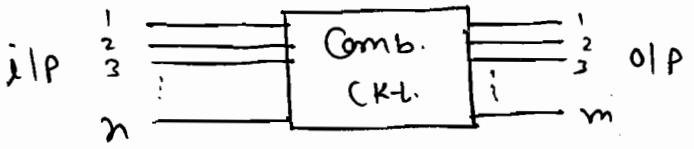
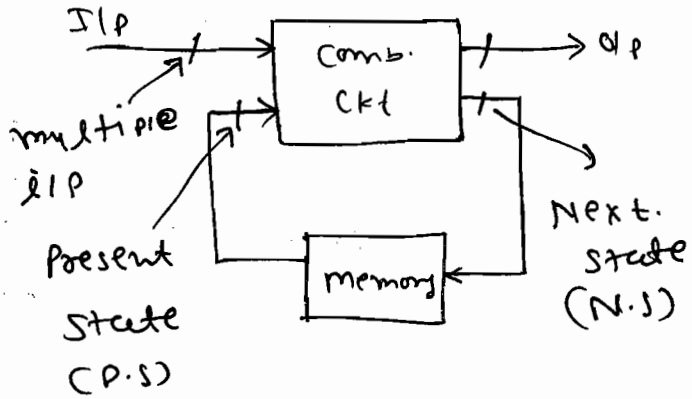
Digital circuits

- Combinational Ckt
- Sequential Ckt.



→ Outputs = F (present I/P)

E.g. = Code Converter,
Adders & Subtractor,
Decoder, MUX,
Mag. Comparators, PLA,
∴ Rom



→ Mealy seq. Ckt.

O/P = F (present state, present inputs)

Next state = F (present state, present i/p.)

→ Moore seq Ckt.

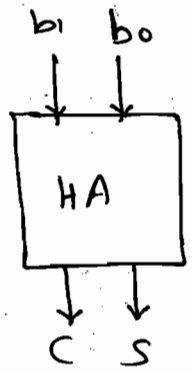
O/P = F (present state only)

N.S. = F (present Inputs, present state).

E.g. Shift register, Counters, calculators, MP, PC.

* Arithmatic Combinational Ckt.

(1) Half Adder:



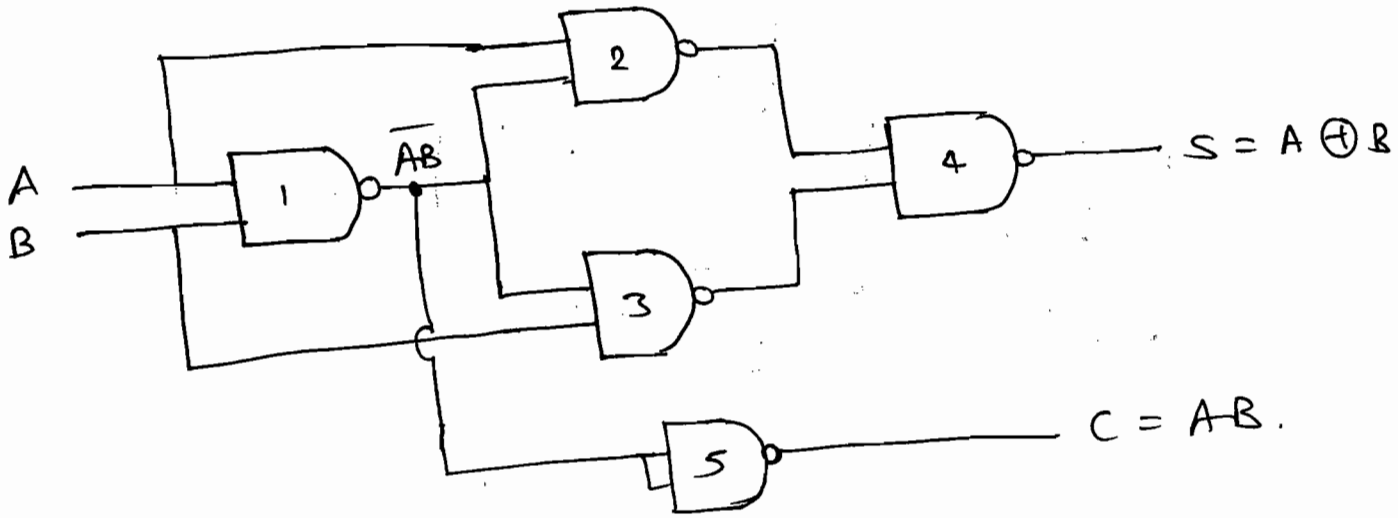
A B	S C
0 0	0 0
0 1	1 0
1 0	1 0
1 1	0 1

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

$S = A \oplus B.$

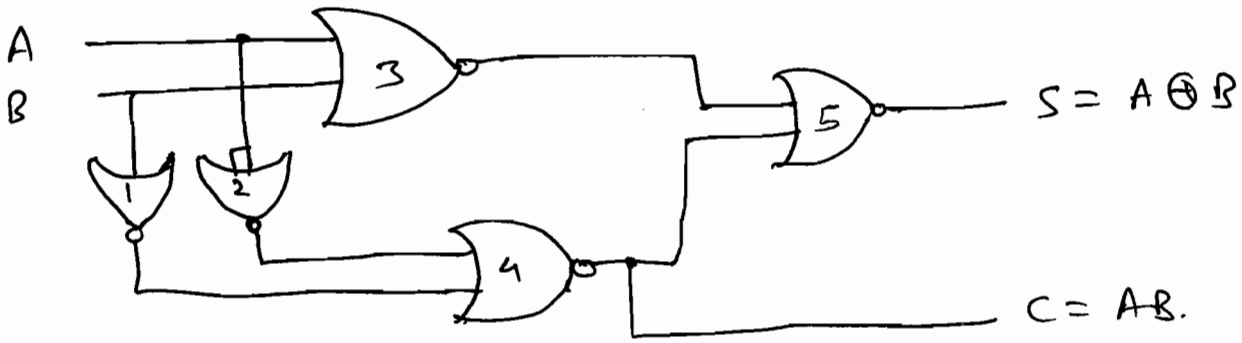
$C = A \cdot B.$

⇒ Half Adder using NAND Gates.



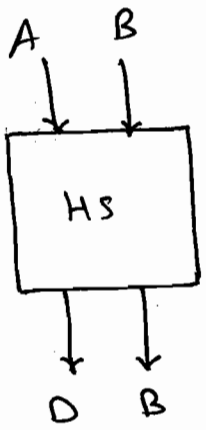
So, (5) NAND gate required

⇒ Half Adder using NOR gate



So, (5) NOR gate required

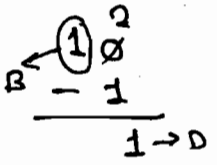
(2) Half Subtractor:



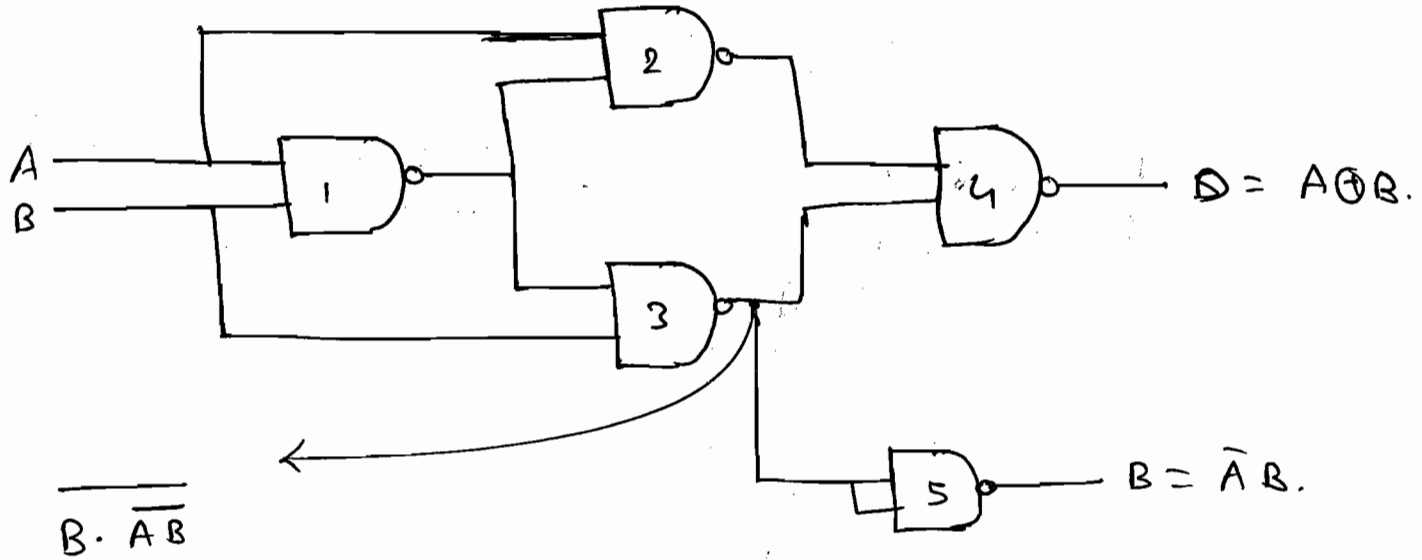
A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$D = A \oplus B$

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



⇒ Half Subtractor using NAND gate only:



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

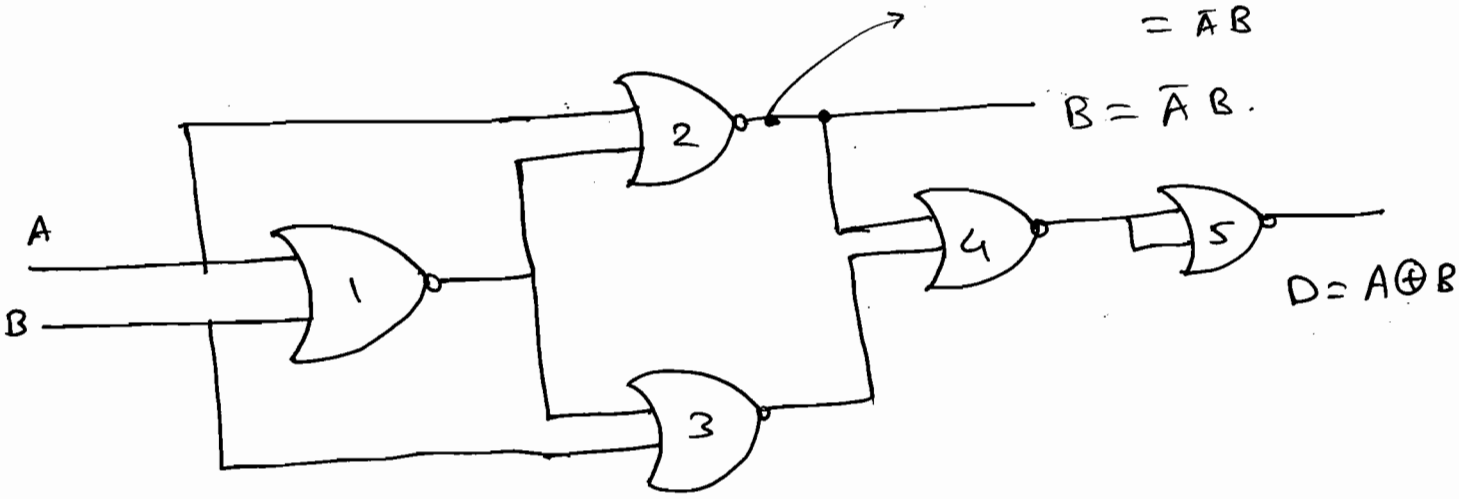
$= \bar{B} + AB$

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

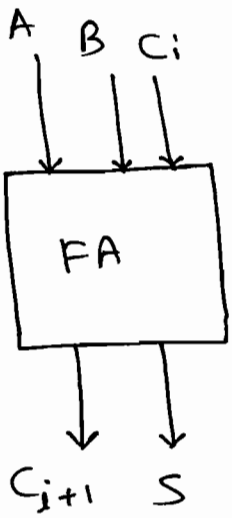
⇒ Half Subtractor using NOR gate only.

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

$= \bar{A}B$

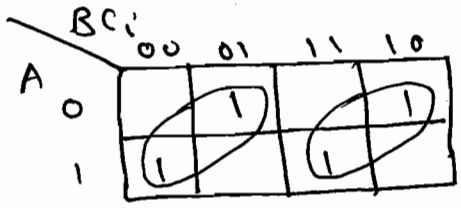


(3) Full Adder: (1 bit adder).



A	B	C_i	S	C_{i+1}	
0	0	0	0	0	
0	0	1	1	0	
0	1	0	1	0	Carry Propogation
0	1	1	0	1	
1	0	0	1	0	
1	0	1	0	1	
1	1	0	0	1	Carry Generation
1	1	1	1	1	

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



$S = A \oplus B \oplus C_i$ (or) $A \odot B \odot C_i$

$$C_{i+1} = \Sigma m(3, 5, 6, 7)$$

$$= \bar{A}BC_i + A\bar{B}C_i + AB\bar{C}_i + ABC_i$$

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

$$\therefore \boxed{C_{i+1} = C_i(A \oplus B) + AB}$$

$$\boxed{C_{i+1} = AB + BC_i + AC_i}$$

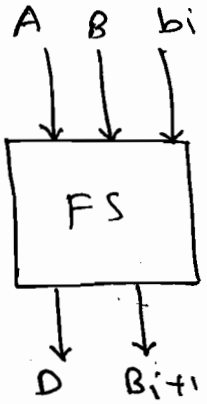
NOTE: Full adder

(i) Require 9 NAND gate

(ii) Require 12 NOR gate.

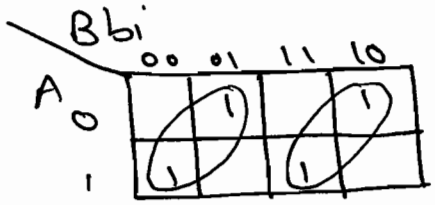
(4) Full Subtractor:

→



A	B	bi	D	bi+1
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

→ $D(A, B, bi) = \Sigma m(1, 2, 4, 7)$.



$$D = A \oplus B \oplus bi \quad (\text{or}) \quad A \odot B \odot bi.$$

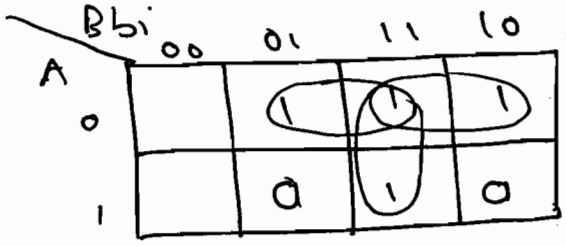
→ $bi(A, B, bi) = \Sigma m(1, 2, 3, 7)$.

$$b_{i+1} = m_1 + m_2 + m_3 + m_7$$

$$= \bar{A}\bar{B}bi + \bar{A}B\bar{bi} + \bar{A}Bbi + ABbi$$

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

$$b_{i+1} = bi(A \odot B) + \bar{A}B.$$



$$\therefore \boxed{b_{i+1} = \bar{A}B + Bbi + \bar{A}bi}$$

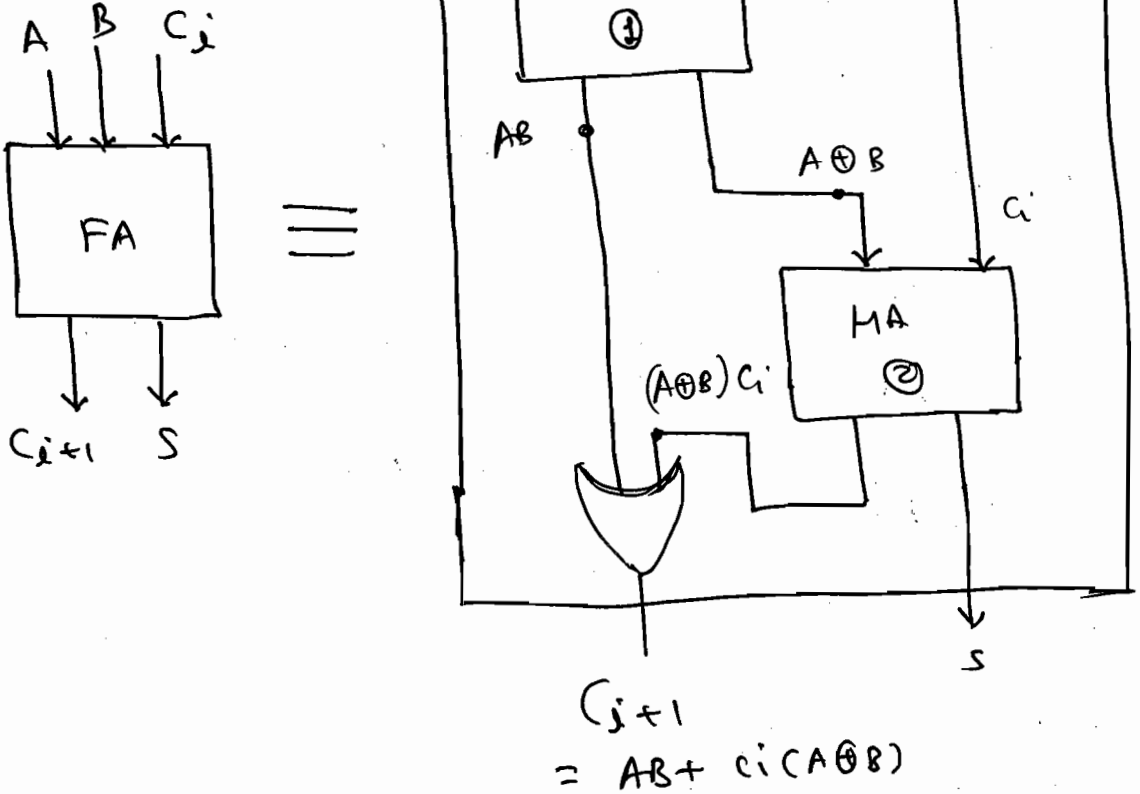
NOTE: Full subtractor

(i) Required 9 NAND gate

(ii) Required 12 NOR gate.

