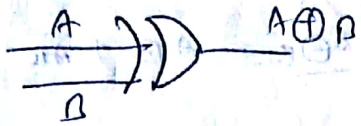
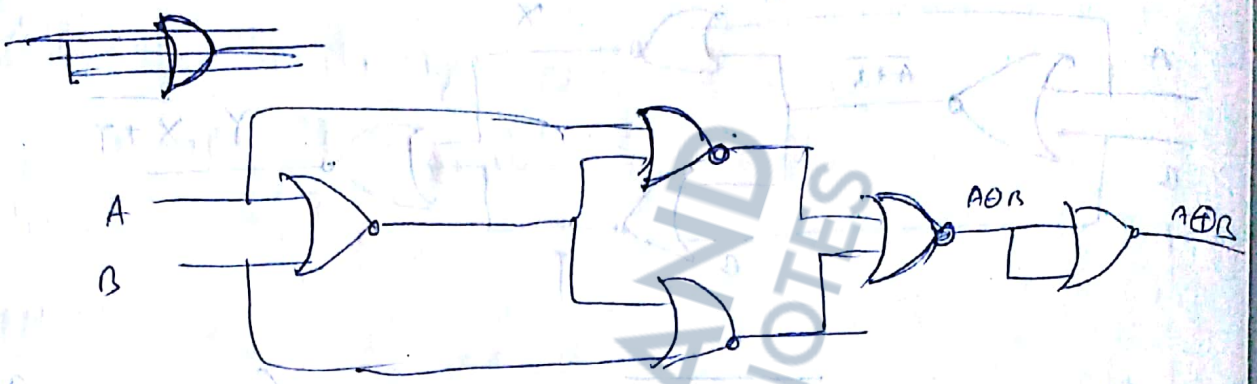
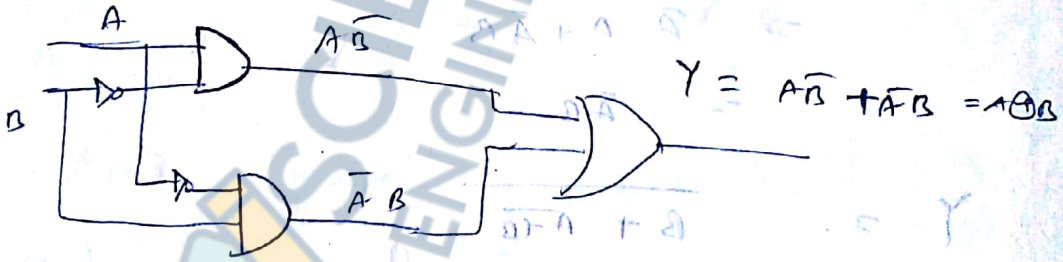
(vi) EX-OR Gate





Q Draw X-OR gate using basic gates

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



Standard Forms of Boolean Expression:-

There are 2 forms of Boolean expressions, These 2 forms are

(i) SOP (Sum of the Product) form.

(ii) POS (Product of the Sum) form.

SOP form

This is an expression in which AND terms (product) ORed (summed) together.

Ex:- $F = \bar{A}BC + AB\bar{C} + A\bar{B}C + \bar{B}C$

POS form :-

This is an expression in which OR terms (sum) ANDed (product) together.

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

Conversion of a General expression to SOP form

Any logic expression can be changed to SOP form by applying Boolean Algebra technique

Ex-1, $A(B+CD) = AB + ACD$

Ex-2 $AB + B(CD+EF)$

$= AB + BCD + BEF$

Ex-3:

$$\frac{\overline{(A+B)+C}}{}$$

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

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

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

The Standard SOP form:- [Canonical form]

A Standard SOP expression is one in which all the variables in the domain appear in the each product term.

Step-I:- Multiply each non standard product term by a term made up of sum of a missing variable and its complement. This results in 2 product terms.

Step-2:- Repeat step I, until all resulting product terms contain all variables in the domain in either complemented or uncomplemented form.

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

We know $A + \bar{A} = 1$

$B + \bar{B} = 1$

$C + \bar{C} = 1$

$D + \bar{D} = 1$

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

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

Consider

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

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

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

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

(3) $A\bar{B}\bar{C}D$ → It contains 4 variables.

∴ Finally,

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

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

Binary representation of a standard SOP form :-

Ex:- If $A\bar{B}.C.\bar{D} = 1$, what is $A\bar{B}CD$?

Ans :- $A.\bar{B}.C.\bar{D} = 1$,

Since these product is 1, individual terms must be 1. If any one would have zero, then their product would be zero.

$A = 1$

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

$C = 1$

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

$A\bar{B}.CD = 1010$

(Tolme)

Proof

A SOP expression is equal to 1 only if one or more of the product terms in the expressions is equal to 1.

(Because $A \cup 1 = 1$) [At least one expression = 1 ex: $0 \cup 1 = 1$, $0 \cup 1 \cup 1 = 1$]

The Standard POS form (Cannonical POS form)

A Standard POS expression is one in which all the variables in the domain appear in each sum term in the expression.

Steps

1. Add to each nonstandard product term a term made up of the product of the missing variable and its complement. This results in 2 sum terms. (because we can add 0 to anything without changing its value).
2. Apply the rule $A+BC = (A+B)(A+C)$
3. Repeat step 1 until all resulting sum terms contain all variables in the domain in either complemented or uncomplemented form.

Ex: Convert the following Boolean expression into standard POS form.

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

Ans:

$$A+\bar{B}+C = \underbrace{A+\bar{B}+C+}_{A}\ \underbrace{D\bar{D}}_{BC}$$

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

$\because D\bar{D}=0$, we can add.

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

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

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

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

→ $(A + B + \overline{C} + D)$ it contains all the variables.

∴ Binary

$(A + \overline{B} + C)(\overline{B} + C + \overline{D})(A + \overline{B} + \overline{C} + D)$ can be converted into standard pos form as

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

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

Converting standard SOP to standard POS

Step 1:- Evaluate each product term in the SOP expression. That is, determine the binary numbers that represent the product term.

Step 2:- Determine all of the binary number not included in the evaluation of Step 1.

Step 3:- Write the equivalent sum term for each binary number from step 2 and express in

POS form.

Ex:- $\bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC$

Ans:- $000 + 010 + 011 + 101 + 111$

i.e 0, 2, 3, 5, 7

So left 1, 4, 6

001, 100, 110

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

(In POS form $0 \rightarrow A$, $1 \rightarrow \bar{A}$

In SOP form $0 \rightarrow \bar{A}$, $1 \rightarrow A$)

Q) Convert POS → SOP $(A+B+\bar{C})(\bar{A}+B+C)(\bar{A}+\bar{B}+C) \rightarrow$? SOP

Q) Convert $\bar{A}B + A\bar{B} = ?$ into POS.

→ (A POS expression is equal to 0 only if one or more of sum terms in the expression is equal to 0)

Ex:- Determine the binary values of the variables for which following POS expression is equal to 0.

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

Ans :- The total expression = 0, if any one of them is 0.

If $A+B+C+D = 0$ $\Rightarrow A=0, B=0, C=0, D=0$

If $A+\bar{B}+\bar{C}+D = 0$ $\Rightarrow A=0, B=1, C=1, D=0$

If $\bar{A}+\bar{B}+\bar{C}+\bar{D} = 0$ $\Rightarrow A=1, B=1, C=1, D=1$

Min term :-

Minterm is a product terms which contain all the variables either in true form or complement form.

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

(min term.)

N:B - For n i/p variable, we have

2^n min terms.

$4 \rightarrow 2^4 \rightarrow 16$ minterms.

Minimal SOP form :- It is an SOP expression containing minm product term which can't be further simplified.

Max term:-

Maxterm is Sum term which contains all the variables given in the expression either in true form or in complement form.

Ex:- $F(A,B,C) = (A+B+C)(B+\bar{C})$

Minimal POS form:- It is a POS expression containing minm sum term which can't be further simplified.

Min term & Max term representation

For 3 i/p binary variable function, we have $2^3 = 8$ min terms as well as max terms.

x y z	min term (m_i)	Max term (m_i)
0 0 0	$\bar{x}\bar{y}\bar{z}$ (m_0)	$(x+y+z)$ M_0
0 0 1	$\bar{x}\bar{y}z$ (m_1)	$(x+y+\bar{z})$ M_1
0 1 0	$\bar{x}y\bar{z}$ (m_2)	$(x+\bar{y}+z)$ M_2
0 1 1	$\bar{x}yz$ (m_3)	$(x+\bar{y}+\bar{z})$ M_3
1 0 0	$x\bar{y}\bar{z}$ (m_4)	$(\bar{x}+y+z)$ M_4
1 0 1	$x\bar{y}z$ (m_5)	$(\bar{x}+y+\bar{z})$ M_5
1 1 0	$xy\bar{z}$ (m_6)	$(\bar{x}+\bar{y}+z)$ M_6
1 1 1	xyz (m_7)	$(\bar{x}+\bar{y}+\bar{z})$ M_7

Rough

→ For n input variables we have 2^n min as well as max term.

→ Min term is a product term whose value equal to 1 i.e denoted by

$$\sum_{i=0}^{2^n-1} m_i = 1$$ i.e either of them 1.

Ex-1

$$f(A,B,C) = \sum(0,2,4,6)$$

i.e $= 000 + 010 + 100 + 110$

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

→ Find the truth table?

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

→ Max term is a sum term whose value equal to 0, i.e denoted by

$$\prod_{i=0}^{2^n-1} M_i = 0$$ i.e either of them is 0.

Find the truth table.

Ex-1 $f(A,B,C) = \prod(0,2,4,7)$

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

$= (000)\ (010)\ (100)\ (111)$

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

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

→ Min term & Max term are complement to each other.

$\sum(0,2,4,6) = \prod(1,3,5,7)$

→ Minimization,

Ex - 1 (Simplify the following expression)

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

Ans:

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

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

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

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

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

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

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

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

$$= A\bar{B} + AC \quad \text{(Ans)} \rightarrow \text{(Minimized expression)}$$

formula

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

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

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

$$= 0 + 0 + 0$$

$$= 0$$

$A\bar{A} = 0$

$B\bar{B} = 0$

$C\bar{C} = 0$

(iii) $\bar{x}\bar{y}z + yz + xz$

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

$$= z[x + y + \bar{x}\bar{y}]$$

$$= z[(x + y) + \overline{x + y}]$$

$\equiv Z[\quad]$

$= Z \cdot [1]$

$= Z$ (Ans)

$\therefore A + \bar{A} = 1$

(iv)

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

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

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

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

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

$= AB + \bar{B}C$

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

one term is eliminated (i.e. AC)

Trick :- See one term is complemented i.e. $\bar{B}C$ for ans. Keep the complemented term & its corresponding uncomplemented term, eliminate the other one.

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

→ Complemented term

→ Corresponding uncomplemented term.

Keep these two.

Ans → $AB + \bar{B}C$.

$= Z \cdot [1]$

$= Z$ (Ans)

$\because A + \bar{A} = 1$

(iv)

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

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

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

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

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

$= AB + \bar{B}C$

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

one term is eliminated (i.e. AC)

Trick :- See one term is complemented i.e. $\bar{B}C$ for ans. Keep the complemented term & its corresponding uncomplemented term, eliminate the other one.

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

$\bar{B}C$ → Complemented term

B → Corresponding uncomplemented term.

Keep these two.

Ans → $AB + \bar{B}C$.

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

Ans:

$$AB + \bar{A}C + BC \cdot 1$$

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

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

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

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

(vi) Prove

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

(vii) Prove that

$$ABCD + AB\bar{C}D + ABC\bar{D} + AB\bar{C}D + ABCDE + AB\bar{D}E = AB$$

Ans:-

$$ABCD + AB(\bar{C}D) + ABC\bar{D} + AB\bar{C}D + ABCDE + AB\bar{D}E$$

$$= ABC(D + \bar{D}) + AB(\bar{C} + \bar{D}) + AB\bar{C}D + ABCDE + AB\bar{D}E$$

$$= ABC + AB\bar{C} + AB\bar{D} + AB\bar{C}D + ABCDE + AB\bar{D}E$$

$$= ABC + AB\bar{C}(1 + D) + AB\bar{D}(1 + E) + ABCDE$$

$$= ABC + AB\bar{C} + AB\bar{D} + ABCDE$$

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

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

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

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

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

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

Prove that

(ii) $\overline{AB + BC + CA} = \overline{AB} + \overline{BC} + \overline{CA}$

Proof:

$\overline{AB + BC + CA}$

$= \overline{AB} \cdot \overline{BC} \cdot \overline{CA}$ (Demorgan's thm $\overline{A+B} = \bar{A} \cdot \bar{B}$

$= (\bar{A} + \bar{B}) \cdot (\bar{B} + \bar{C}) \cdot (\bar{C} + \bar{A})$ (Again demorgan's thm $\overline{A \cdot B} = \bar{A} + \bar{B}$)

↑ multiply ↑

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

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

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

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

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

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

$A \cdot A = A$

$A + A = A$

$\overline{AB + BC + CA} = \bar{A}\bar{B} + \bar{B}\bar{C} + \bar{A}\bar{C}$ | $\because$ Gn $A + A = A$

(Proved)

Alternative Method for Converting SOP → POS

1) Take the Complement of given SOP expression & expand using Demorgan's thm.

2) Simplify the above expression using thm of Boolean algebra.

3) Take once again Complement of the simplified expression obtained in Step 2, to get the POS form

Ex:- $F = A\bar{B} + \bar{A}B$

Step I $\bar{F} = \overline{A\bar{B} + \bar{A}B}$

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

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

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

Step II $\bar{F} = \bar{A}A + \bar{A}\bar{B} + AB + B\bar{B}$ ($A\bar{A} = 0$, $B\bar{B} = 0$)

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

Step III $\bar{\bar{F}} = \overline{AB + \bar{A}\bar{B}}$

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

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

$A = A \cdot A$, $A = A + A$

$\Rightarrow \boxed{F = (\bar{A} + \bar{B})(A + B)}$ POS form

POS → SOP

Step 1 :- Expand the POS expression

Step 2 :- Simplify, using theorems of Boolean algebra.

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

Step 1 :-

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

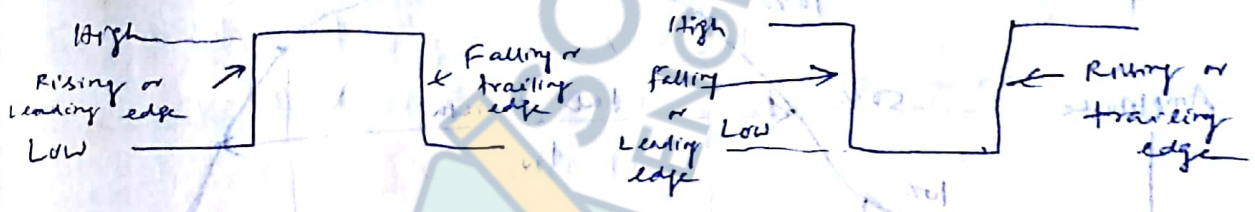
(0) (0)

Step 2 :-

$$= \bar{A}B + A\bar{B} \quad (\text{SOP form})$$

Q/ Design $A+B\bar{C}$ using NAND / NOR.

Digital Waveform :-



(a) +ve going pulse (b) -ve going pulse.

→ A positive going pulse is generated when the voltage (current) goes from its normally low level to its high & then backs to low level.

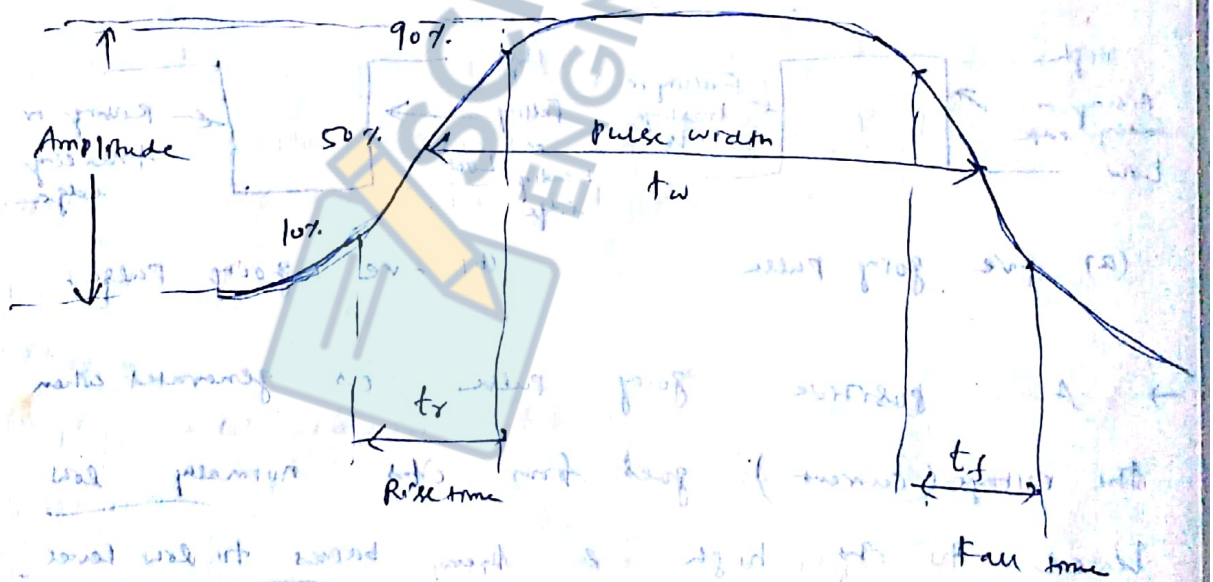
→ A -ve going pulse is generated when the voltage goes from its normally High level to its low level & back to high level.

t_r (Rise time)

The time required for the pulse to go from its low level to high level is called rise time (t_r). Practically, it is the measure of time from 10% of pulse amplitude to 90% of pulse amplitude.

t_f (Fall time)

The time required for the transition from High level to Low level is called the fall time (t_f). Practically it is the measure of time from 90% to 10% of pulse amplitude.



Duty cycle = $\frac{t_w}{T}$ = $\frac{\text{Pulse width}}{\text{Time Period}}$

$f = \frac{1}{T}$ i.e frequency = $\frac{1}{\text{Time Period}}$

Pulse Width :-

It is a measure of the duration of the pulse and is often defined as the time interval between 50% point on the rising & falling edge.

20

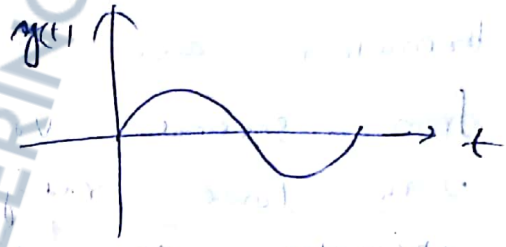
For 2010 Reut

1) (a) Analog, digital, discrete time signal ?

Analog :-

The analog signal is that type of signal which varies continuously with certain interval of time.

e.g. $y(t) = \sin t$



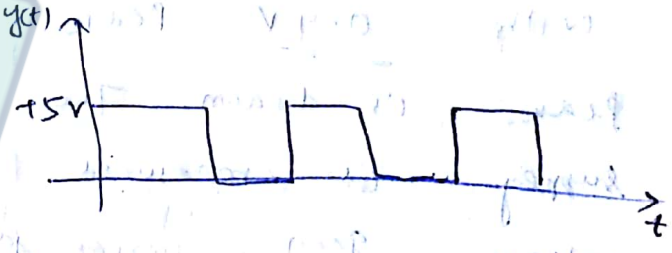
Digital :-

Digital signal is that type of signal which represented as sequence of numbers i.e magnitude at an instant of time.

e.g :- Pulse train shown.

High = +5V (Consider logic 1)

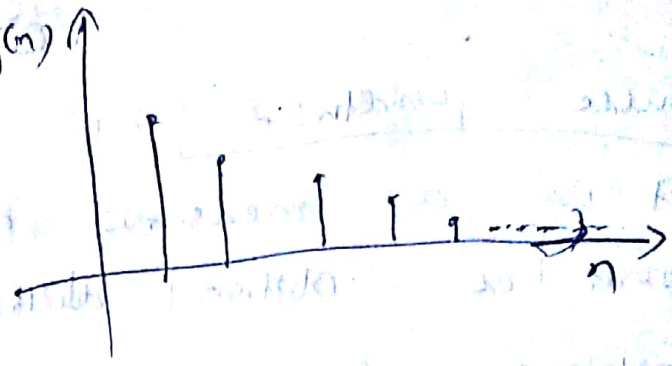
low = 0V (Consider logic 0)



Discrete-Time Signal :-

It is defined at certain specific values of time. The time instant need not be equidistant in practice. They are usually taken as equally ~~spaced~~ spaced intervals for computational convenience and mathematical help.

$$y(n) = \begin{cases} 0.2^n, & \text{for } n \geq 0 \\ 0, & \text{otherwise} \end{cases}$$



n = integer.

b) P-N junction diode made up of which material (Si, Ge, GaAs) will have highest thermal stability? Why?

Ans :- P-N junction diode made up of Si material will have highest thermal stability, because Silicon diodes operated in Avalanche breakdown are available with maintaining voltages from several volts to several hundred volts with power rating up to 50 watts. They can operates in high temperature.

Internal :- 4) An amplifier operating from ±5V supplies provides a 3.2 V peak sine wave across 50Ω load; when provided with 0.4 V peak i/p from which 2.0 mA peak i/s drawn. The avg current in each supply is measured to be 30 mA. Find voltage gain, current gain & power gain expressed in dB. as well as supply power, amplifier dissipation & amplifier efficiency.

Logic Gates & Boolean Algebra

1) KTU 2016 Prove that

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

Ans :-

$$\overline{AB+AC} + A\bar{B}C$$

$$= \overline{AB} \cdot \overline{AC} + A\bar{B}C \qquad \because \overline{A+B} = \bar{A}\cdot\bar{B}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(i) $\overline{AB+AC} + A\bar{B}C$

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

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

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

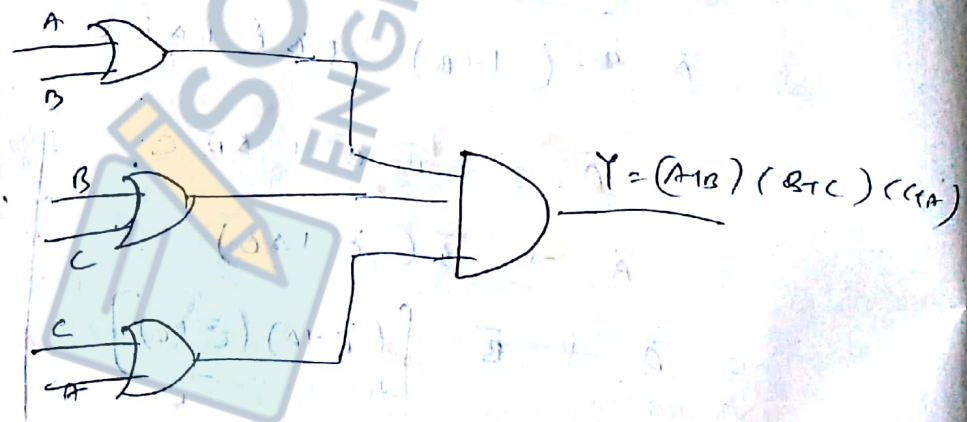
(Proved)

(2) (a) Draw the logic diagram for the boolean expression

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

(b) Simplify the above equation & draw the logic diagram for simplified expression.

Ans :-



(ii) $(A+B)(B+C)C(C+A)$

$= (AB+AC+B.B+BC)(C+A)$

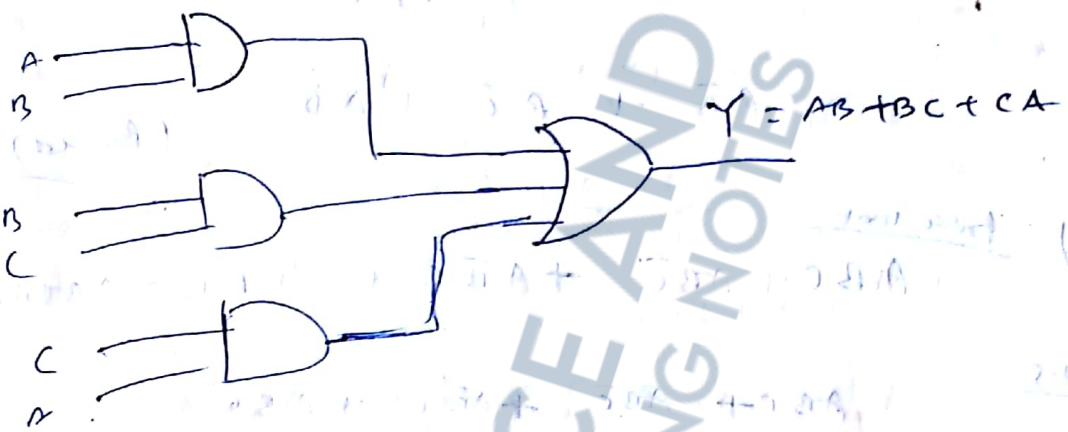
$= (AB+AC+B+B\bar{C})(C+A)$

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

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

$= BC + BA + A \cdot C \cdot C$ | $C \cdot C = C$

$= AB + BC + AC$



3) Prove that

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

Proof

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4) Prove that

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

L.H.S

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

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

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

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

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

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

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

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

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

$= AB + C(A+B)$

$= AB + AC + BC$

$= R.H.S$ (Proved)

1] Find the complement of the function given below and Implement using logic gates.

