
A	B	Y
0	0	Y_1 [0-3]
0	1	Y_2 [4-7]
1	0	Y_3 [8-11]
1	1	Y_4 (12-15)

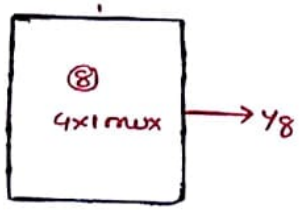
4a) A 32×1 mux is having the select variables A, B, C, D and 'E', where 'A' is MSB and so on 'E' is LSB, is to be realised using 4×1 mux. Identify all the select variables A, B, C, D & 'E' are accomodated and how many no. of multiplexers are required.

Ans:

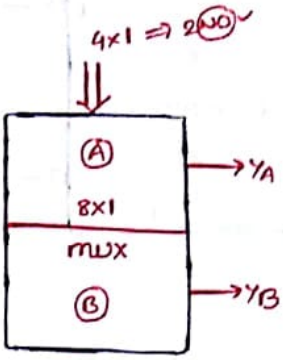
At level ① no. of 4×1 mux required = $\frac{32}{4} = 8$



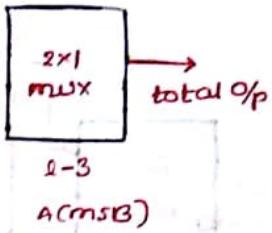




D,E
level-a



B,C
level-②



1-3
A(MSB)

a) 4x1 mux ⟶ 10 NO; 2x1 mux ⟶ 1 NO.

b) 4x1 mux ⟶ 11 NO.

5(a) Realize n_1 (4x16) decoder using

a) 3x8 Decoder

b) 2x4 Decoder

(n_2)

A B C D

MSB LSB

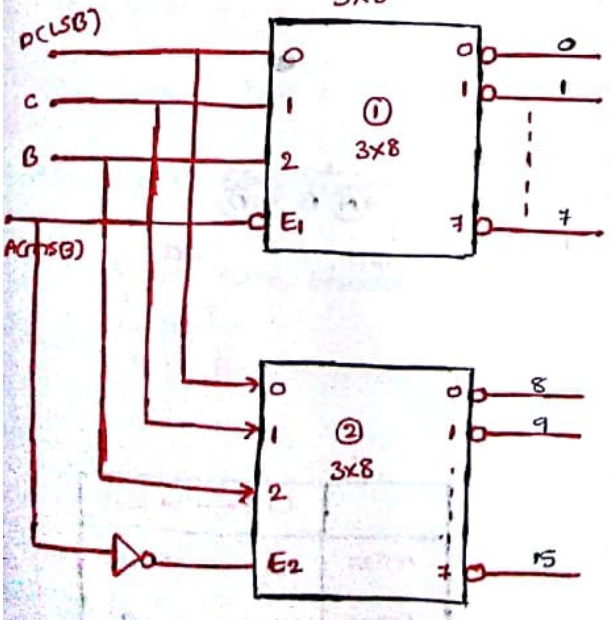
⟶ No. of decoders required = n_1/n_2

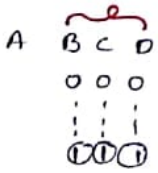
(at level 1)

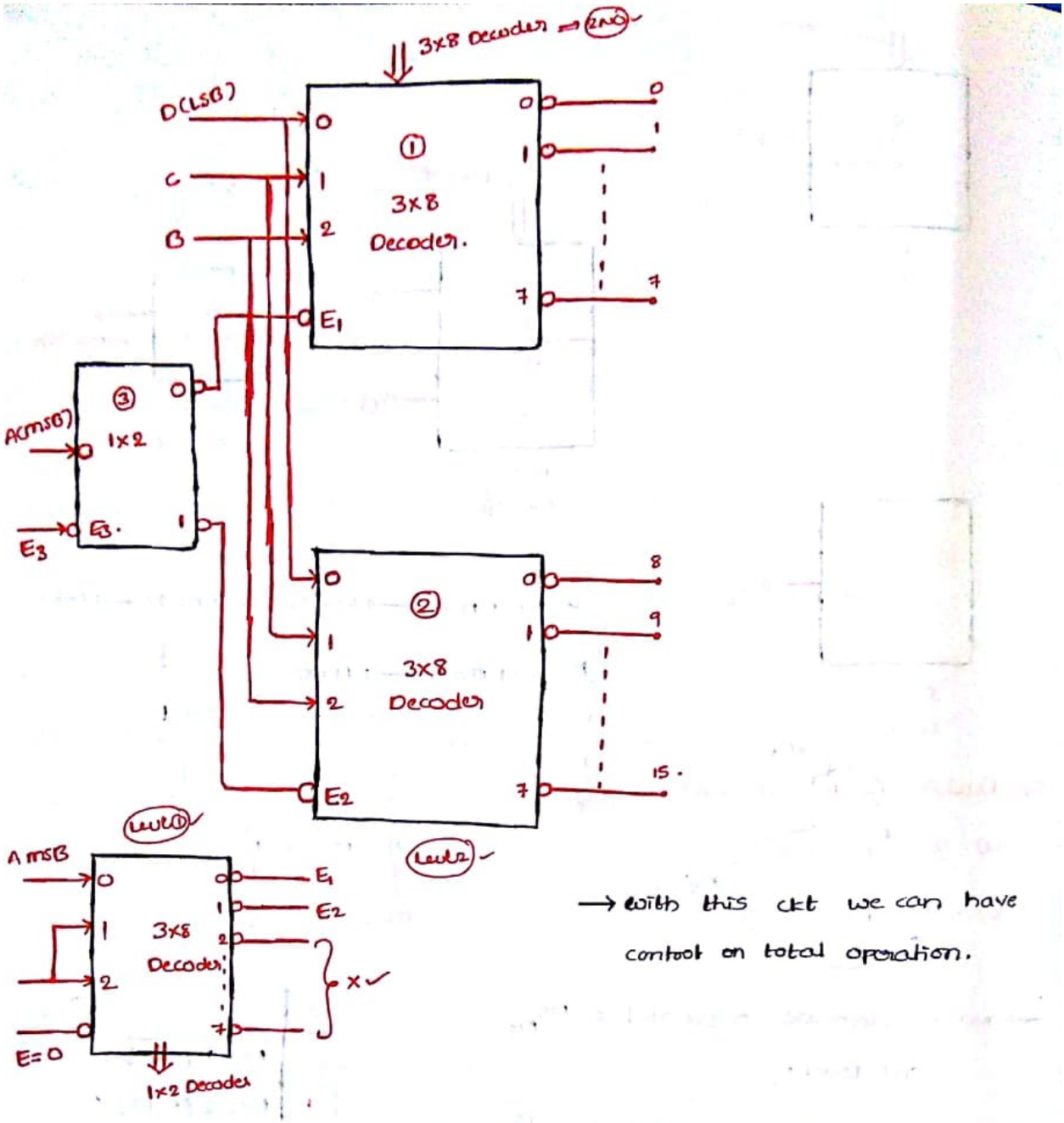
a) No. of decoders required = $16/8 = ②$ ✓

A	
0	Dec 1 [0-7] ✓
1	Dec 2 [8-15] ✓

3x8 Decoder ⇒ 2 NO ✓



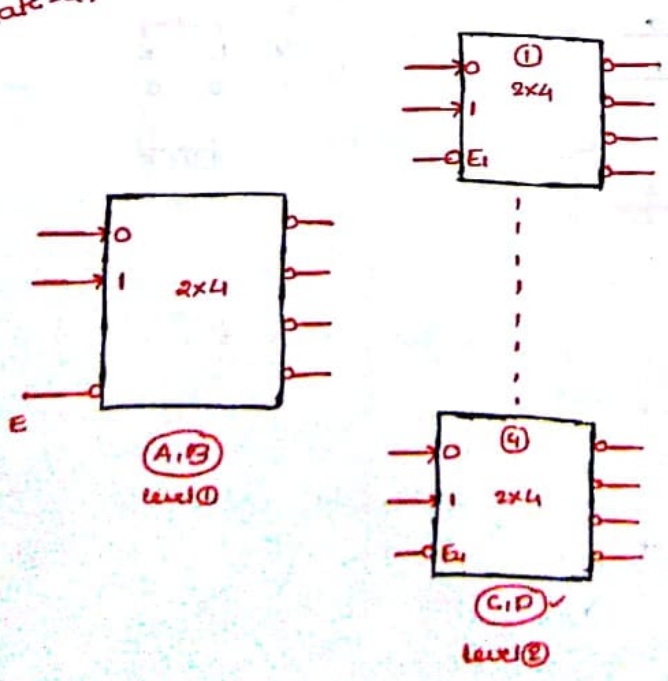


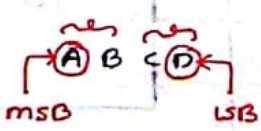


→ with this ckt we can have control on total operation.