Ex-1 Implement Full Adders using HAs and Logic gate:

Ans:



$C_{i+1} = AB + c_i(A \oplus B)$

1 Full adder = 2 Half adder + 1 OR gate

Ex-2 How many HA required to implement the following f^n.s.

$F_1 = \bar{A}C + AB\bar{C} + \bar{B}C.$

$F_2 = A \oplus B \oplus C.$

$F_3 = \bar{A}BC + A\bar{B}C.$

Ans: (i) $F_1 = \bar{A}C + AB\bar{C} + \bar{B}C.$

$= \overline{AB} \cdot C + AB\bar{C}$

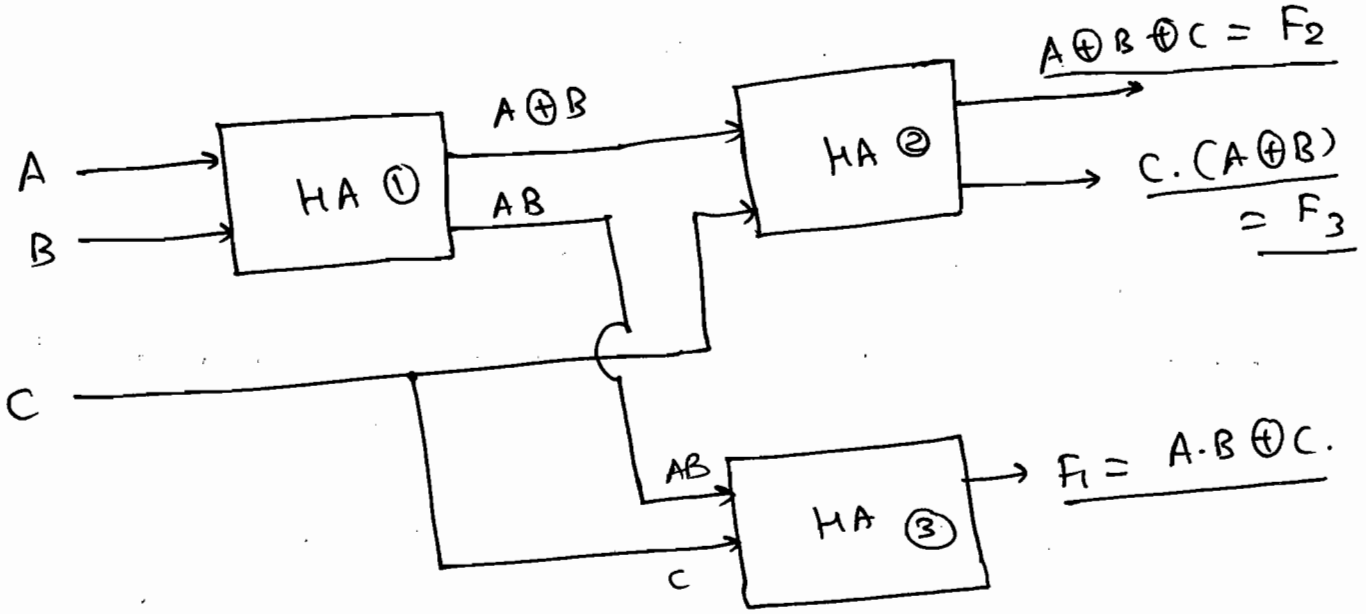
$= ((A \oplus B)) \cdot C$ $\quad X = \overline{AB}.$

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

$\boxed{F_1 = A \cdot B \oplus C}$ →

Refer (ii) $\boxed{F_2 = A \oplus B \oplus C.}$

(iii) $\boxed{F_3 = (\bar{A}B + A\bar{B})C = C \cdot (A \oplus B)}$



* Binary Adders:

(1) Parallel binary Adder:

(2) Carry Look Ahead Adder:

(3) Serial Adder.

(1) Parallel binary Adder: (4 bit):

→ IC = 74LS83

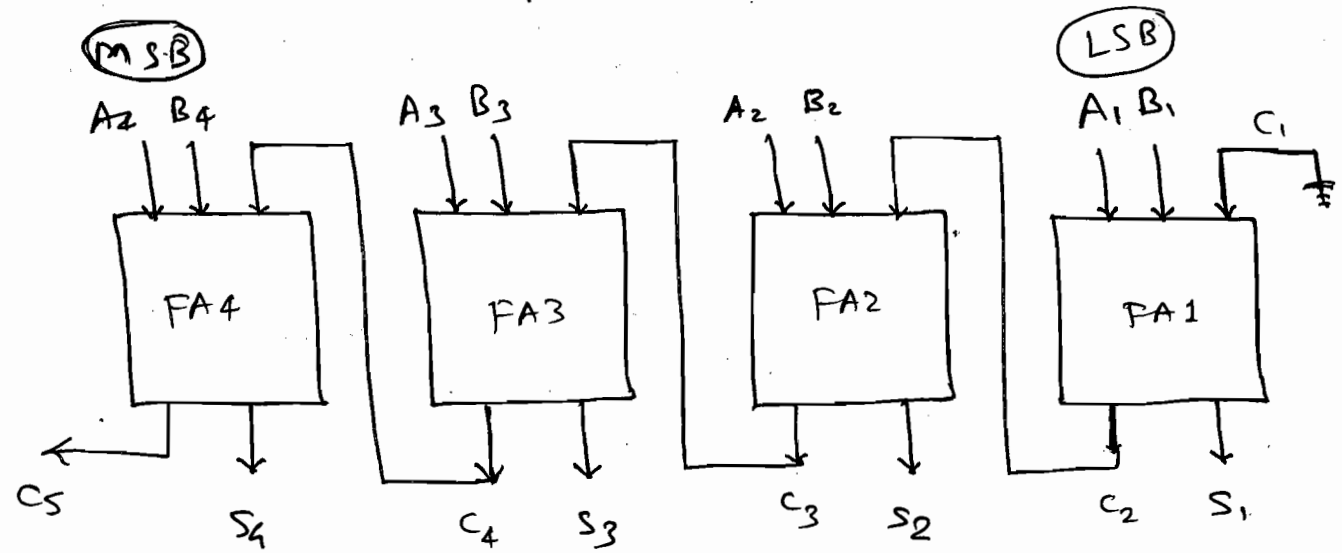
→

$$\begin{array}{rccccc} A = & A_4 & A_3 & A_2 & A_1 & \\ + & B_4 & B_3 & B_2 & B_1 & \\ \hline C_5 & S_4 & S_3 & S_2 & S_1 & \end{array}$$

PA, HA (OR) FA

→ 3 FA + 1 HA (OR) 4 FAs

→



Q: How many HA required to implement 4-bit IPL adder

Ans: 7 HA & 3 OR gate

Q.

→ In a 4 bit parallel binary adder a FA takes 32 ns to produce the sum and 14 ns to produce the carry. Determine
(i) Time required for addition.
(ii) the adition rate of the adder.

Ans: a)

Time required for Addition in N-Bit parallel adder = $T = (N-1)t_c + \max(t_s, t_c)$.

$\Rightarrow T = (4-1)14 + \max(32, 14)$.

$= 42 + 32$

$T = 74$ ns.

b) Addition Rate = $\frac{1}{T} = \frac{1}{74 \times 10^{-9}}$ Hz.

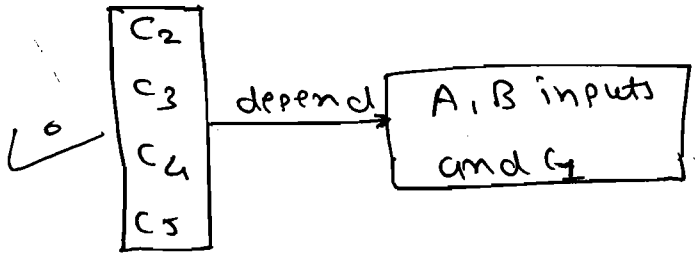
→ This can be used upto 4 bit.

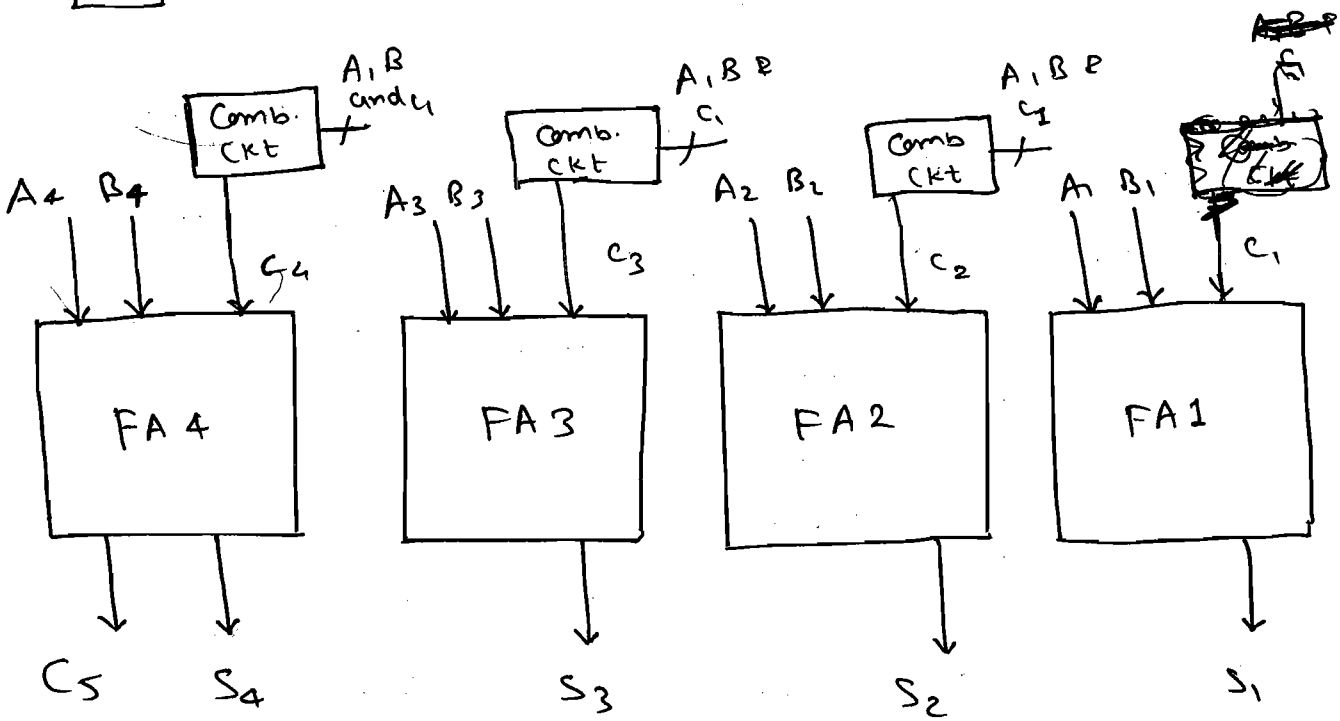
→ Disadvantages:

→ as the size of the Adder increases the Speed operation decreases as the carry has to propogate through all the FAs to overcome this we use Carry Look ahead Adder.

(2) Carry Look Ahead Adders:

→ ⊛ Principle:





→ $C_{i+1} = C_i (A \oplus B) + AB$

$P_i = A \oplus B =$ Propogation of carry

$G_i = AB =$ Carry Generation

$\therefore C_{i+1} = C_i P_i + G_i$

$\therefore \rightarrow \boxed{C_2 = C_1 P_1 + G_1}$ — ①

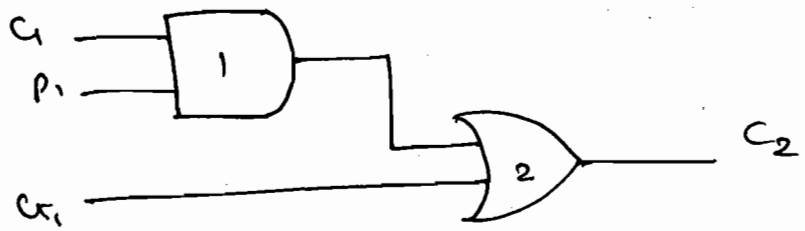
$\rightarrow C_3 = C_2 P_2 + G_2$

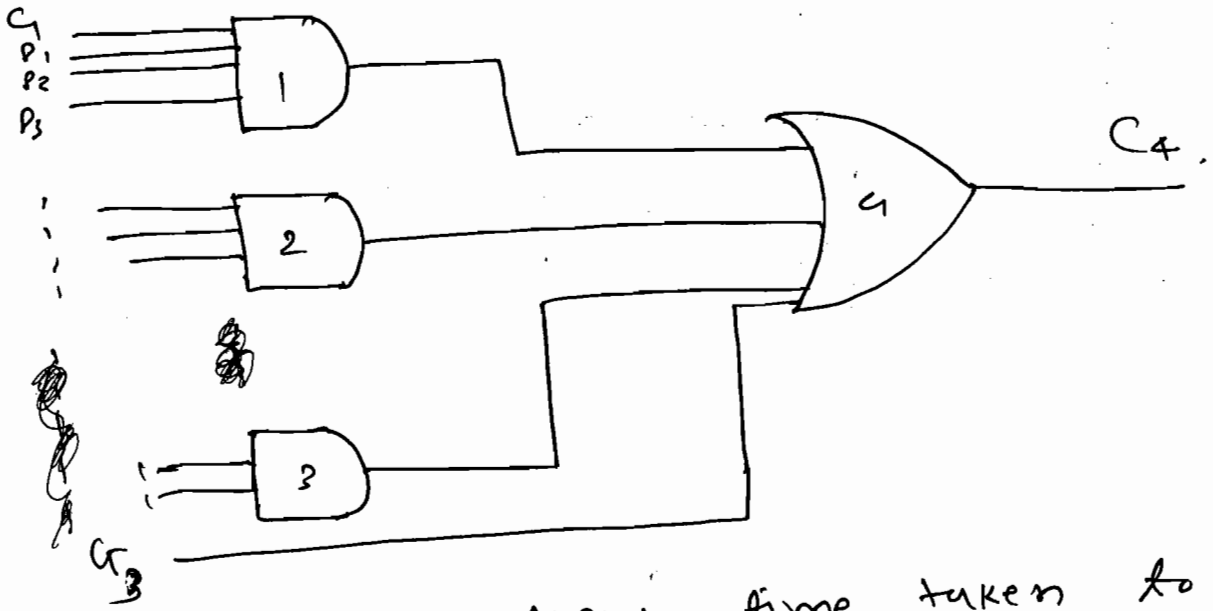
$\boxed{C_3 = C_1 P_1 P_2 + G_1 P_2 + G_2}$ — ②

→ $C_4 = C_3 P_3 + G_3$

→ $C_4 = C_1 P_1 P_2 P_3 + C_{r_1} P_2 P_3 + C_{r_2} P_3 + C_{r_3}$ —③

*





→ In Carry Look-Ahead time taken to generate the carries C_2, C_3, C_4 is same as they are impilmented by 2 level Logic.

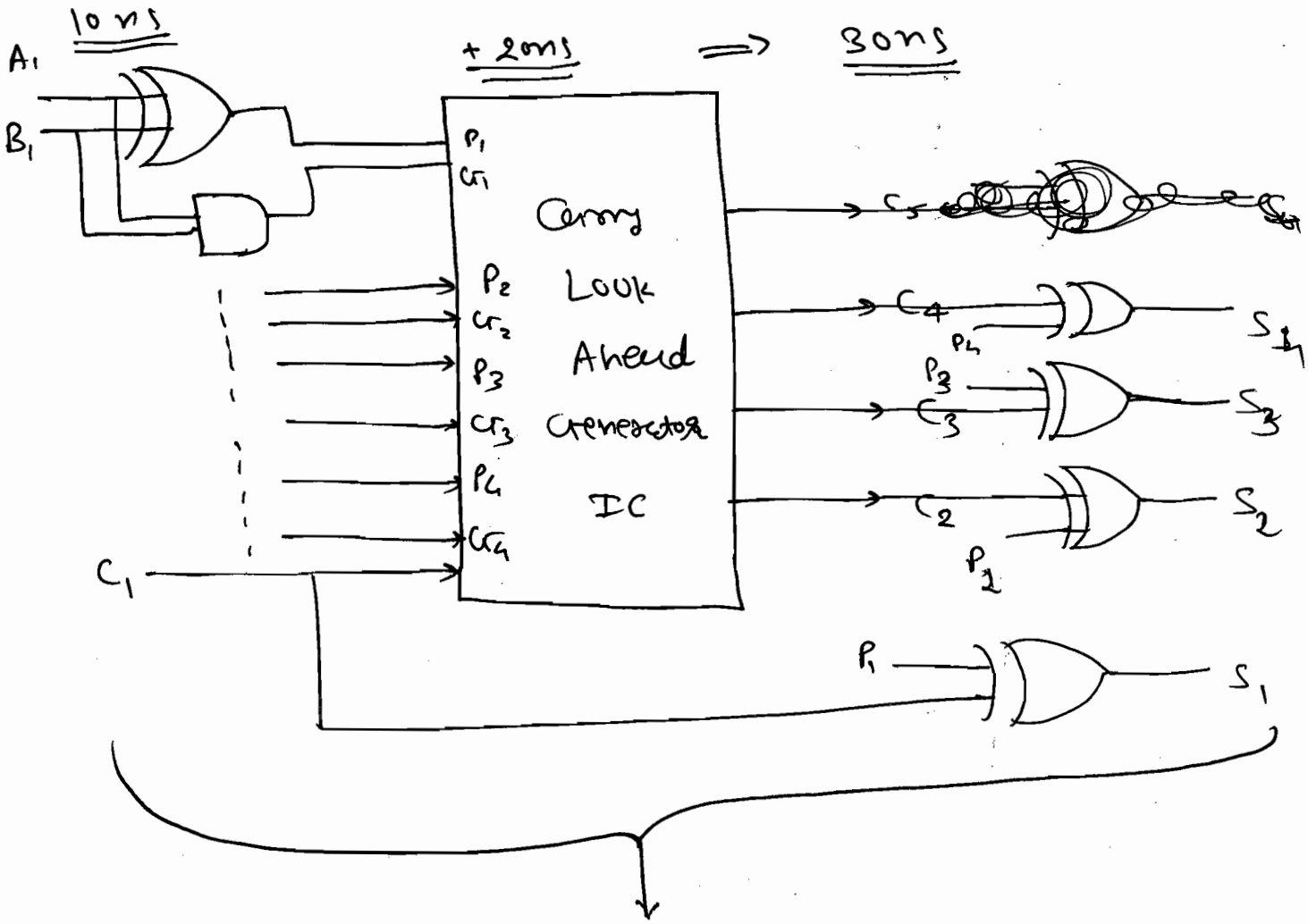
* Advantage:

→ Its Speed of operation is very high and doesn't depend on the size of the adder.