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

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

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

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

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

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

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

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

$$= x + yz + \bar{x}z$$

$$= (x+\bar{x})(x+z) + yz$$

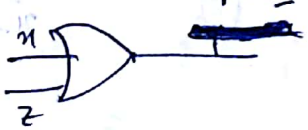
$$= (x+z) + yz$$

$$= x + z(1+y)$$

$$\bar{F} = x + z$$

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

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



6) Simplify the following expression

$$F(x,y,z) = (x+y)\left[\overline{x(\bar{y}+\bar{z})}\right] + xy + \bar{x}\cdot\bar{z}$$

$$= (x+y)\left[\overline{(\bar{z}+\bar{y})\,x}\right] + \bar{x}\cdot\bar{z} + xy$$

$$= (x+y)\left[\bar{x} + \overline{(\bar{z}+\bar{y})}\right] + \bar{x}\cdot\bar{z} + xy$$

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

$$= x + x\bar{z}\cdot\bar{y} + xy + \bar{x}\cdot\bar{z}$$

$$= x(1+\bar{z}\cdot\bar{y}+y) + \bar{x}\cdot\bar{z}$$

$$= x + \bar{x}\cdot\bar{z}$$

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

$$= x + \bar{z}$$

(Ans)

7) If $\bar{A}B + A\bar{B} = C$

Then prove $\bar{A}C + A\bar{C} = B$

Proof:

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

$$= A\left(\overline{\bar{A}B + A\bar{B}}\right) + \bar{A}\left(\bar{A}B + A\bar{B}\right) \quad \Big| \quad \because C = \bar{A}B + A\bar{B}$$

$$= A\left(\overline{\bar{A}B} \cdot \overline{A\bar{B}}\right) + \bar{A}\cdot\bar{A}\cdot B + \bar{A}\cdot AB \;(\to 0)$$

$$= A\left[\left(\bar{\bar{A}} + \bar{B}\right)\left(\bar{A} + \bar{\bar{B}}\right)\right] + \bar{A}B$$

$$= A\left[\left(A + \bar{B}\right)\left(\bar{A} + B\right)\right] + \bar{A}B$$

$$= A\left[A\bar{A} + AB + \bar{B}\bar{A} + \bar{B}B\right] + \bar{A}B \quad (A\bar{A}, \bar{B}B \to 0)$$

$$= A\left[AB + \bar{A}\bar{B}\right] + \bar{A}B$$

$$= AAB + A\bar{A}\bar{B} + \bar{A}B \quad (A\bar{A} \to 0)$$

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

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

$$= B \quad \text{(Proved)}$$

8) Simplify the logic expression & draw the logic diagram for the simplified expression.

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

P.T.O

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

$$= (\bar{x} + xy\bar{z})\left[1 + (x + \bar{x}\bar{y}\bar{z})\right]$$

$$= \underbrace{\bar{x}}_{A} + \underbrace{xy}_{B}\underbrace{\bar{z}}_{C}$$

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

$$= \cancel{\bar{x} + y\bar{z}}$$

$$= \bar{x} + y\bar{z} \quad \text{(Ans)}$$

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

express as

(a) Minimal SOP (ii) minimal POS

Ans:

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

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

The above expression is itself minimal SOP, since further reduction is not possible.

∴ (i) Minimal SOP = $AB + A\bar{C} + D$.

(ii)

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

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

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

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

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

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

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

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

$= [\bar{A}(1 + C + \bar{B}) + \bar{B}C] \cdot \bar{D}$

$= [\bar{A} + \bar{B}C]\bar{D}$

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

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

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

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

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

= Minimal POS form.

6) Minimize the following expression

Ex: $F = \sum(4,5,6,7)$

Ans:

$$F = \sum(4,5,6,7)$$

$$= 100 + 101 + 110 + 111$$

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

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

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

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

$$\boxed{F = A}$$

11) Convert following expression into POS form.

$$F = \prod(0,1,2,6)$$

Ans: $F = \prod(3,4,5,7)$

(∵ $F = m_0 + m_1 + m_2 + m_6$

$$\bar{F} = \overline{m_0 + m_1 + m_2 + m_6}$$

$$= \bar{m}_0 \cdot \bar{m}_1 \cdot \bar{m}_2 \cdot \bar{m}_6$$

$$\bar{F} = m_3 + m_4 + m_5 + m_7$$

$$F = \overline{m_3 + m_4 + m_5 + m_7}$$

$$= \bar{m}_3 \cdot \bar{m}_4 \cdot \bar{m}_5 \cdot \bar{m}_7$$

$F = M_3 \cdot M_4 \cdot M_5 \cdot M_7$

$F = 011 \cdot 100 \cdot 101 \cdot 111$

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

12) Simplify the following boolean expression

$F = \Sigma(0,2,4,6)$

~~$F = 000 + 010 + 100 + 110$~~

$F = 000 + 010 + 100 + 110$

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

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

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

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

$= \bar{C}$ (Ans).

13) Realize the following expression using

(a) Basic logic gates.

(b) universal logic gates.

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

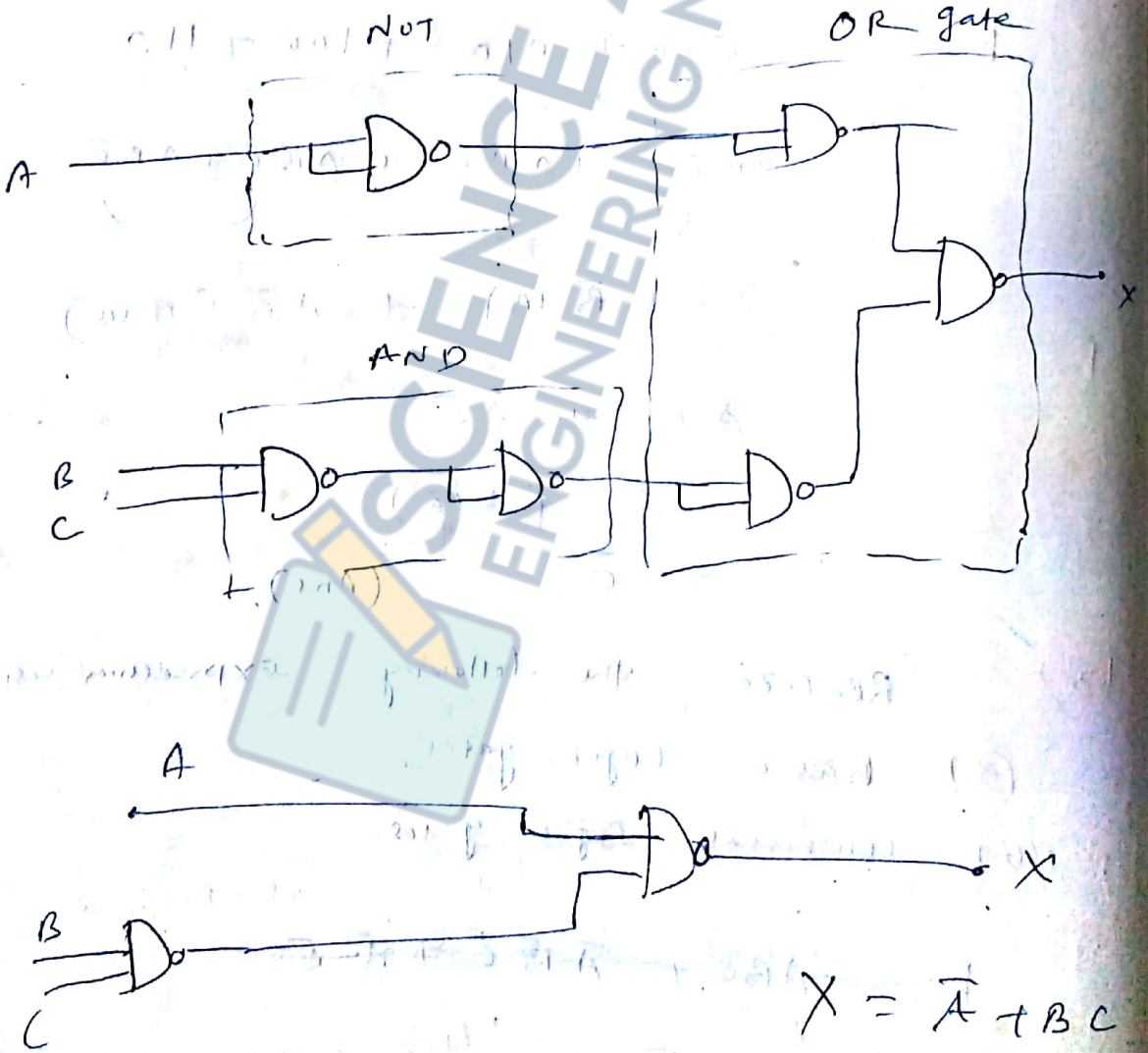
$F = \bar{A} + BC$

(a) using Basic logic gates:-

①

$F = \bar{A} + BC$

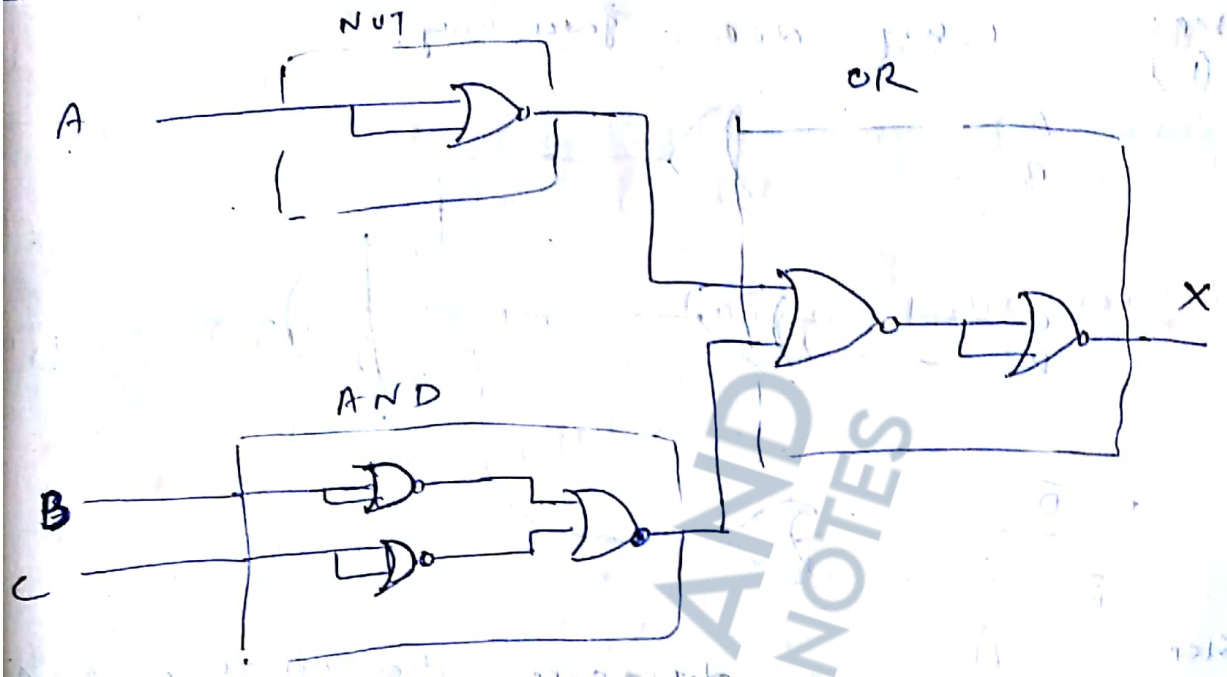
Replacing each gate by corresponding NAND gate



$X = \bar{A} + BC$

(∵ 2 Not gates cancel each other $\bar{\bar{A}} = A$)

Replacing each gate by NOR gate



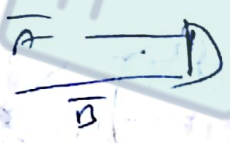
$$X = \bar{A} + BC$$

Note:-

→ Bubbled AND = NOR

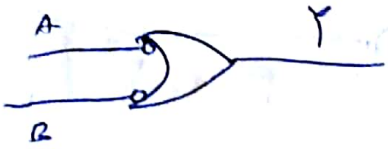
→ Bubbled OR = NAND

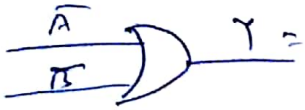
Bubbled AND :-



$Y = \bar{A} \cdot \bar{B} = \overline{A+B}$ = NOR

Bubbled OR

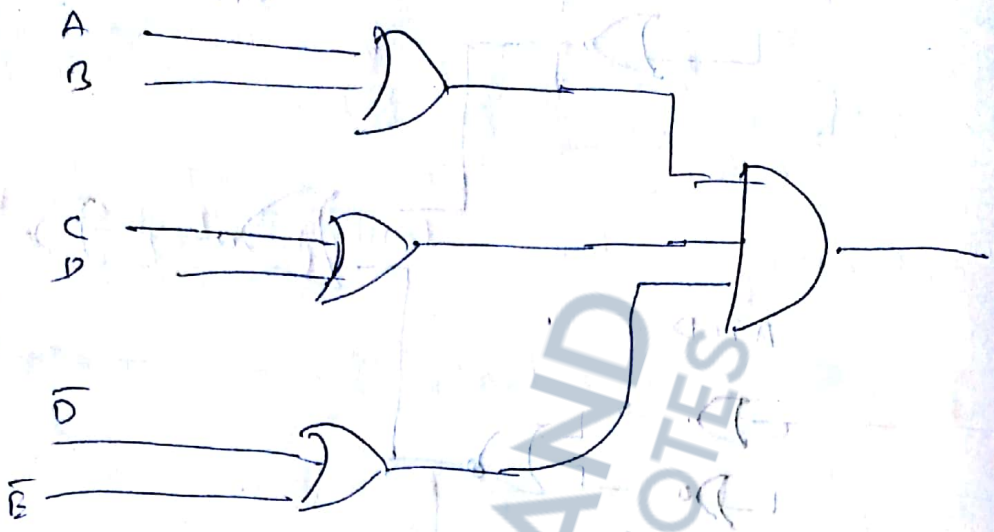




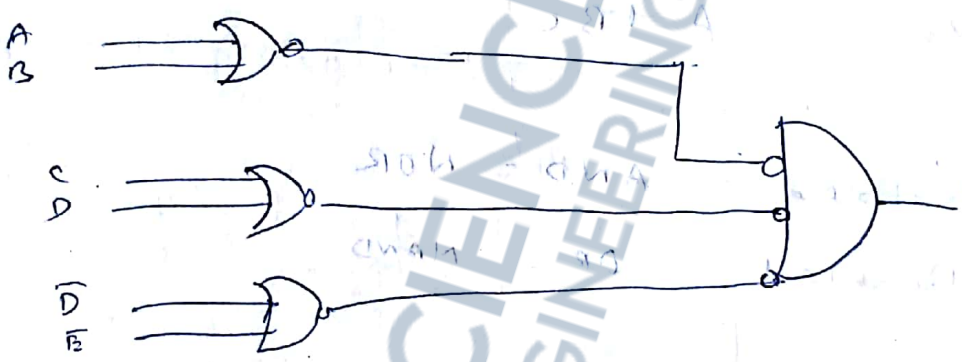
$Y = \bar{A} + \bar{B} = \overline{AB}$ = NAND.

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

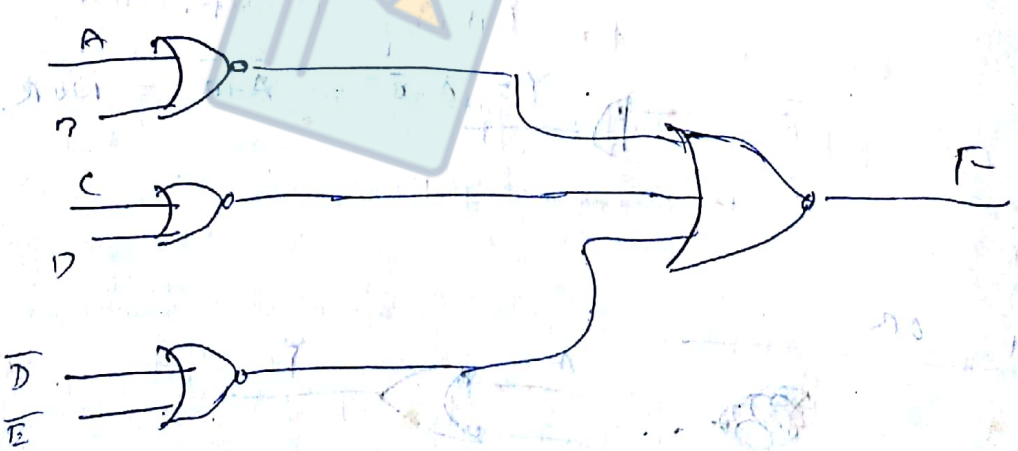
Step (1) Using NOR gates only!



Step (2) Adding 2 Not gate does not change ($\bar{\bar{A}} = A$)



(3) Bubbled AND = NOR



Another Method of Using NAND and NOR

Q) Implement $A+B\bar{C}$ Using Minm NAND gates & Minimum NOR gates.

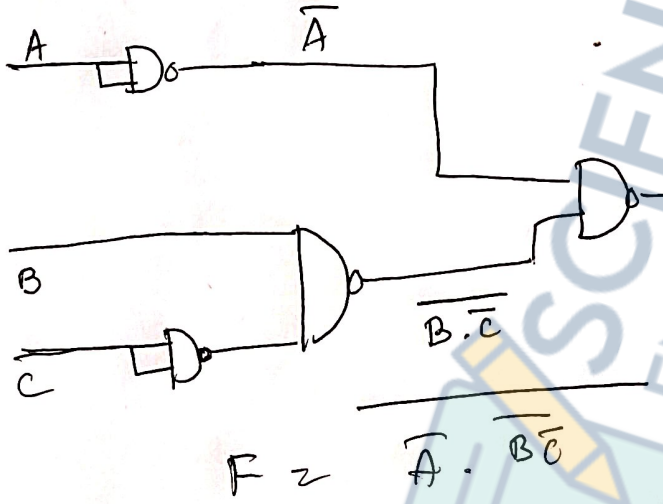
Ans :

Minimum NAND gates
(express on product term bar)

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

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

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



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

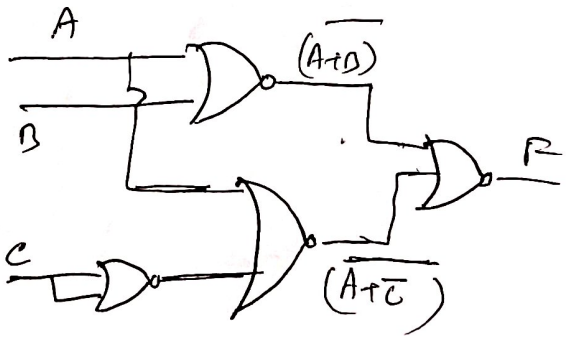
Minimum NOR gates
(express on sum term bar)

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

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

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

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



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

Minimization of Boolean function

1) Using Karnaugh Map or (K-Map)

The Karnaugh map is used for simplifying Boolean expressions to their minimum form. A minimized SOP expression contains the fewest possible terms with fewest possible variables per term.

Mapping a Standard SOP expression

Step 1:- Determine the binary value of each product term in the Standard SOP expression.

Step 2:- As each product term is evaluated, place 1 on the K-map in the cell having the same value as the product term.

Ex:1 Map the following SOP expression on a K-map.

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

Ans:- $\bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} + ABC$

001 010 110 111

Binary values →

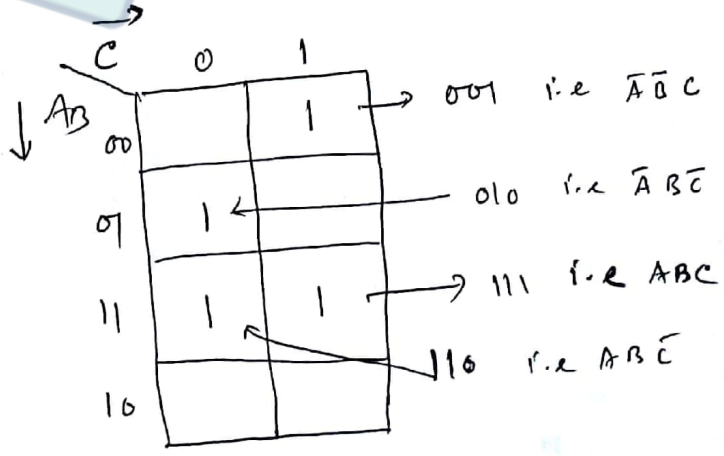


fig:- 3 Variable K-Map

2) Map

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

Ans:- 0011 0100 1101 1111 1100 0001 1010

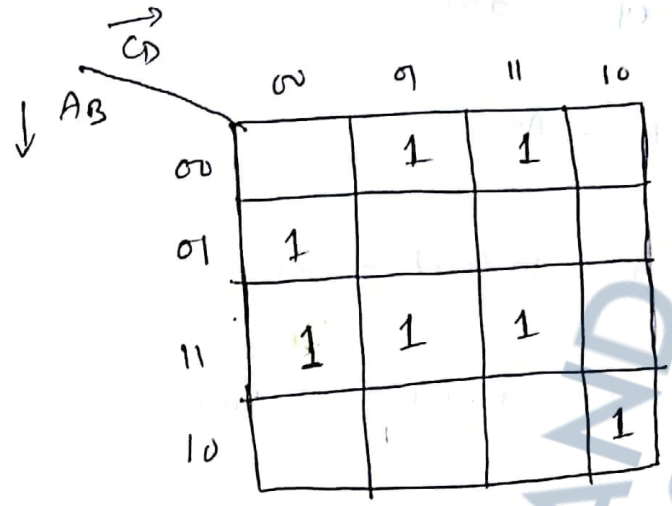


fig:- 4 Variable K-Map

Note :-1) The cell in a K-Map are arranged so that there is only a single variable change between adjacent cells. Adjacency is defined by a single variable change. Cells that differ by only one variable are adjacent.

Ex:-	010 cell is adjacent to	000, 110	Reason: 1 bit change
		001, 111	2 bit change.

[fig 3]

010 but not "

010 → 000, 010 → 110 (1 bit change)

010 ↔ 001, 010 → 111 (2 bit change)

2) A Boolean expression must first be in standard form before it is used for a K-Map. If an expression is not in standard form, then it must be converted to standard form.

Ex-2

Map $\bar{A} + A\bar{B} + AB\bar{C}$ on a K-Map.

A:- First Convert the expression into Standard SOP form.

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

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

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

2nd Method

Given SOP expression $\bar{A} + A\bar{B} + AB\bar{C}$

To Convert into Standard SOP form.

for $\bar{A}$ → the missing terms are B & C.

So first write the binary value of the Variable $\bar{A}$; Then attach all possible values for missing Variables B & C. as follows

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

← (2)

(1) & (2) are Equal.

Similarly

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

$\rightarrow A\bar{B}\bar{C}$ — (3)

$\rightarrow A\bar{B}C$

Combining (2) & (3), we have

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

Mapping

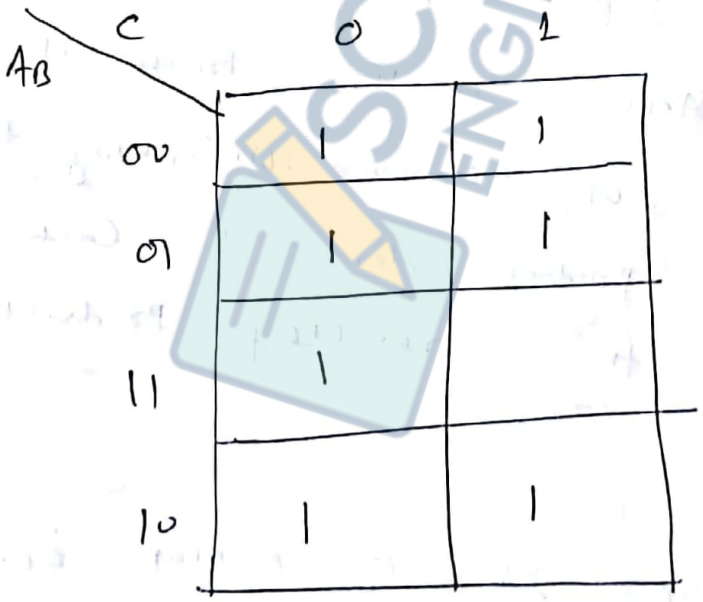
$\bar{A} + A\bar{B} + AB\bar{C}$

i.e

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

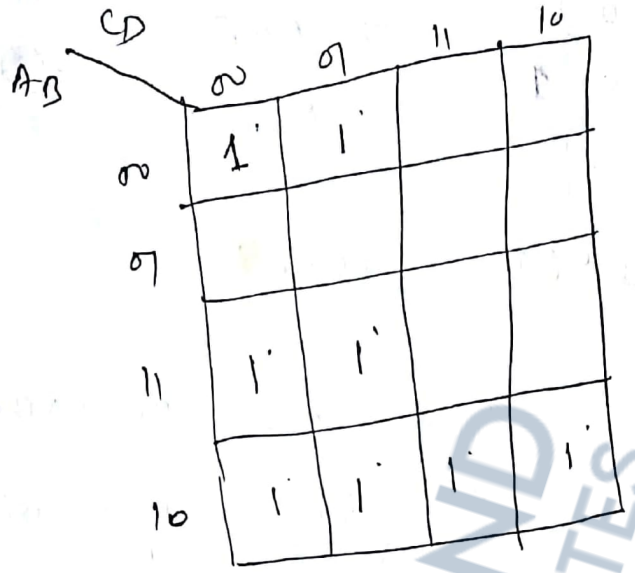
$A\bar{B}$
1 0 0
1 0 1

$AB\bar{C}$ → 1 1 0



Ex: 3) Mapping the following SOP expression to K-Map.

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



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

1010 0001 1011

0	00	0	10	00	110	0	
0	00	1	10	01	110	1	
0	00	0	10	10			
1	00	1	10	11			

K-Map Simplification of SOP expression

After an SOP expression has been mapped, there are 3 steps in the process of obtaining a minimum SOP expression: (A) Grouping the 1s, (B) determining the product term for each group, and (C) Summing the resulting product term.

(A) Grouping of 1's

You can group 1s on K-Map according to the following rules by enclosing those adjacent cells containing 1s. The goal is to maximize the size of the groups and minimize the number of groups.

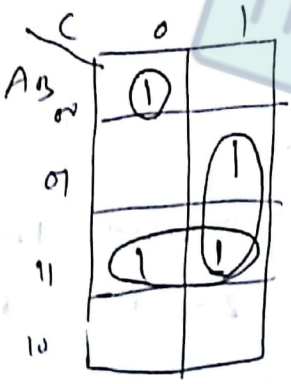
1. A group must contain either 1, 2, 4, 8, or 16 cells, which are all power of two. In the case of a 3-variable map, $2^3 = 8$ is the maximum group.