b) No. of decoders required = $16/4$ = (4)

Gate-01





	L_1 ⟶ Last
MUX	LSB ⟶ MSB
Decod-er	MSB ⟵ LSB

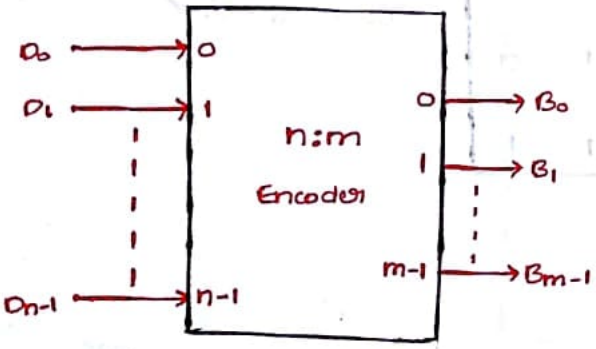
Encoder:

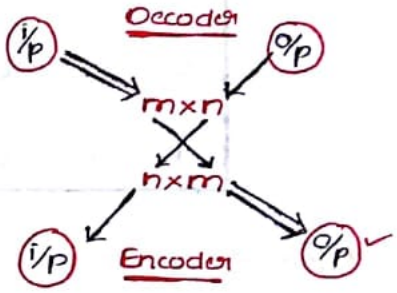
→ The opposite of the "Decoding" process is called "Encoding" and is performed by a logic circuit called an "Encoder".

→ An encoder has 2^n i/p lines, only one of which is activated at a given time and produces an n bit o/p code, depending on which i/p is activated.

m - i/p's — 'n' o/p's.

∴ Here the i/p's are active-High, which means they are normally low.





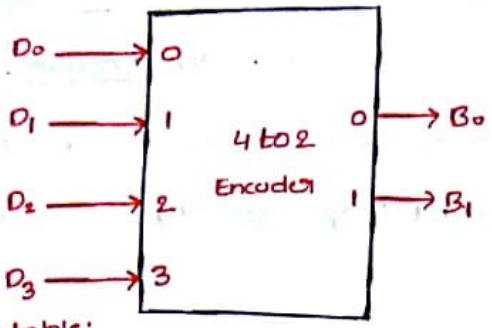
$$\therefore 2^m \geq n$$

(m) → smallest possible integer

Types of Encoders:

01. 4 to 2 line Encoder (Nibble to Binary)
02. 8 × 3 line Encoder (octal to Binary)
03. 10 : 4 line Encoder (Decimal to BCD)
04. 16 × 4 line Encoder (Hex-Decimal to Binary)

4 to 2 line Encoder:



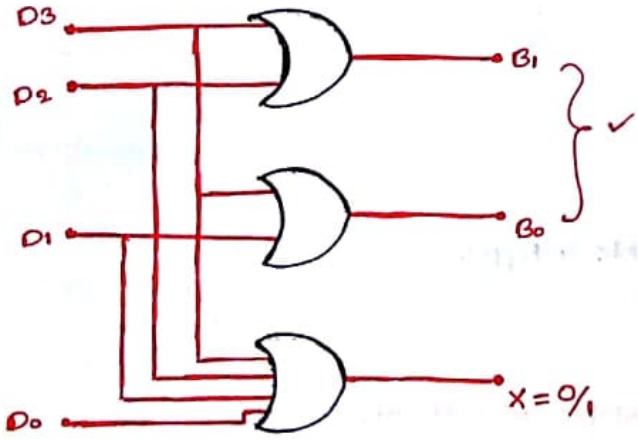
Truth table:

B_3	D_2	D_1	D_0	B_1	B_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

$B_1 = D_3 + D_2$ ✓

$B_0 = D_3 + D_1$ ✓

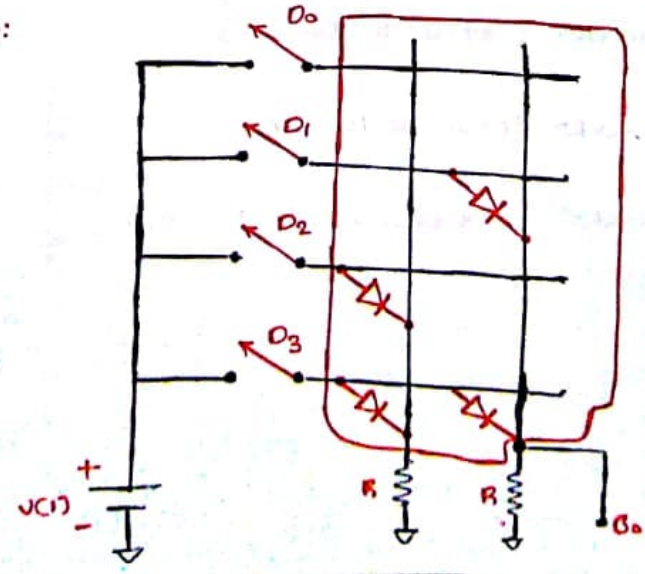
Logic diagram:

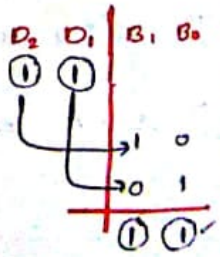


✓

→ We can design → circuits by using multimeter transistors

Using diodes:





✓

→ Encoder is used as interface b/w keyboard and C.P.U

priority Encoder: (Low priority):

→ This is the modified version of simple encoder which elemnates the draw back of encoder (when more than one i/p is activated at a time)

→ A "priority Encoder" includes the necessary logic to ensure that when ② or more i/p's are activated, the o/p code will corres-pond to the "Highest-number-i/p"

D_3	D_2	D_1	D_0	B_1	B_0
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

$B_1 = D_3 + \bar{D}_3 D_2$

$B_1 = D_3 + D_2$

0 1 = 1 ✓

$B_0 = D_3 + \bar{D}_3 \bar{D}_2 D_1$

$B_0 = D_3 + \bar{D}_2 D_1$

0 0 = 0 ✓

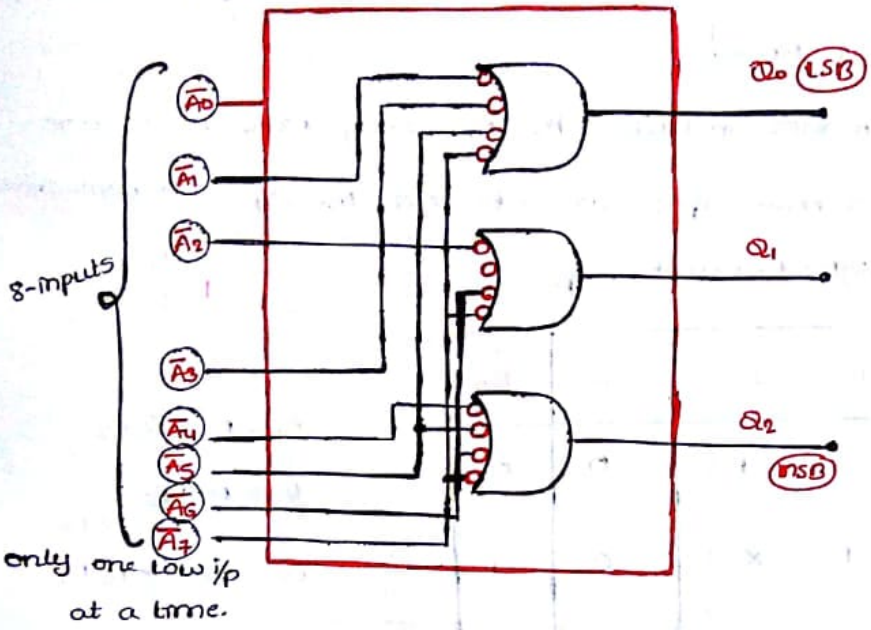
(ii) High priority:

D_3	D_2	D_1	D_0	B_1	B_0
X	X	X	1	0	0
X	X	1	0	0	1
X	1	0	0	1	0
1	0	0	0	1	1

② Octal-to-Binary Encoder: (8 to 3)

i.e 8 line to 3-line Encoder

⟶ 8×3 encoder with active low i/p's



Truth Table:

i/p's								o/p's		
$\bar{A}_0$	$\bar{A}_1$	$\bar{A}_2$	$\bar{A}_3$	$\bar{A}_4$	$\bar{A}_5$	$\bar{A}_6$	$\bar{A}_7$	Q_2	Q_1	Q_0
×	1	1	1	1	1	1	1	0	0	0
×	0	1	1	1	1	1	1	0	0	1
×	1	0	1	1	1	1	1	0	1	0
×	1	1	0	1	1	1	1	0	1	1
×	1	1	1	0	1	1	1	1	0	0
×	1	1	1	1	0	1	1	1	0	1
×	1	1	1	1	1	0	1	1	1	0
×	1	1	1	1	1	1	0	1	1	1

$Q_0 = \bar{A}_1 + \bar{A}_3 + \bar{A}_5 + \bar{A}_7$ ✓

$Q_1 = \bar{A}_2 + \bar{A}_3 + \bar{A}_6 + \bar{A}_7$ ✓

$Q_2 = \bar{A}_4 + \bar{A}_5 + \bar{A}_6 + \bar{A}_7$ ✓

Problems

01. The minimal function that can detect a "divisible by 3" 8421 BCD code digit (representation is $D_8\ D_4\ D_2\ D_1$) is given by:

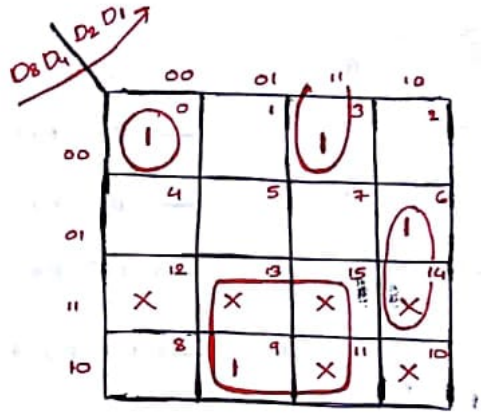
a) $D_8D_1 + D_4D_2 + \bar{D}_8D_2D_1$ ✓ b) $D_8D_1 + D_4D_2\bar{D}_1 + \bar{D}_4D_2D_1 + \bar{D}_8\bar{D}_4\bar{D}_2\bar{D}_1$

c) $D_8D_1 + D_4D_2 + \bar{D}_8\bar{D}_4\bar{D}_2\bar{D}_1$ d) $D_4D_2\bar{D}_1 + D_4D_2D_1 + D_8\bar{D}_4D_2D_1$

Sol: BCD → valid (10) demds

not valid (6) ✓

D.NO.	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)



$Y = \bar{D}_8\bar{D}_4\bar{D}_2\bar{D}_1 + \bar{D}_4D_2D_1 + D_4D_2\bar{D}_1 + D_8D_1$

02. A 2-bit binary multiplier can be implemented using.

a) 2 i/p ANDs only.

b) 2 i/p X-OR's & 4-i/p AND gates only. ✓

c) two (2) i/p NORs and one x-NOR gate.

d) XOR gates and shift registers.

Sol:

X = 10 → (2)

Y = 01 → (1) =(2)

10 — And operation

00 → EXOR operation

010 =(2) ✓

03. For the circuit shown in the following I_0-I_3 are i/p's to the 4:1 multiplexer. R(MSB) and S are control bits.

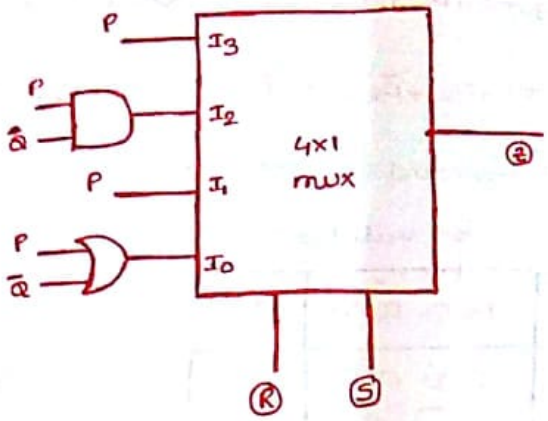
→ the o/p (Z) can be represented by

a) $PQ + P\bar{Q}S + \bar{Q}\bar{R}\bar{S}$

b) $P\bar{Q} + PQ\bar{R} + P\bar{Q}\bar{S}$

c) $P\bar{Q}\bar{R} + \bar{P}QR + PQRS + \bar{Q}\bar{R}\bar{S}$

d) $PQ\bar{R} + PQR\bar{S} + P\bar{Q}\bar{R}\bar{S} + \bar{Q}R\bar{S}$



Sol: $(P+\bar{Q})$

	R	S	
$I_0 = P+\bar{Q}$	0	0	$\longrightarrow \bar{R}\bar{S}(P+\bar{Q})$
$I_1 = P$	0	1	$\longrightarrow \bar{R}S(P)$
$I_2 = P+Q$	1	0	$\longrightarrow R\bar{S}(P+Q)$
$I_3 = P$	1	1	$\longrightarrow RS(P)$

$Z = P\bar{R}\bar{S} + PQR\bar{S} + PR\bar{S} + (P+\bar{Q})\bar{R}\bar{S}$

$Z = \Sigma m\,(0,1,8,9,11,12,13,14,15)$

$\longrightarrow \bar{R}\bar{S}P + \bar{R}\bar{S}\bar{Q} + \bar{R}SP + R\bar{S}P + R\bar{S}\bar{Q} + RSP$

$\longrightarrow \bar{S}P(\bar{R}+R) + \bar{S}P + \bar{R}\bar{S}\bar{Q}$

$\longrightarrow \bar{S}P(\bar{R}+Q) + \bar{S}P + \bar{R}\bar{S}\bar{Q}$

$\bar{S}P\bar{R} + \bar{S}PQ + \bar{S}P + \bar{R}\bar{S}\bar{Q}$

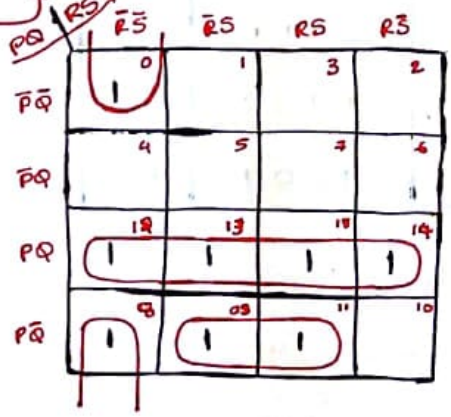
$P(\bar{S}\bar{R} + \bar{S}) + \bar{S}PQ + \bar{R}\bar{S}\bar{Q}$

$P(\bar{S}+\bar{R}) + \bar{S}PQ + \bar{R}\bar{S}\bar{Q}$

$P\bar{S} + P\bar{R} + \bar{S}PQ + \bar{R}\bar{S}\bar{Q}$

$P(\bar{S}+Q) + P\bar{R} + \bar{R}\bar{S}\bar{Q}$

$PQ + P\bar{S} + P\bar{R} + \bar{R}\bar{S}\bar{Q}$



$Z = PQ + P\bar{Q}S + \bar{Q}\bar{R}S$

Conv:

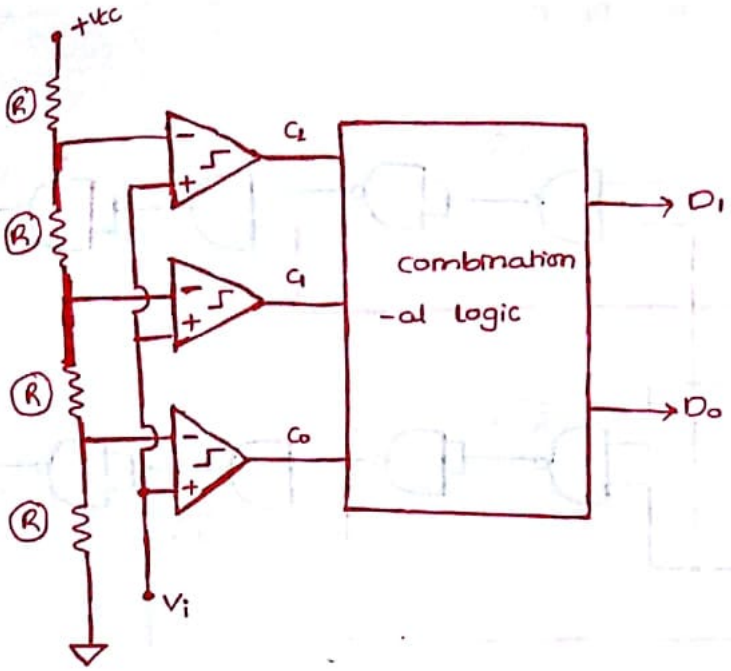
04) The circuit diagram of a 2-bit A to D converter is shown in fig. The combinational logic is to be designed to provide a natural binary representation using D_1 & D_0 for the analog i/p V_i. D_1 is to be the most significant bit.

a) Draw the Karnaugh maps for D_1 & D_0 interms of Q_2, Q and Q_0

b) obtain the minimal sum of products expression for D_1 & D_0

c) Realize the logic for D_1 and D_0 using 2-i/p NAND gates only

d) Find the resolution of the A to D converter. (Q-87 (8m))



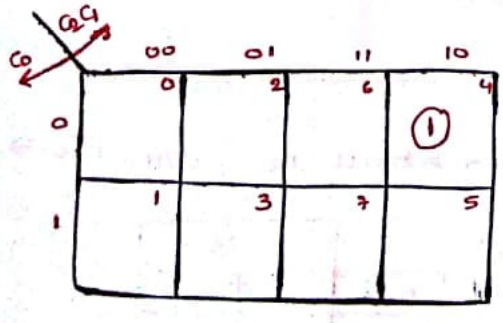
Sol: For the combinational logic circuit D_1 and D_0 are obtained as follows

D.NO.	C_2	C_1	C_0	D_1	D_0
1	0	0	1	0	0
2	0	1	0	0	1
4	1	0	0	1	0

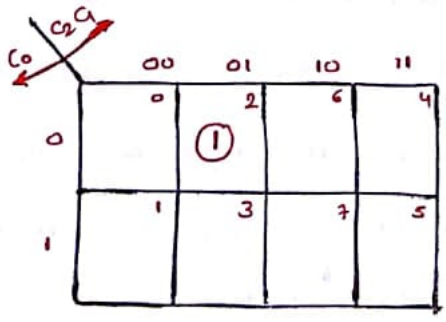
$D_1(C_2, C_1, C_0) = \Sigma m(4)$

$D_0(C_2, C_1, C_0) = \Sigma m(2)$

a) K-maps:



$D_1 = C_2\bar{C}_1\bar{C}_0$



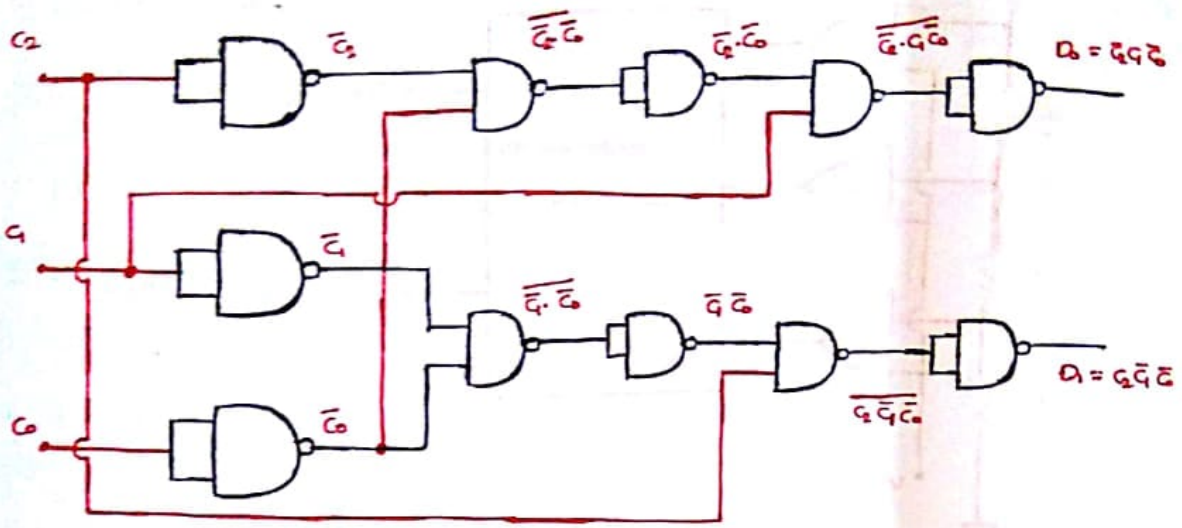
$D_2 = \bar{C}_2C_1\bar{C}_0$

b) minimum sum of products expressions

$D_1 = C_2\bar{C}_1\bar{C}_0$

$D_2 = \bar{C}_2C_1\bar{C}_0$

c) Logic NAND gates for D_1 & D_0



d) Resolution $= \frac{1}{2^n - 1} = \frac{1}{2^2 - 1} = \frac{1}{3} = 0.33.$

05) A chemical reactor has three sensors indicating the following conditions

i) pressure (P) is low @ high.

ii) Temperature (T) is low @ high.

iii) Liquid levels (L) is low @ high.

It has ② controls - Heater (H) which is either on @ off and an inlet (V) which is open @ close. The controls are operated as per table.

a) using the convention High = 1, low = 0, on = 1, off = 0, open = 1 & closed = 0 draw the k-maps for Ⓗ & Ⓥ.

b) obtain the minimal product of sums expressions for Ⓗ & Ⓥ

c) Realize the logic for Ⓗ & Ⓥ using two 4-bit/pt mux with T & L as control i/p's. use Ⓣ as MSB

I/p's			O/p's	
P	T	L	H	V
low	low	low	off	open
low	low	high	on	closed
low	high	low	off	open
low	high	high	off	closed
high	low	low	on	open
high	low	high	on	closed
high	high	low	off	closed
high	high	high	x	x

Sol: a) High = 1

Low = 0

ON = 1

OFF = 0

open = 1

closed = 0

I/P's			O/P's	
P	T	L	H	V
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	x	x

V \ PT	00	01	11	10
0	0	0	0	
1		0	x	

$H = \bar{T}(P+L)$

V \ PT	00	01	11	10
0			0	
1	0	0	x	0

$V = \bar{L}(\bar{T}+\bar{P})$

b)

(H)

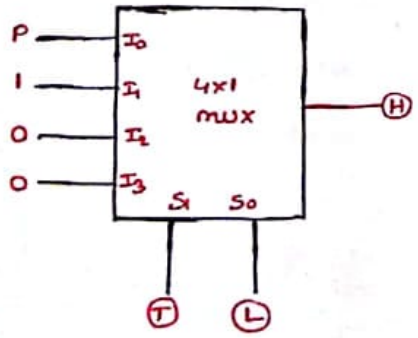
	S_1 (T)	S_0 (L)			
	I_0	I_1	I_2	I_3
$\bar{P}$ 0	0	(1)	2	3
P 1	(4)	(5)	6	7
	P	1	0	0

(V)

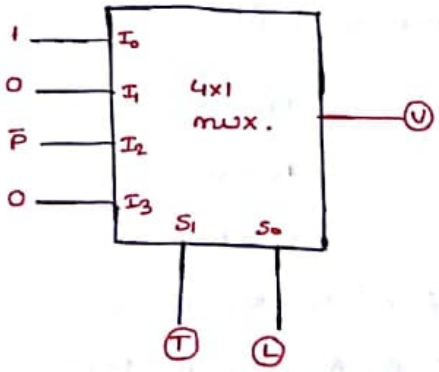
	S_1 (T)	S_0 (L)			
	I_0	I_1	I_2	I_3
$\bar{P}$ 0	(0)	1	(2)	3
P 1	(4)	5	6	7
	1	0	$\bar{P}$	0

c) 4x1 mux to realise H & V

(H)



(V)