* Disadvantage:

→ It has more hardware complexity. To overcome this we use carry look ahead generator IC.

⇒ Carry Look Ahead generator IC



Carry Look Ahead Adder

$S_i = A_i \oplus B_i \oplus c_i$

$S_i = P_i \oplus c_i$

$S_1 = P_1 \oplus C_1$

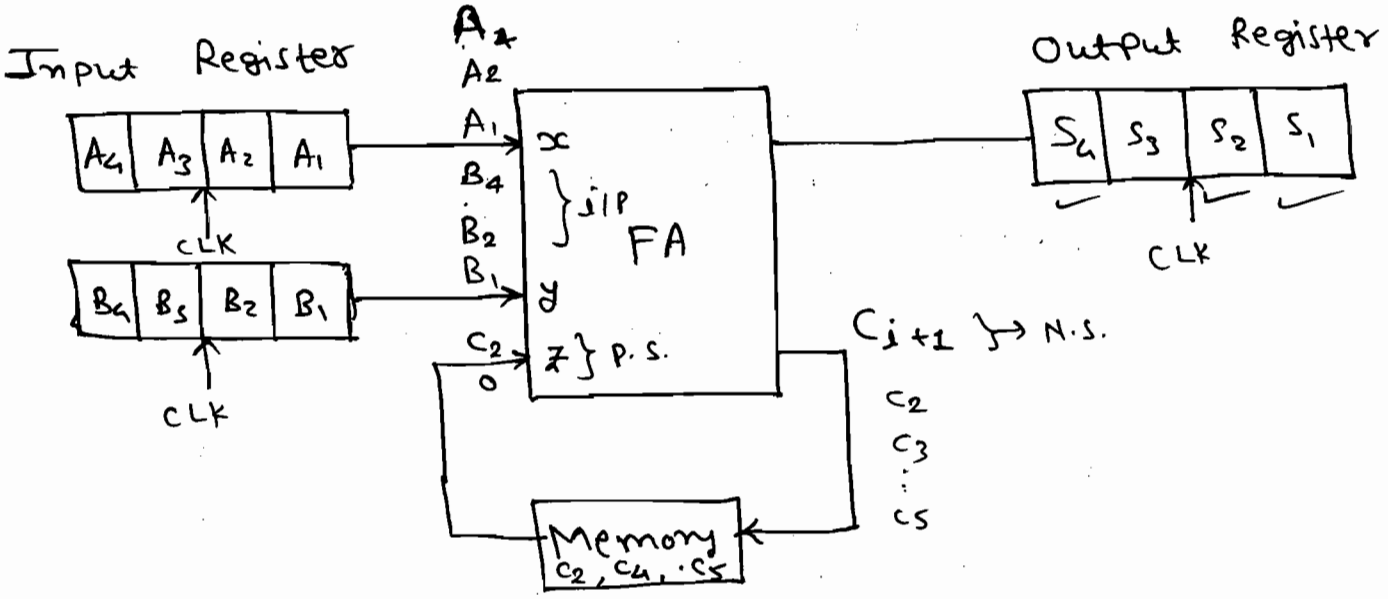
$S_2 = P_2 \oplus C_2$

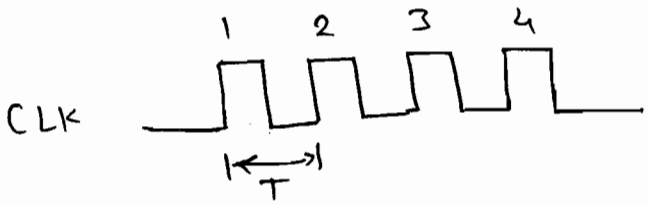
$S_3 = P_3 \oplus C_3$

$S_4 = P_4 \oplus C_4$.

(3) Serial Adder: (sequential ckt)

→

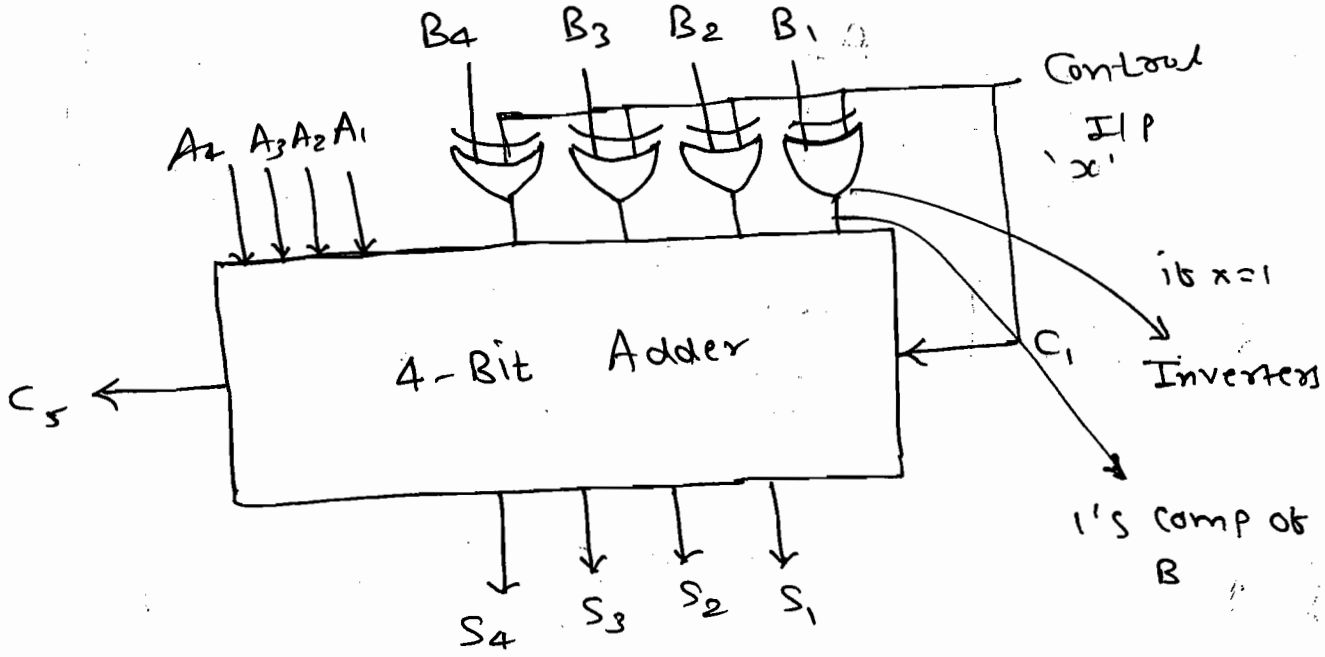




T = CLK period

Time for 4-bit serial addition = 4T

* 4-bit Parallel Adder / Subtractor.



(1) If $\boxed{x=1}$ ⇒ Ex-OR gate → Inverters

→ A + 1's Comp of B + $(C_1 = 1)$

→ A + (2's Comp of B)

⇒ $\boxed{A-B}$ → Binary Subtraction

(2) If $\boxed{x=0}$ ⇒ Ex-OR gate ⇒ Buffer

→ A + B + $(C_1 = 0)$

= $\boxed{A+B}$ → Binary Adding.

Ex-1 Determine the fn of the above circuit if the inputs are as given below:

(a) $x=1$; A = Ex-3 code ; B = 0011 ⇒ Function = ?

$\Rightarrow A-B = Ex-3 - 0011$

$= Ex-3 + (1100) + 1$

$=$ BCD code.

Ex-3 to BCD code converter.

∴ let, Ex-3 of 4 → 0111 ⇒ 0100 → BCD

↑ Ex-3

$$\begin{array}{r} 0111 \\ +\ 1100 \\ +\quad\ 1 \quad (\because C=1) \\ \hline \otimes\ 0100 \end{array}$$

$\Rightarrow 4_{10} = (0100)_{BCD}$.

(b) $x=1$, A = 1001 ; B = BCD ⇒ Function = ?

Ans: A = 1001. , let, B = BCD of 3

B = 0011.

$$\begin{array}{r} A \\ +\ 1 \end{array} \quad A-B = \begin{array}{r} 1001 \\ +\ 1100 \\ +\quad\ 1 \\ \hline 1\ 0110 \end{array}$$

$= 6_{10}$ which is 9's comp. of 3_{10}

∴ So, BCD to 9's Comp. of BCD.

A + 1's comp of B + $C_1 = 1$.

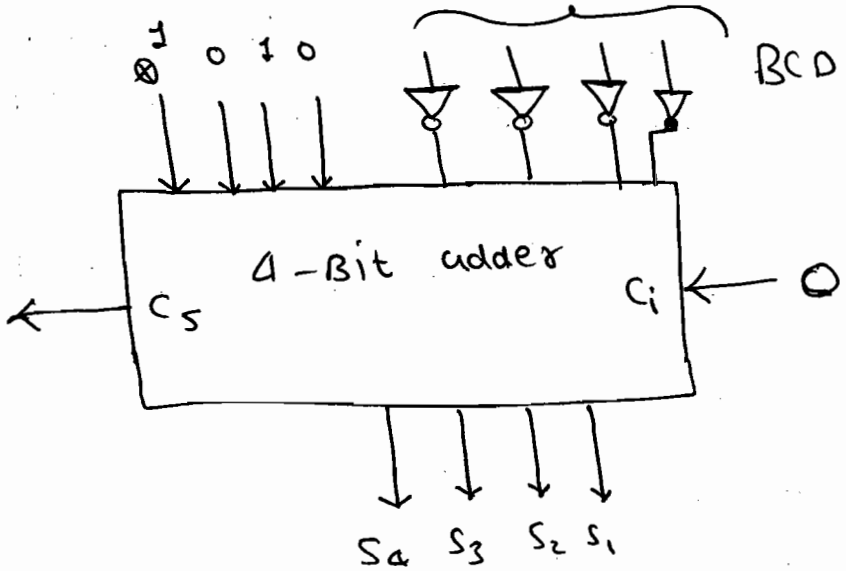
= A + (2's comp of B).

= A − BCD.

= 9_{10} − BCD

= 9's complement of BCD.

(c)



$\Rightarrow$ 1010 + 1's Comp. of BCD + $\boxed{C_i = 0}$

$= 1001 + 1 +$ 1's Comp of BCD

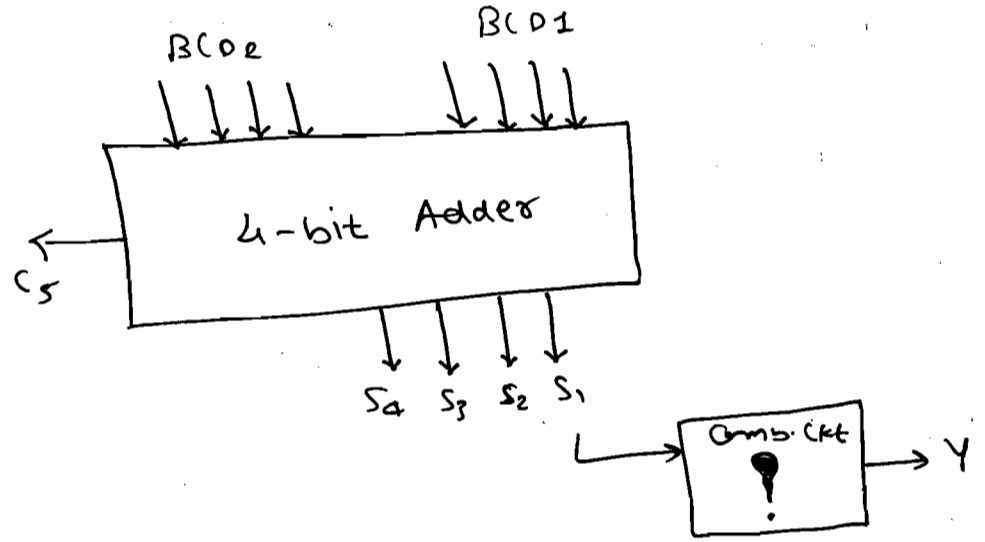
$\Rightarrow$ 1001 + 2's Comp. of BCD.

$= 9_{10} -$ BCD.

$\rightarrow$ 9's complement of ckt.

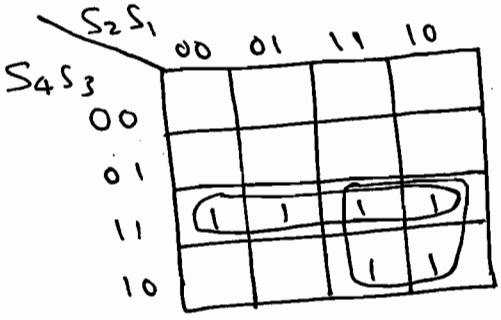
NOTE: The above 9's Complement ckt can be Converted to a 10's Complement either by choosing $C_1 = 1$ (or) by choosing A value as A=1011

* BCD Adder



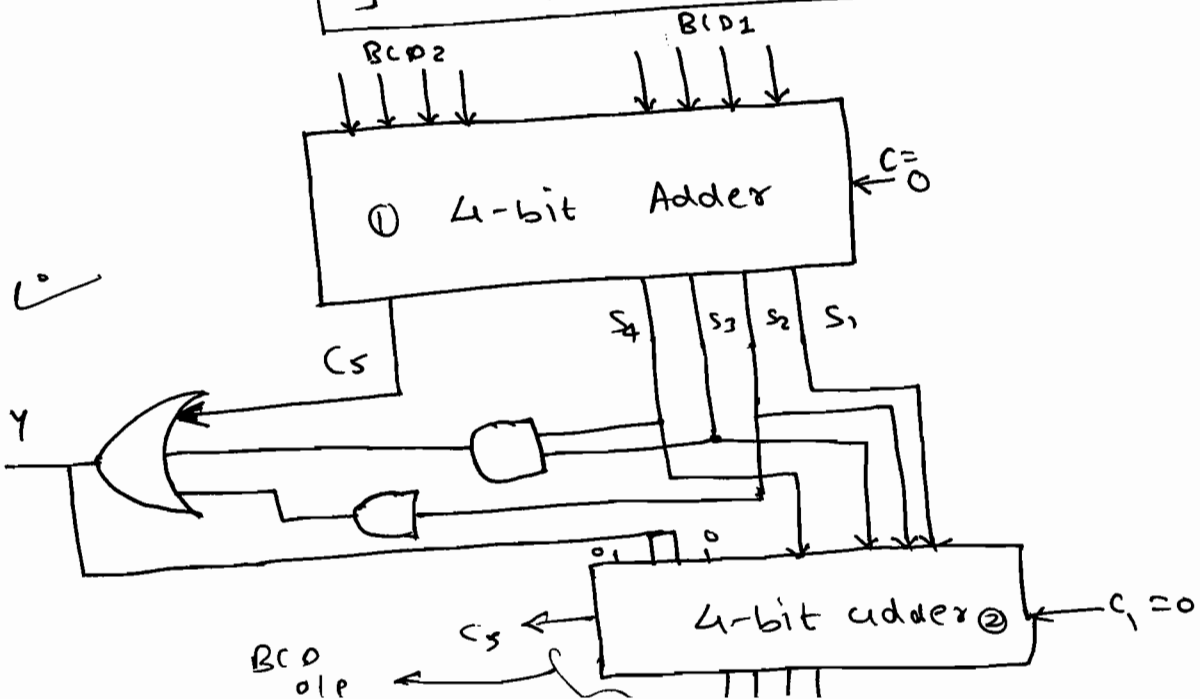
* Output is invalid BCD if

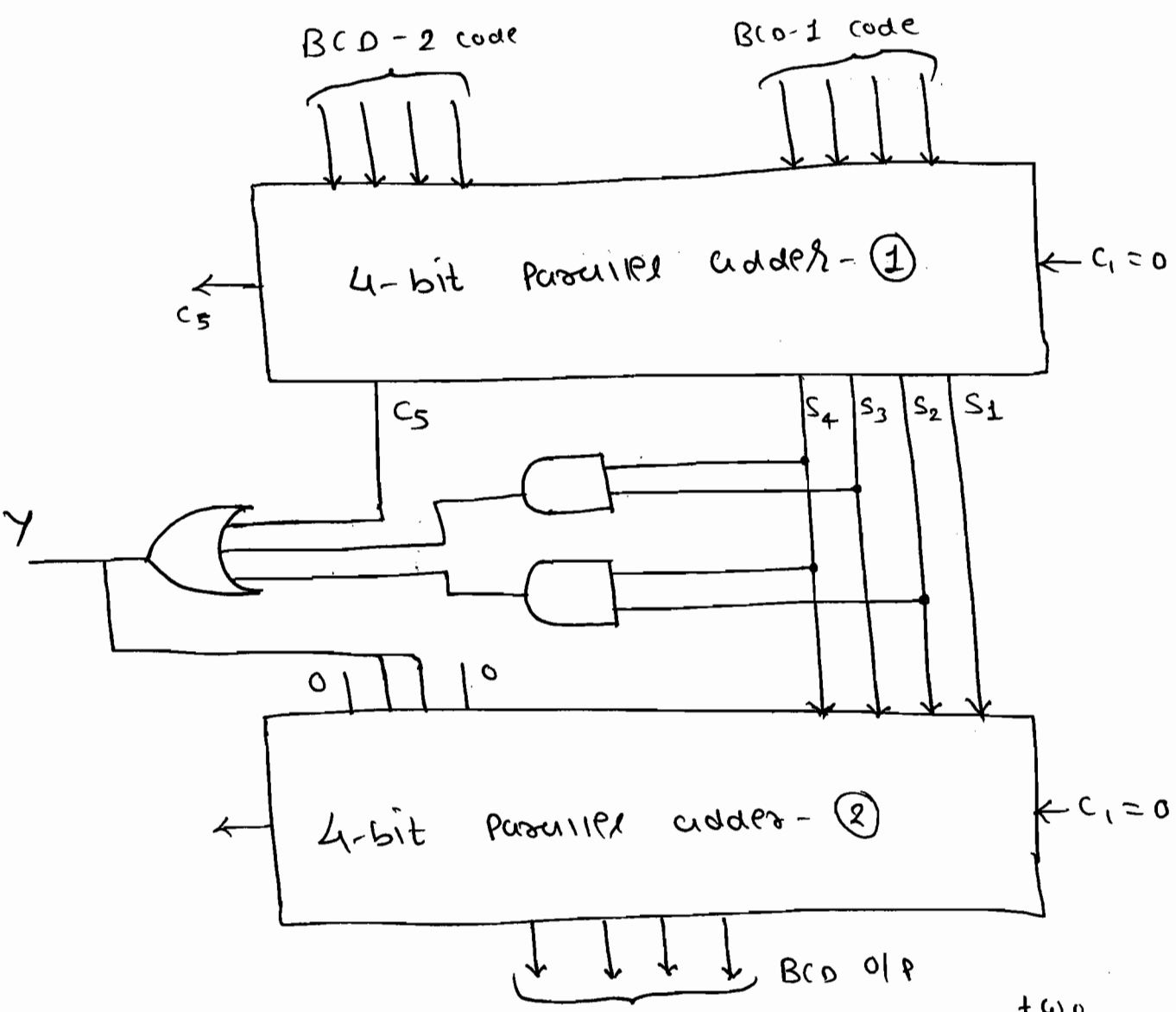
$$S_4 S_3 S_2 S_1 > 9 \quad (or) \quad C_5 = 1.$$