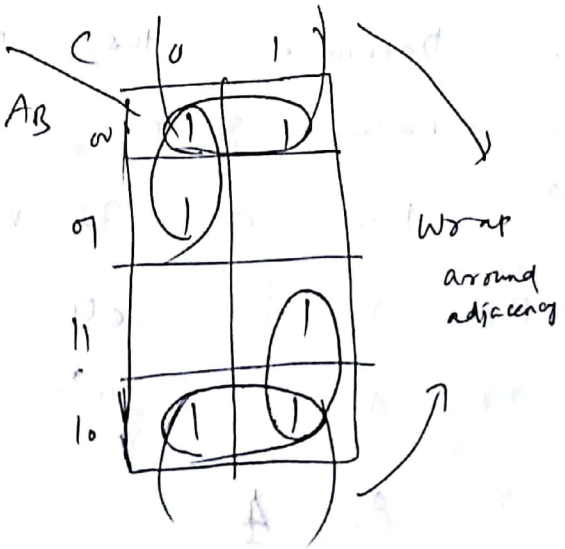
2. Each cell in a group must be adjacent to one or more cells in that same group, but all cells in the group don't have to be adjacent to each other.

3. Always include the largest possible number of 1s in a group in accordance with rule 1.

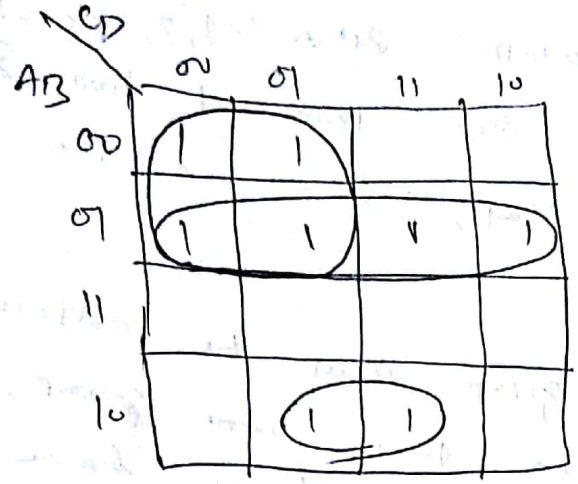
4. Each 1 on the map must be included in at least one group. The 1s already in a group can be included in another group as long as the overlapping groups include non common 1s.

Ex 4:

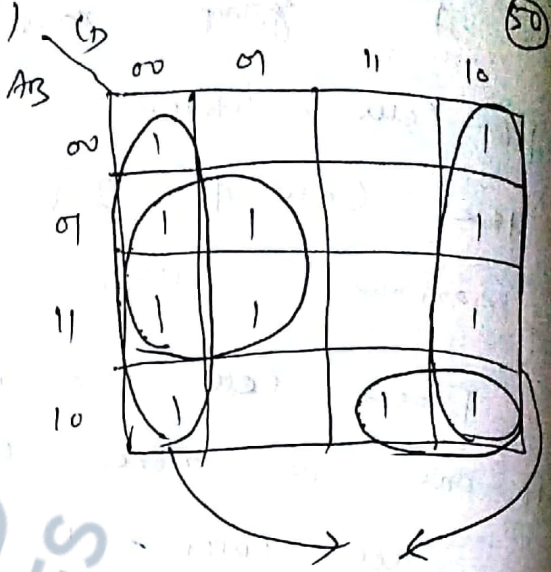




6)



7)



Wrap around adjacency

(B) Determining the minimum Sop expression from the map

1. Group the cells that have 1s. Each group of cells containing 1s creates one product term composed of all variables that occurs in only one form (either uncomplemented or complemented) within the group. Variables that occure both uncomplemented and complemented within the group are eliminated. These are called Contradictory variables.

2. Determine the minm product term for each group.

(a) For a 3-variable map:

(1) A 1-cell group yields 3-variable product term.

(2) A 2- " " " 2- " " "

(3) A 4- " " " 1 " " "

(4) A - 8 " " " a value of '1' for the expression.

(b) For a 4-variable Map

(1) A 1-cell group yields a 4-variable product term.

(2) A 2-cell group yields 3-variable product term.

(3) A 4-cell " " 2 " " "

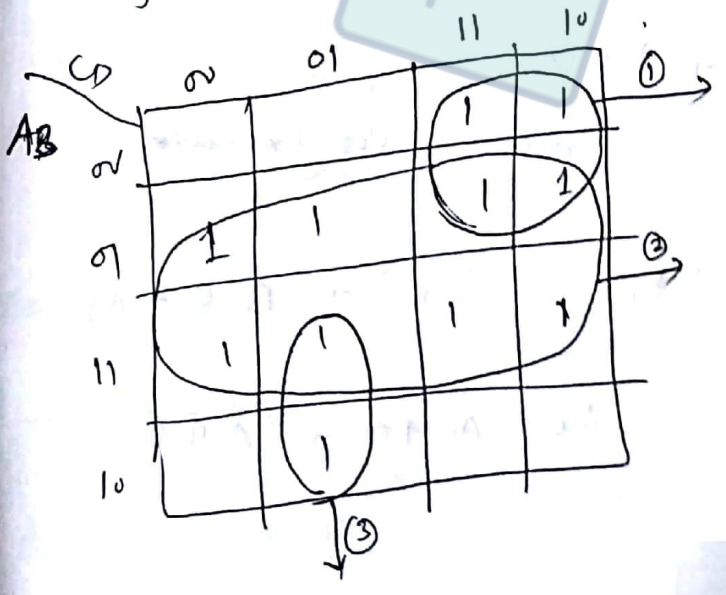
(4) A 8-cell " " 1 " " "

(5) A 16 " " " a value of '1' for the expression.

(C) Summing the resulting product term

When all the minimum product terms are derived from K-map, they are summed to form the min SOP expression.

Ex: 8 : Determine the product term for the K-map as shown below.



Ans:- $\sum m(2,3,4,5,6,7,9,12,13,14,15)$

①

For this group D is changing 1 → 0

B is changing 0 → 1

A = const = 0 = $\bar{A}$

C = const = 1 = C

∴ product term = $\bar{A}C$

For group (2)

A is changing 0→1

C is " .

D " " .

Only Constant is $B = 1 = B$.

∴ Product term is B.

< Since 8 cells for a 4 variable map we got 1-Variable term >

For group (3)

$\bar{C}D$ is Const., B is changing 1→0.

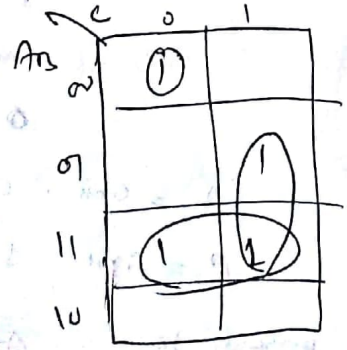
A is constant.

∴ Product term $A\bar{C}D$.

∴ The resulting minimum SOP expression is the sum of these product terms:

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

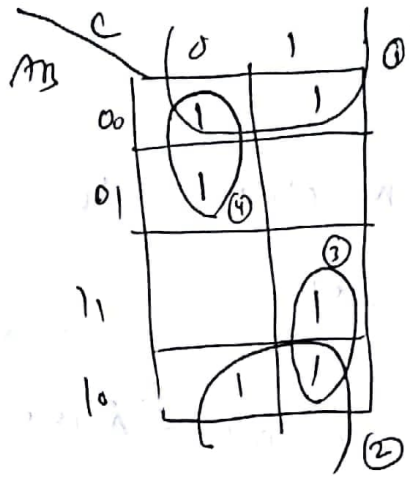
Ex :- Determine the minm SOP expression.



Ans :- $\bar{A}\bar{B}\bar{C} + BC + AB$

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

Ex - 10



① & ② Combine

Combined bcz $\bar{A}\bar{B}$ & $A\bar{B}$

$\bar{B}$ is const. A is changing

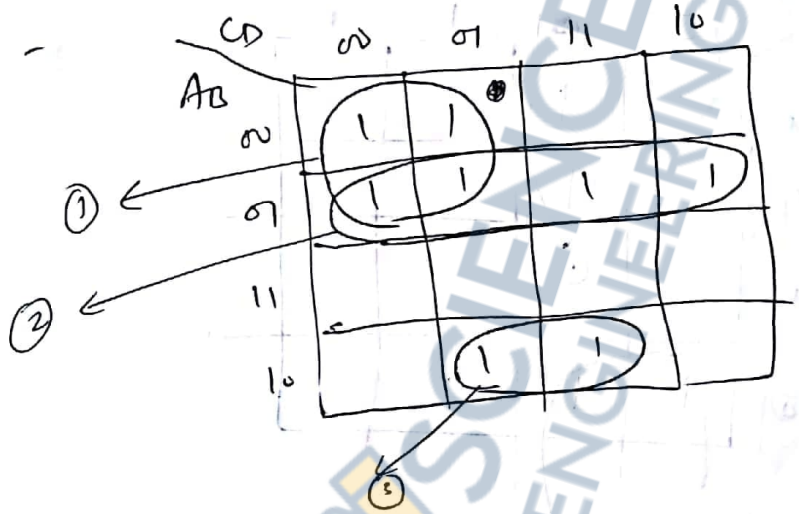
① & ② → $\bar{B}$

③ → AC

④ → $\bar{A}\bar{C}$

$\therefore$ SOP = $\bar{B} + AC + \bar{A}\bar{C}$

ex - 11 :-



group

① → $\bar{A}\bar{C}$

② → $\bar{A}B$

③ → $\bar{A}\bar{B}D$

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

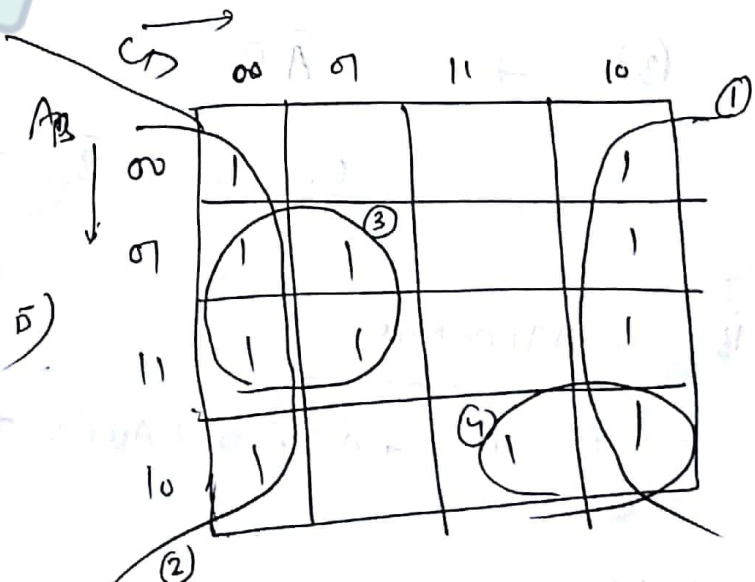
ex - 12

group

① & ② → $\bar{D}$ (∵ $\bar{C}\bar{D}$ & $C\bar{D}$)

③ → $B\bar{C}$

④ → $A\bar{B}C$



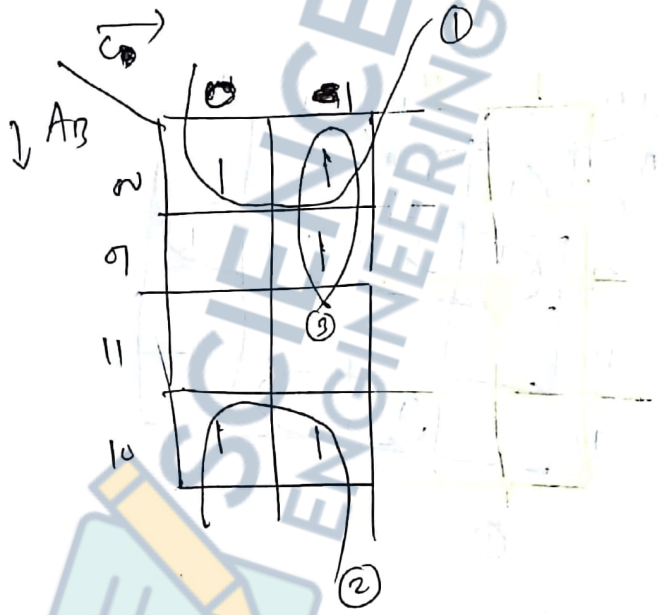
SOP → $\bar{D} + B\bar{C} + A\bar{B}C$

Ex :- 14) Use K-Map to minimize the Standard SOP.

Ans: $A\bar{B}C + \bar{A}BC + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C}$

Ans: The binary values of the expression are

$101 + 011 + 001 + 000 + 100$



① & ② → $\bar{A}\bar{B}$ & $A\bar{B}$ → $\bar{B}$

③ → $\bar{A}C$

∴ SOP = $\bar{B} + \bar{A}C$

Ex 15) Minimize

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

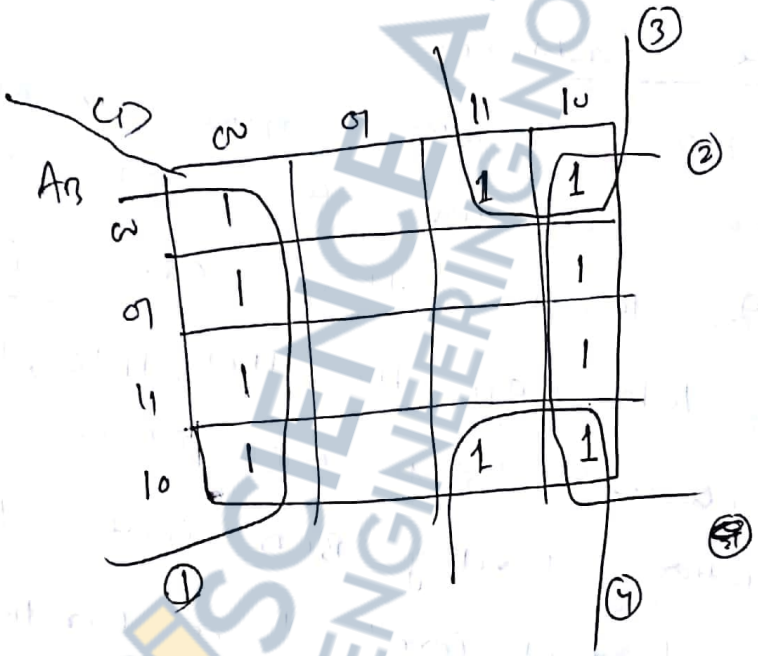
Ans:- Converting $\bar{B}\bar{C}\bar{D}$ to Standard SOP

0	000	$\rightarrow \bar{A}\bar{B}\bar{C}\bar{D}$
1	000	$\rightarrow A\bar{B}\bar{C}\bar{D}$

So Binary numbers corresponding to SOP are

0000] 0100, 1100, 0011, 1011, 0010, 0110,
1000] 1110, 1010

↑
$\bar{B}\bar{C}\bar{D}$



① & ② → $\bar{C}\bar{D}$ & $C\bar{D}$ → $\bar{D}$

③ & ④ → $\bar{A}\bar{B}C$ & $A\bar{B}C$ → $\bar{B}C$

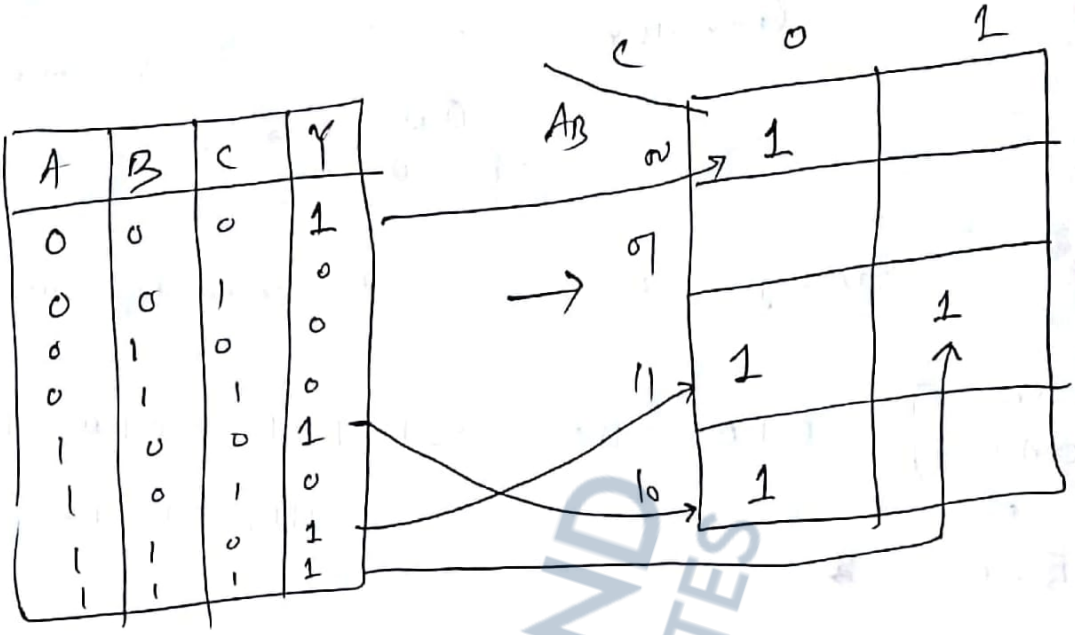
∴ SOP = $\bar{D} + \bar{B}C$

Mapping Directly from truth table.

The ones in the O/P Column of a truth table can be mapped directly onto K-Map into the cells corresponding to the values of the associated I/P variable combinations.

Ex:

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



Don't Care Conditions:-

Sometimes a situation arises in which some o/p variable combinations are not allowed. For example, in BCD there are 6 invalid combinations: 1010, 1011, 1100, 1101, 1110, 1111. Since these unallowed states will never occur in an application involving BCD code, they can be treated as "don't care" terms w.r.to their effect on the o/p. That is, for these 'don't care' terms either a 1 (SOP) or a 0 (POS) may be assigned to the o/p; it really does not matter since they will never occur.

The 'don't care' terms can be used to advantage on the K-map. Figure 1(b) shows that for each 'don't care' term, an X is placed in the cell. When grouping the 1s, the Xs can be treated as 1s to make a

larger grouping of as '0's if they can't be used to advantage. The larger a group, the simpler the resulting term will be.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

fig 2 (a)

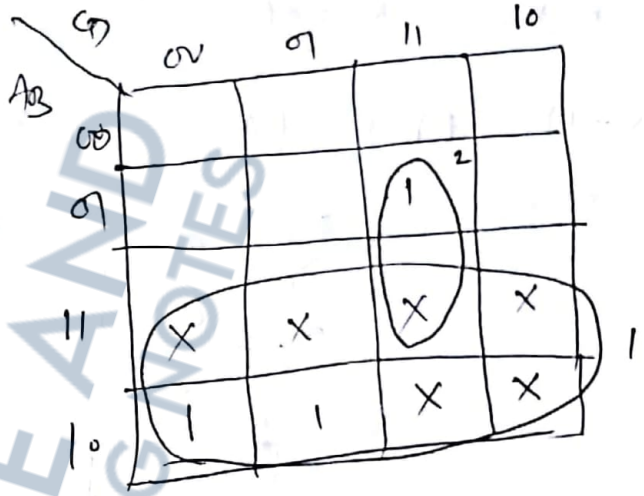


fig 1 (b)

① → A

② → BCD

Note:- If taking don't care helps in minimizing take it otherwise don't take them as they add an extra term.

∴ SOP → A + BCD

If don't care is not taken → $A\bar{B}\bar{C} + \bar{A}BCD$

i.e the advantage of using don't care terms to get simplest expression.

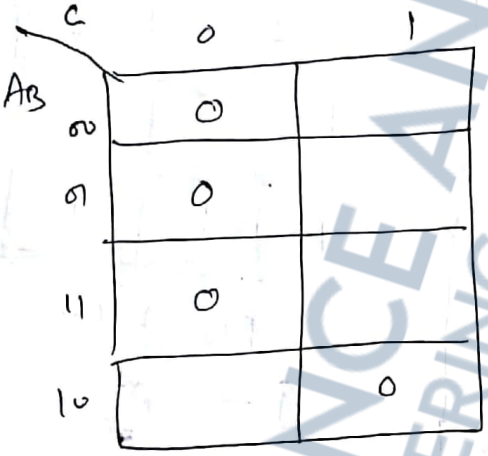
K-Map POS Minimization

For a POS expression in standard form, a '0' is placed on the K-map for each sum term in the expression. e.g. the sum term $(A+\bar{B}+C)$, a '0' goes in the 010 cell on a three variable map.

Steps

1. Determine the binary value of each sum term in the standard POS expression.
2. As each sum term is evaluated, place a '0' on the K-map in corresponding cell.

Ex:-1) Map $(A+B+C)$ $(A+\bar{B}+C)$ $(\bar{A}+\bar{B}+C)$ $(\bar{A}+B+\bar{C})$



Binary Values

$A+B+C \rightarrow 000$

$A+\bar{B}+C \rightarrow 010$

$\bar{A}+\bar{B}+C \rightarrow 110$

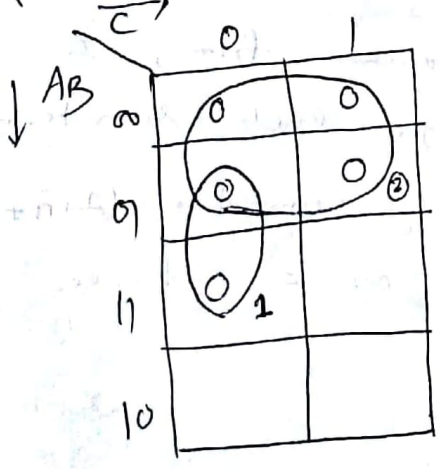
$\bar{A}+B+\bar{C} \rightarrow 101$

Ex 2) Use K-Map to minimize the following standard POS

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

Ans :- The Combinations of binary values of the expression are

$(0+0+0)$ $(0+0+1)$ $(0+1+0)$ $(0+1+1)$ $(1+1+0)$



① → $(\bar{B}+C)$

② → A

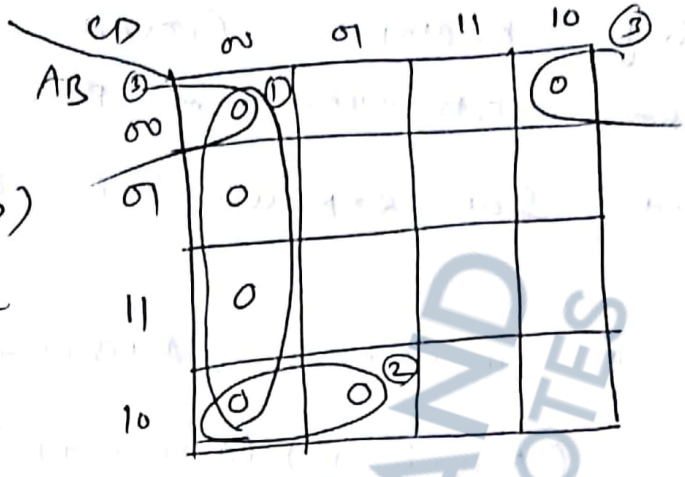
POS → $A(\bar{B}+C)$

8) Minimize POS

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

0 0 1 0 1 0 0 1 0 1 0 0 1 1 0 0

Ans:-



Other terms

0 0 1 0
1 0 0 1
0 1 0 0
1 1 0 0

First Convert $(B+C+D)$ to standard POS form.

A B C D

0 → 0 0 0 0 → $A\ B\ C\ D$

1 → 1 0 0 0 → $\bar{A}\ B\ C\ D$

Mapping 6 terms in the K map, we have.

Grouping the 0's in the cells, we have.

① → $(C+D)$

② → $(\bar{A}+B+C)$

③ → $(A+B+D)$

∴ POS minimized → $(C+D)\ (A+B+D)\ (\bar{A}+B+C)$

Converting POS → SOP, SOP → POS Using K-Map.

For a POS expression, all the cells that don't contain 0s contain 1s, from which the SOP expression is derived.

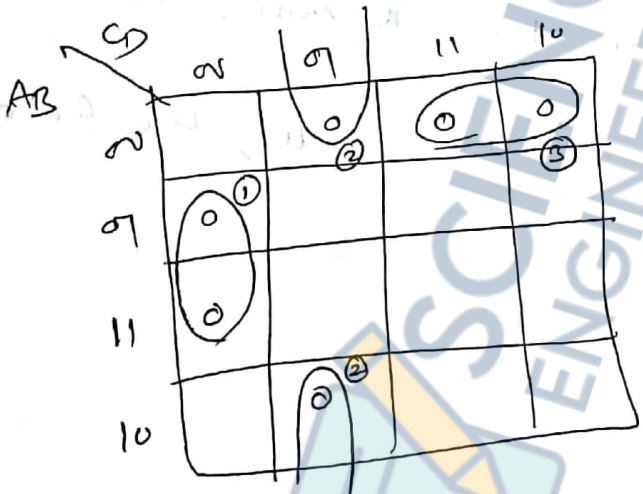
Likewise, for an SOP expression, all the cells that don't contain 1s contain 0s, from which the POS expression is derived.

Ex:- 4) Using K-Map convert the following POS expression into (i) minimum POS expression, (ii) Standard SOP expression (iii) minimum SOP expression.

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

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

Ans :- First Map the POS into the K-Map.



Map

1	1	0	1
0	1	0	0
0	0	0	1
0	0	1	1
1	0	0	1
0	0	1	0

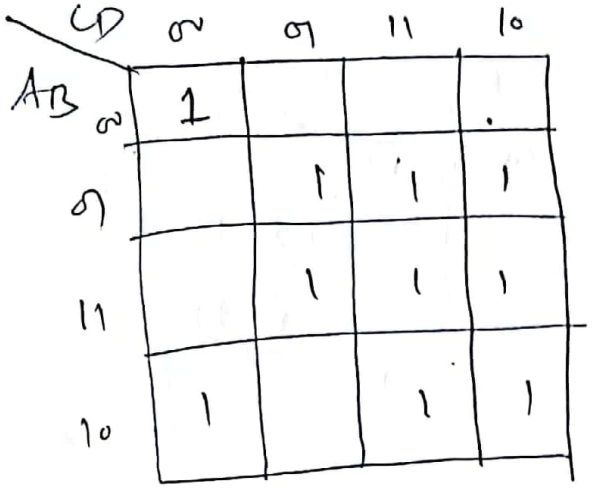
① → $(\bar{B}+C+D)$

② → $(B+\bar{C}+\bar{D})$

③ → $(A+B+\bar{C})$

Minimized POS → $(A+B+\bar{C})(\bar{B}+C+D)(B+C+\bar{D})$

(ii) To get the Standard SOP, Place the 1s in the cells that don't contain 0s.