4-96

06) It is desired to generate the following three boolean functions

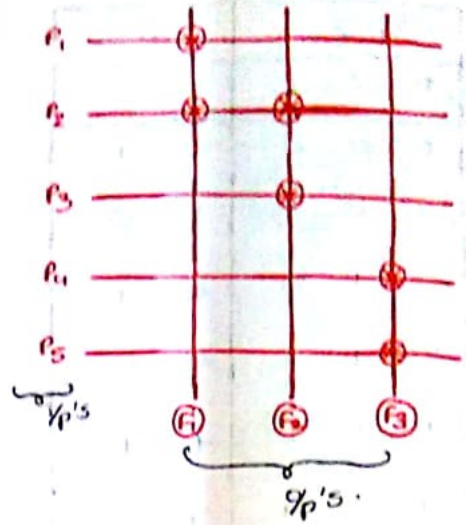
$F_1 = a\bar{b}c + \bar{a}b\bar{c} + bc$

$F_2 = a\bar{b}c + ab \pm \bar{a}\bar{b}c$

$F_3 = \bar{a}\bar{b}\bar{c} + abc + \bar{a}c$

By using an OR gate array as shown in figure, where P_1 to P_5 are the product terms in one or more of the variables $a, \bar{a}, b, \bar{b}, c, \bar{c}$

write down the terms ① ② ③ ④ ⑤

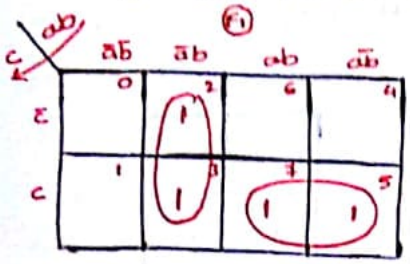


Sol: $f_1 = a\bar{b}c + \bar{a}b\bar{c} + bc$ ⟶ (1)

$f_2 = a\bar{b}c + ab + \bar{a}b\bar{c}$ ⟶ (2)

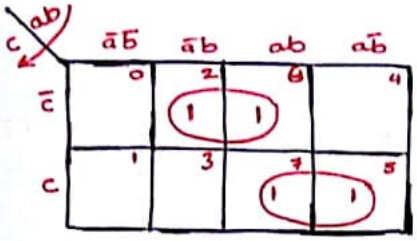
$f_3 = \bar{a}b\bar{c} + abc + \bar{a}c$ ⟶ (3)

(f1)



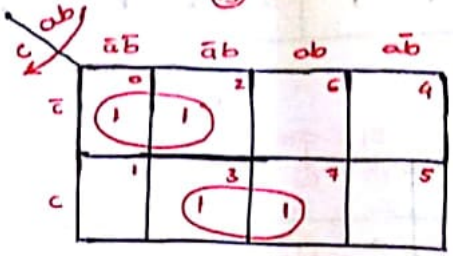
$f_1 = ac + \bar{a}b$ ⟶ (a)

(f2)



$f_2 = b\bar{c} + ac$ ⟶ (b)

(f3)



$f_3 = bc + \bar{a}\cdot\bar{c}$ ⟶ (c)

from OR gate functions are

$f_1 = P_1 + P_2$ ⟶ (I)

$f_2 = P_3 + P_4$ ⟶ (II)

$f_3 = P_4 + P_5$ ⟶ (III)

compare I, II, III and a, b, c

then $P_1 = \bar{a}b$

$P_2 = ac$

$P_3 = b\bar{c}$

If $P_4 = bc$ then $P_5 = \bar{a}\bar{c}$

If $P_4 = \bar{b}\bar{c}$ then $P_5 = bc$ ✓

5-97 EE:

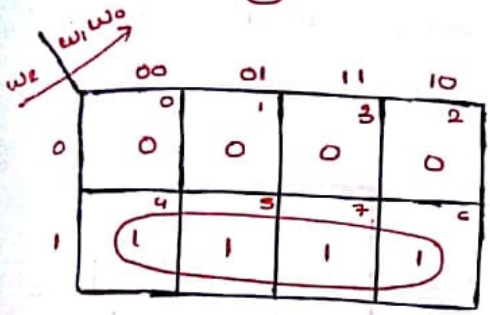
07. A 3-I/p 2-O/p priority encoder has the following truth table x's indicate don't care conditions. Realize the logic using NAND gates and inverters.

ω_2	ω_1	ω_0	y_1	y_0
0	0	0	0	0
0	0	1	0	0
0	1	x	1	0
1	x	x	1	1

Sol: 3 i/p 2 o/p priority Encoder.

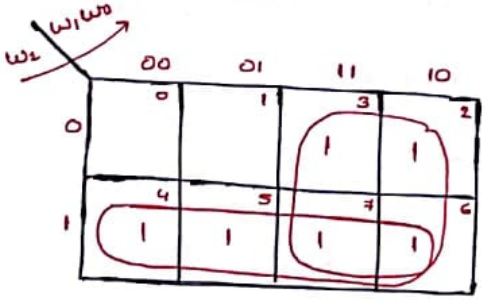
	w_2	w_1	w_0	y_1	y_0
m_0	0	0	0	0	0
m_1	0	0	1	0	0
m_2-m_3	0	1	x	1	0
m_4-m_7	1	x	x	1	1

(Y0)



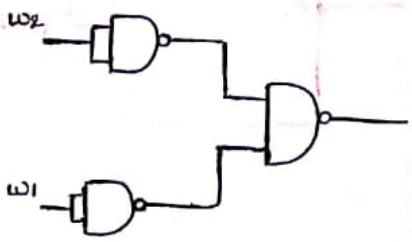
$Y_0 = w_2$

(Y1)



$Y_1 = w_2 + w_1$

i)



$Y_1 = \overline{\overline{w_2}.\overline{w_1}} = w_2 + w_1$

ii) w_2 —— Y_0 ✓

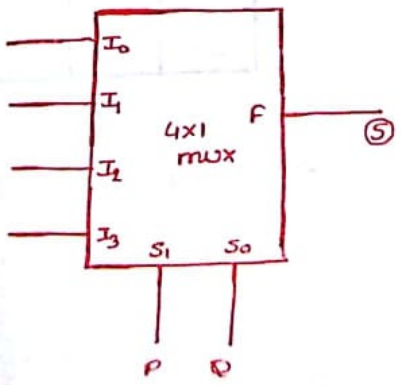
Q8) figure 4x1 mux to be used to implement the sum (S) of a 1-bit full adder with i/p bits P & Q and the carry i/p Cin. which of the following combinations of i/p's to I_0, I_1, I_2 & I_3 of the mux will realize the sum (S)?

a) $I_0 = I_1 = C_{in}$; $I_2 = I_3 = \overline{C_{in}}$

b) $I_0 = I_1 = \overline{C_{in}}$; $I_2 = I_3 = C_{in}$

✓ c) $I_0 = I_3 = C_{in}$; $I_1 = I_2 = \overline{C_{in}}$

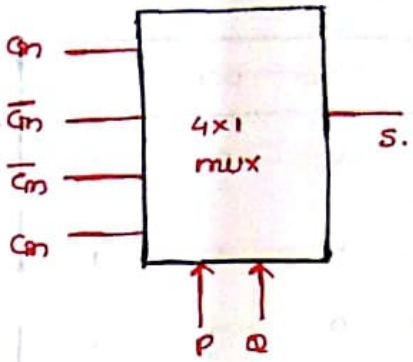
d) $I_0 = I_3 = \overline{C_{in}}$; $I_1 = I_2 = C_{in}$



Sol: for full adder, the sum of o/p is $S(P,Q,C_{in}) = \Sigma m(1,2,4,7)$

to implement this sum of minterm expression using 4x1 mux ✓

	I_0	I_1	I_2	I_3
$\overline{Cm}$	0	(2)	(4)	6
Cm	(1)	3	5	(7)
	Cm	$\overline{Cm}$	$\overline{Cm}$	Cm



09) A 3 line to 8 line decoder, with active low o/p's is used to implement a 3-variable Boolean function as shown in figure.