→ i.e. output is invalid if

$$Y = S_4 S_3 + S_4 S_2 + C_5.$$





→ Hence, one BCD adder requires two 4-bit Parallel adders

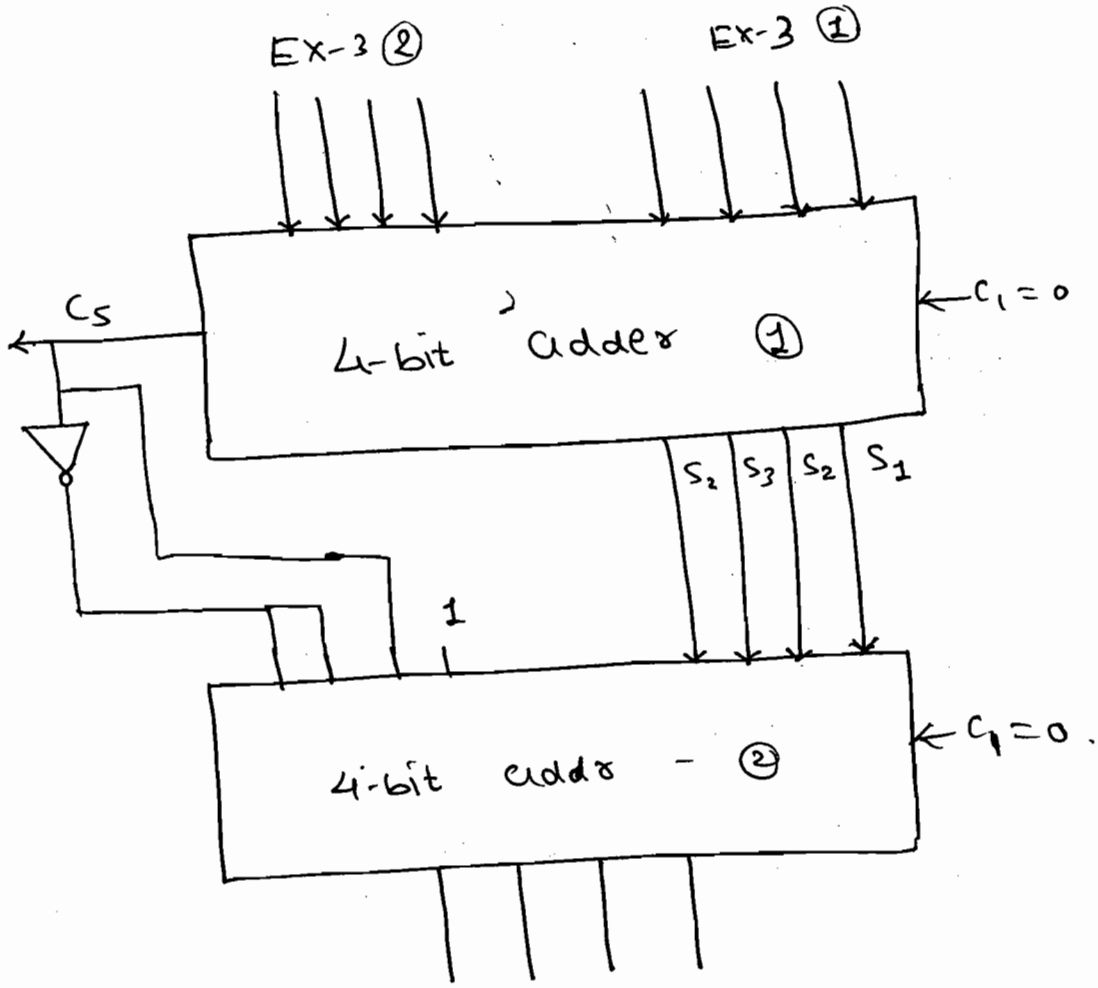
* Ex-3 Adder

(1) If $C_5 = 1 \Rightarrow$ Add $\overline{C_5}\ \overline{C_5}\ C_5$ = 0 0 1 1

(2) If $C_5 = 0 \Rightarrow$ Subtract 0 0 1 1

= Add 2's Comp of 0011

= Add $\overline{C_5}\ \overline{C_5}\ C_5$ = 1 1 0 1



* 2-Bit Magnitude Comparator:

→ The truth table of n bit magnitude Comparator is not preferred for the design of the no. of rows & in the table of 2^{2n}.

n-bit mag Comparator ⟶ No. of rows $= 2^{2n}$
$= 2^{2n}$

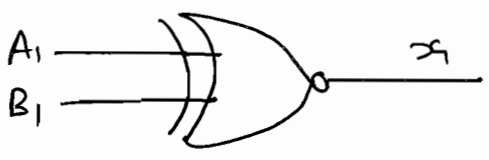
*Q
→ How many possible way in 2 bit + mag. Comparator that A > B?

Ans:

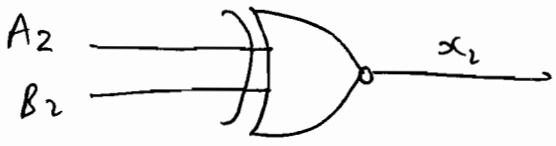
A_2 A_1	B_2 B_1	
0 0	—	
0 1	0 0	→ ①
1 0	0 0 0 1	→ ②
1 1	0 0 0 1 1 0	→ ③

So, 6 possibility that A > B.

*



$x_1 = 1$ if $A_1 = B_1 \Rightarrow x_1 = 1$

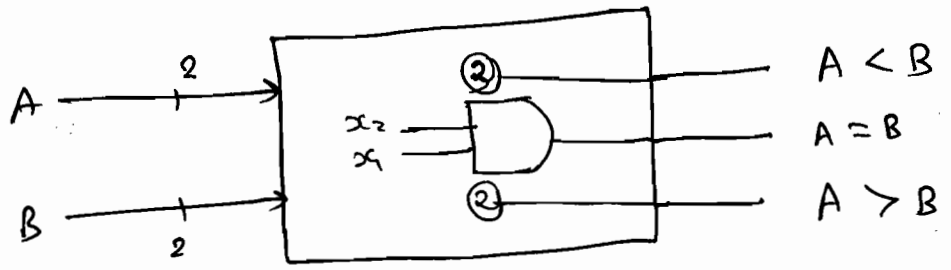


$x_2 = 1$ if $A_2 = B_2 \Rightarrow x_2 = 1$.

① $A = B$ if $A_2 = B_2$ and $A_1 = B_1$

i.e $A = B$ if $x_2 = 1$ and $x_1 = 1$.

i.e $A = B$ if $x_1 \cdot x_2 = 1$.



$A = A_2 A_1$.
$B = B_2 B_1$.

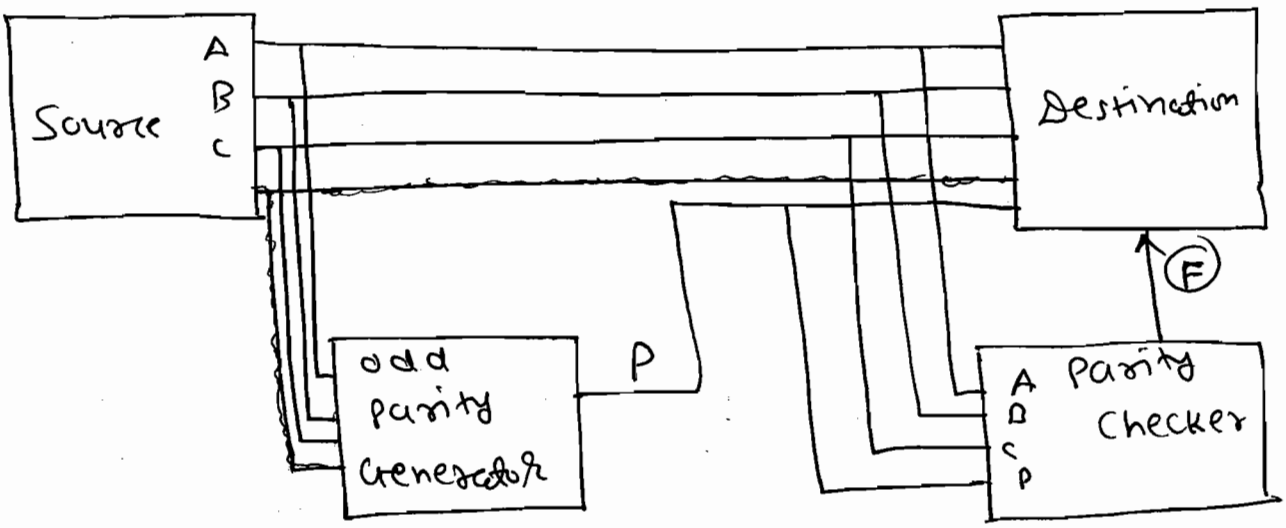
② $A > B$ if $A_2 > B_2$ (or) $A_2 = B_2$ and $A_1 > B_1$.

i.e. $A > B$ if $\boxed{A_2 \bar{B}_2 + x_2 \cdot A_1 \bar{B}_1 = 1}$ — ②

③ $A < B$ if $A_2 < B_2$ (or) $A_2 = B_2$ and $A_1 < B_1$.

i.e $A < B$ if $\boxed{\bar{A}_2 B_2 + x_2 \cdot \bar{A}_1 B_1 = 1}$ — ③.

* Odd parity generator and parity Checker.



P is choosen so that
A, B, C, P → odd parity.

F=1 is even parity
F=0 is odd parity
occur for A, B, C, P.

→ odd parity generator:

A	B	C	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

$P(A,B,C) = \Sigma m(0,3,5,6)$

$P = A \oplus B \odot C$

(or)

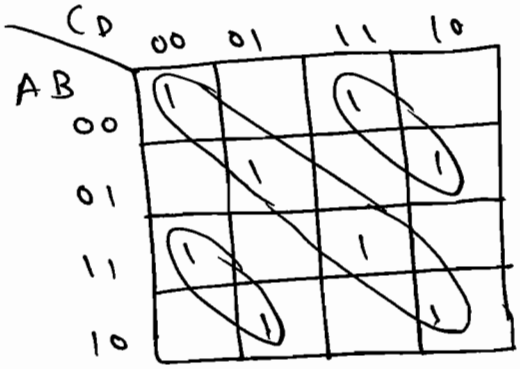
$P = A \odot^{B} \oplus C$.

Parity checker.

A	B	C	P	F
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

$F = \Sigma m(A, B, C, P)$

$= \Sigma m(0, 3, 5, 6, 9, 10, 12, 15)$.



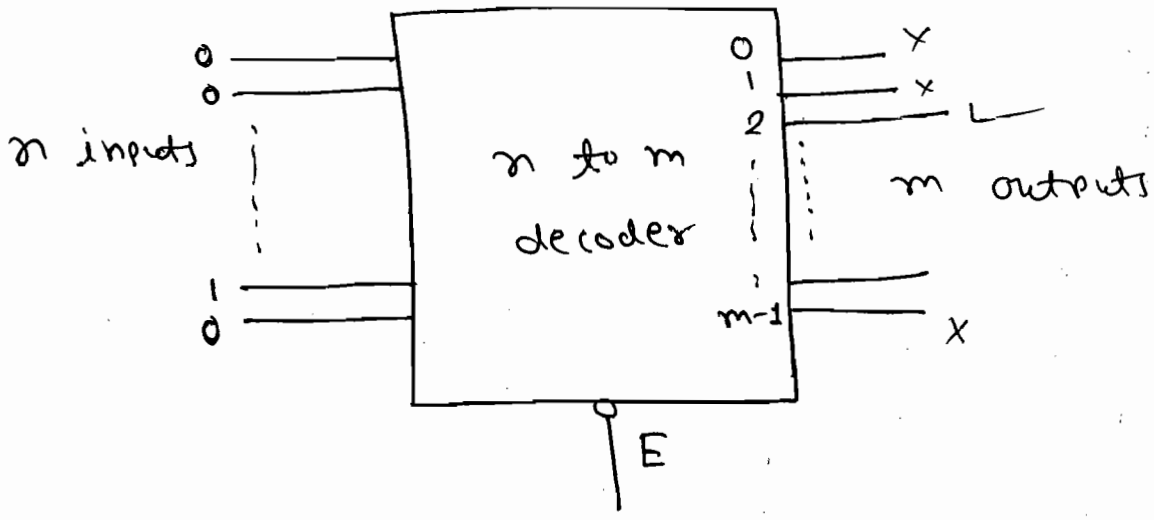
* All minterms have Even no of 0's.

$\Rightarrow$ $\boxed{F = A \odot B \odot C \odot P.}$

* Decoder:

→ It converts the binary information on i/p lines to one of many O/P lines.

→ n to m decoder = (1 out of m decoder).

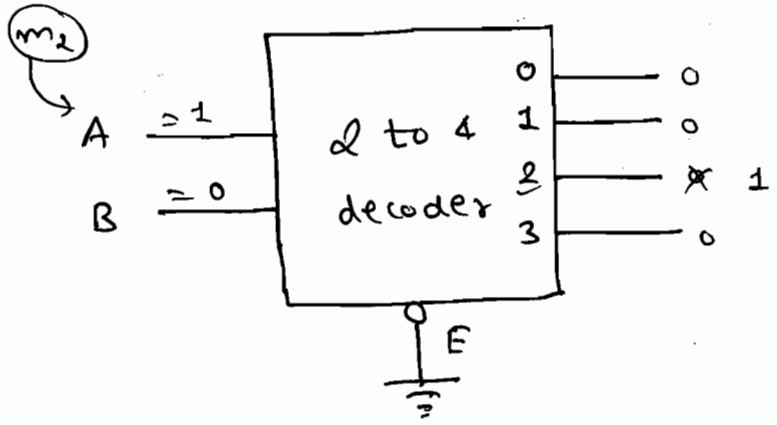


E=0 => Decoder is enabled

E=1 => Decoder is disabled

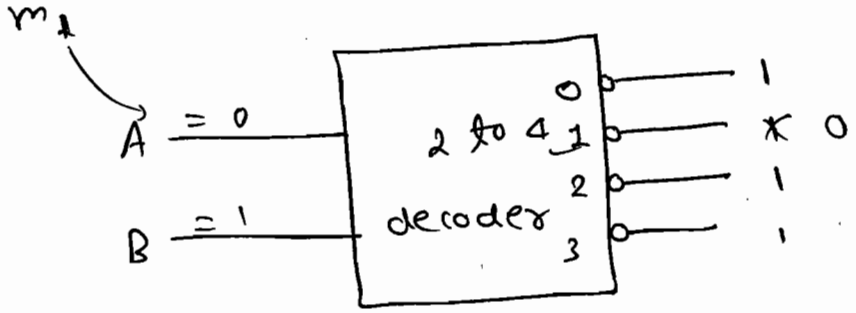
$$m \leq 2^n$$

* 2 to 4 decoder (Active High O/P).

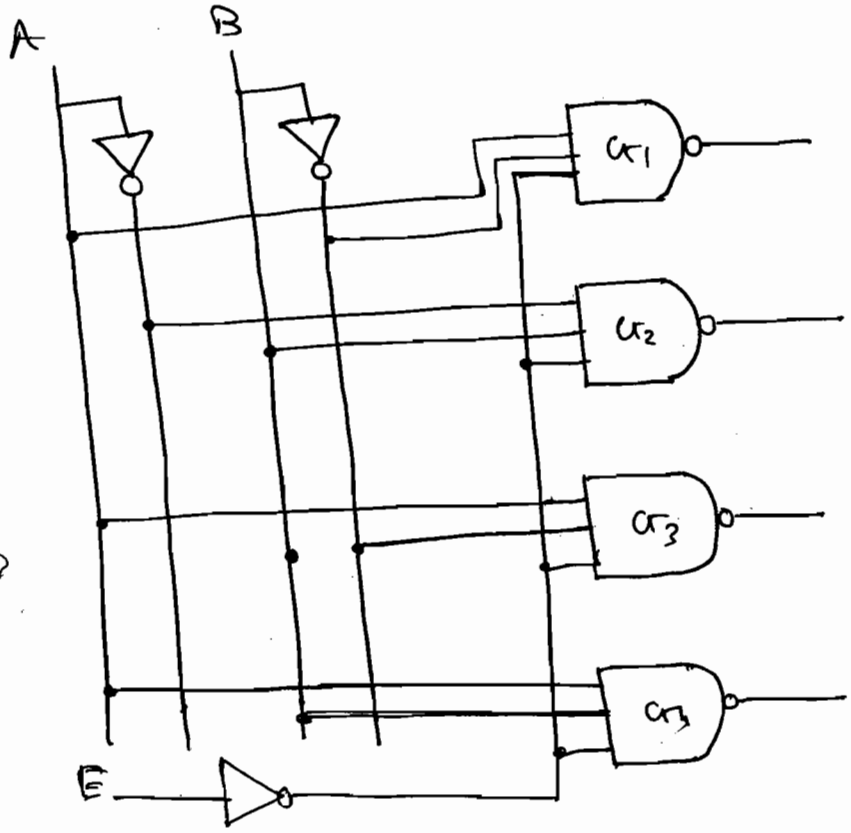


⊛ 2 to 4 decoder (Active Low O/P).