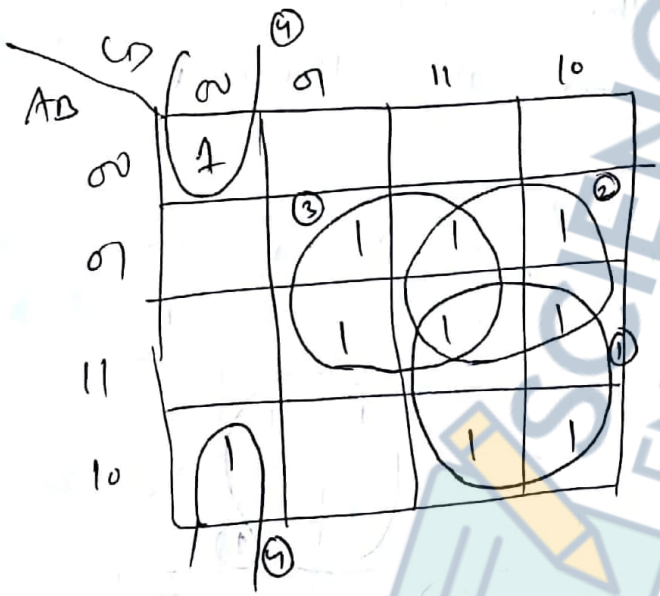
Standard SOP

$\bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + \bar{A}BC\bar{D}$

$+ AB\bar{C}D + ABCD + ABC\bar{D}$

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

(iii) Minimum SOP



① → AC

② → BC

③ → BD

④ → $\bar{B}\bar{C}\bar{D}$

SOP → $AC + BC + BD + \bar{B}\bar{C}\bar{D}$

Ans:

Note:-

1) Pair → A group of two logical 1's in K-map.
It eliminates one variable in the o/p expression.

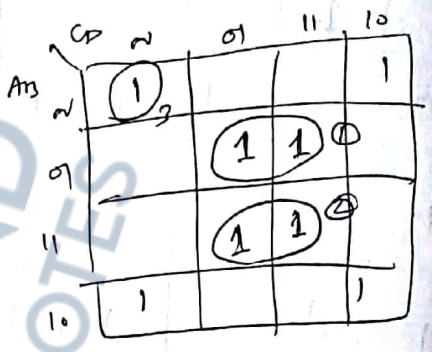
Quad → A group of 4 logical 1's in K-map.
It eliminates 2 variables in o/p expression.

~~Octet~~ Octet → group of '8', eliminates 3 variables.

Implicant

A group of 1

An Implicant is a product term obtained by combining 1 or 2 or 4 or 8 adjacent squares in the map.

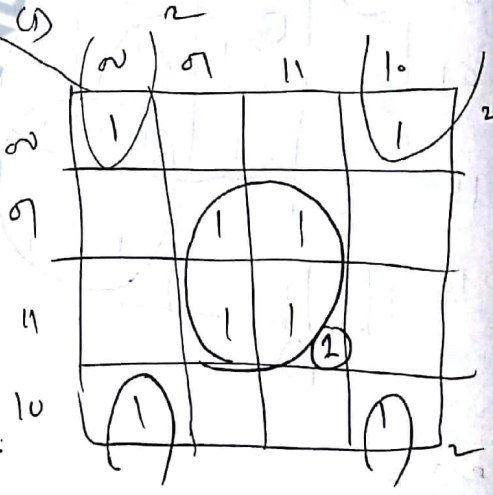


Ex:-

① & ②, ③ are called Implicants.

Prime implicant:-

A prime implicant is a product term obtained by combining the maximum possible number of adjacent squares in the map.



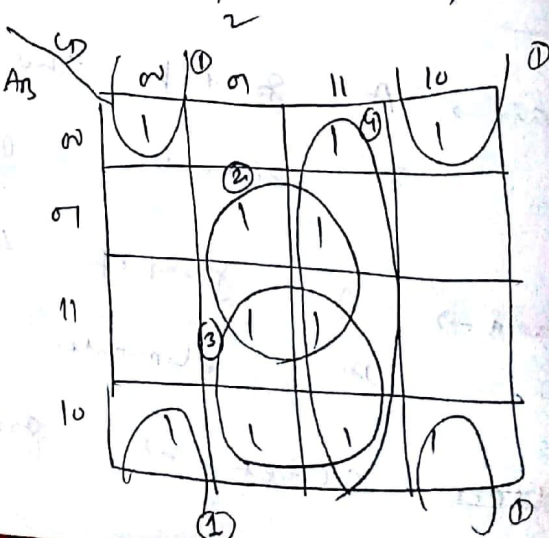
① → $\bar{B}D$

② → $\bar{B}\bar{D}$

are called the prime implicants.

Ex:-5

→



① → Combining the corners

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

② → BD

③ → AD

④ → CD

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

Taking different groups, we have

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

$= BD + \bar{B}\bar{D} + \bar{B}C + AD$

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

Essential Prime Implicants

The prime implicant which contains at least one '1' which can't be covered by any other prime implicants is called an essential prime implicant (EPI).

The prime implicant whose each 1 is covered by at least one EPI is called redundant prime implicant (RPI).

A prime implicant which is neither an (EPI) or (RPI) is called a selective prime implicant. (SPI).

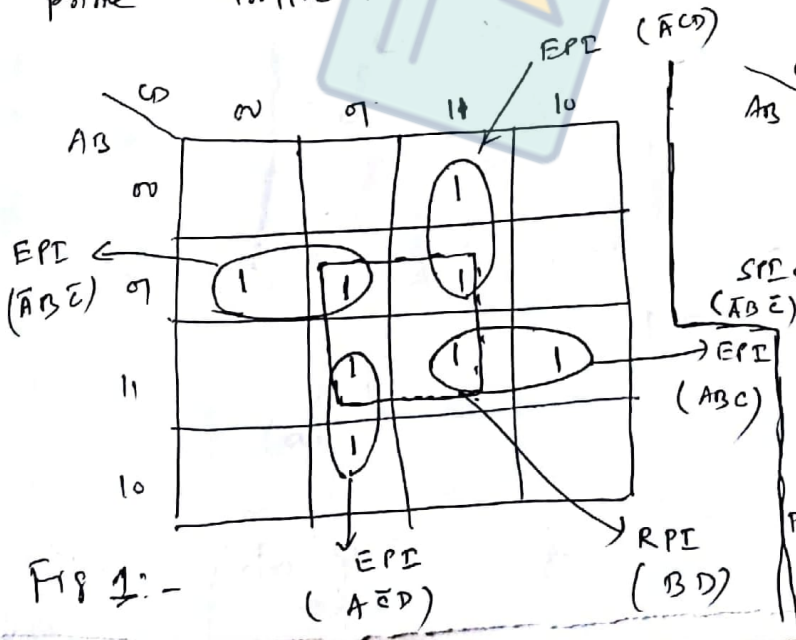


Fig 1:-

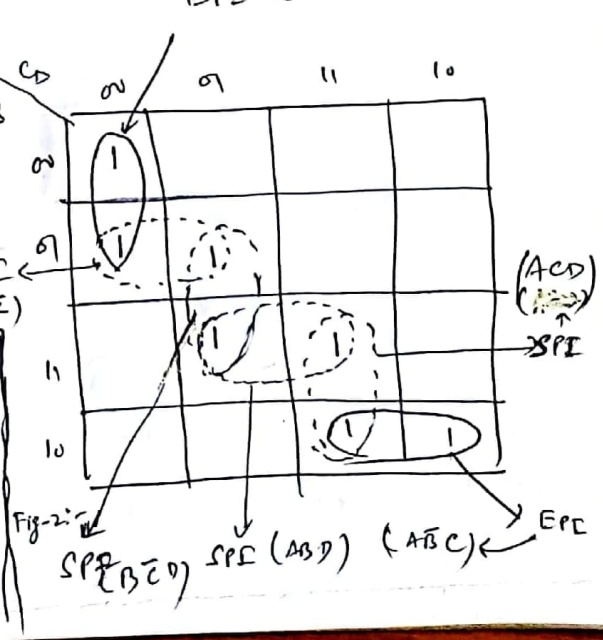


Fig-2:-

In figure 2 :- EPI → $\bar{A}C\bar{D}$, $A\bar{B}C$

SPI → $\bar{A}BC$, $B\bar{C}D$, $AB\bar{D}$, ACD

Redundant group. (More details)

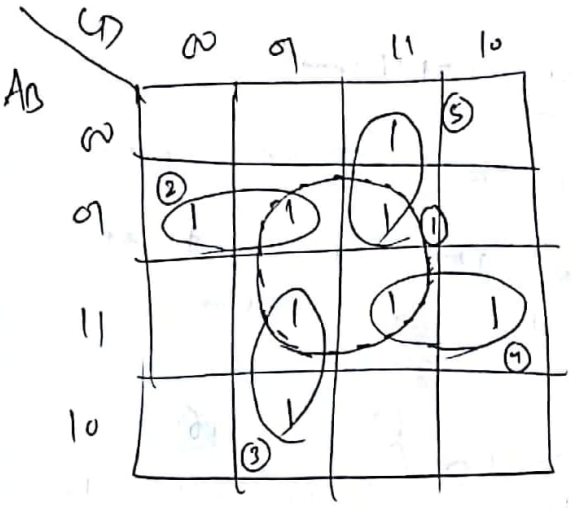
Options

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

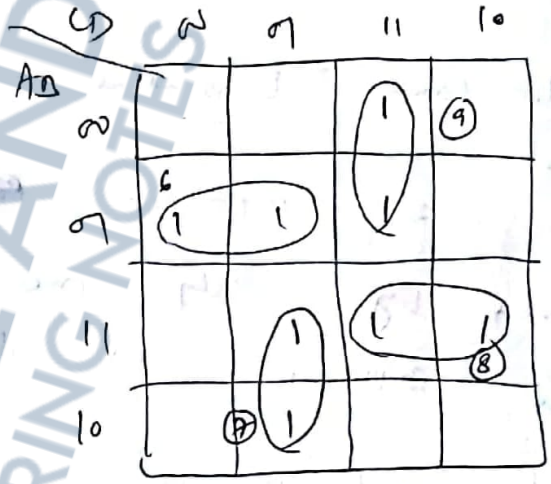
or $\bar{A}C\bar{D} + A\bar{B}C + B\bar{C}D + AB\bar{D}$

or $\bar{A}C\bar{D} + A\bar{B}C + B\bar{C}D + ACD$

(EPI) (SPIs)



Case – I



Case – II

In Case I, Suppose group 1 is taken, then group ②, ③, ④, ⑤ are taken. So we have 5 terms in the SOP expression.

But in Case II, group ⑥, ⑦, ⑧, ⑨ Cover all the 1s. So no need of quad.

→ So group ①, is a redundant group and it is unnecessary.

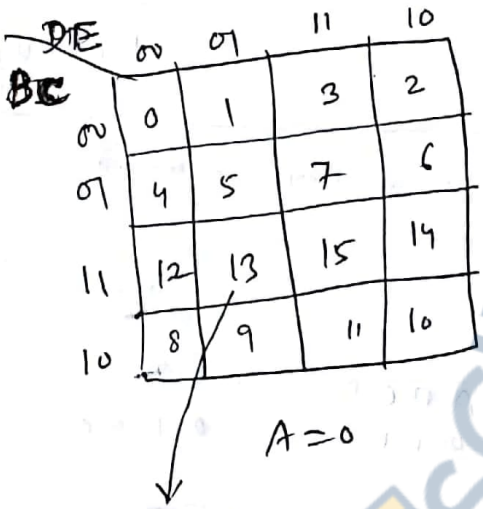
Note :- 1) In Selective prime implicant Case we have options, either of the case

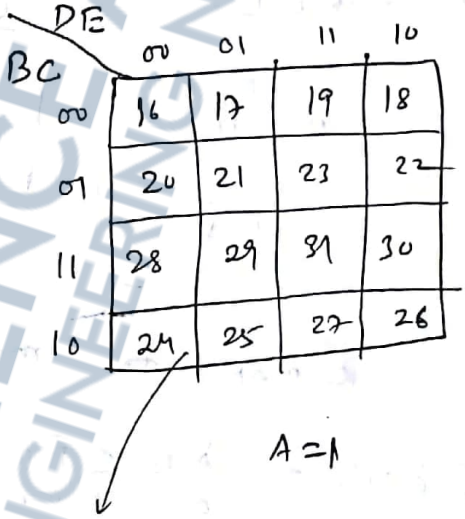
We can take.

But in redundant group case we don't have any option. We have to eliminate the redundant group.

Five Variable Map

Two 4-variable maps (16 cells each) are used to construct a 5-variable Map.



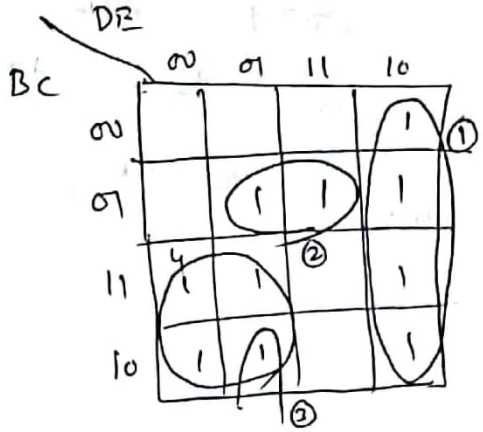


Because A B C D E
0 11 01 → 13

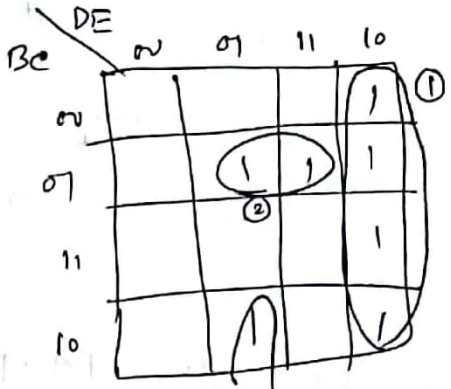
Because A B C D E
1 1 0 0 0 → 24

The Best way to visualize cell adjacencies between the two 16-cell maps is to imagine that A=0 map is placed on the top of the A=1 map. Each cell in the A=0 map is adjacent to the cell directly below it in the A=1 map.

ex:-1)



A=0



A=1

① → ~~DE~~ DĒ

② → B̄ C E

③ → B C̄ D̄ E

④ → Ā B D̄

The total SOP expression

$$SOP = \cancel{B}\ D\bar{E} + \bar{B}CE + \bar{A}B\bar{D} + B\bar{C}\bar{D}E$$

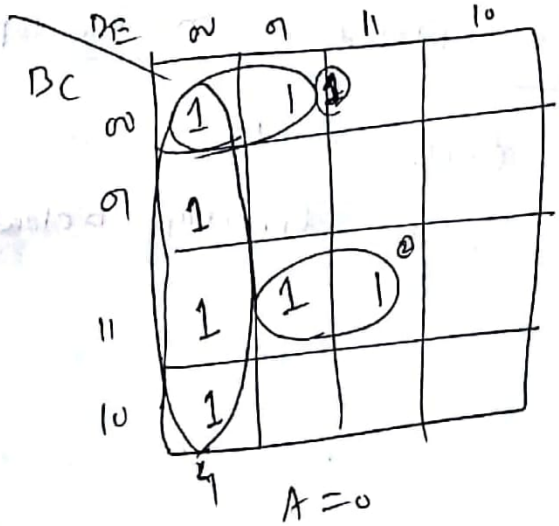
★ Don't Cwc example in extra note

2) Use K-Map to minimize

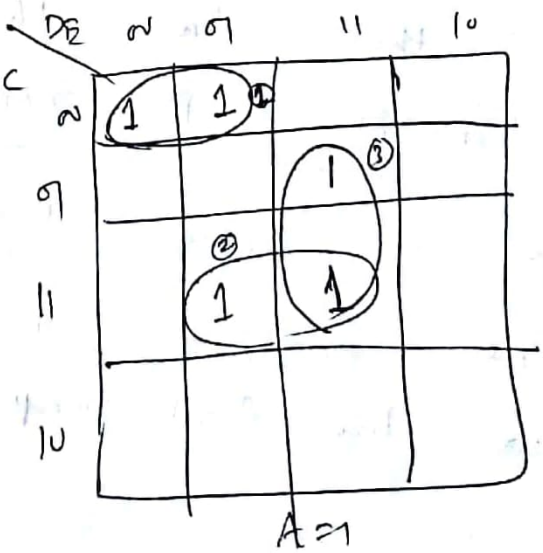
$X = \bar{A}\bar{B}\bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}C\bar{D}\bar{E} + \bar{A}BC\bar{D}\bar{E} + \bar{A}B\bar{C}\bar{D}\bar{E}$
$00000 \quad 00100 \quad 01100 \quad 01000$

$+ \bar{A}\bar{B}\bar{C}\bar{D}E + \bar{A}BCDE + \bar{A}BC\bar{D}E + A\bar{B}\bar{C}\bar{D}\bar{E}$
$00001 \quad 01101 \quad 01111 \quad 10000$

$+ A\bar{B}\bar{C}\bar{D}E + ABC\bar{D}E + ABCDE + A\bar{B}CDE$
$10001 \quad 11101 \quad 11111 \quad 10111$



A=0



A=1

① → $\bar{B}\,\bar{C}\,\bar{D}$

② → ~~A B~~ BCE

③ → $ACDE$

④ → $\bar{A}\,\bar{D}\,\bar{E}$

∴ SOP = $\bar{A}\bar{D}\bar{E} + \bar{B}\bar{C}\bar{D} + BCE + ACDE$

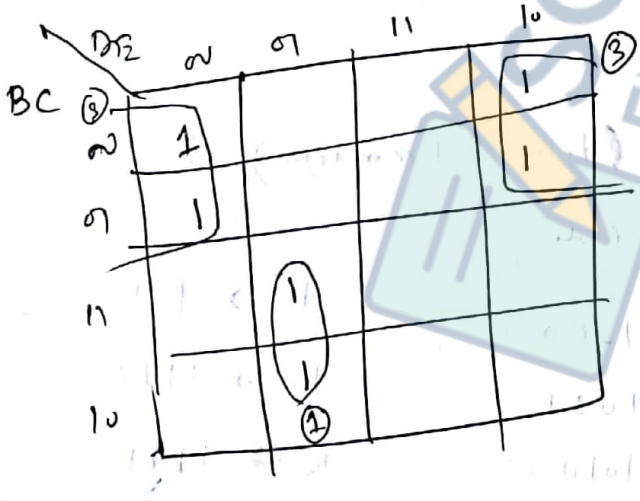
Simplify

$F(A,B,C,D,E) = \Sigma(0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)$

∴ 00000, 00010, 00100, 00110
01001, ~~11001~~ 01101

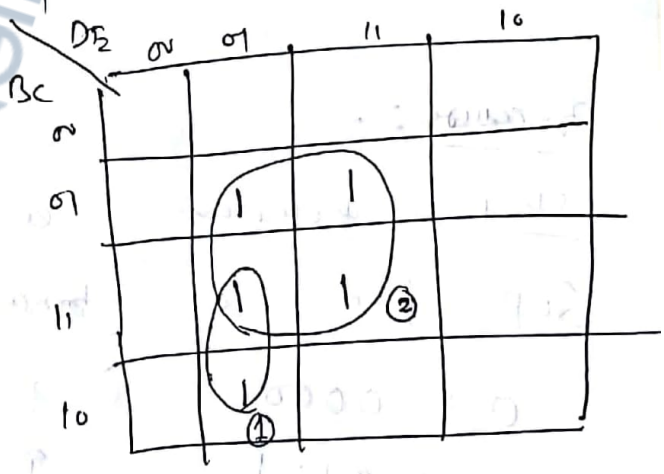
10101, 10111, 11001,
11101, 11111

A = 0



A = 0

A = 1



A = 1

① → $B\bar{D}E$

② → ACE

③ → $\bar{A}\,\bar{B}\,\bar{E}$

SOP → F = $\bar{A}\bar{B}\bar{E} + B\bar{D}E + ACE$

(Ans)

Digital ckts. & Design

Digital ckts

- Combinational logic ckts
- Sequential logic ckts.

→ A Combinational ckt consists of logic gates whose o/p at any time are determined from present combination of i/ps.

→ A Combinational ckts performs an operation that can be specified logically by a set of Boolean functions.

→ Sequential ckts employ storage elements in addition to logic gates. Their o/ps are a function of previous i/ps. As a consequence, the o/ps of a sequential ckts depend not only on present values of i/ps, but also past i/ps and the ckt. behavior must be specified by a time sequence of i/ps and internal states.

→ Comparison of Combinational & sequential logic ckts.

Combinational Logic ckt

1) Its o/p depend on the present values of the i/p only.

2) It does not have memory element.

3) It does not have feedback path from o/p to i/p

4) It does not have a clock signal.

5) A combinational ckt consists of logic gates.

6) Its action does not depend on clock transition.

7) Its ckt is simpler than the ~~Combinational~~ Sequential logic ckts.

8) Ex → Adder, Subtractor, MUX, Demux, Decoder, Encoder etc.

Sequential Logic ckt

1) Its o/p not only depend on the present values of i/ps, but also depends on past i/ps.

2) It has memory element.

3) It has feedback path from o/p to i/p.

4) It has a clock signal

5) A sequential logic gate consists of combinational logic ckts and memory elements.

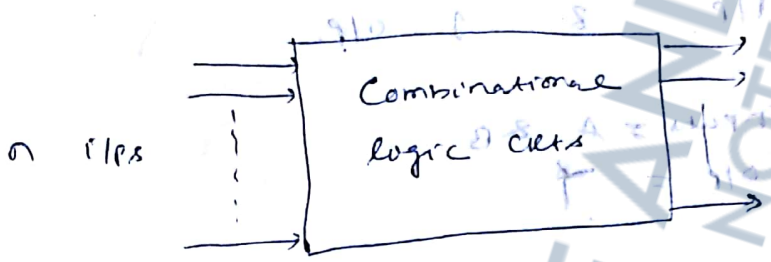
6) Its action depend on clock transition.

7) Its ckt is more complex than that of combinational logic ckts.

8) Ex → Flip-flops, Counters, Shift-registers.

Combinational Logic Ckt :-

A Combinational Ckt. Consists of i/p variables, logic gates and o/p variables. The logic gates accept signal from the i/ps and generate signal to the outputs.



The n i/p binary variables come from an external source. The m o/p variables go to external destination.

Design procedures -

1) From the specification of the ckt, determine the required number of i/ps and o/ps.
2) Assign i/p & o/p variables.
3) Derive truth table that defines the required relationship between i/ps and o/ps.
4) Obtain the simplified expression for o/p as function of i/p variables.
5) Draw the logic diagram & verify the correctness of the design.

Ex :- 1) Design a 2 i/p Digital ckt whose o/p is 1 if all i/ps are same. The o/p is 0 otherwise. Design the digital ckts.

(a) Using Basic gates b) Using NAND gates.

Ans :-

Step 1 :- 2 i/p & 1 o/p.

Step 2 :-

Inputs = A & B

o/p = Y

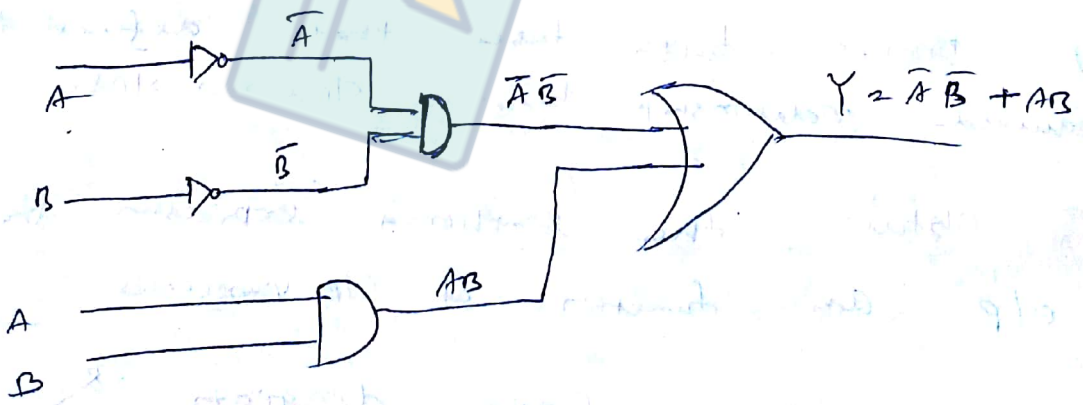
Step-3

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

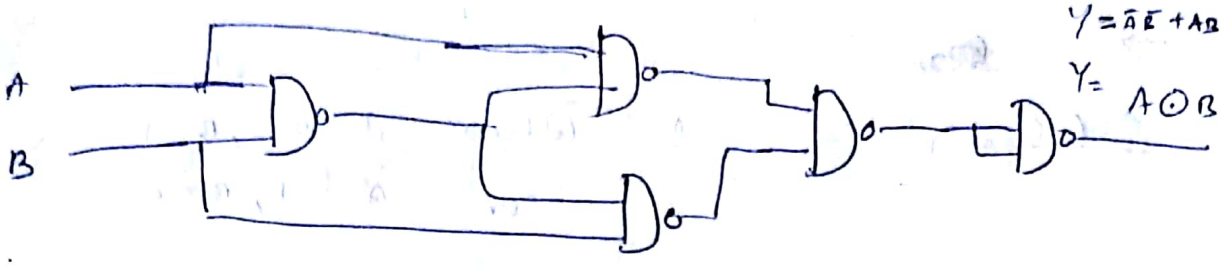
Step-4

$Y = \bar{A}\bar{B} + AB$ (X-NOR)

Step-5 :-



(a) using Basic gates.



Basic Adders :-

→ The most basic arithmetic operation is addition of 2 binary digits.

$0+0=0$
$0+1=1$
$1+0=1$
$1+1=10$ (carry)

```
  1
+ 1
-----
  1 0
```

→ The first three operation produces a sum of one digit and 4th one produces a carry.