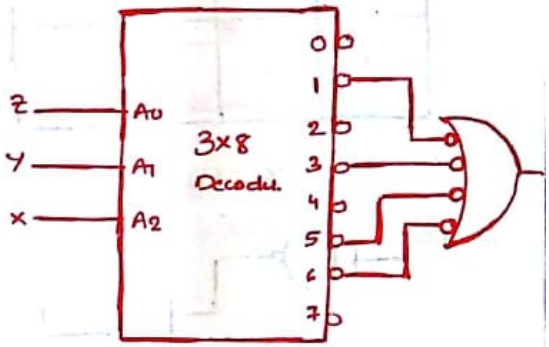
→ The simplified form of Boolean fun F(A,B,C) implemented in "product of sum" form. will be (G-08 EE)

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

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

c) $(\bar{x}+\bar{y}+\bar{z})(\bar{x}+y+z)(x+\bar{y}+z)(x+y+\bar{z})$

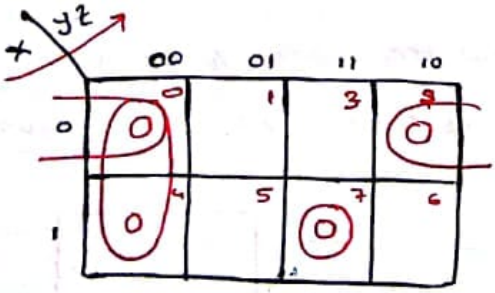
d) $(\bar{x}+\bar{y}+\bar{z})(\bar{x}+y+\bar{z})(x+y+z)(x+\bar{y}+\bar{z})$



Sol: The Negative-OR gate = NAND gate

$F(x,y,z) = \Sigma m(1,3,5,6)$

$F(x,y,z) = \pi M(0,2,4,7)$



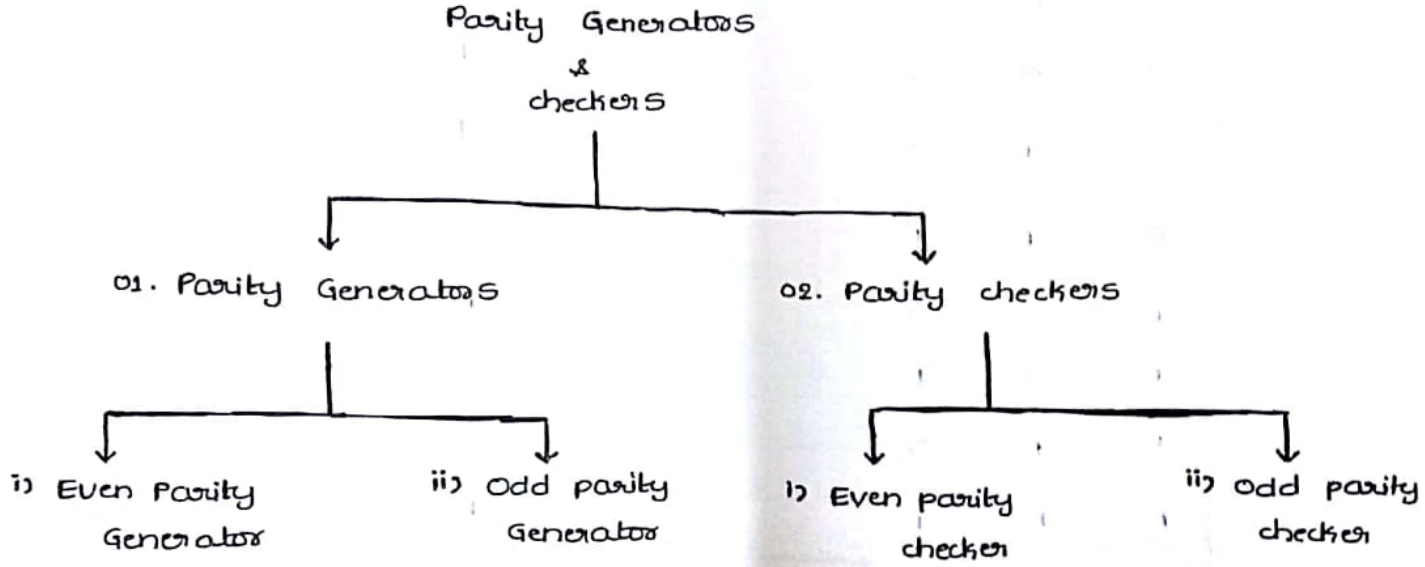
$F = (y+z)(\bar{x}+\bar{y}+\bar{z})(x+z)$

Parity Generators & checkers:

Parity Bit: In digital systems, when binary data is transmitted and processed, data may be subjected to noise so that such noise can alter 0's $\xrightarrow{to}$ 1's

1's $\xrightarrow{to}$ 0's.

To detect the errors, we have to add the parity bit to the data code word. Now the msg containing the data bits along with Parity bit is transmitted from transmitter node to receiver node.



01) Parity Generator: A parity Generator is a combinational logic circuit that generates the parity bit in the transmitter.

→ In even parity, the added parity bit will make the total number of 1's even.

→ In odd parity, the added parity bit will make the total number of 1's odd.

Basic Principle:

the logic involved in the implementation of parity circuits is that sum of odd number of 1's is always "1".

sum of even number of 1's is always "0".

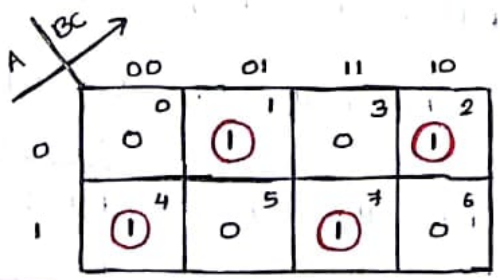
→ For this detection and correction can be implemented by using "EX-OR & EX-NOR" gates.

i) Even Parity Generator:

In even parity bit scheme, the parity bit is "0" if there are even number of 1's in the data stream. and the parity bit is "0" if there are odd number of 1's in the data stream.

3-bit msg			Even parity bit generator (P)
A	B	C	P
0	0	0	0
0	0	0	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

K-map for (P):



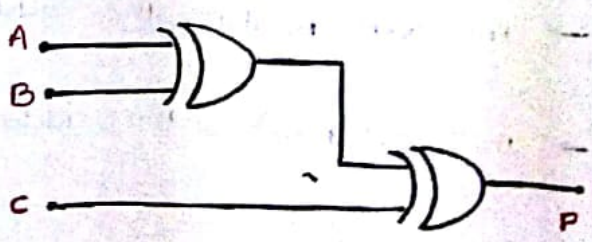
$P = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

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

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

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

Even Parity Generator ckt:



A, B, C, P ⇒ 4-bit even parity code.

Odd parity Generator:

The added parity bit will make the total number of 1's odd.

In odd parity bit scheme,

the parity bit is "1" if there are even number of 1's in the data stream

the parity bit is "0" if there are odd number of 1's in the data stream

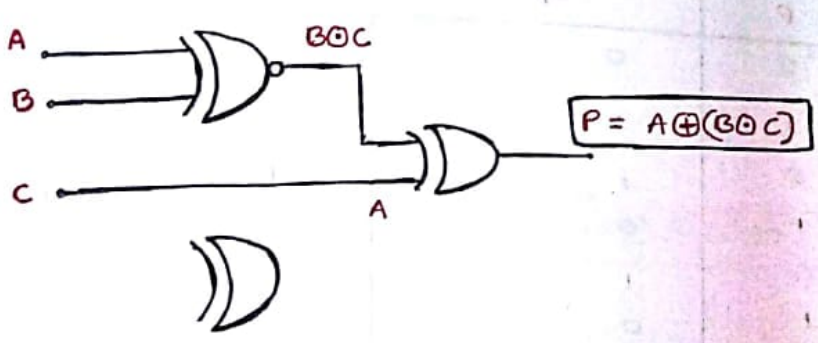
3 bit msg A	B	C	odd parity bit Generator (P)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

K-map for (P):

A \ BC	00	01	11	10
0	1 (0)	0 (1)	1 (3)	0 (2)
1	0 (4)	1 (5)	0 (7)	1 (6)

$P = A \oplus (B \odot C)$

Odd Parity Generator Ckt:



for Even Parity Generator ⟶ EX-OR

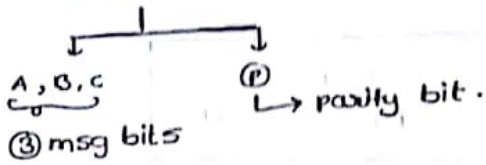
for odd Parity Generator ⟶ EX-OR & EX-NOR

02) Parity checker: A circuit that checks the parity in the Receiver is called parity checker.

→ for Even Parity checker ⟶ no. of i/p bits must always be even.

for odd Parity checker ⟶ no. of i/p bits must always be odd.