Most widely used in practice. because it generate less noise compare to Active High Decoder.



⇒ Internal circuits:



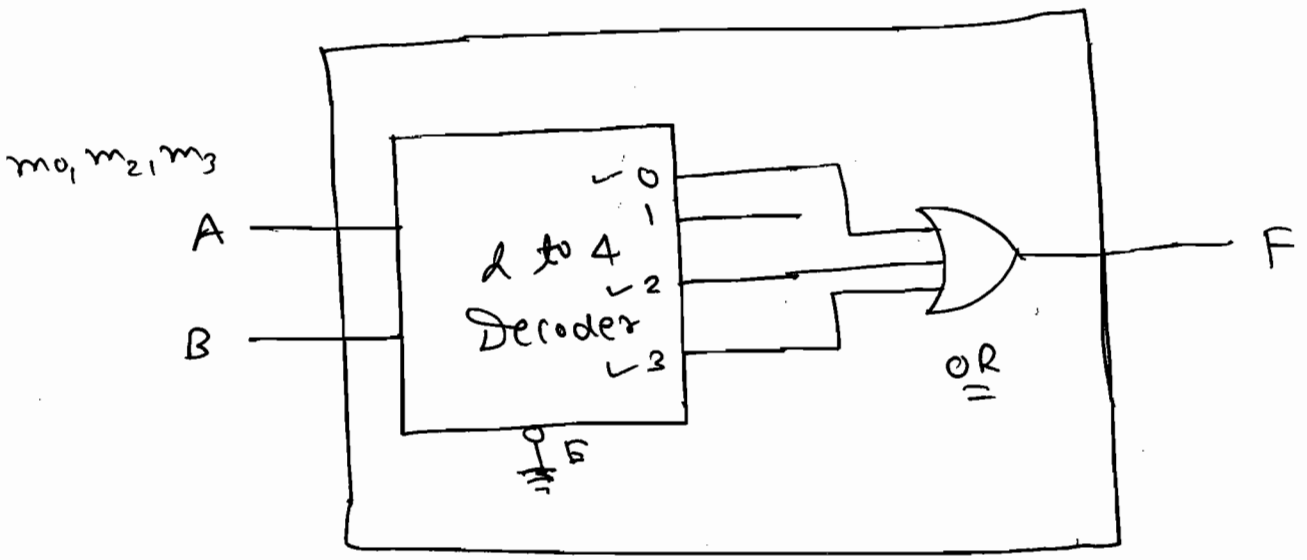
* Decoders
  => AND, NAND gates.

Ex-1 Implement $F(A,B) = \Sigma m(0,2,3)$. using a decoder (a) with active high o/p.s (b) with active Low o/p.s.

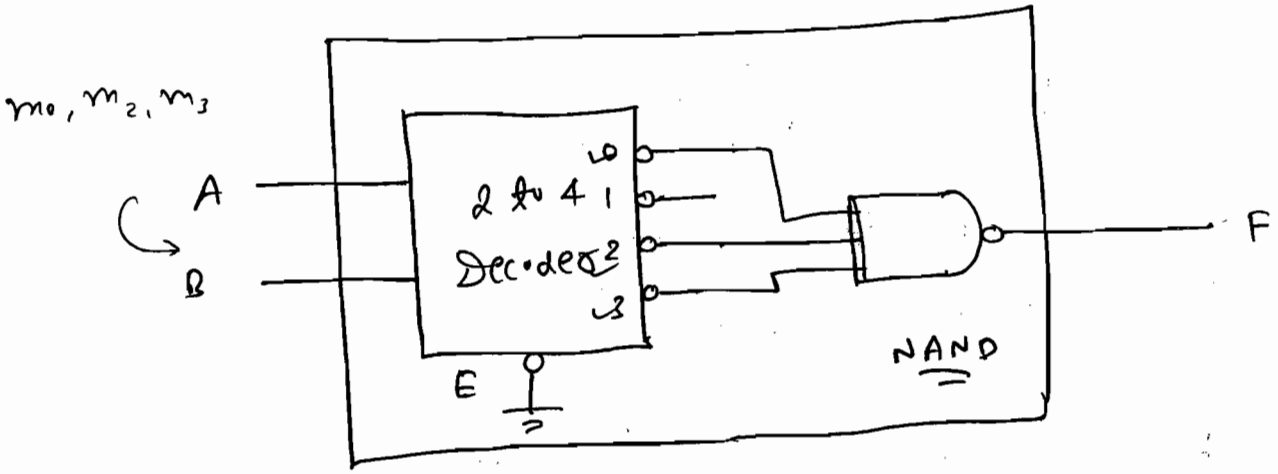
Ans: (a) With active high o/p.

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

m_0
m_2
m_3



(b) With active Low o/p.

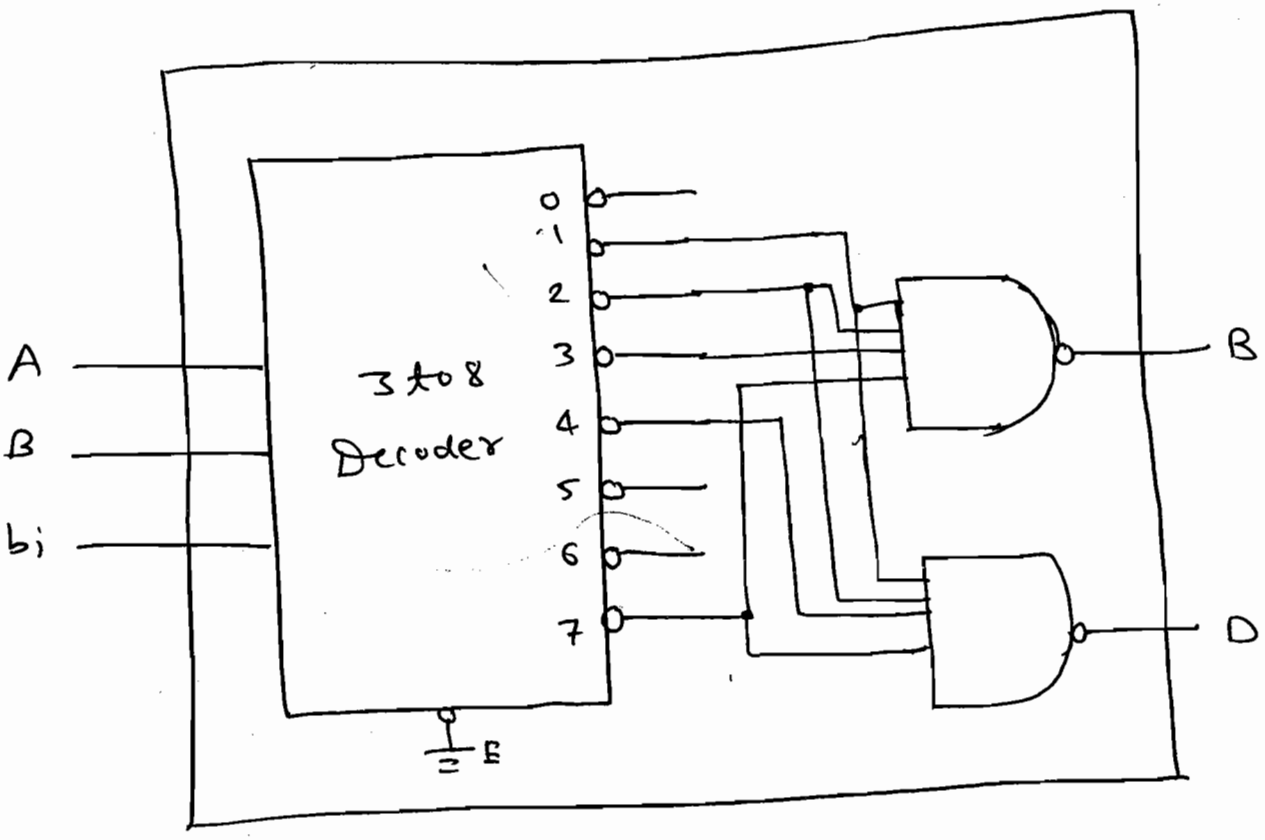


Ex-2 Implement a full Subtractor using a decoder with active Low outputs.

Ans:

$D(A, B, bi) = \Sigma m(1, 2, 4, 7)$.

$B(A, B, bi) = \Sigma m(1, 2, 3, 7)$.



Ex 3 How many 2 to 4 decoder required to construct 1 out of 32 decodoer.

Ans: Short cut:

Required No. of O/P

Given no. of O/P

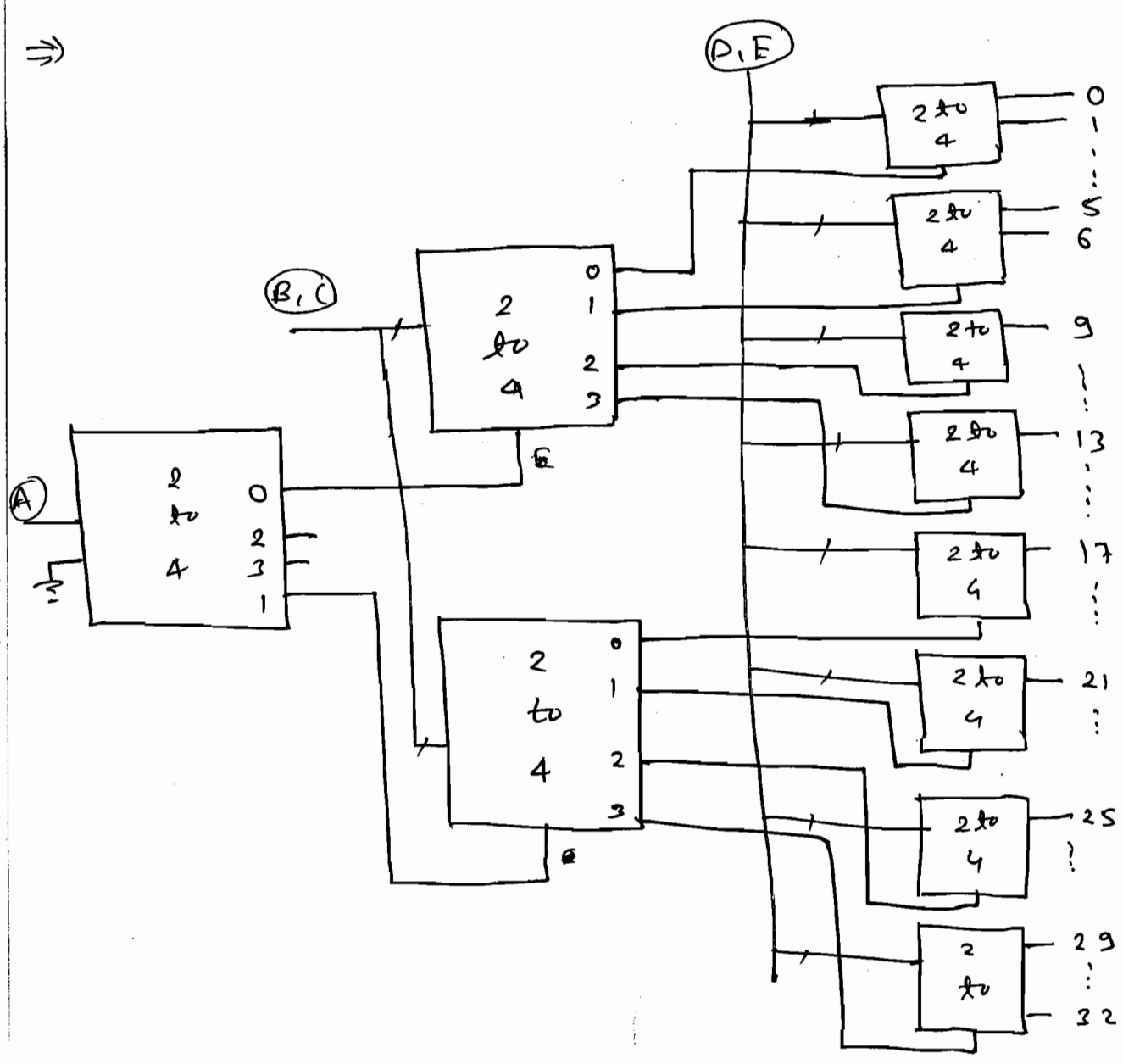
No. of Decoders

$\frac{32}{4} = 8$

$\frac{8}{4} = 2$

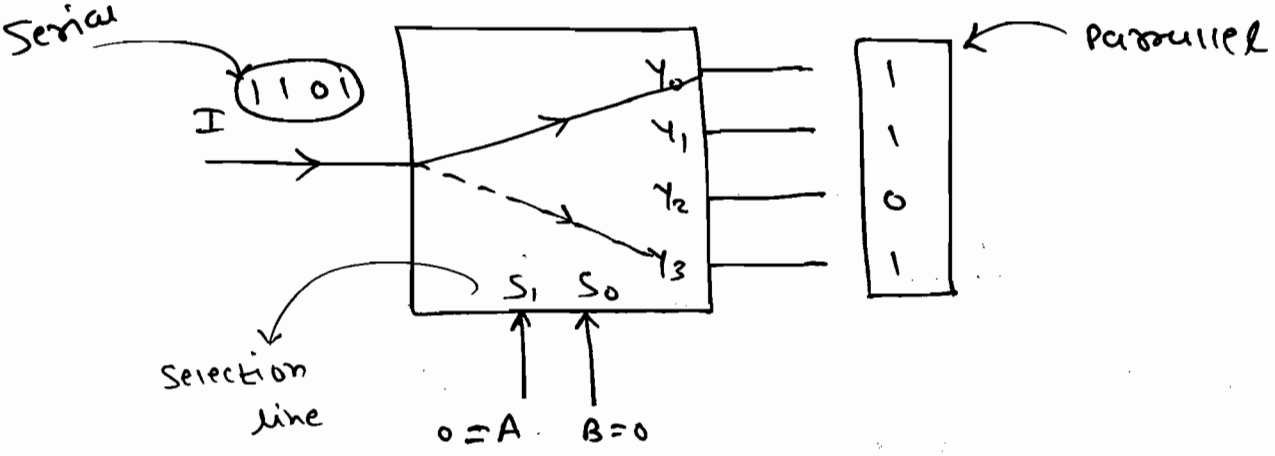
$\frac{2}{4} = 0.5 \simeq 1$

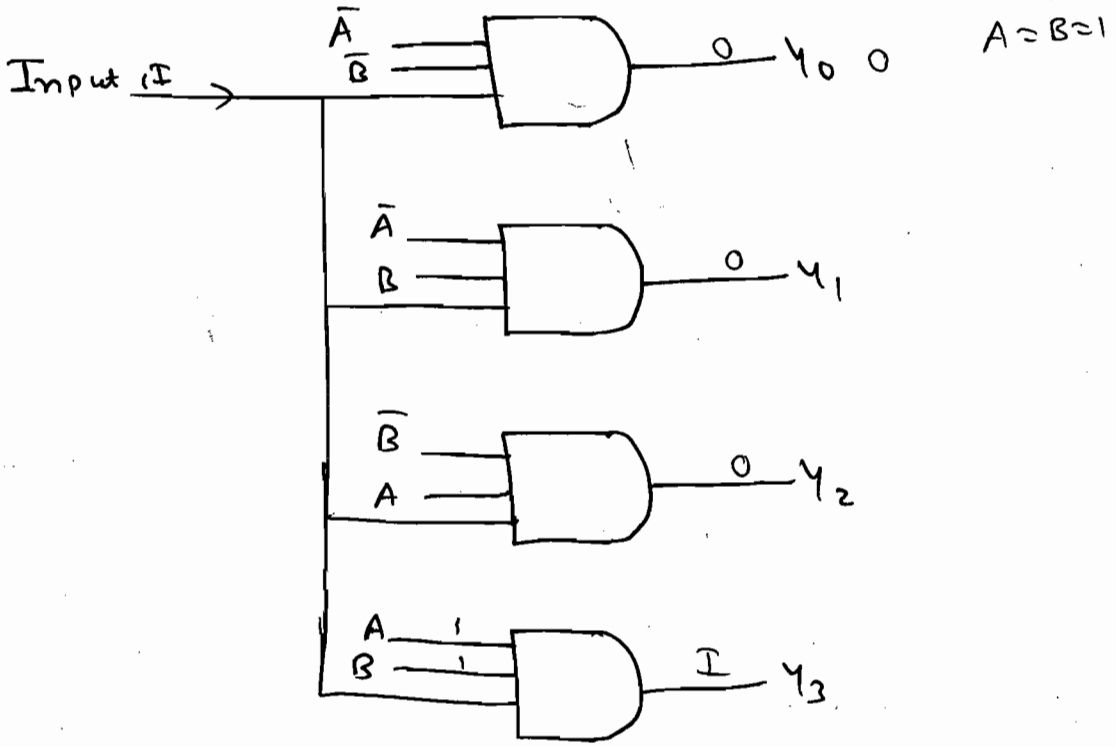
$\therefore$ 8 + 2 + 1 = 11 } 3 levels.



* Demultiplexer (one to many ckts, Serial to parallel converter)

1 : 4 Demux





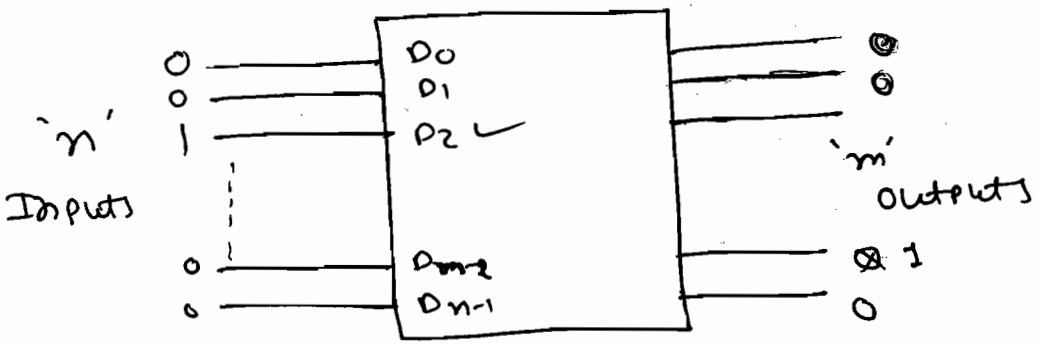
→ A Demux is similar to a decoder.