→ A combinational ckt. that performs the addition of 2 bit is called half adder.

→ One that performs addition of 3 bit is called full adder.

Half Adder :-

→ Half-adder is a combinational logic ckt, which perform 2 bits addition.

→ The result produces a sum as well as carry.

→ The simplified expression can be directly obtained from the truth table.

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

<Truth table for half adder>

→

S (Sum) is 1 when A = 0, B = 1
or A = 1, B = 0

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

C (Carry) is 1, when A = 1, B = 1

$\therefore C = AB$

Block diagram:-

fig (a)

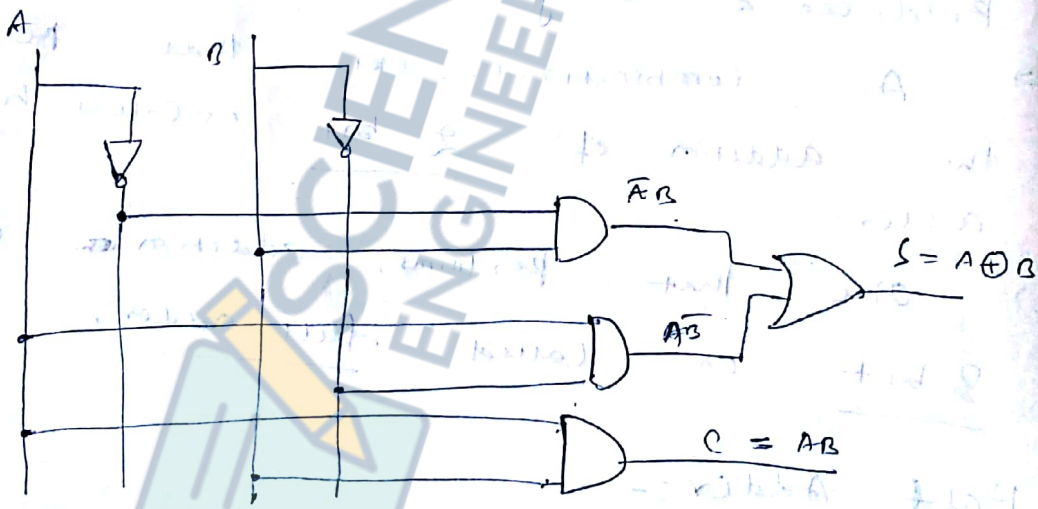
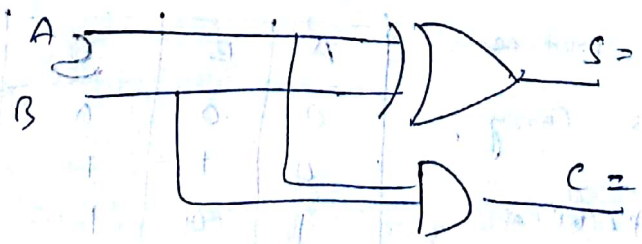


fig (b) Using Basic logic gates.



(c) Using EX-OR and AND gate

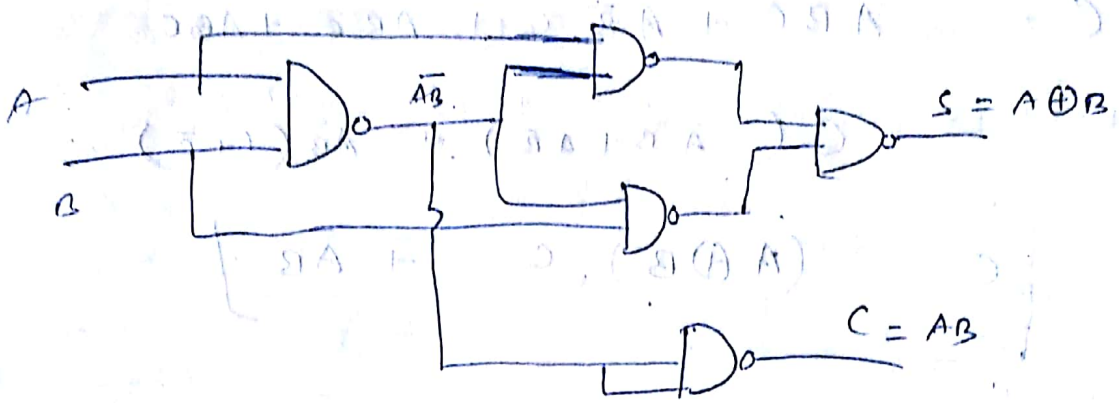


fig (1) :- Using NAND gates only.

Full Adder :- <Start - Previous theory of half adder>

Full adder is combinational logic ckt which perform 3 bit binary addition, which produces a sum as well as carry.

A	B	C	S (Sum)	C (Carry)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

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

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

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

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

$$= A \oplus Y$$

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

$B \oplus C = Y$

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

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

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

$$\boxed{C = (A \oplus B).C + AB}$$

(or)

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

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

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

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

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

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

$$= B(A+C) + A\bar{B}C \qquad \left| \because A + BC = (A+B)(A+C) \right.$$

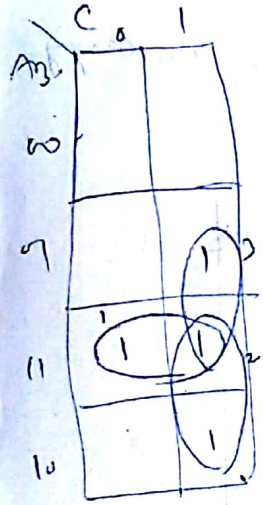
$$= AB + BC + A\bar{B}C$$

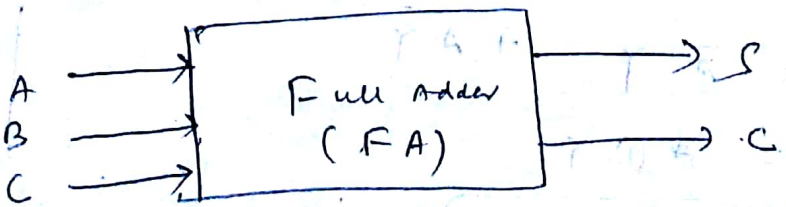
$$= AB + BC(B + A\bar{B})$$

$$= AB + BC((B+A)(B+\bar{B})) \qquad \left| \because A + BC = (A+B)(A+C) \right.$$

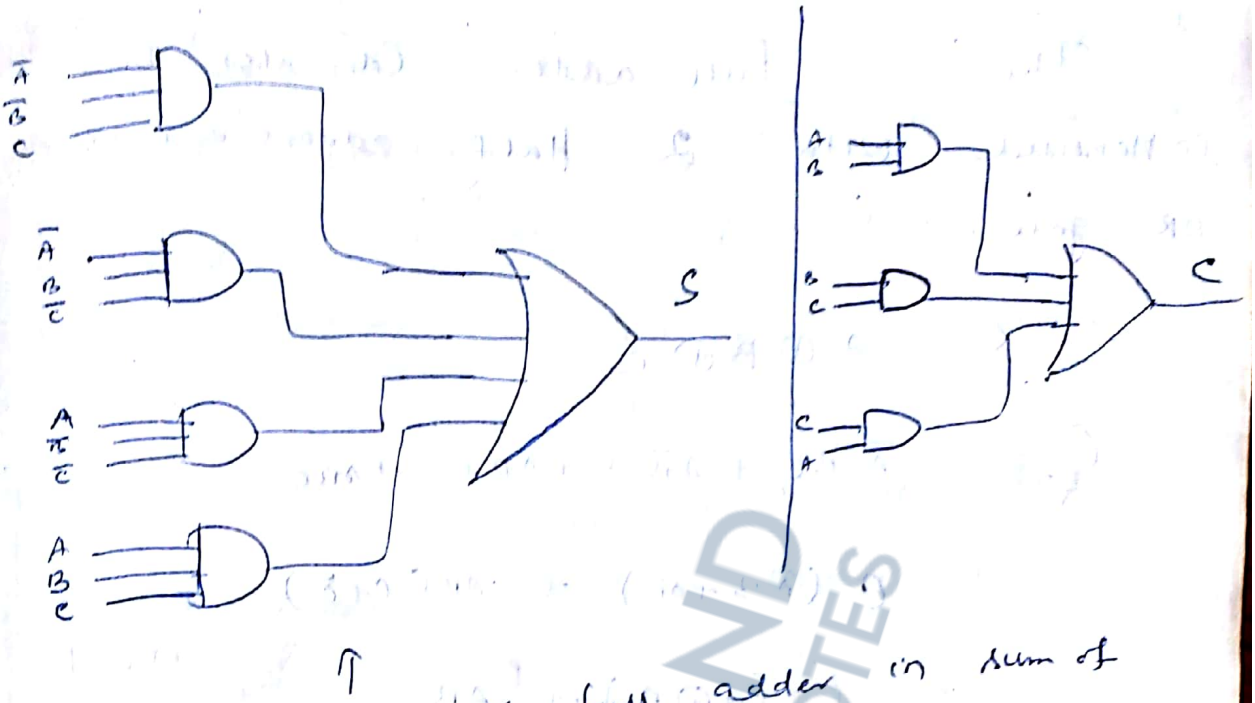
$$= AB + C(A+B)$$

$$\boxed{C = AB + BC + AC}$$

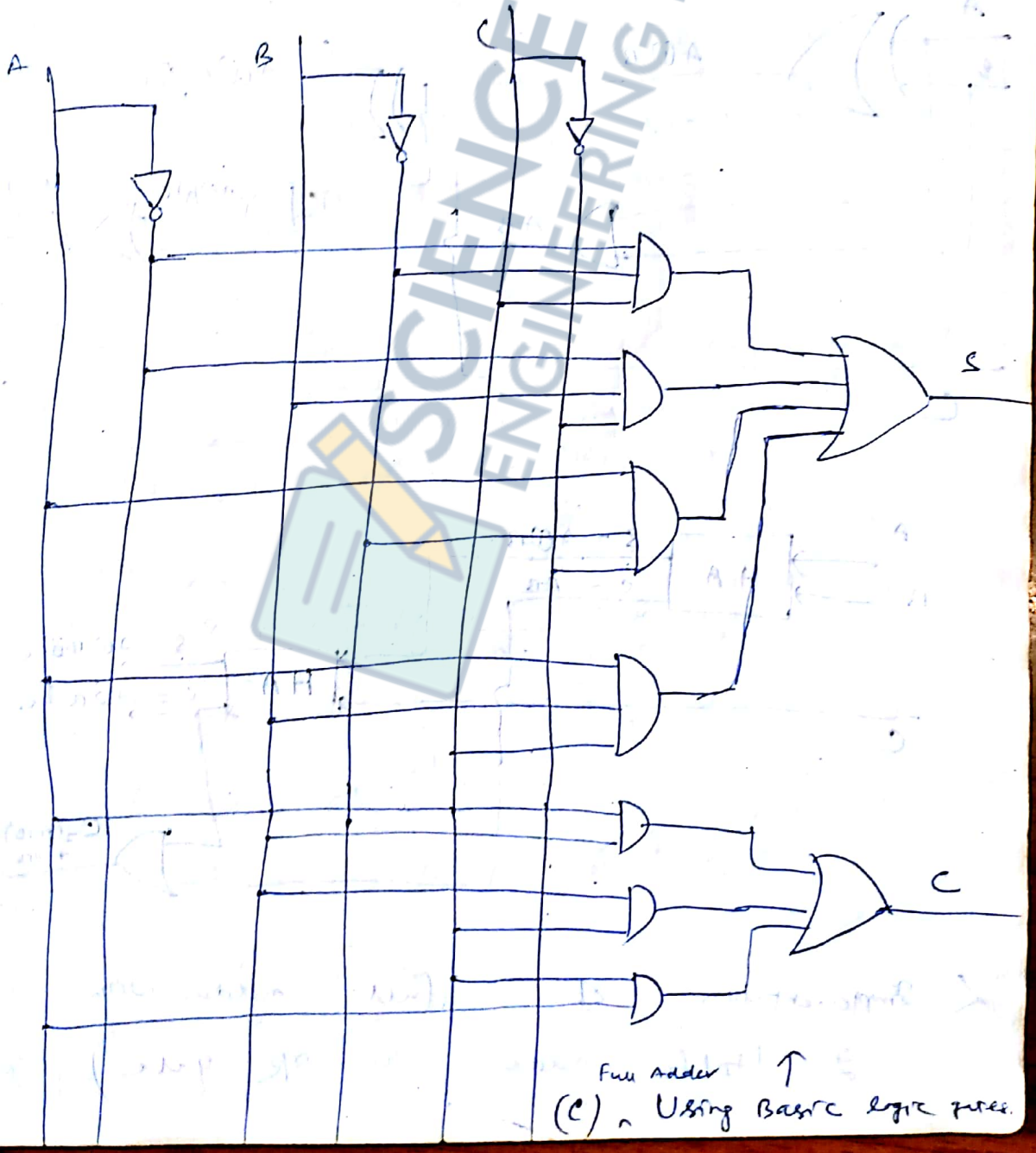




(a) Block diagram.



(b) Implementation of full adder in sum of product (SOP)



(c). Using Basic logic gates.

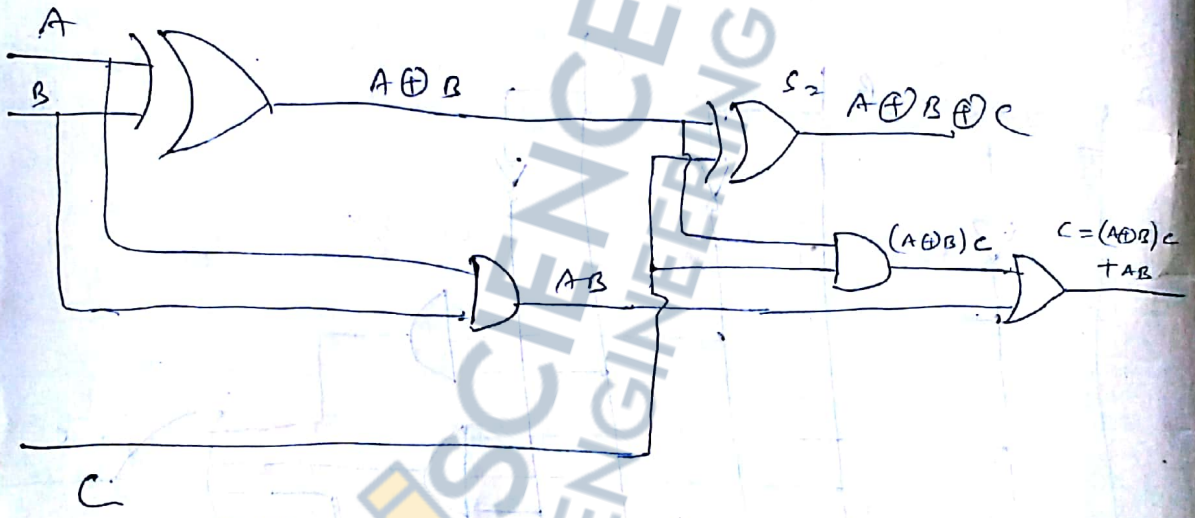
The full adder can also be implemented with 2 Half adders and one OR gate.

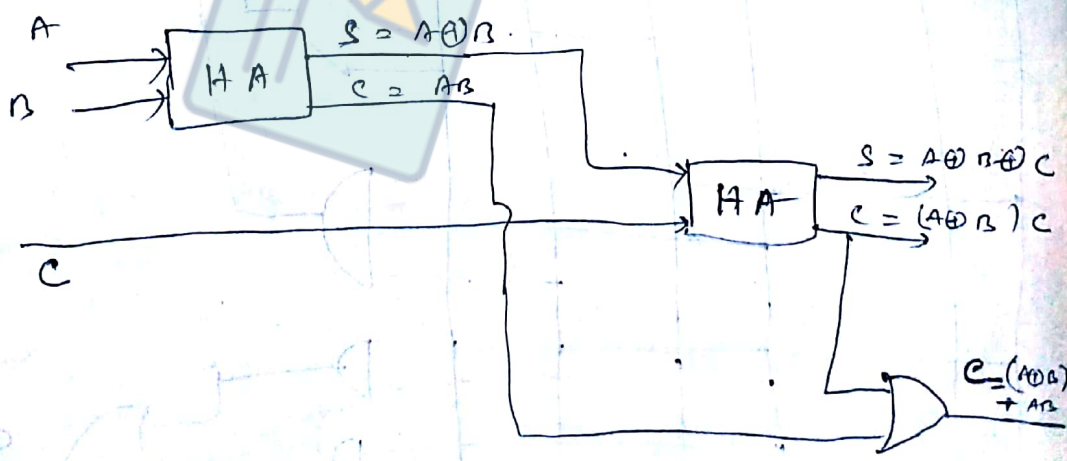
$$S = A \oplus B \oplus C$$

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

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

$$= C(A \oplus B) + AB$$





∠ Implementation of full adder with 2 Half adder an OR gate)

Basic Substractor :-

→ The simple substraction consists of 4 possible elementary operations :-

$0-0=0$, $0-1=1$, Borrow 1.

$1-0=1$, $1-1=0$

→ The Combinational ckt that performs the ~~subtraction of one digit and the difference~~ substraction of 2 bit is called half substractor.

→ one that performs the substraction of 3 bits is called full substractor.

→ 2 half substractor can be used to implement a full substractor.

Half Substractor :-

· Half substractor is a combinational ckt, which performs a substraction, produces a difference of 2 bits as well as borrow.

Truth Table :-

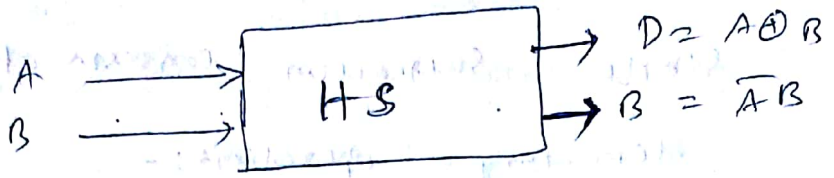
A	B	D (Difference)	B (Borrow)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

100
-011

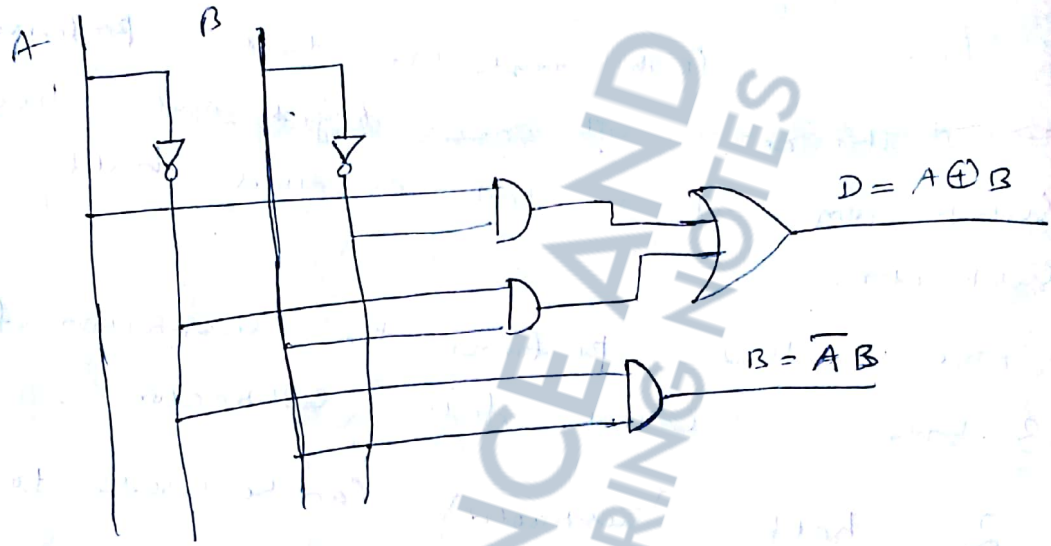
$\therefore D = \bar{A}B + A\bar{B} = A \oplus B$

$B = \bar{A}B$

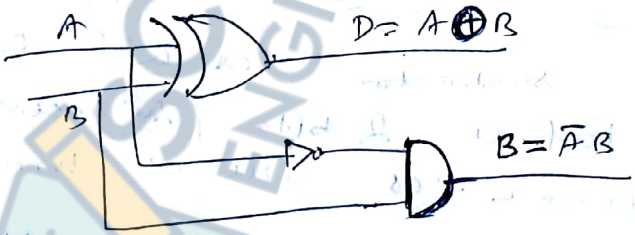
(Borrow)

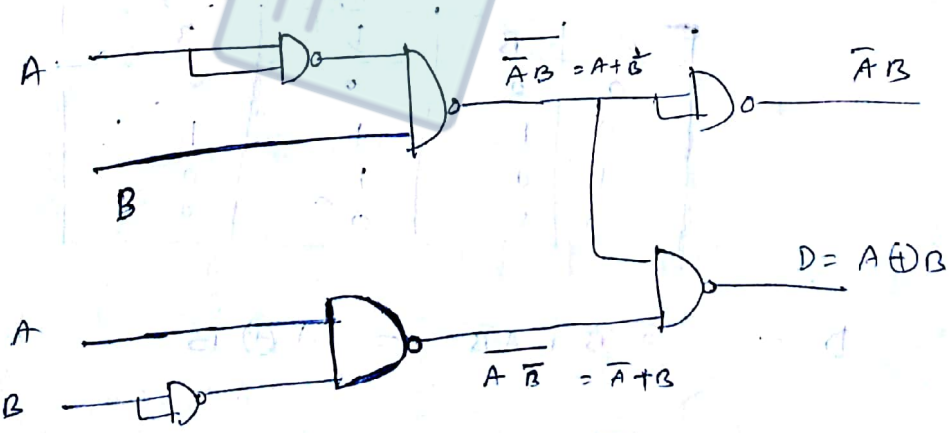


(a) Block diagram.



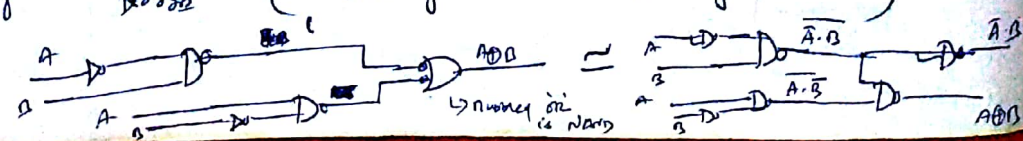
(b) Using basic logic gates





Hints: - using Bubbled or

(Using NAND gate only)



Proof:- $D = \overline{(A+\bar{B}).(\bar{A}+B)}$

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

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

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

$= A \oplus B$ (Proved)

Full Substractor :-

Full Substractor is a Combinational logic ckt which perform 3 bit binary substraction, which produces a difference as well as a borrow.

Truth Table:-

A	B	C	D (Difference)	B (Borrow)
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

From the truth table :-

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

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

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

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

$= A \oplus Y = A \oplus B \oplus C$

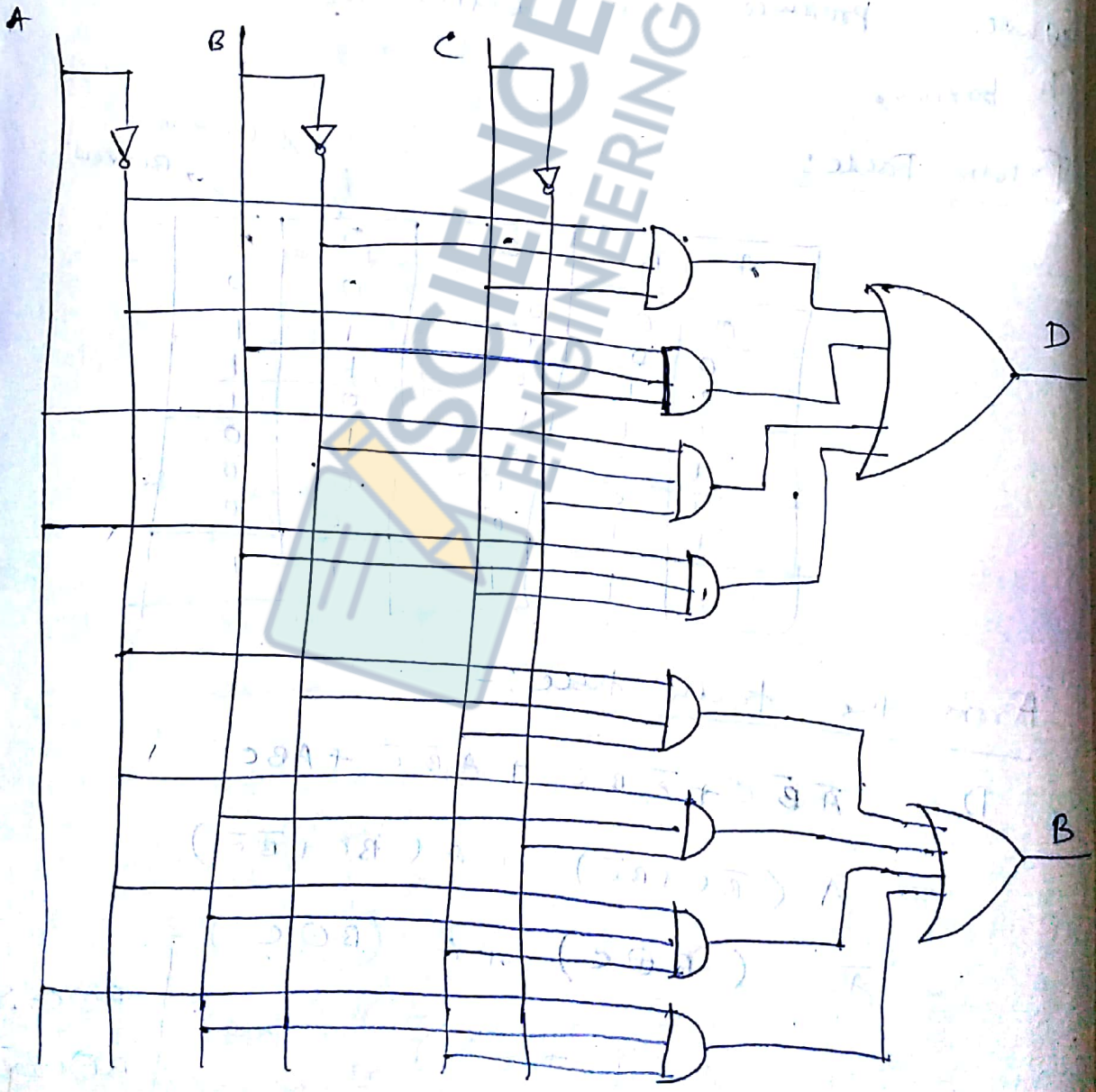
$B \oplus C = Y$

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

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

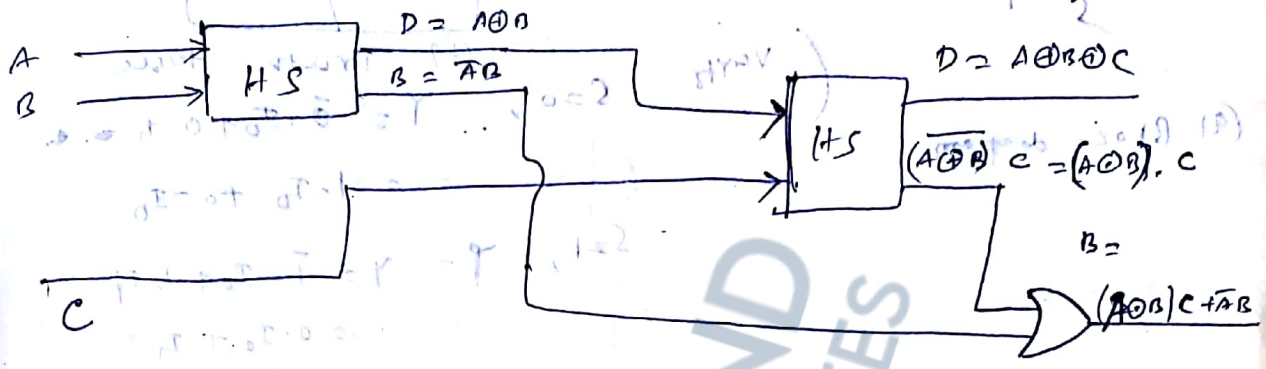
$$= C(A \odot B) + \bar{A}B$$

The Block diagram for a full subtractor is shown



(Full Subtractor Using Basic logic gates)

Full Subtractor can also be realized with 2 half Subtractor and one OR gate.