→ The Parity checker 'ckt having ④ i/p's

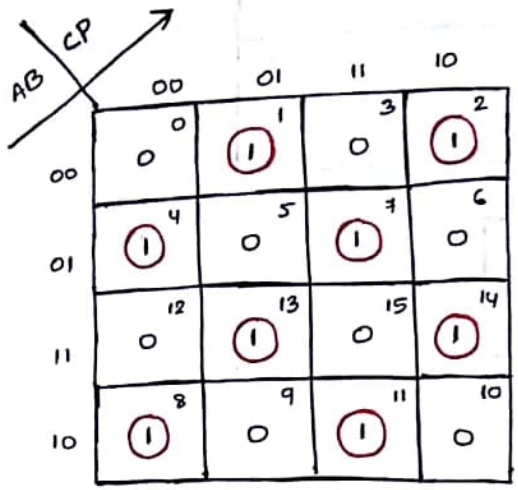


i) Even Parity checker: ④ bits are applied as i/p to the parity checker circuit which checks the possibility of error on the data. Since the data transmitted with even parity, four bit's received at ckt must have an even number of 1's.

At receiver	PEC
Even no. of 1's	1 ⟶ no error.
odd no. of 1's	0 ⟶ Error occured

	4 bit RX msg				Parity Error Check
	A	B	C	P	Cp
0	0	0	0	0	0
1	0	0	0	1	1
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	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	0	1
14	1	1	1	1	1
15	1	1	1	1	0

K-map for C_P :



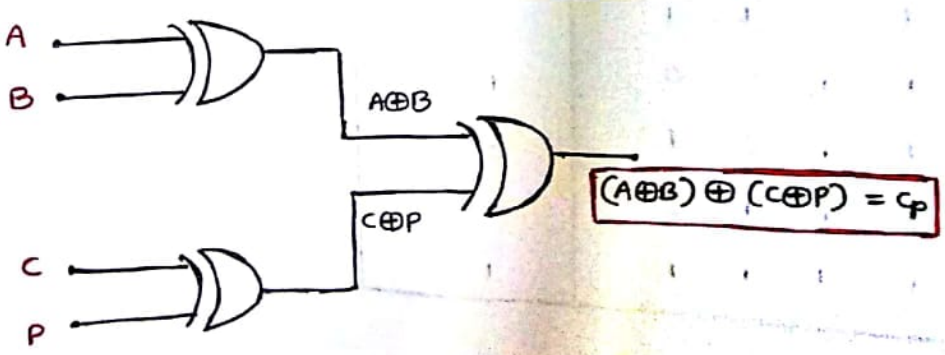
$$C_P = PEC = \bar{A}\bar{B}[\bar{C}P + C\bar{P}] + \bar{A}B[\bar{C}\bar{P} + CP] + AB[\bar{C}P + C\bar{P}] + A\bar{B}[\bar{C}\bar{P} + CP]$$

$$C_P = \bar{A}\bar{B}[C \oplus P] + \bar{A}B[\overline{C \oplus P}] + AB[C \oplus P] + A\bar{B}[\overline{C \oplus P}]$$

$$C_P = [\bar{A}\bar{B} + AB][C \oplus P] + [\bar{A}B + A\bar{B}][\overline{C \oplus P}]$$

$$\boxed{C_P = [A \oplus B] \oplus [C \oplus P]}$$

Logic circuit:

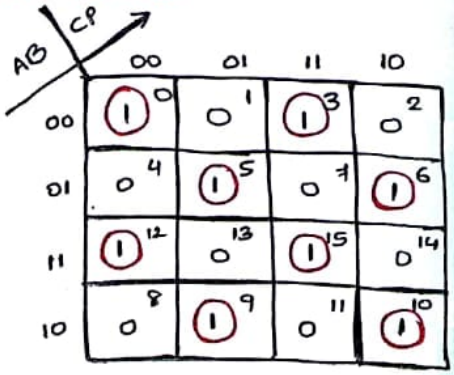


ii) odd Parity checker:

At Rx	PEC (Cp)
Even no. of 1's	0 → Error occurred
odd no. of 1's	1 → no error

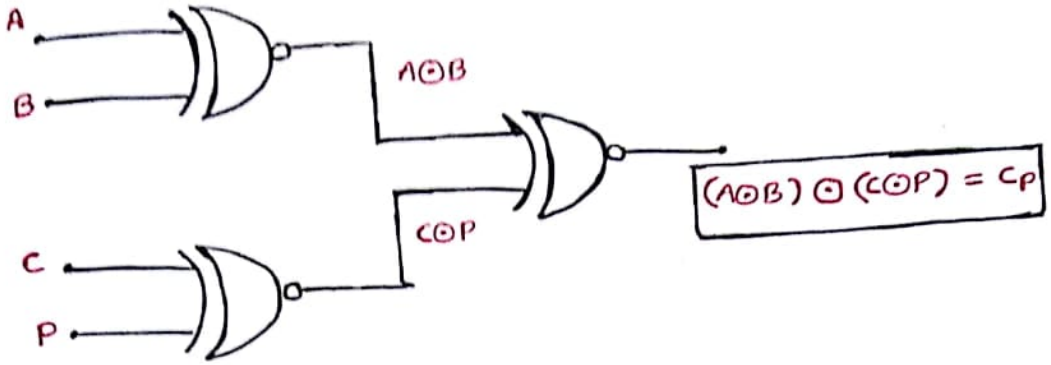
4 bit Received msg				PEC
A	B	C	P	Cp
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

K-map for (Cp):

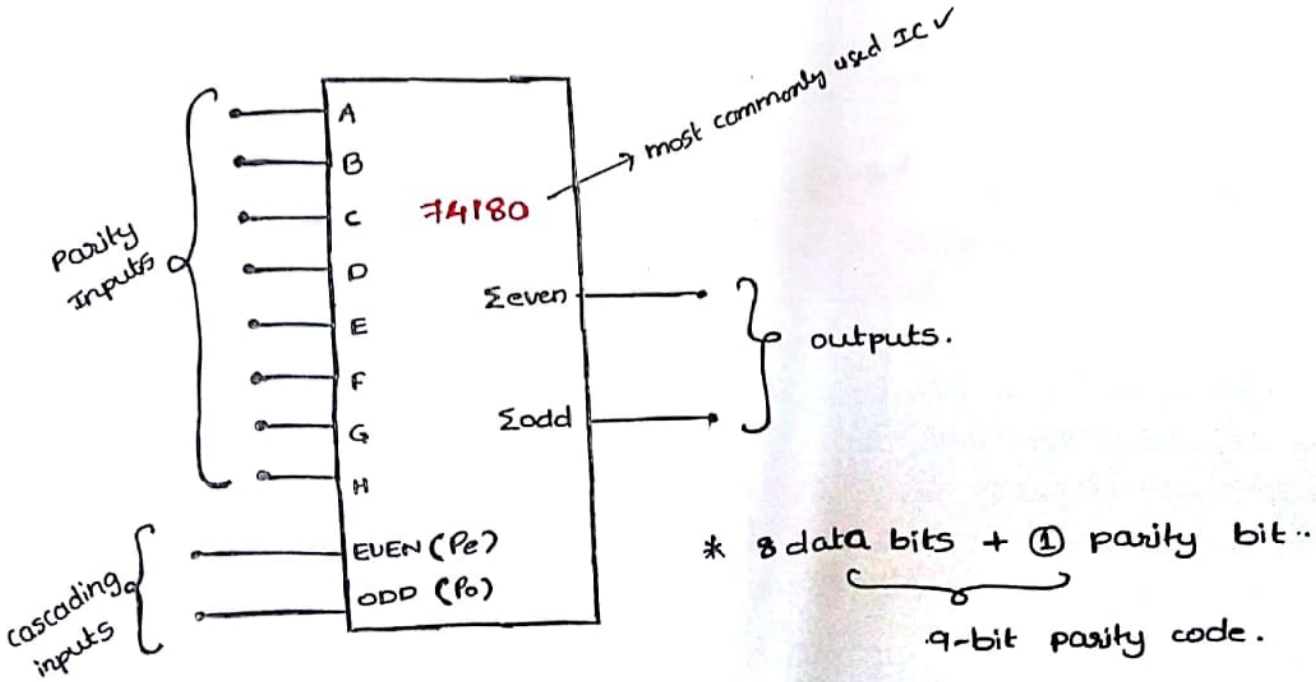


$$PEC = (Cp) = (A \odot B) \odot (C \odot P)$$

Logic circuit :



* Parity Generator / checker IC's :



* 8 data bits + (1) parity bit.