→ A 1:4 Demux is converted to a 2 to 4 Decoder by making two changes.

(1) Selection of Demux are converted to the i/ps of 2 to 4 decoder.

(2) The demux input I is treated as active high enable of 2 to 4 decoder.

✱ Encoder:



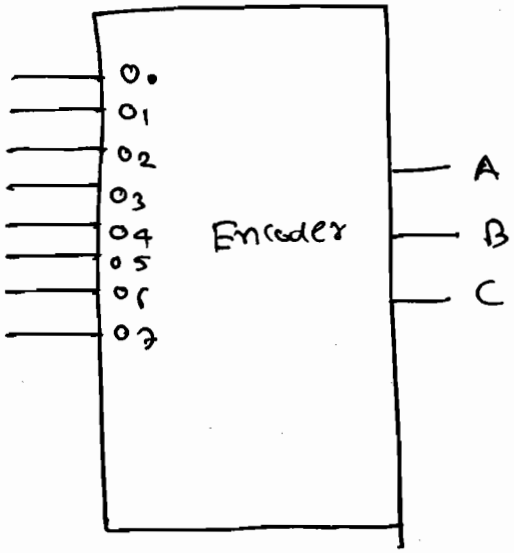
Input selected ⟶ Output code

If $D_1 = 1$ ⟶ 000.... 01

If $D_2 = 1$ ⟶ 000.... 10.

If $D_1 = 1, D_2 = 1$ ⟶ * Coding can not be performed.

Ex-1 Design octal to Binary encoder.



O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0	A	B	C
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$A = O_4 + O_5 + O_6 + O_7.$

$B = O_2 + O_3 + O_6 + O_7.$

$C = O_1 + O_3 + O_5 + O_7.$

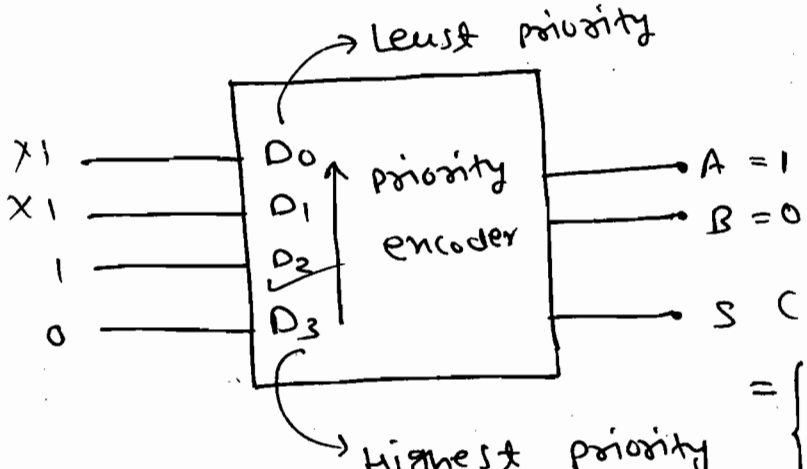
NOTE:

→ The encoder follows the **OR Logic**.

→ The limitation of encoder is it can't performed the coding if more than 1 input is active simultaneously. To overcome this, we use priority encoder.

* Priority Encoder:

⇒



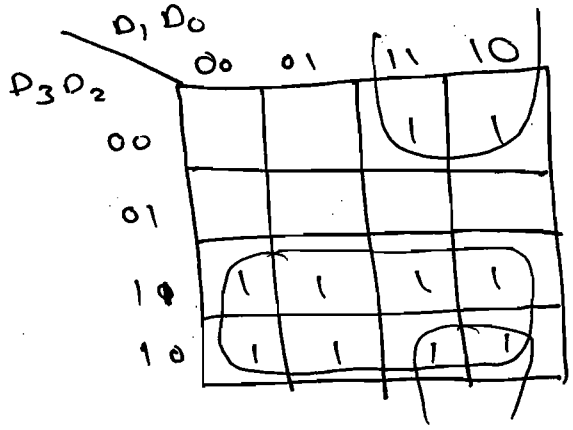
$= \begin{cases} 1 & \text{if any i/p is applied.} \\ 0 & \text{if no i/p is applied.} \end{cases}$

⇒

	D_3	D_2	D_1	D_0	A	B	S
m_0 ←	0	0	0	0	0	0	0
m_1 ←	0	0	0	1	0	0	1
m_2, m_3 ←	0	0	1	X	0	1	1
m_4 to m_7 ←	0	1	X	X	1	0	1
m_8 to m_{15} ←	1	X	X	X	1	1	1

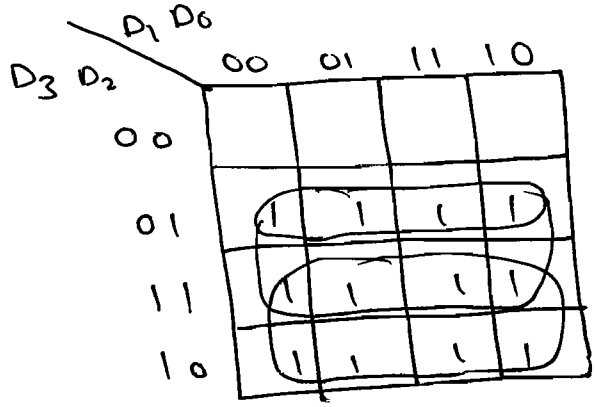
(b) Priority Encoder Table.

→ $B(D_3, D_2, D_1, D_0) = \Sigma m(2, 3, 8, 9, 10, \cdots 15)$.



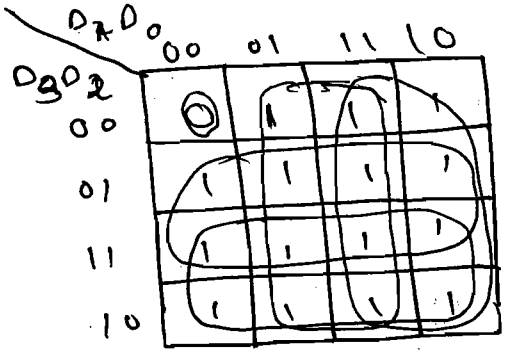
$B = D_3 + \overline{D_2} D_1$

→ $A(D_3, D_2, D_1, D_0) = \Sigma m(4, 5, 6, 7, \cdots 15)$.



$A = D_2 + D_3$

→ $S = \Sigma m(1, 2, 3, \cdots 15)$.



$\therefore S = \overline{D_3} + \overline{D_2} + \overline{D_1} + D_0$

$S = D_3 + D_2 + D_1 + D_0$

NOTE:

74LS138 ⇒ { 3 to 8 decoder (Active low outputs).

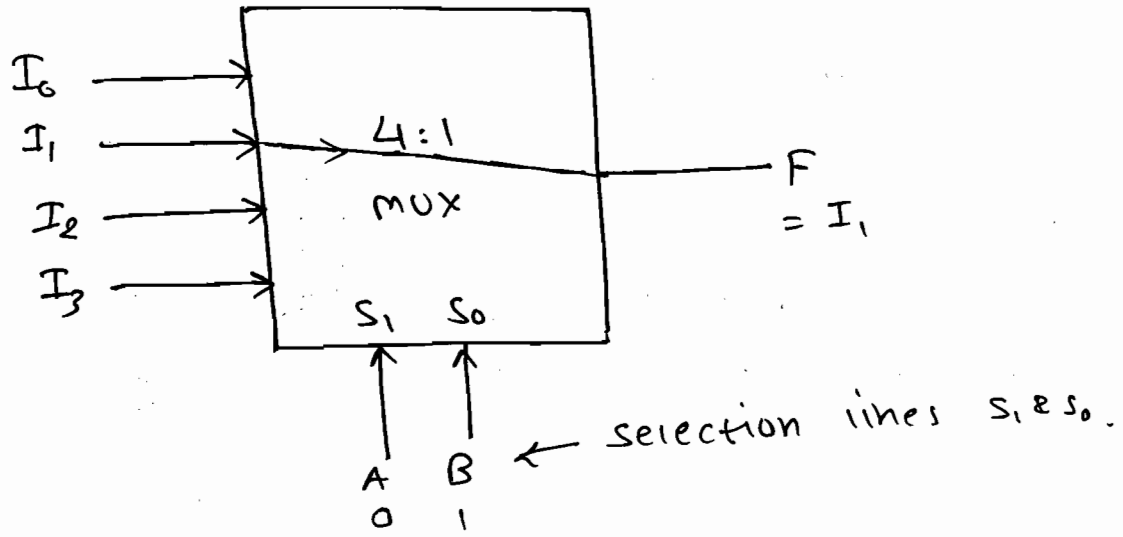
74LS148 ⇒ { 8-input, 3-output priority Encoder.

* Multiplexer:

(Many to one circuit, parallel to serial converter).

*

4:1 MUX.



∴

A	B	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

⇒ 4:1 MUX :→ $F = \bar{A}\bar{B}I_0 + \bar{A}BI_1 + A\bar{B}I_2 + ABI_3$.

$\therefore$ $\boxed{F = m_0 I_0 + m_1 I_1 + m_2 I_2 + m_3 I_3.}$

* E.g.: Given $I_0 = I_1 = 1$; $I_2 = I_3 = 0$.

$\therefore$ $F = m_0 \cdot 1 + m_1 \cdot 1 + 0 + 0.$

$\therefore$ $F = m_0 + m_1$

$\therefore$ $\boxed{F = \Sigma m (0,1).}$

Ex-1 Implement the Sum o/p of half Adder using 4:1 MUX.

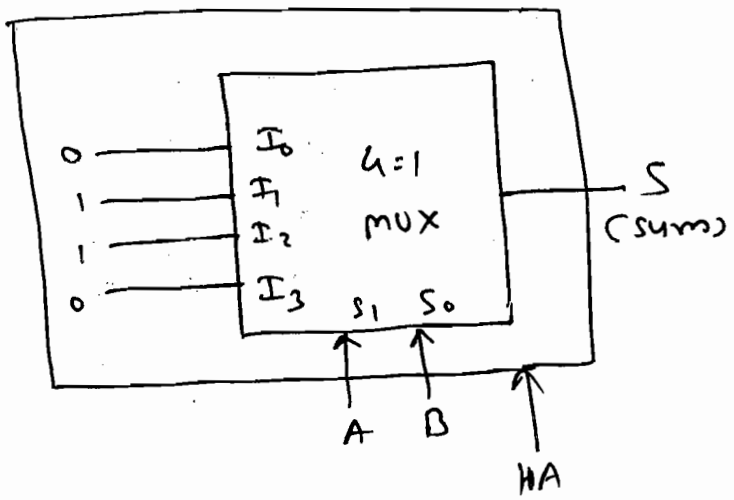
Ans:

$S(A,B) = m\Sigma(1,2).$

$S(A,B) = m_1 + m_2$

Choose $I_1 = I_2 = 1$ &

$I_0 = I_3 = 0$



Ex-2 which of the following is implemented by the following mux.

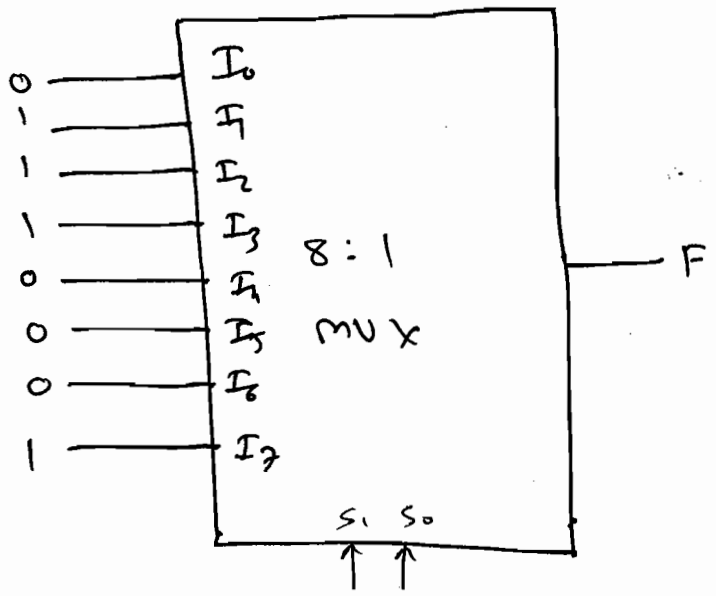
(a) Sum output of Full adder.

(b) Carry output of Full adder.

(c) Difference output of Full Subtractor.

✓(d) Borrow output of Full Subtractor.

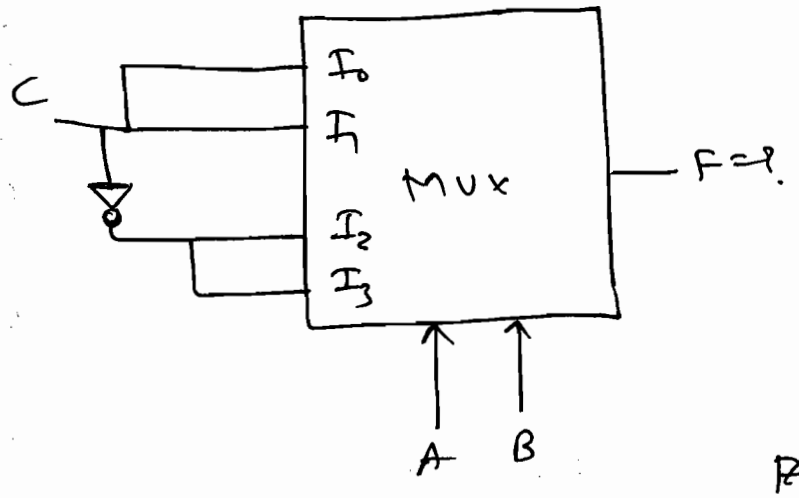
(6) Sum of the given address.



$F = \Sigma m(1,2,3,7)$

Ex-2 What are the Logic gates represented by the following mux circuits.

(a)



$F = m_0 C + m_1 C + m_2 \bar{C} + m_3 \bar{C}$

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

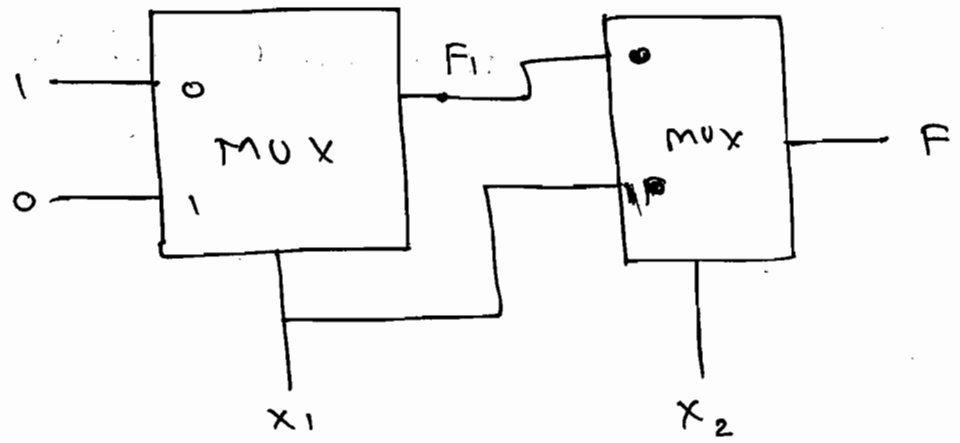
$F = (\bar{A}\bar{B} + \bar{A}B)C + (A\bar{B} + AB)\bar{C}$

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

$F = A \oplus C$

∴ Ex-OR gate

(b)



$\therefore \quad F_1 = m_0 = \bar{x}_1$

$F_1 = \bar{x}_1$

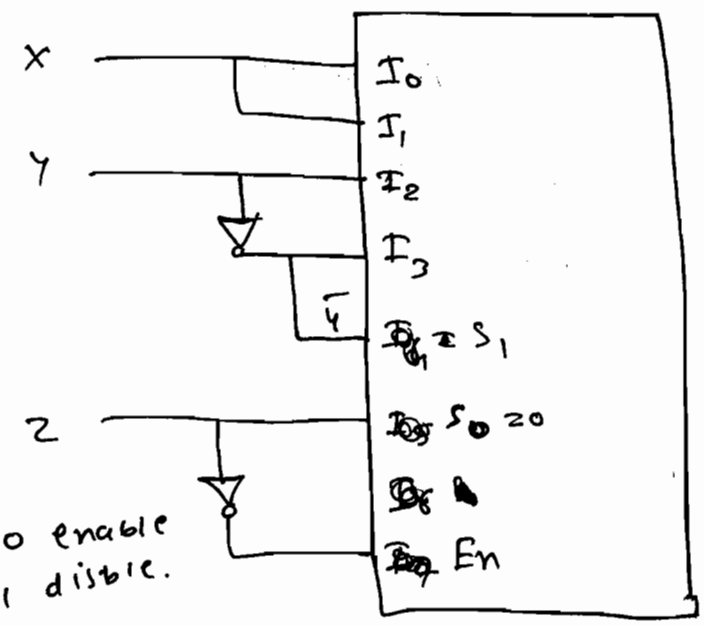
$\therefore \quad F = F_1 \bar{x}_2 + x_2 x_1$

$= \bar{x}_2 \cdot \bar{x}_2 + x_2 x_1$

$\therefore \boxed{F = x_1 \odot x_2}$

Ex- 4 Determine the output of the following MUX.

Ans:



$F = (m_0 + m_1) X + m_2 Y + m_3 \bar{Y}$

$= (\bar{S}_1 \bar{S}_0 + \bar{S}_1 S_0) X + S_1 \bar{S}_0 Y + S_1 S_0 \bar{Y}$

$= \bar{S}_1 X + S_1 \bar{S}_0 Y + S_1 S_0 \bar{Y}$

$= \bar{S}_1 X + S_1 Y + 0$

$= XY + 0$

$= XY$

$\Rightarrow$ $\boxed{F = x \cdot y \cdot \bar{z}}$

$\Rightarrow$ $z = 0$ → enable.

$z = 1$ → disable.