2nd

Multiplexers :-

→ Multiplexer (MUX) is a combinational logic ckt. that receives information from many i/p s and directs this information to the o/p. → Ex: Enable IV a Remote

→ The i/p information that is selected is controlled by a selector lines.

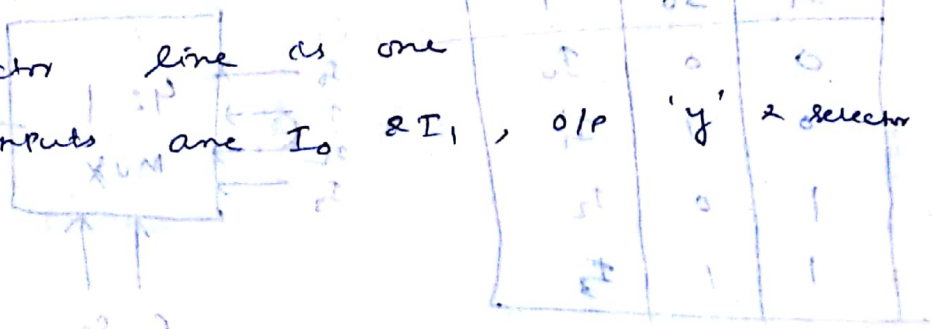
→ A MUX with 2^n i/ps requires n selector lines.

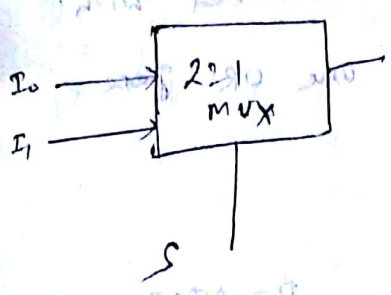
→ Mux is also sometimes known as many to one or it as known as data selector.

2 : 1 MUX

→ It has 2 i/ps and one o/p, so no. of selector line is one

→ Inputs are I_0 & I_1, o/p 'y' & selector line 's'.





(a) Block diagram

$Y = \bar{S} I_0 + S I_1$

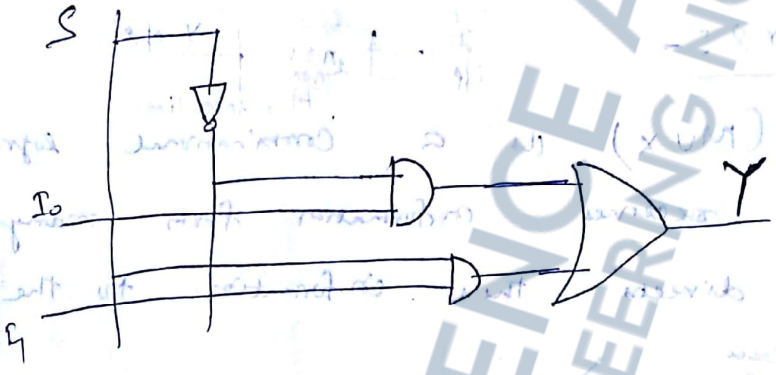
S	Y
0	I_0
1	I_1

(b) Truth table

Verify

$S=0, \quad Y = \bar{0} \cdot I_0 + 0 \cdot I_1$
$= 1 \cdot I_0 + 0 = I_0$

$S=1, \quad Y = \bar{1} \cdot I_0 + 1 \cdot I_1$
$= 0 \cdot I_0 + I_1$
$= I_1$



(c) Logic ckt.

4:1 MUX

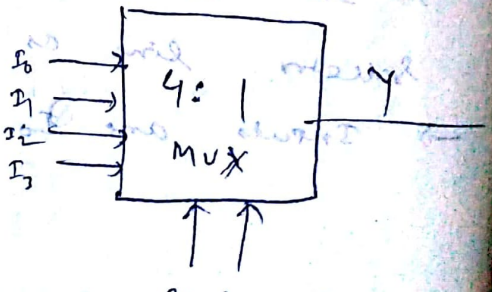
→ It has 4 I/Ps and one O/P. So number of selector line (S) is 2. ($\because 4 = 2^2$)

→ Inputs are I_0, I_1, I_2, I_3 and Selector lines S_1, S_0.

→

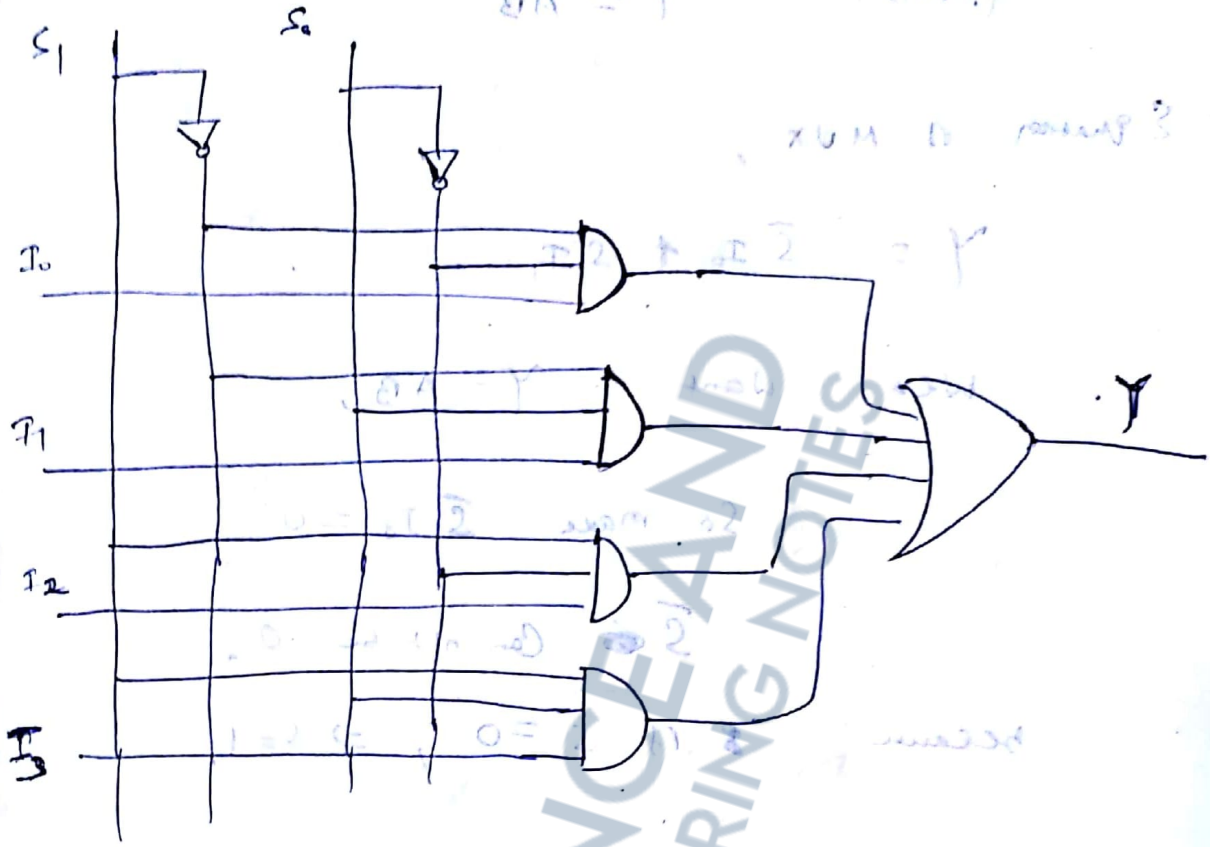
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(a) Truth table



(b) Block diagram

$$Y = \bar{S_1}\,\bar{S_0}\,I_0 + \bar{S_1}\,S_0\,I_1 + S_1\,\bar{S_0}\,I_2 + S_1\,S_0\,I_3$$



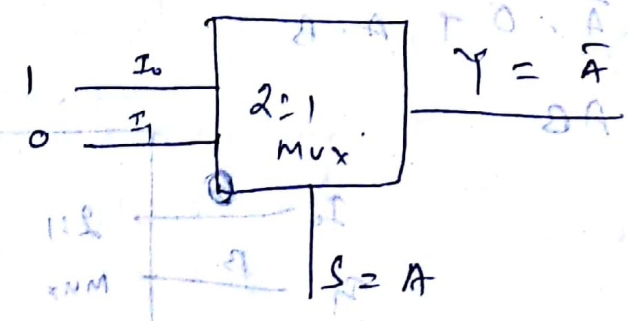
Ex-1 :- Implement Inverter $Y = \bar{A}$ using MUX

Ans :- $Y = \bar{S}\,I_0 + S\,I_1$

If $S = A$, $I_1 = 0$, $I_0 = 1$

$$Y = \bar{A}.1 + A.0$$

$$Y = \bar{A}$$



Ex-2 Implement 2 i/p AND gate using MUX

Ans :

(AND) $Y = AB$

Equation of MUX,

$$Y = \bar{S} I_0 + S I_1$$

We want $Y = AB$,

So make $\bar{S} I_0 = 0$

$\bar{S}$ can not be 0.

because if $\bar{S} = 0$, $\Rightarrow S = 1$

$$S I_1 = 1 \cdot I_1 = I_1$$

But we want $S I_1 = AB$.

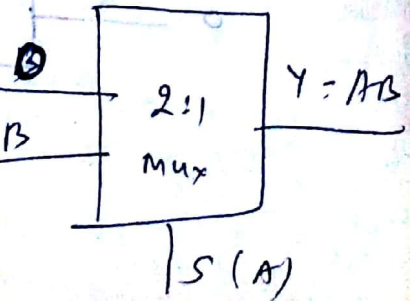
$\therefore$ Since $\bar{S} \neq 0$, $I_0 = 0$ (because we want $\bar{S} I_0 = 0$)

Lets take $\begin{cases} S = A \\ I_1 = B \\ \bar{S} = \bar{A} \\ I_0 = 0 \end{cases}$

$$Y = \bar{S} I_0 + S I_1$$

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

$$Y = AB$$



Or

$S = B$

$I_1 = A$

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

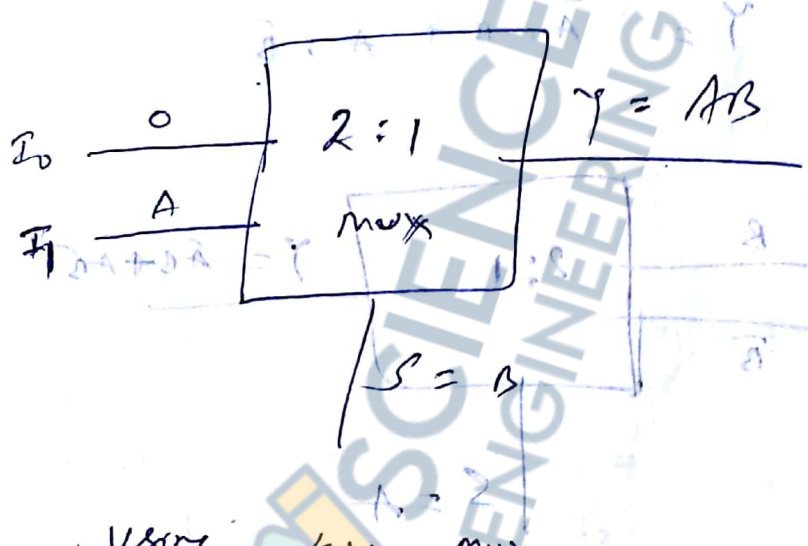
$I_0 = 0$

$Y = \bar{S} I_0 + S I_1$

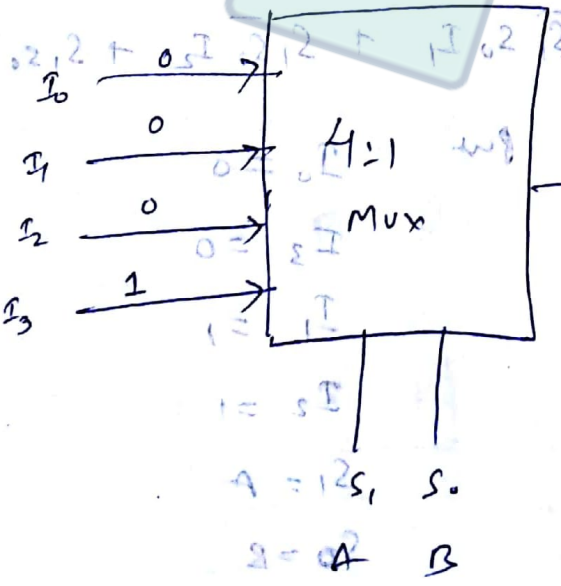
$= \bar{B} \times 0 + B \cdot A$

$= BA$

$= AB$



Using 4:1 mux



$Y = \bar{S_1} \bar{S_0} I_0 + \bar{S_1} S_0 I_1 + S_1 \bar{S_0} I_2 + S_1 S_0 I_3$

$Y = \bar{A} \cdot \bar{B} \cdot 0 + \bar{A} \cdot B \cdot 0 + A \cdot \bar{B} \cdot 0 + AB \cdot 1$

$= AB$ (AND gate)

Ex-3 ①

Implement 2 I/P XOR gate using MUX.

Ans :

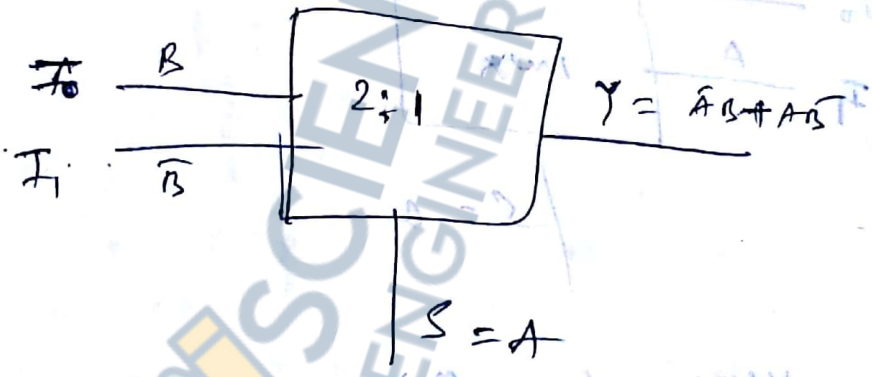
$$Y = \bar{A}B + A\bar{B}$$

Boolean Eqn

$$Y = \bar{S} I_0 + S I_1$$

Let $S = A$, $I_0 = B$, $I_1 = \bar{B}$

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



Using 4:1 MUX

$$Y = S_1 \bar{S_0} I_0 + \bar{S_1} S_0 I_1 + S_1 \bar{S_0} I_2 + S_1 S_0 I_3$$

If we put $I_0 = 0$

$I_3 = 0$

$I_1 = 1$

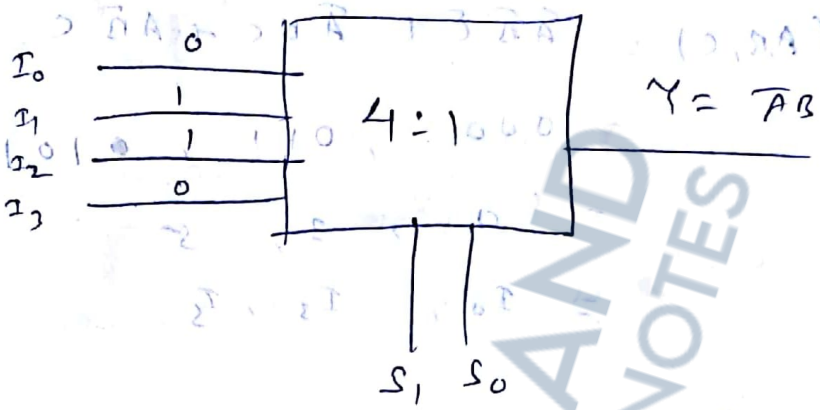
$I_2 = 1$

$S_1 = A$

$S_0 = B$

$$Y = 0 + \bar{A}B \cdot 1 + A\bar{B} \cdot 1 + 0$$

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



XNOR : Similar $Y = \bar{A}\bar{B} + AB$

$$Y = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$$

If we put, $I_1 = 0$

$I_2 = 0$

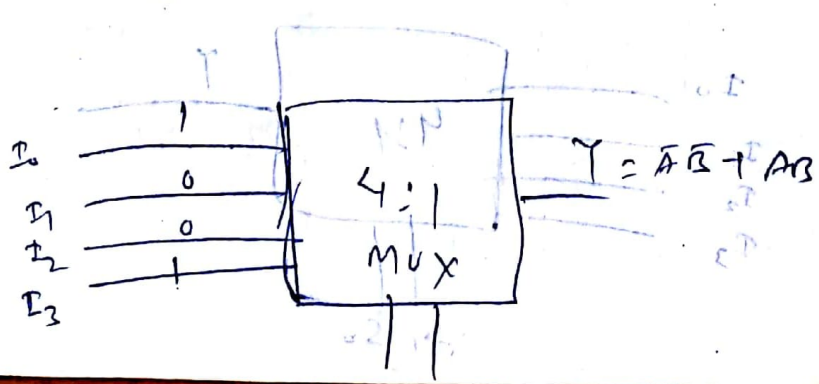
$I_0 = 1$

$I_3 = 1$

$S_1 = A$

$S_0 = B$

$$Y = \bar{A}\bar{B} \cdot 1 + 0 + 0 + AB \cdot 1$$



Ex :- 4 Implement

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

using 8:1 MUX

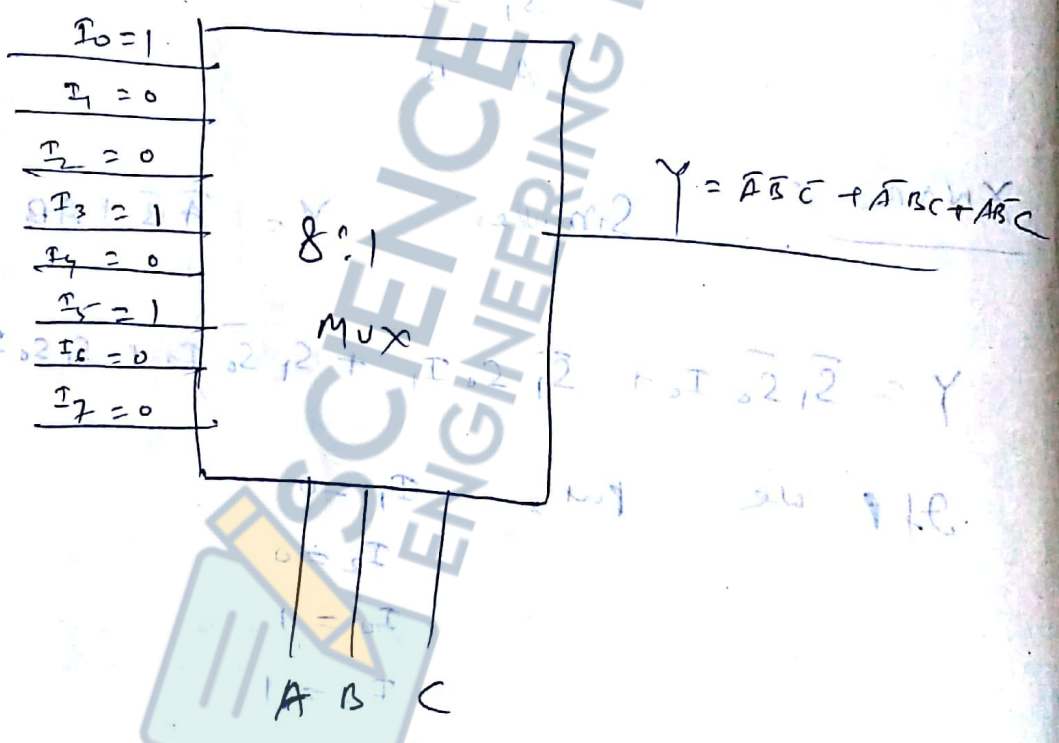
Ans :

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

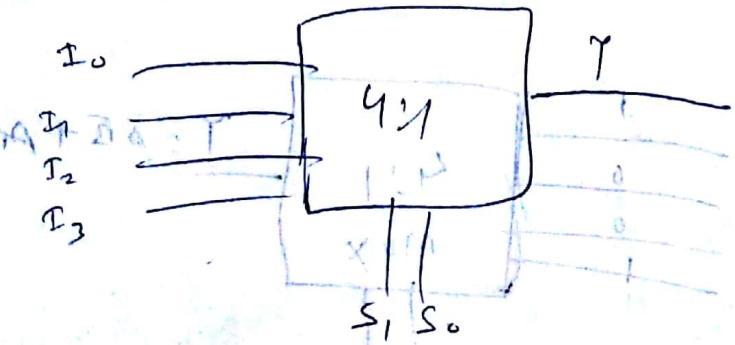
$$= 000, 011, 101$$

$$= 0, 3, 5$$

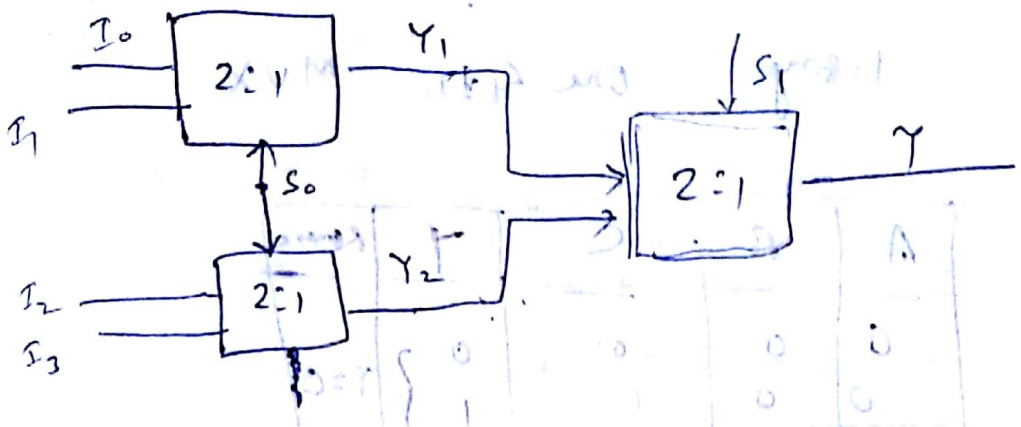
$$= I_0, I_3, I_5$$



Ex :- 5 Realize 4:1 MUX using 2:1 MUX



S_1	S_0	
0	0	Y_1
0	1	Y_1
1	0	Y_2
1	1	Y_2



< We have 4 i/p pins & 2 select pin. Use three 2:1 MUX to Construct the functionality of 4:1 MUX >

S_1	S_0		Y
0	0	—	I_0
0	1	—	I_1
1	0	—	I_2
1	1	—	I_3

(Hw) 1) 8:1 mux using 4:1 mux
2) 8:1 " " 2:1 mux

BIUT 2019
Ex - 5

Implement the following boolean function with a 4:1 MUX.

$$f(x, y, z) = \Sigma (1, 2, 6, 7)$$

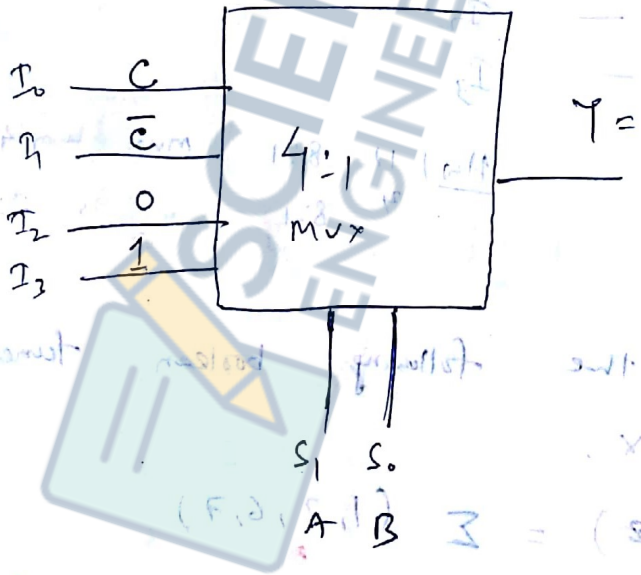
Ans: The truth table for the boolean function is given below

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

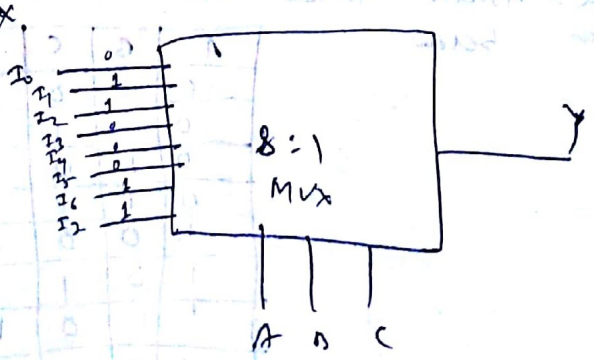
①

We have to implement the given truth table using one 4:1 MUX.