9-bit parity code.

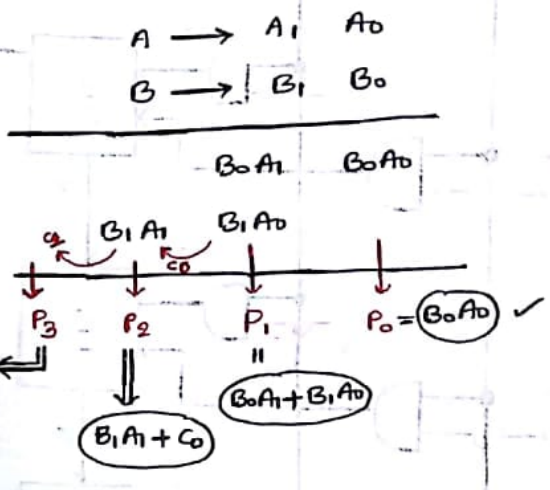
Array Multiplier: An array multiplier is a combinational circuit that is used for the multiplication of two binary numbers by employing an array of full adders and half adders. This array is used for the nearly simultaneous addition of the various product terms involved.

→ To form the various product terms, an array of AND gates is used before the Adder array.

Ex: 2×2 array multiplication

It require ⟹ ④ AND gates

② Adders required.



Hard ware Implementation:

for m×n array multiplier:

required

- array of AND gates → m×n — AND gates
- Array of Adders → (m−1)n Adders
  - ↳ at least (m−2)·n full Adders.
  - rest of (n) can be either half Adder or full adder with $C_{in}=0$.

Delay: m×n bit multiplication:

Here T_A ⟶ AND gate propagation delay.

T_C ⟶ carry propagation delay.

T_S ⟶ sum propagation delay.

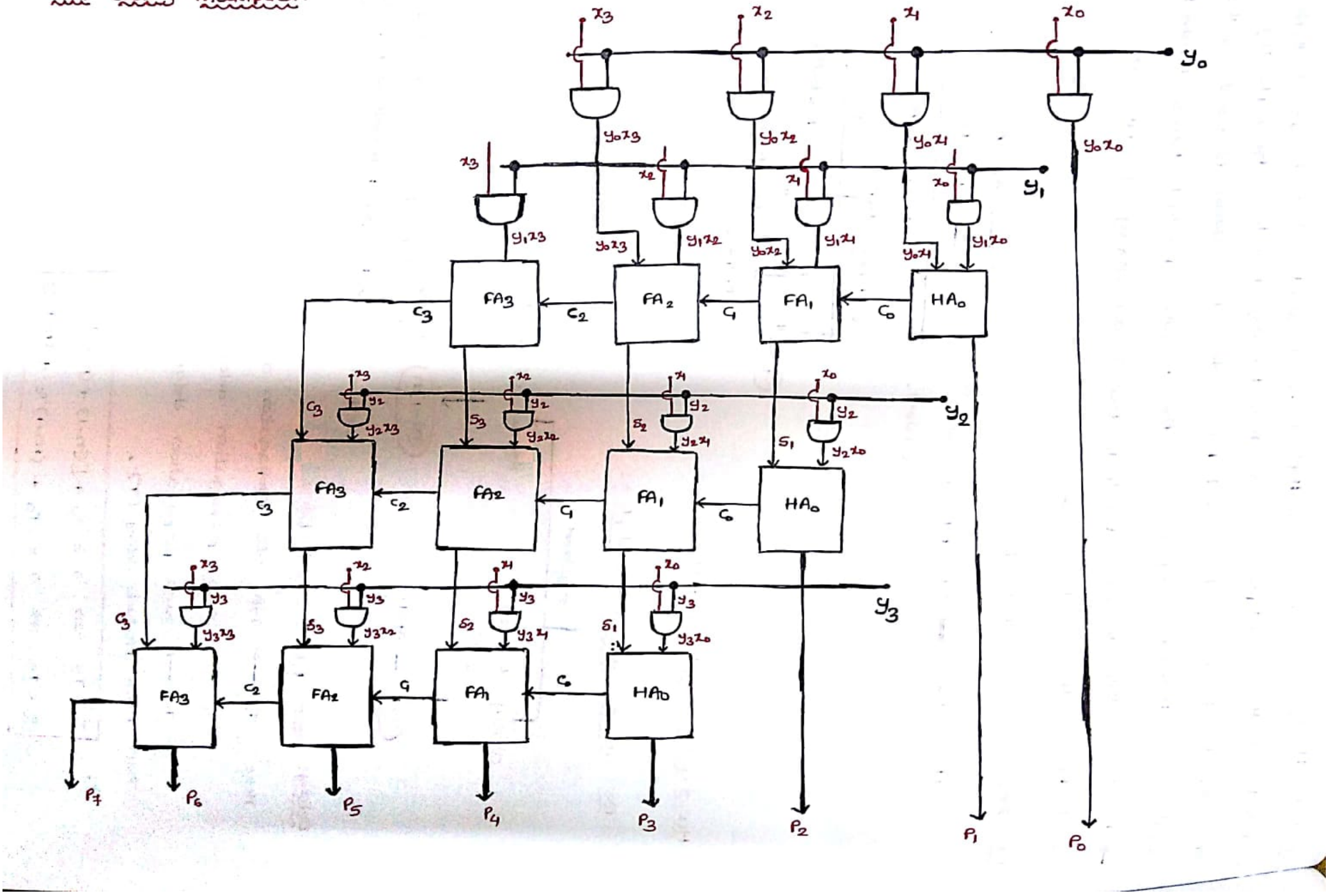
Then final product delay (T),

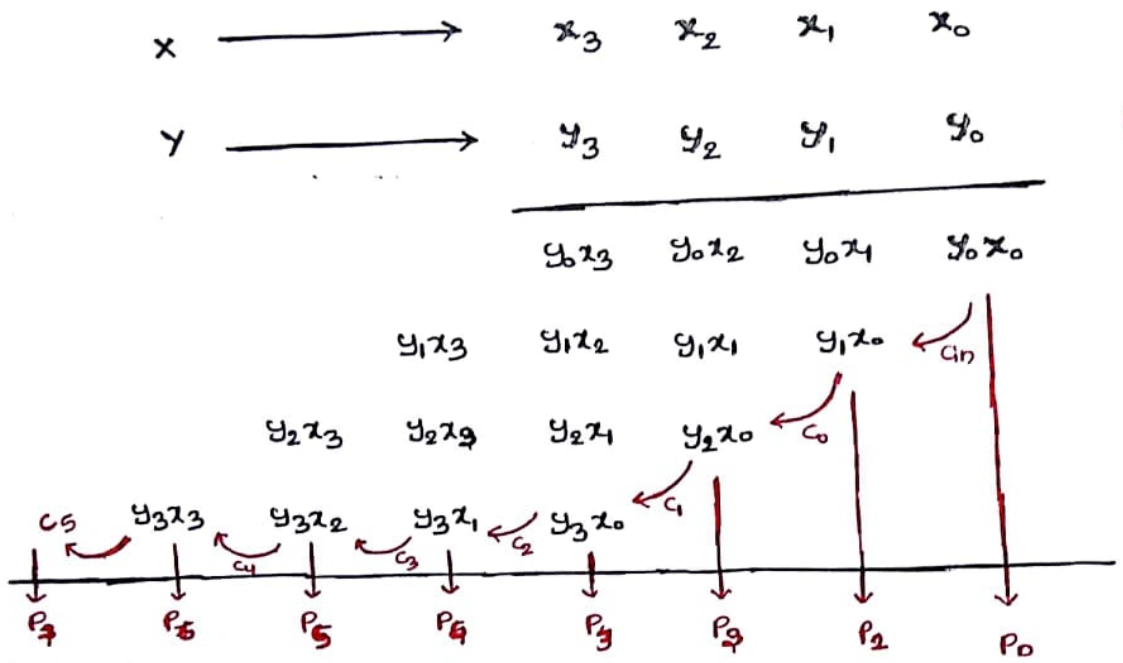
If

$T_C > T_S \Rightarrow T = T_a + [(m-1)+(n-1)]\cdot T_C$

$T_C < T_S \Rightarrow T = T_a + (m-1)T_C + (n-1)T_S$

4x4 array multiplier:





(4×4 array multiplication)