Ex-5 How many 4:1 MUX required to construct 128 : 1 MUX.

Ans: $\frac{\text{Required No. of Inputs}}{\text{Given No. of Ip}}$

No. of MUX

$= \frac{128}{4} = (32) \longrightarrow 32$

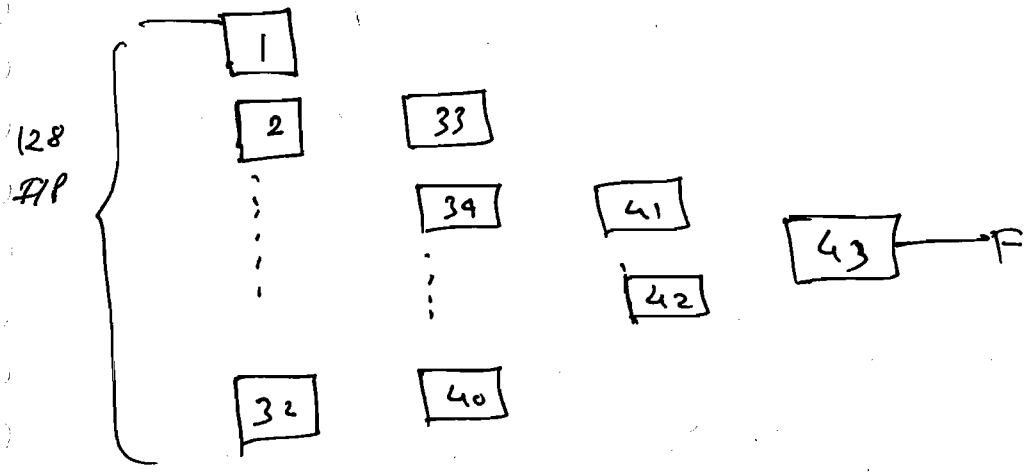
$+$

$\therefore \frac{32}{4} = (8) \longrightarrow 8$

$\frac{8}{4} = (2) \longrightarrow +2$

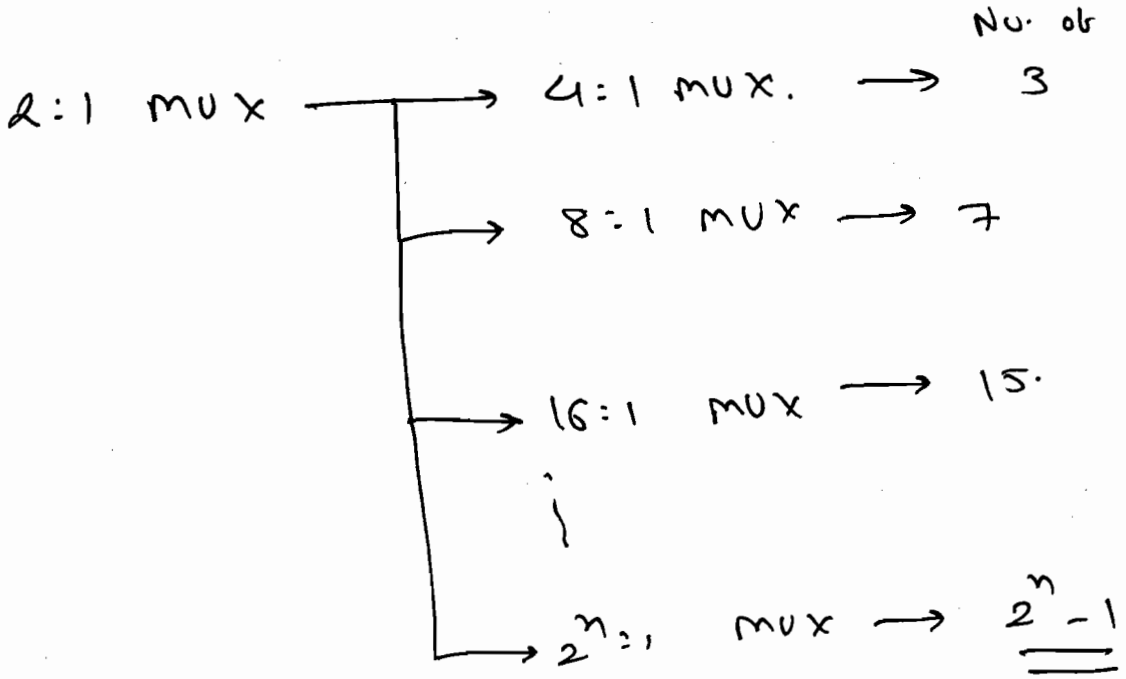
$\frac{2}{4} = 0.5 \longrightarrow +1$

$43.$



(b) 2:1 MUX ⟶ 2^n:1 MUX.

→



So, $\boxed{2^n - 1}$ 2:1 MUX required for 2^n:1 MUX.

Ex-5 How many 2^1:1 MUX are req.
to realize (a) AND gate
(b) OR gate
(c) Ex-OR gate.

Ans:

$F = \bar{A} I_0 + A I_1$

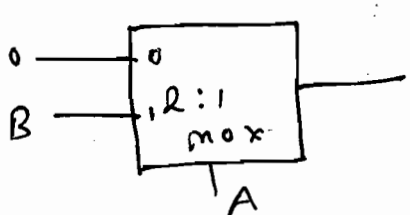
$F = AB$

$F = AB + 0 \cdot \bar{A}$

$F = 0 \cdot \bar{A} + B \cdot A$

$F = \bar{A} I_0 + A I_1$

So, $I_0 = 0$, $I_1 = B$.



(b) OR gate:

$\therefore F = \bar{A} I_0 + A I_1$

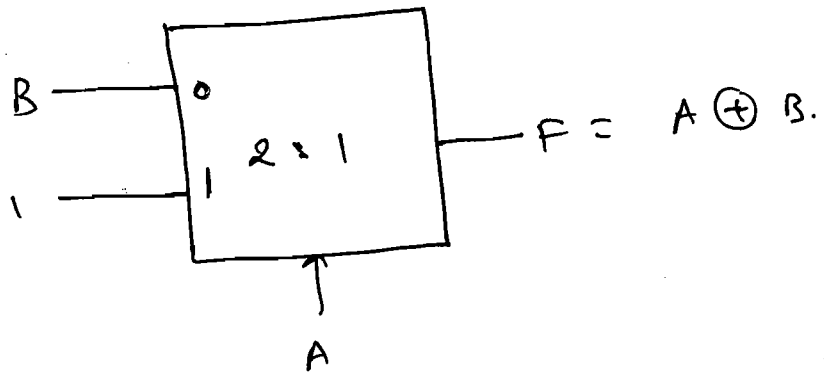
$F = A + B.$

$= A + \bar{A} B$

$F = \bar{A} B + A \cdot 1.$

$\therefore F = \bar{A} I_0 + A \cdot I_1$

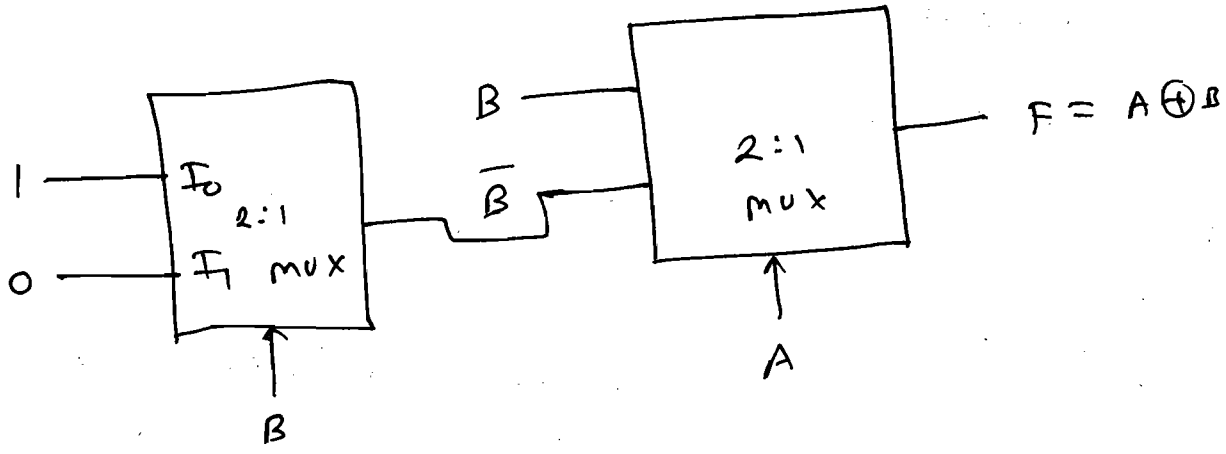
$\therefore$ So, Choose $I_0 = B$, $I_1 = 1.$



(c) Ex-OR gate:

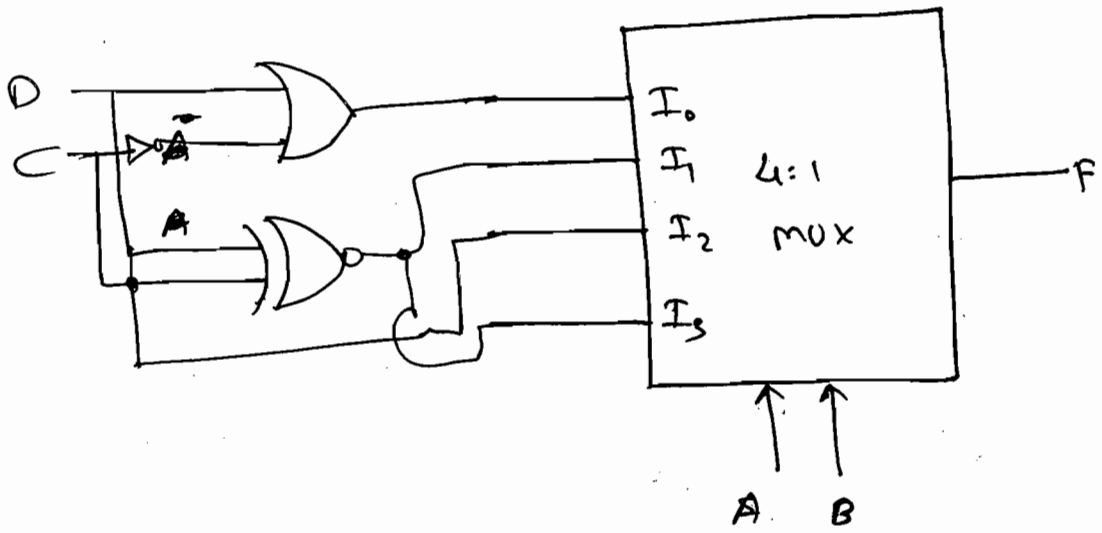
$\rightarrow F = \bar{A} B + A \bar{B} = A \oplus B.$

($\bar{A} B$: $B \uparrow$, I_0; $A \bar{B}$: $\bar{B} \uparrow$, I_1)



Ex-6 Implement $F(A,B,C,D) = \Sigma m(0,1,3,5,6,10,11,13,14)$ using 4:1 MUX.

Ans:



Method - 1

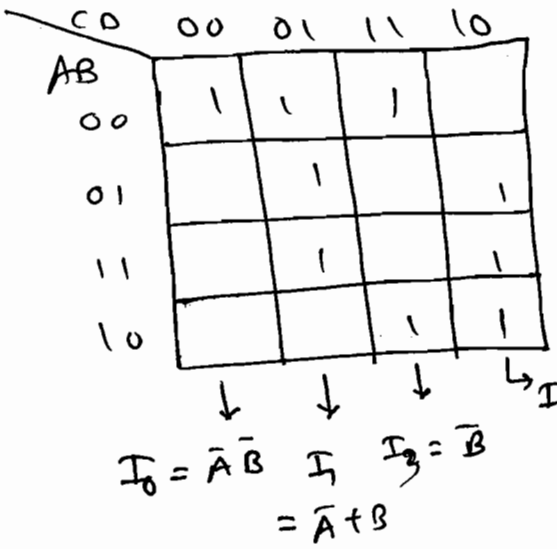
AB =	00	01	10	11
	I_0	I_1	I_2	I_3
00 $\bar{C}\bar{D}$	(0)	4	8	12
01 $\bar{C}D$	(1)	(5)	9	(13)
10 $C\bar{D}$	2	(6)	(10)	(14)
11 CD	(3)	7	(11)	15
	$\bar{C}\bar{D} + \bar{C}D + CD = \bar{C} + D$	$C \oplus D$	C	$C \oplus D$.

Method :- 2

If A, B are selection line

CD / AB	00	01	11	10	
00	1	1	1		$\rightarrow I_0 = \bar{C}\bar{D} + \bar{C}D + CD = \bar{C} + D$
01		1		1	$\rightarrow I_1 = \bar{C}D + C\bar{D} = C \oplus D$.
10		1	a	1	$\rightarrow I_2 = \bar{C}D + C\bar{D} = C \oplus D$
11			1	1	$\rightarrow I_3 = CD + C\bar{D} = C$

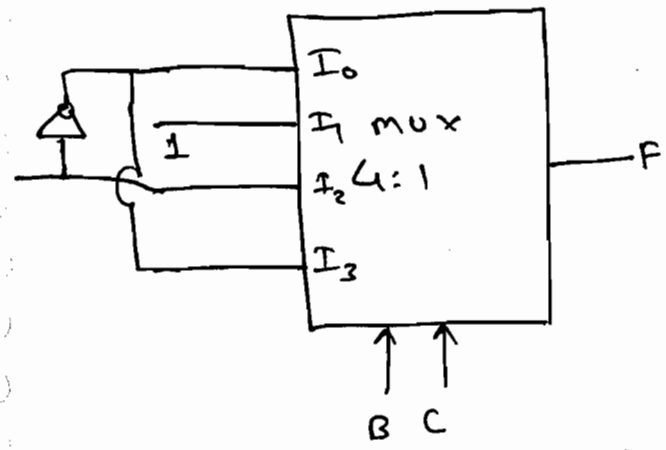
→ NOTE: If C & D are selection lines



Ex-1 $F(A,B,C) = \Sigma m(0,1,3,5,6)$ using 4:1 MUX.

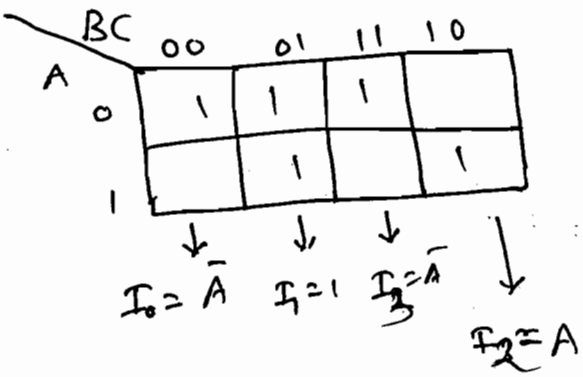
Ans:

Method-I:



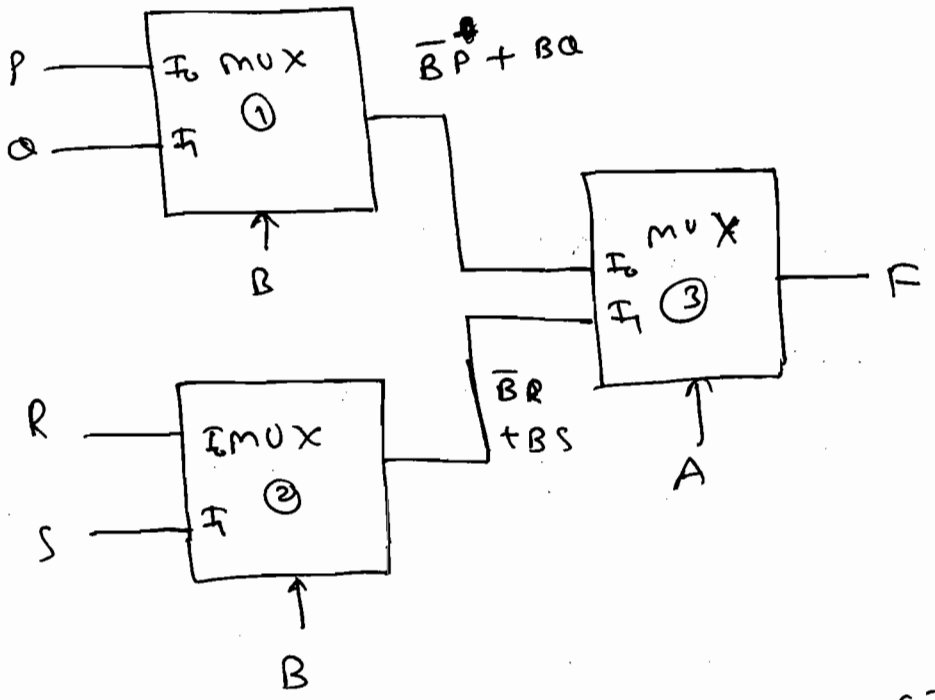
	I_0	I_1	I_2	I_3
$\bar{A}$	(0)	(1)	2	(3)
A	4	(5)	(6)	7
	$\bar{A}$	1	A	$\bar{A}$

Method-II (K-MAP)



Ex-2 In the following MUX tree, determine the values of $P, Q, R, S = ?$

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



$$\therefore F = \bar{A}(\bar{B}P + BQ) + A(\bar{B}R + BS).$$

$$\therefore F = \bar{A}\bar{B}\bar{P} + \bar{A}BQ + A\bar{B}R + ABS.$$

↓

AB \ AC	00	01	11	10
0	0	2	6	4
		1	1	1
1	1	3	7	5
	1			1

$I_0 = C$ $I_1 = \bar{C}$ $I_3 = \bar{C}$ $I_2 = 1$

P Q R S

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

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

Ex-2 Using n:1 MUX we can implement all '$\log_2 N$' variable function and Some of "$\log_2 N + 1$" variable functions [T/F].