A	B	C	Y	Remark
0	0	0	0	Y = C
0	0	1	1	
0	1	0	1	Y = $\bar{C}$
0	1	1	0	
1	0	0	0	Y = 0
1	0	1	0	
1	1	0	1	Y = 1
1	1	1	1	



Using 8:1 mux



O/P is 1,

At 1, 2, 6, 7

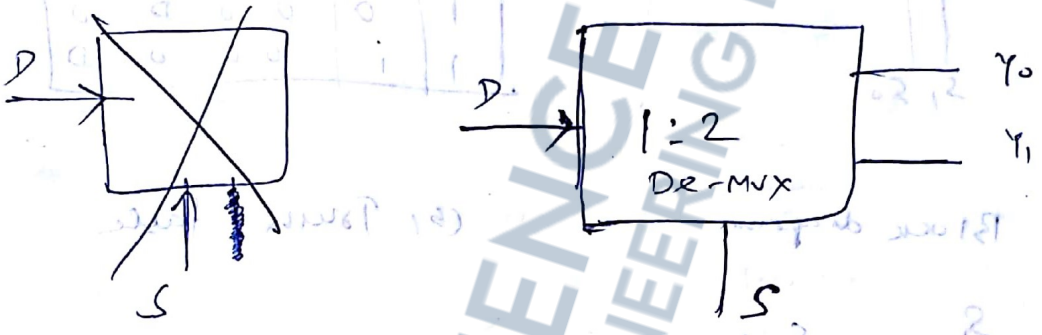
i.e

I_1, I_2, I_6, I_7

Demultiplexer

Demultiplexer (DE-MUX) is a combinational logic ckt that performs the reverse operation of MUX. It has one i/p & many o/ps. The i/p information is transmitted to a particular o/p line depending on the data available at the selector line.

1:2 DE-MUX



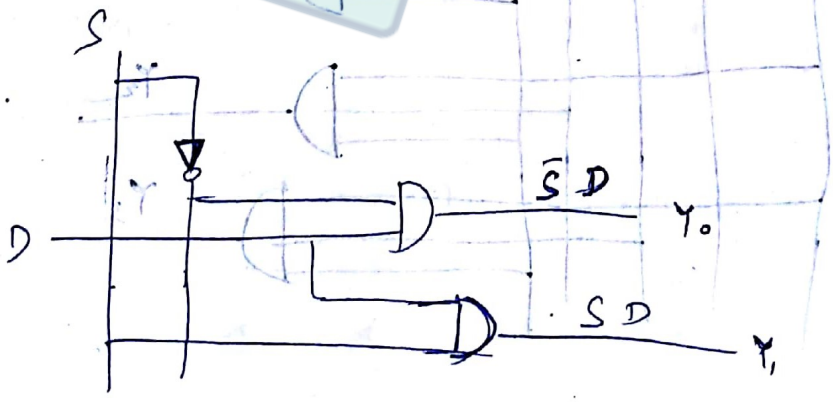
(a) Block diagram

S	Y_0	Y_1
0	D	0
1	0	D

(b) Truth table

$Y_0 = \bar{S} D$

$Y_1 = S D$



(c) Logic ckt.

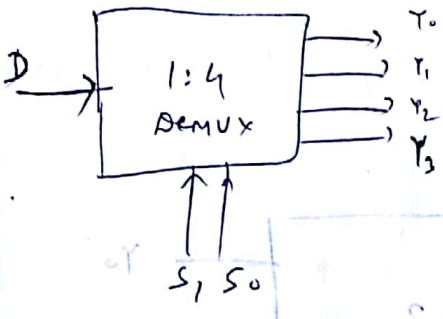
1:4 DE-MUX

Here 1 I/P & 4 O/P, So it has 2M Selector line.

Input → D

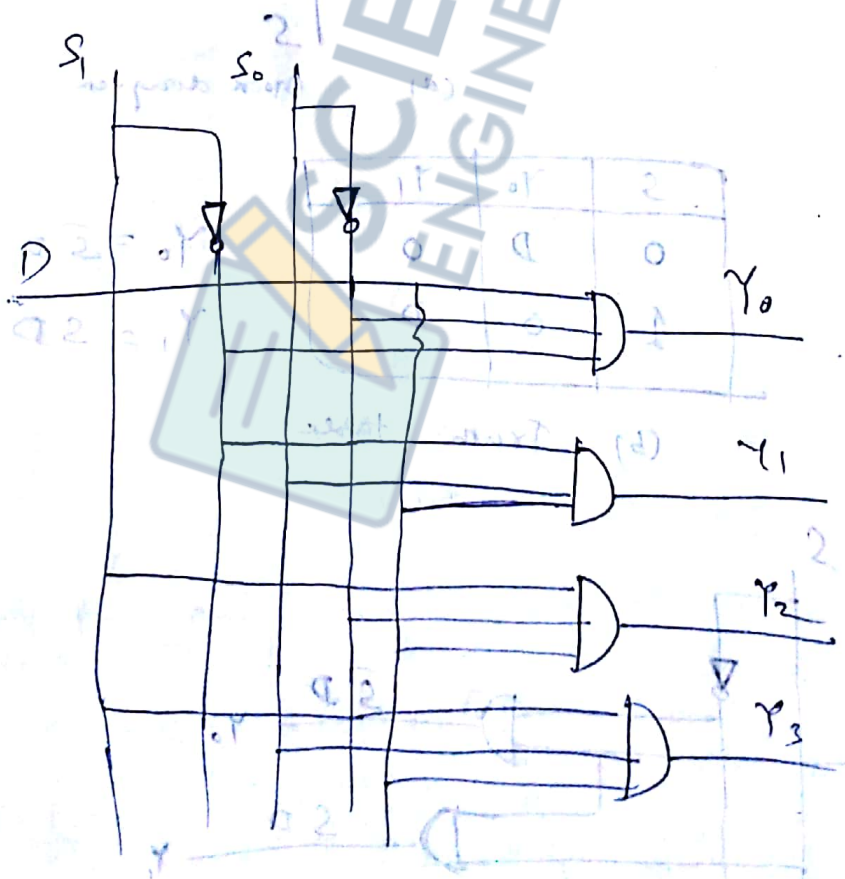
O/P → Y_0, Y_1, Y_2, Y_3

Selector line - S_1 & S_0



S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

(a) Block diagram (b) Truth table



(c) Logic ckt.

$$Y_0 = \bar{S_1}\,\bar{S_0}\,D$$

$$Y_1 = \bar{S_1}\,S_0\,D$$

$$Y_2 = S_1\,\bar{S_0}\,D$$

$$Y_3 = S_1\,S_0\,D$$

~~Pgs~~ -

Multiplexer

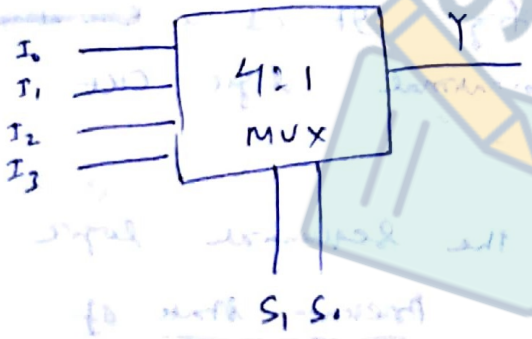
1) It is a data selector.

2) It has many i/p and single o/p.

3) Parallel to serial converter.

4) It has n selection line, 2^n no of i/p, and 1 o/p.

5) Ex:-



Demultiplexer

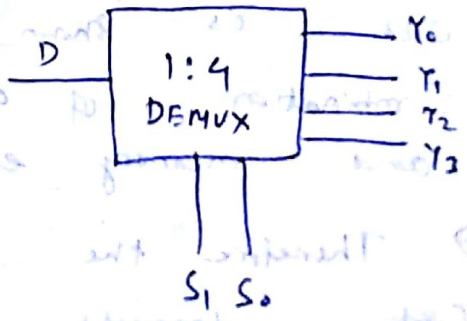
1) It is a data distributor.

2) It has single i/p and many o/p.

3) Serial to parallel converter.

4) It has 1 i/p, n selection line and 2^n o/p.

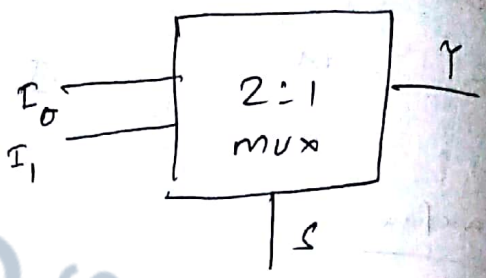
5) Ex:-



MUX as Universal gate

1) NOT gate using 2:1 MUX

A	Y
0	1
1	0



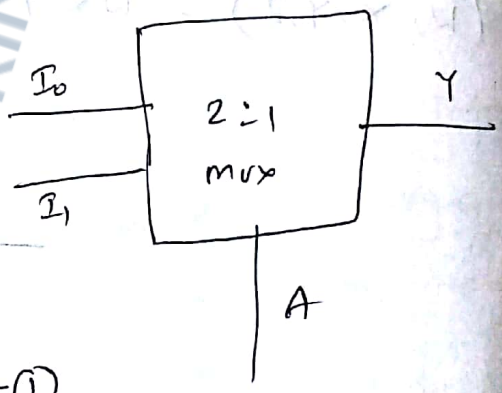
Step 1:-

We have one select line. A select line can take 2 values '0' or '1'. So lets take 'A' as select line because it has value 0 & 1.

Step 2:-

When $A = 0$, $Y = I_0$

$A = 1$, $Y = I_1$



A	Y
0	I_0
1	I_1

—(1)

Step 3:- Relation of Y with A

$$Y = \bar{A}$$

(∵ when $A = 0$, $Y = 1$

when $A = 1$, $Y = 0$.)

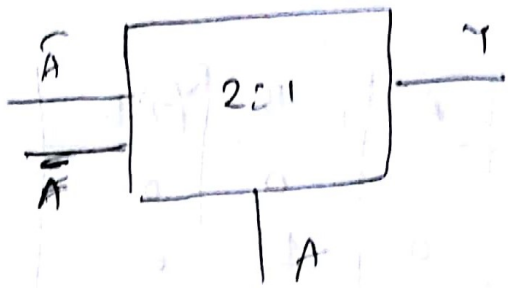
A	Y	Y
0	1	$\bar{A}$
1	0	$\bar{A}$

—(2)

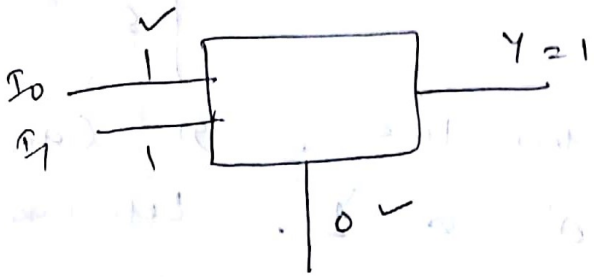
Equating ① & ②

$I_0 = \bar{A}$,

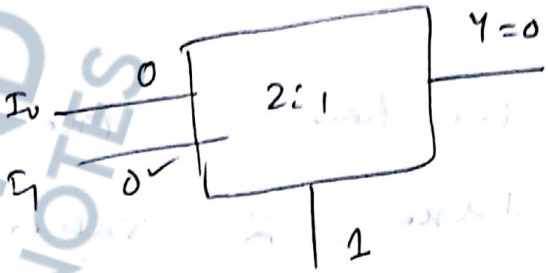
$I_1 = \bar{A}$



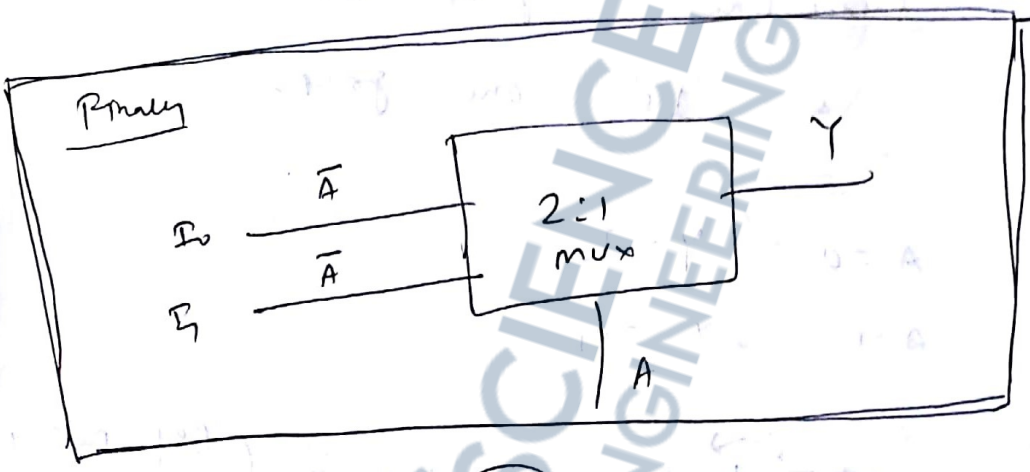
Ex:- When A = 0



When A = 1



∴



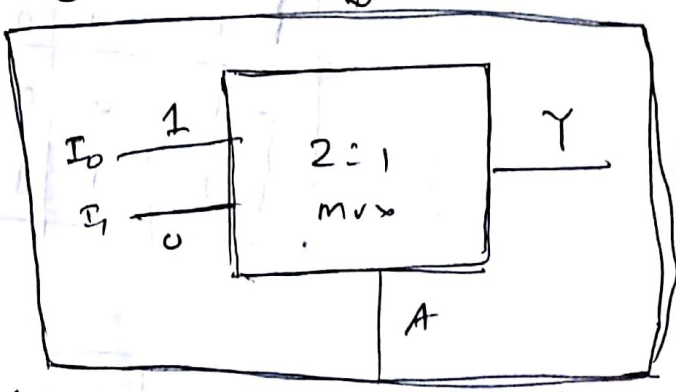
~~By AND gate~~ (or)

Answer 2

Equating ① & ②.

$I_0 = 1$

$I_1 = 0$



Verify

When A = 0, I_0 is selected, ⇒ Y = 1.

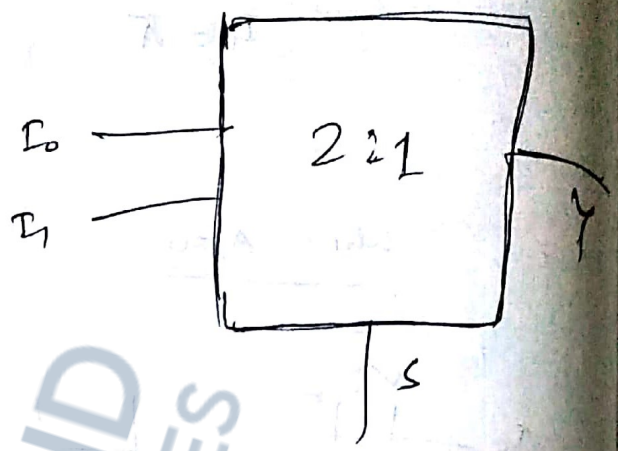
When A = 1, I_1 is selected, ⇒ Y = 0

} Act as NOT gate

Verify; $Y = \bar{A} \cdot I_0 + A \cdot I_1 = \bar{A} \cdot 1 + A \cdot 0 = \bar{A}$

2) AND gate

A	B	Y=AB
0	0	0
0	1	0
1	0	0
1	1	1



We have one select line. It can take 2 values '0' or '1'. Lets take 'A' as select line group the '0s' as one group & '1s' one group.

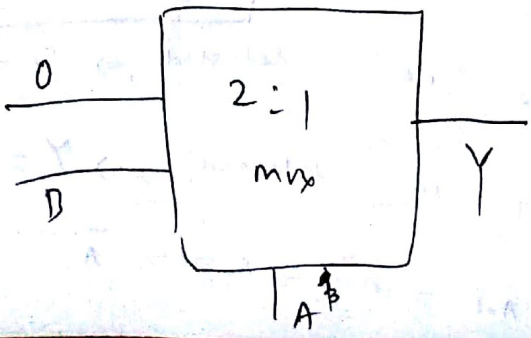
For A = 0, $Y = I_0$

A = 1, $Y = I_1$

A	B	Y	Y	Remark (Rel^n Bet^n Y and B)
0	0	0	I_0	0 → Y = 0, is always independent of B.
0	1	0		
1	0	0	I_1	B → Y = B
1	1	1		

$\therefore I_0 = 0$

$\therefore I_1 = B$



Verify

$$Y = \bar{S} I_0 + S I_1$$

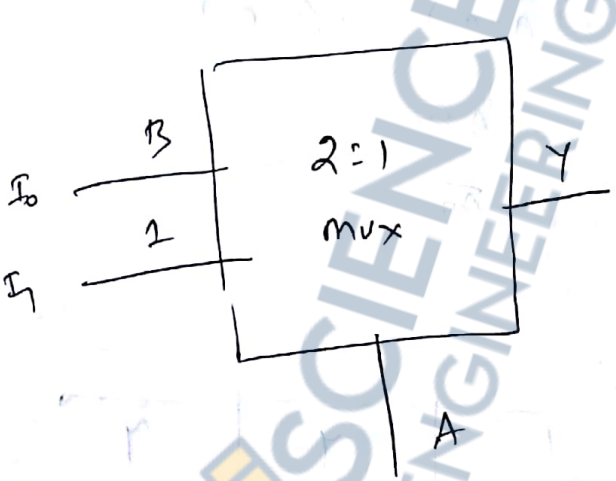
$$= \bar{A} \cdot 0 + A \cdot B$$

$$= AB$$

3) OR gate

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

Y	Y
I_0	B
I_1	1



A2 Select line

Verify ($Y = \bar{S} I_0 + S I_1$)

$$Y = \bar{A} \cdot B + A \cdot 1$$

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

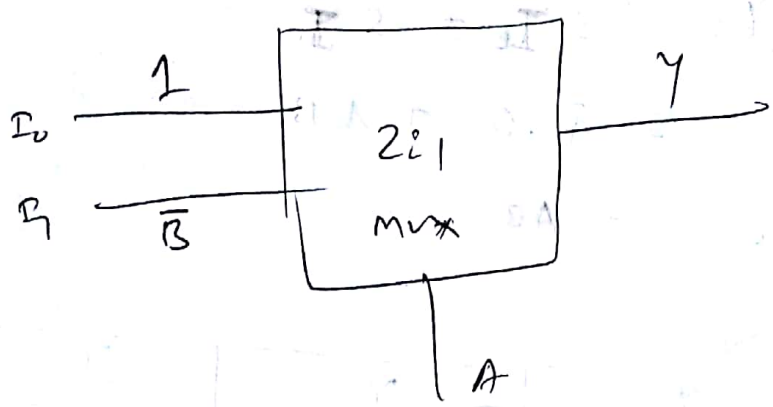
$$= (A + \bar{A})(A + B)$$

$$Y = A + B$$

4) NAND gate:

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

Y	Y
I_0	1
I_1	$\bar{B}$



Verify :

$$Y = \bar{S} I_0 + S I_1$$

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

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

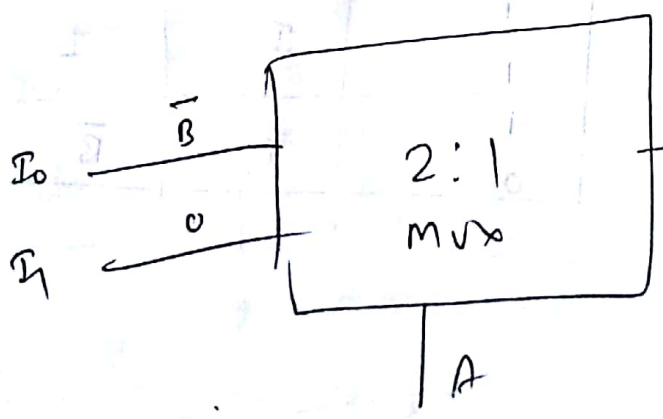
$$= (\bar{A} + A)(\bar{A} + \bar{B})$$

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

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

5) NOR

A	B	Y	Y	Y
0	0	1	I_0	$\bar{B}$
0	1	0		
1	0	0	I_1	0
1	1	0		



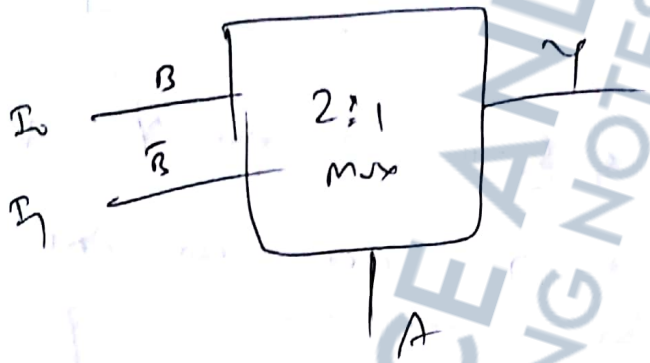
O/P

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

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

6) Ex-OR

A	B	Y	Y	Y
0	0	0	I_0	B
0	1	1		
1	0	1	I_1	$\bar{B}$
1	1	0		



Verify: $Y = \bar{A}.B + A\bar{B} = A \oplus B$.

7) Ex-NOR

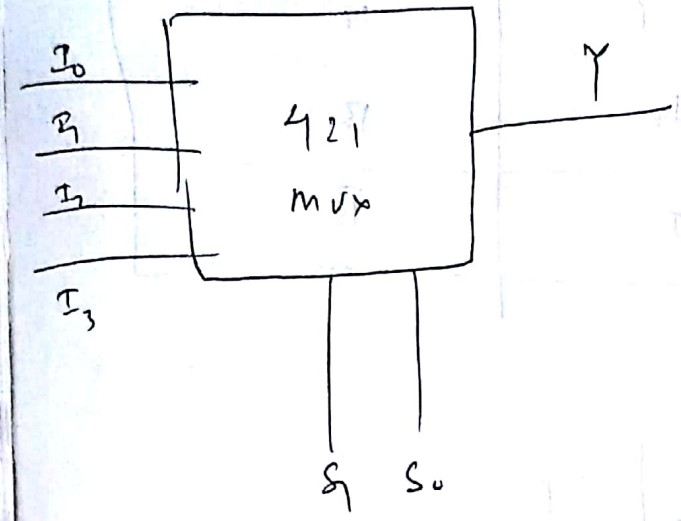
A	B	Y	Y	Y
0	0	1	I_0	$\bar{B}$
0	1	0		
1	0	0	I_1	B
1	1	1		

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

$Y = A \odot B$

(verified)

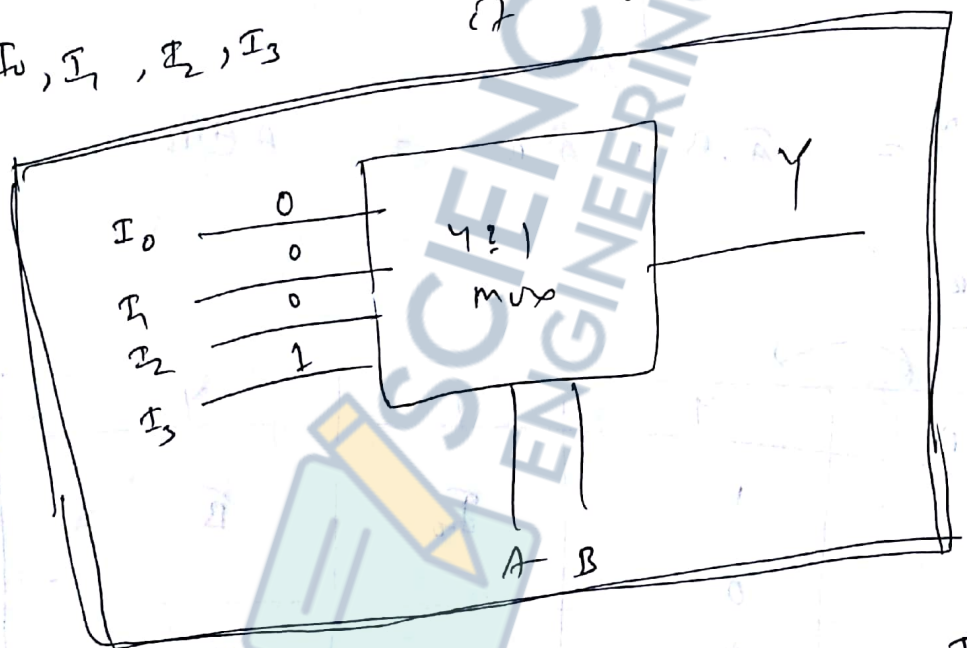
8) Implement $Y = AB$ using 4:1 mux



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

If $S_1 = A$ & $S_0 = B$ are Y values to act as AND gate

I_0, I_1, I_2, I_3



Ex: When $A=0, B=0$, I_0 is selected its value is '0'.