Ans: let, 4:1 MUX

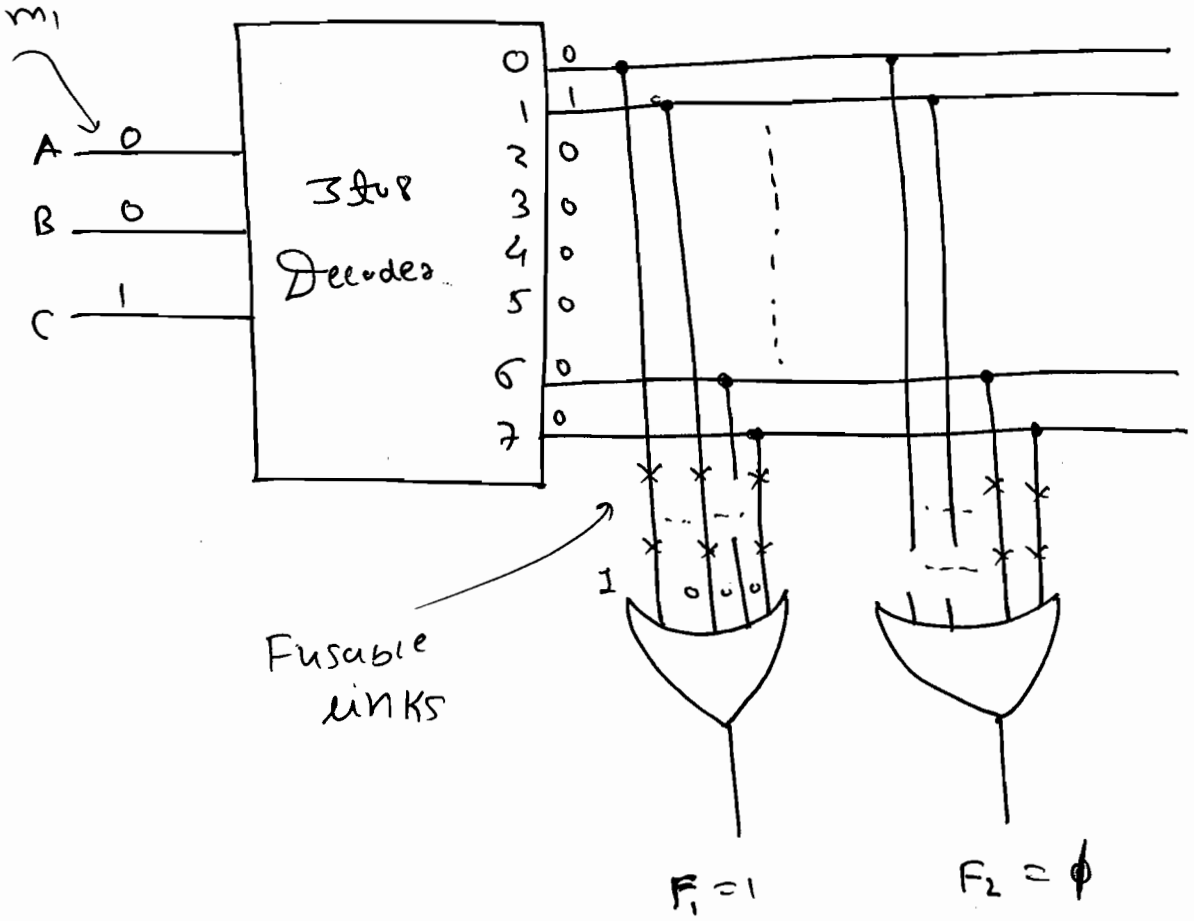
So, $\log_2 4 = 2$

→ True : because the other functions required external logic gates along with MUX.

* Rom (Read only memory).

⇒ Decoder + Prog. OR gates } = Rom.
(Fixed AND gates) (encoder)

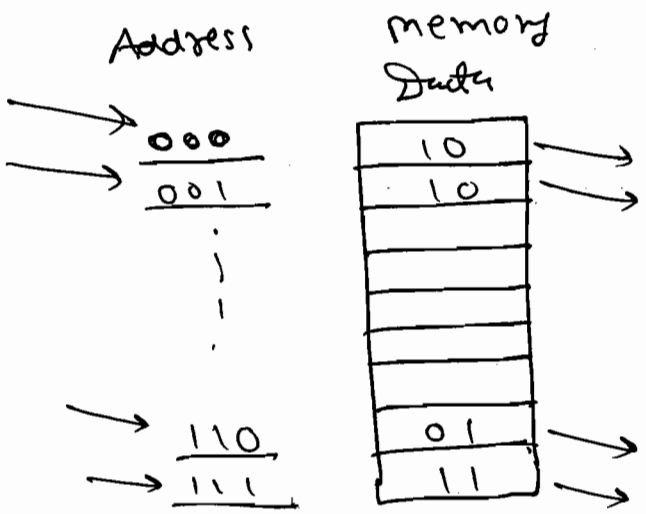
→ Rom is a Combinational circuits and we can use it to implement Sum of minterms expression as shown in the following example:



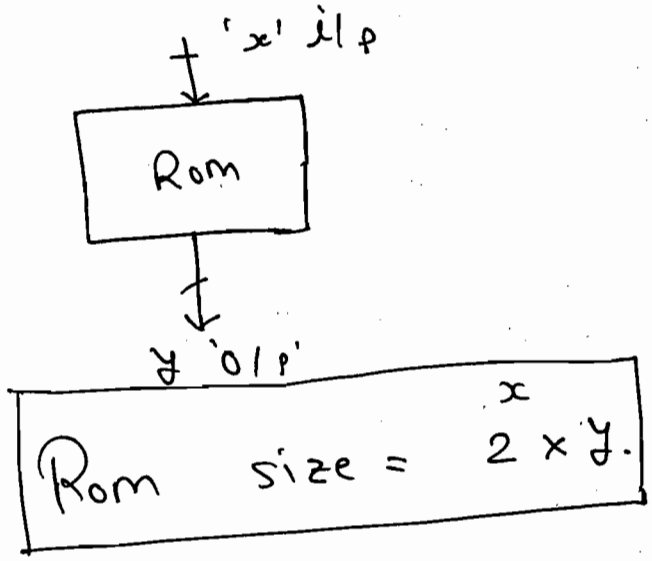
$F_1(A,B,C) = \Sigma m(0,1,7)$. size = $2^3 \times 2 = 16$.

$F_2(A,B,C) = \Sigma m(6,7)$.

⇒ **Read only memory**



⇒



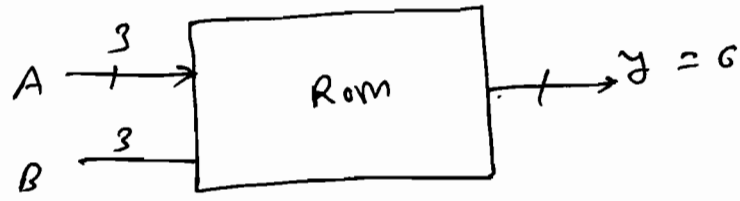
∴

→ Rom size indicates that how many bits can be stored.

Ex-1 Determine the size of the ROM for the following functions.

① Rom as 3 bit binary multiplier.

②



$y = 3+3 = 6$

$y = 6$ $\qquad$ $111_{10} * 111_{10} = 49_{10}$

$2^y > 49$

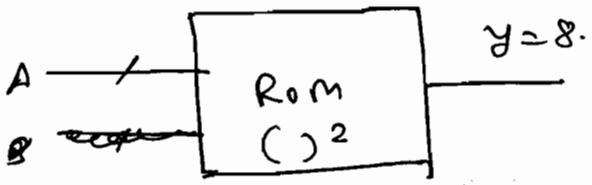
$\boxed{y = 6}$

$\therefore$ Rom size $= 2^x \times y = 2^6 \times 6 = 384.$

② Rom as 4-Bit squarer.

Ans:

$X = 8$



~~64~~

1111 $\qquad$ ~~64 × 64~~

1111 $\qquad$ ~~4 × 4~~ $\qquad$ ~~y=8~~

~~6~~ $\qquad$ ~~2 × 2~~

~~x = 4.~~ $\qquad$ ~~$2^4 \times 8 = 16 \times 8 = 128.$~~

$15 \times 15 = 225_{10}$

$\therefore$

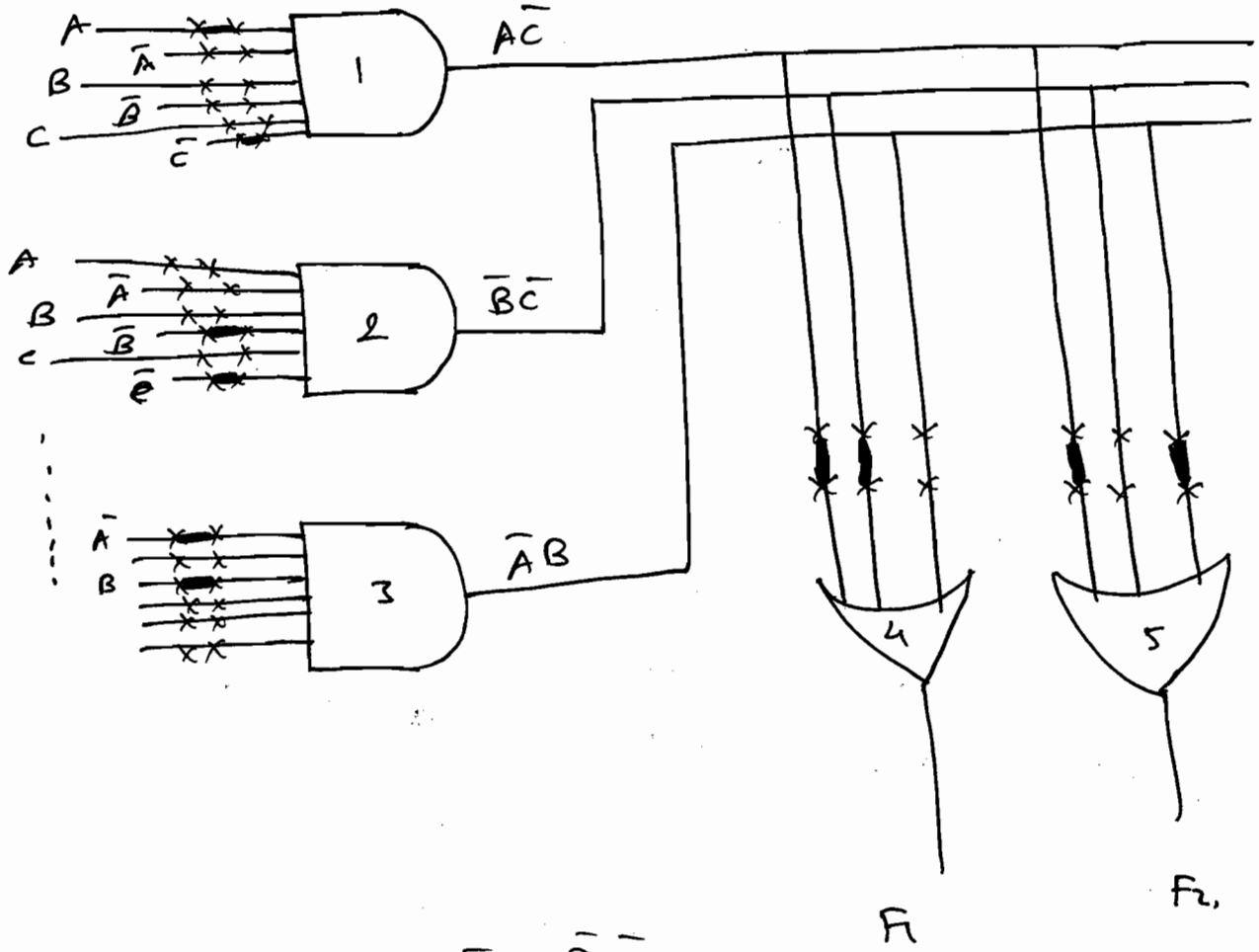
$2^y > 225$

$\boxed{y = 8}$, $x = 4.$

$\therefore$ size $= 2^4 \times 8 = 128.$

* PLA (Prog. Logic Array):

⇒ Prog. AND gates + Prog. OR gates.



$F_1(A,B,C) = A\bar{C} + \bar{B}\bar{C}$

$F_2(A,B,C) = \bar{A}B + A\bar{C}$.

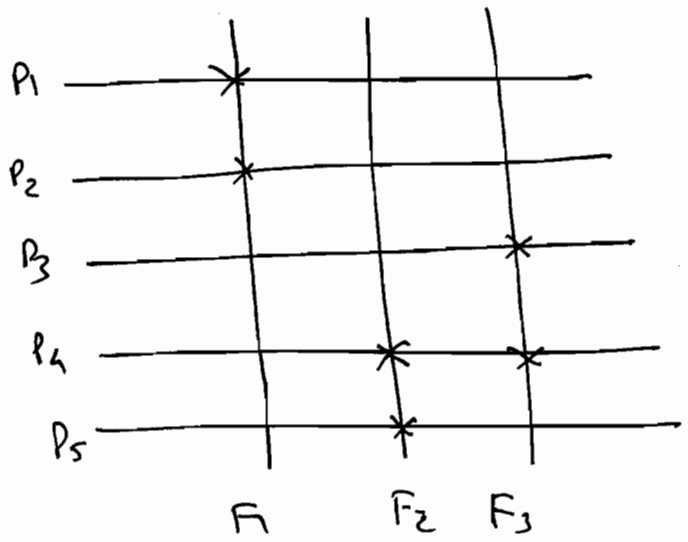
Product terms: ⟶ $P_1 = A\bar{C}$

$P_2 = \bar{B}\bar{C}$

$P_3 = \bar{A}B$.

PLA size ⇒ (3 inputs, 3 Product terms, 2 outputs).

Ex-1 In the following PLA Determine the product terms P_1, P_2, P_3, P_4, P_5



$F_1(A,B,C) = \bar{A}B\bar{C} + A\bar{B}C + AB.$

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

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

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

$F_1(A,B,C) = B\bar{C} + AB + AC = B\bar{C} + AC$

$F_2(A,B,C) = \bar{A}\bar{B}\bar{C} + ABC + \bar{A}C$

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

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

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

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

$F_2 = \bar{A}\bar{C} + \bar{A}B + BC = \bar{A}\bar{C} + BC$

$F_3 = \bar{A}B\bar{C} + BC + A\bar{B}C.$

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

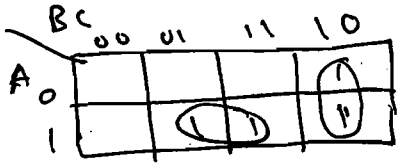
$= AC + BC + \bar{A}B\bar{C}$

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

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

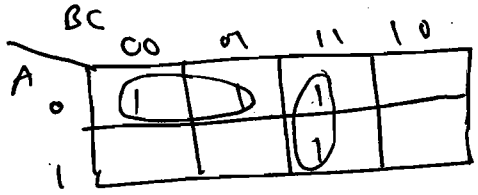
$= AC + \bar{A}B.$

$F_1 =$



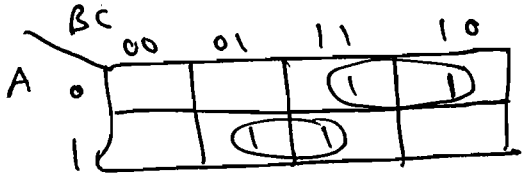
$F_1 = \underline{AC} + B\bar{C}$

$F_2 =$



$F_2 = \bar{A}\,\bar{B} + BC.$

$F_3 =$



$F_3 = \underline{AC} + \bar{A}B.$

$\therefore \quad P_2 = AC \;,\; P_1 = B\bar{C} \;,\; P_3 = \bar{A}B.$

$P_4 = \bar{A}B \;,\; P_5 = BC.$ or

$P_4 = \bar{A}B \;,\; P_4 = BC.$

size = (3, 5, 3)
(i/p, product term, o/p)

* PAL : Prog. AND gate + Fixed OR gate.

GAL : Generic Array logic

= Prog. AND gates + fixed OR gate
+ Prog output Logic
(E.g. Tri state O/P, Normal d/p etc).

Ex-1 How many invalid i/p combinations occur at input of BCD adder?

Ans: No. of Invaid BCD Combination at Input.

= Total Input Combinations
- No. of valid Combinations.

= (16x16) - (10x10)

= 156.

* Overflow:

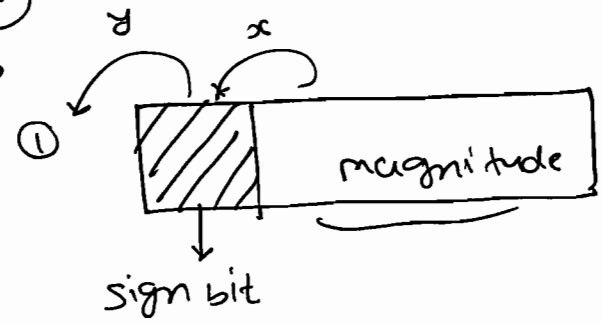
→ overflow is used in signed arithmatic.

→ overflow occurs whenever the result exceeds the range of signed no.s.

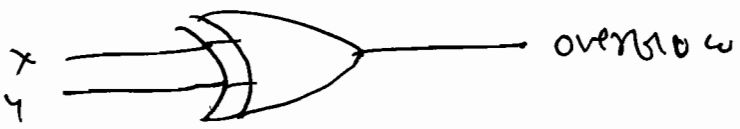
→ The overflow condition can be varified by the following methods:

(1)

→



If x=1 and y=0
(or)
x=0 and y=1
→ overflow occured.



(2) Overflow occurs if 2 +ve nos are 113 added the result is -ve or 2 -ve nos are added the result is +ve.

E.g. Consider 2's comp. nos.

(a)
$x=0$, $y=1$

1.01 (-3)

$+ \quad 1.10$ (-2)

$\otimes \quad 0\ \cancel{0}\ 1$

Overflow occured (or) Two -ve nos are added Result is +ve

(b)
$y=0$ (1) → $x=1$

010 (2)

$+ \quad 011$ (3)

101 (-3)

Overflow occured (OR) Two +ve nos are added Result is -ve

(c)
$y=1$ (1) → $x=1$

110 (-2)

$+ \quad 110$ (-2)

$\otimes \quad 1\ 0\ \cancel{0}$ ~~(-4)~~ (-3).

No overflow

Ex-1

x

$+ \quad y$

z

overflow

If $x=1$, $y=1$ but $z=0$

(or)

$x=0$, $y=0$ but $z=1$.

Expression for overflow i.e. is ⇒ $\boxed{\bar{x}\cdot\bar{y}\,z + x\cdot y\cdot\bar{z}}$