When $A=0, B=1$, I_1 selected, $Y_B = 0$

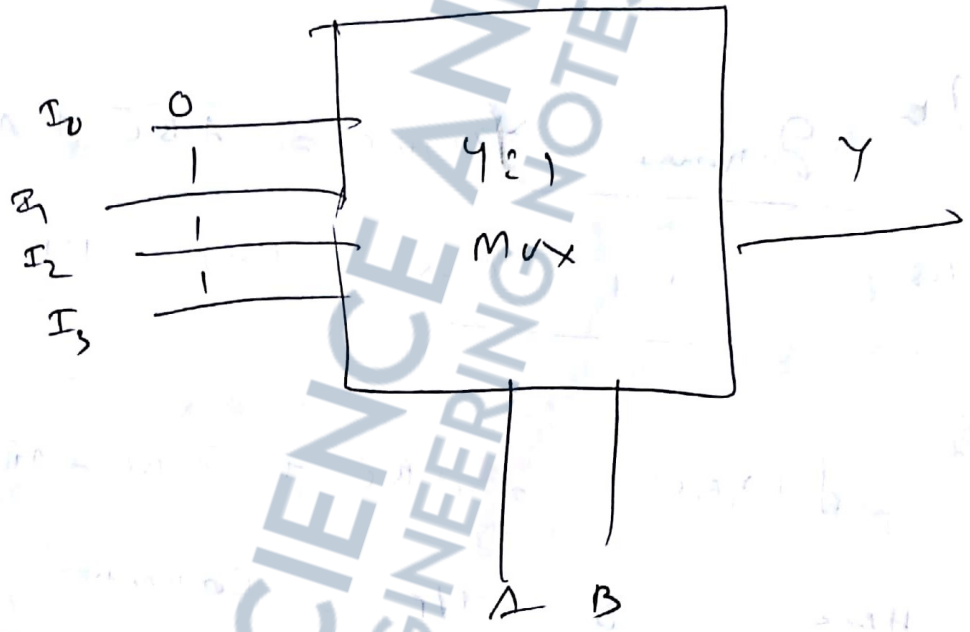
$A=1, B=0$, I_2 " , $Y=0$

$A=1, B=1$, I_3 " , $Y=1$.

∴ It behaves as AND gate.

9) Similarly other gates can be implemented using 4:1 MUX. With A, B select lines & I_0, I_1, I_2, I_3 one fourth table values of that gate.

Ex: A + B



Implementation of any function using MUX

→ First Check what are the i/p combinations required.

→ ~~If ——— Then~~

(Ex:- AND gate : 4 combinations)

→ Then Check what are the i/p line available.

If 2:1 mux → 2 i/p line → I_0, I_1

If 4:1 MUX → 4 " 4 → I_0, I_1, I_2, I_3

→ If 2 i/p line then we have to make a group.

$A = \begin{cases} 0 \\ 0 \end{cases}$ → 1 group → I_0

$A = \begin{cases} 1 \\ 1 \end{cases}$ → 1 group → I_1

for 4 possible I/P combination.

If 4 I/P line no need of grouping

Just assign the corresponding truth table values — to I_0, I_1, I_2, I_3.

Ex) 10) Implement $f(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}C$

using 8:1 mux, using 4:1 mux, 2:1 mux

Ans:

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

Here 8 I/P combination, If we use a 8:1 mux, just we have to assign the truth table values.

A	B	C	Y
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	0
1	1	1	0

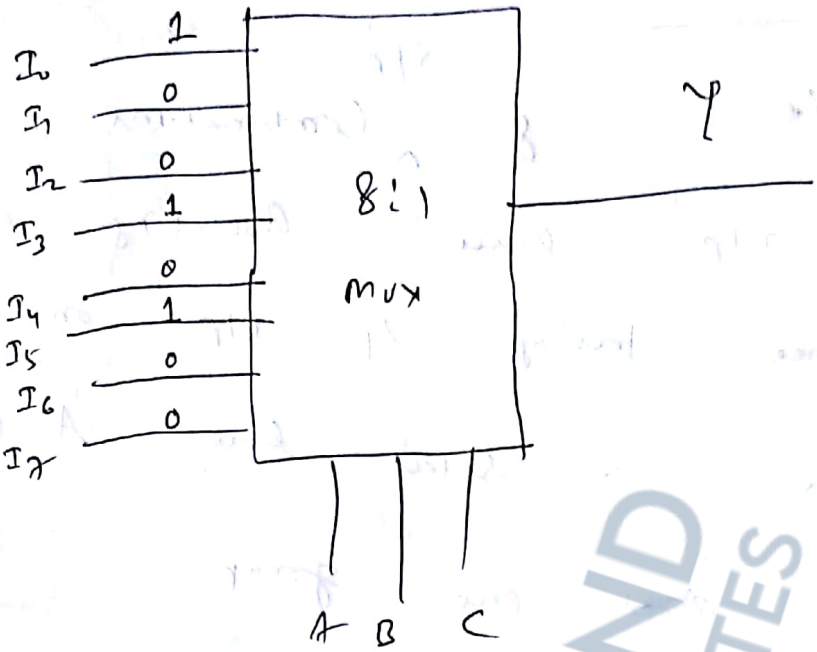
$Y = 000, 011, 101$

$= 0, 3, 5$

$= I_0, I_3, I_5$

$\therefore$ I_0, I_3, I_5 will be '1' and rest will be Zero.

A, B, C are the select lines.



Using 4:1 MUX

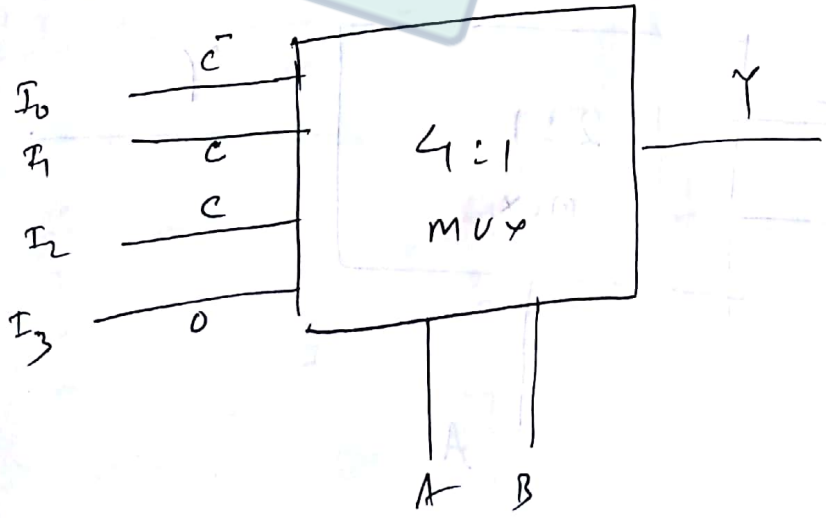
8 i/p Combination we have 4 o/p line. So we have to make grouping.

A	B	C	Y	Y	Y
0	0	0	1	I_0	$\bar{C}$
0	0	1	0		
0	1	0	0	I_1	C
0	1	1	1		
1	0	0	0	I_2	C
1	0	1	1		
1	1	0	0	I_3	0
1	1	1	0		

Lets say 'A' is the select line.

00, 01, 10, 11

4 groups.



Using 2:1 mux

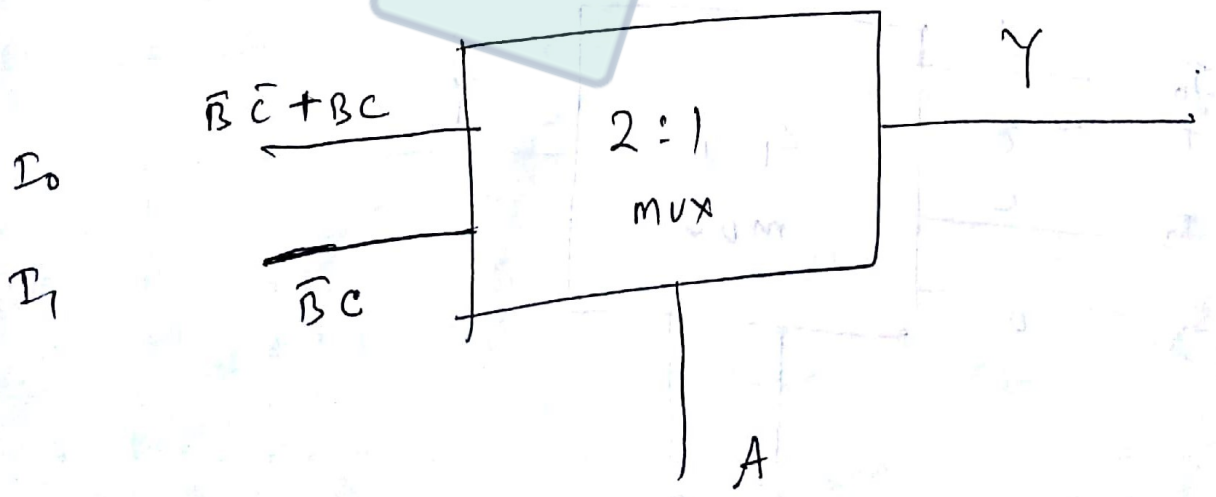
We have 8 i/p combinations, but we have 2 i/p lines. Grouping is made taking 4 i/p one group.

→ We have a select line 'A' (Lets say)

→ A = 0, one group

A = 1, another group.

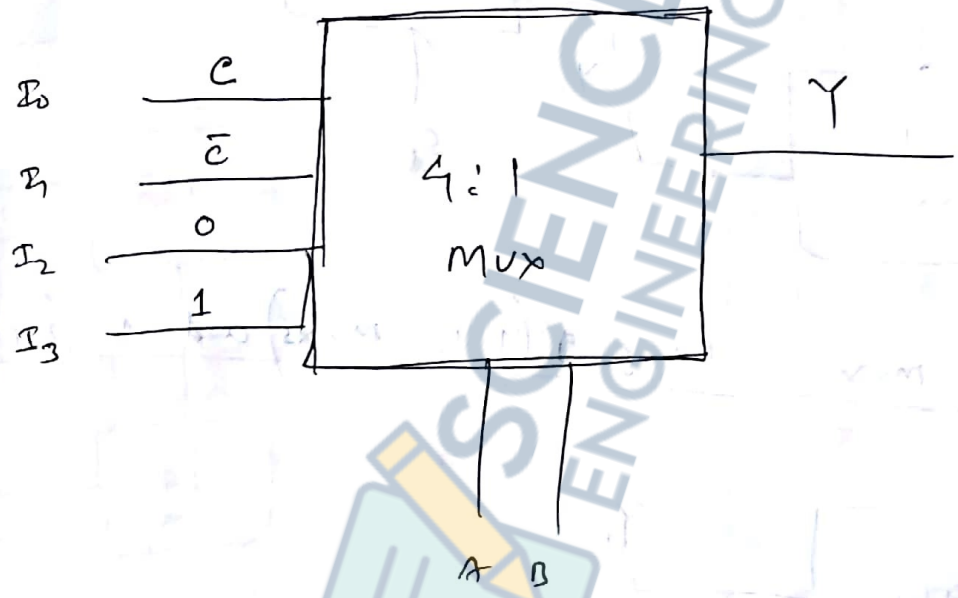
A	B	C	Y	Y	Y (Rel^n of B and C with Y)
0	0	0	1	I_0	$\bar{B}\bar{C} + BC$
0	0	1	0		
0	1	0	0		
0	1	1	1		
1	0	0	0	I_1	$\bar{B}C$
1	0	1	1		
1	1	0	0		
1	1	1	0		



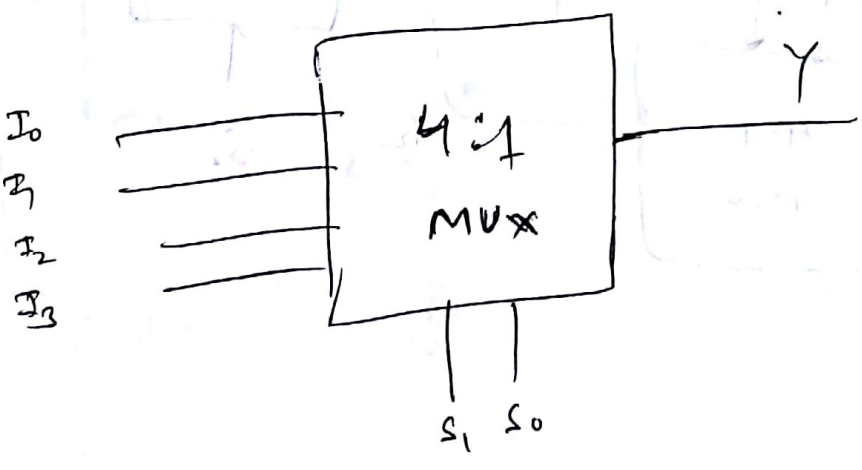
10) BPUT-2009

$f(A,B,C) = \Sigma(1,2,6,7)$ Using 4:1 MUX.

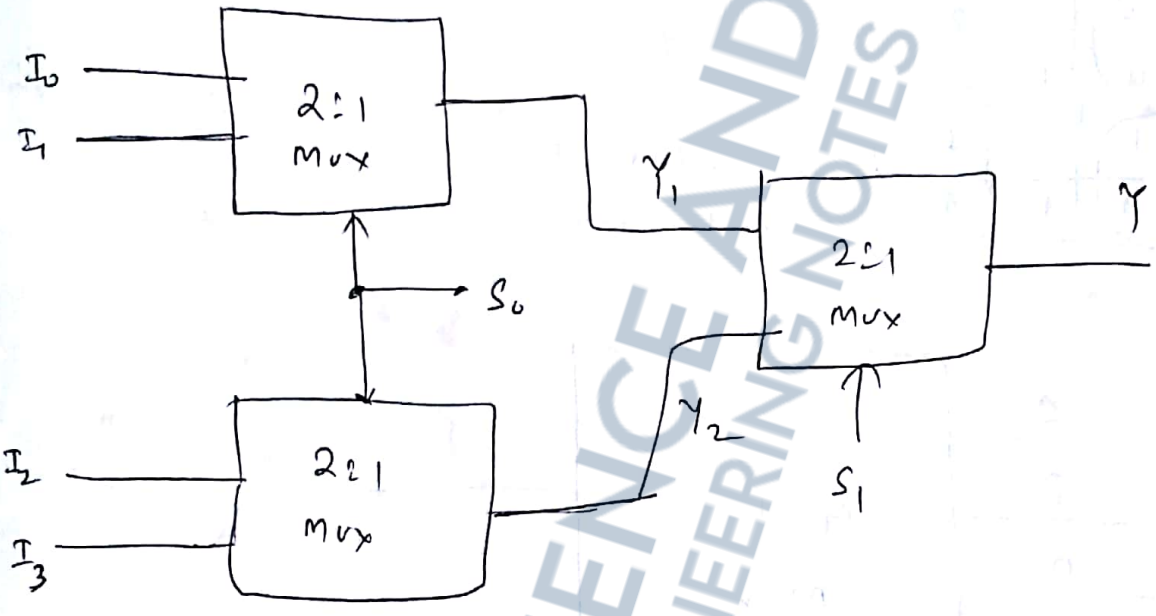
A	B	C	Y	Y	Y
0	0	0	0	I_0	$Y = C$
0	0	1	1		
0	1	0	1	I_1	$Y = \bar{C}$
0	1	1	0		
1	0	0	0	I_2	$Y = 0$
1	0	1	0		
1	1	0	1	I_3	$Y = 1$
1	1	1	1		



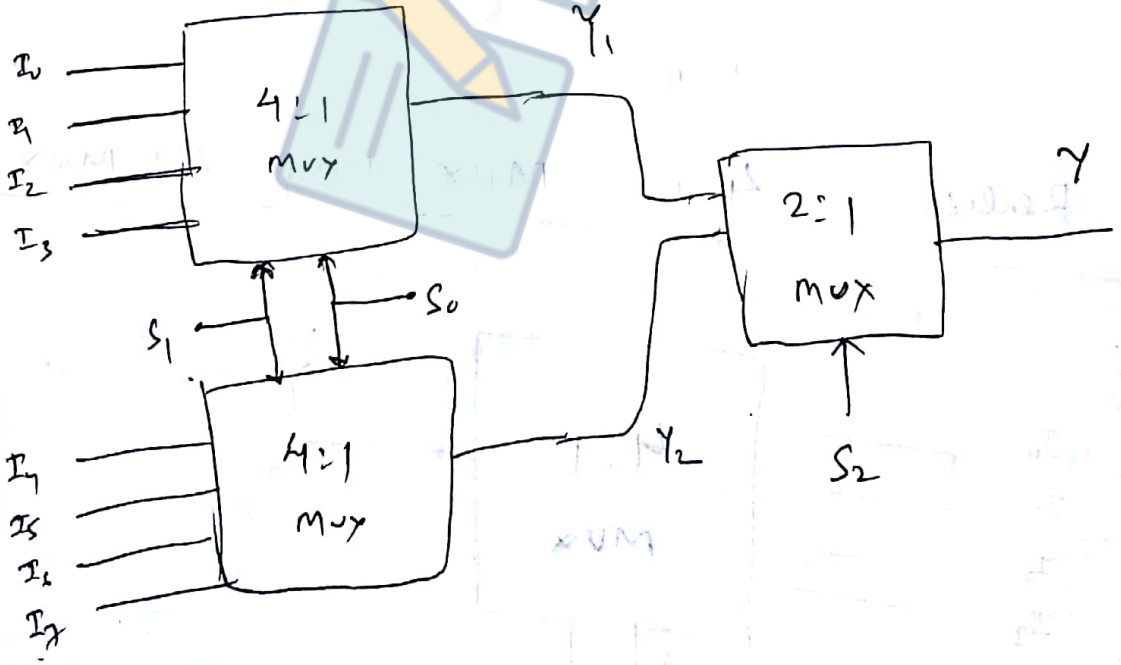
11) Realize 4:1 MUX using 2:1 MUX



S_1	S_0	Y	Y
0	0	I_0	Y_1
0	1	I_1	
1	0	I_2	Y_2
1	1	I_3	



12) 8:1 mux using 2 (4:1 muxs) and 1 (2:1 MUX)



8:1 MUX

S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7

For

$S_2 = 0$, Y_1 is selected.

Y_1 depends on S_1 S_0.

Let's check

$S_2 = 0$, $S_1 = 0$, $S_0 = 0$.

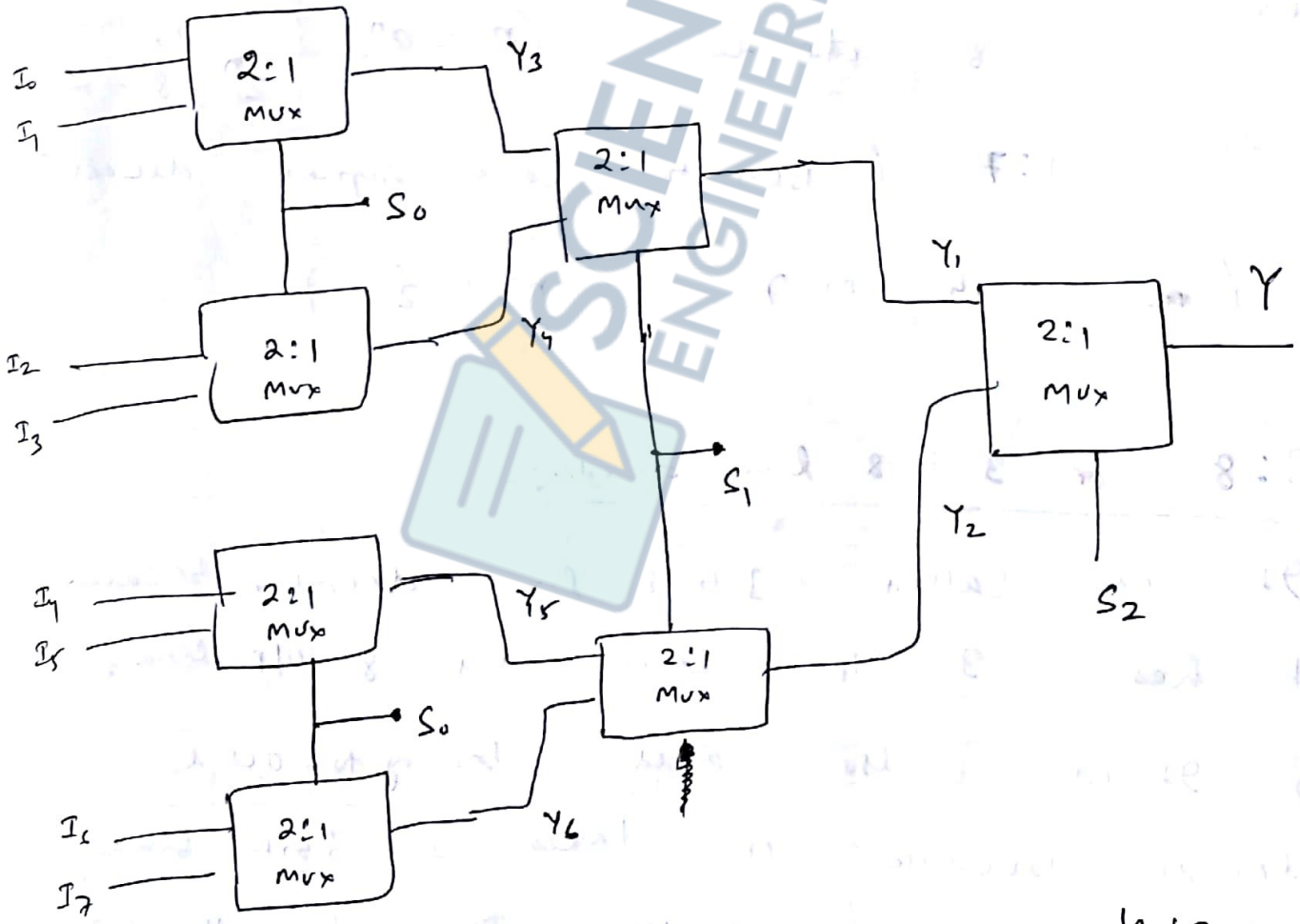
$S_1 = 0$, $S_0 = 0$, $Y_1 = I_0$

$Y_2 = I_4$

Again since $S_2 = 0$,

Y_1 is selected i.e I_0.

(3) 8:1 MUX Using 2:1 MUX



Total 2:1 MUX required = 4+2+1

= 7

Decoders

A decoder is a combinational circuit that converts binary information from 'n' i/p lines to a max^m 2^n unique o/p lines.

If the n-bit coded information has unused combinations, the decoder may have fewer than 2^n o/ps.

So we have n-to-m line decoders, where $m \leq 2^n$.

ex:
1) 3:8 decoder $(m = 2^n)$ | $n = 3, m = 8$, $2^n = 8 = m$
2) 4:7 (BCD to Seven segment decoder)
($n = 4$, $m = 7$, $m < 2^n$)

3:8 or 3-to-8 line decoder:-

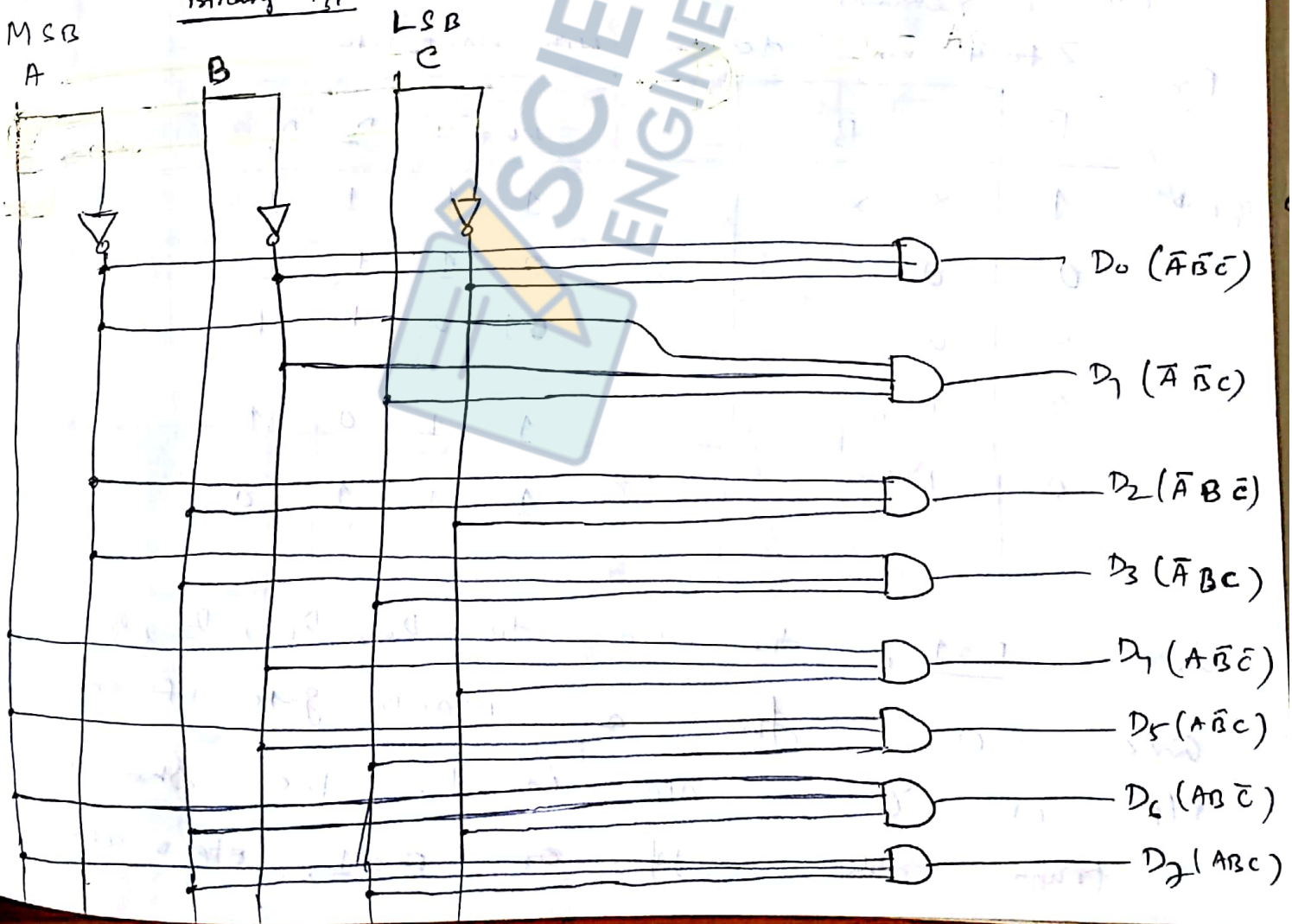
→ It is called 3-to-8 line decoder because it has 3 i/p lines and 8 o/p lines.

→ It is also called binary-to-octal decoder because it takes a 3 bit binary i/p code and activates one of the eight (octal) outputs corresponding to that code.

Truth Table :-

i/ps			o/ps							
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Logic diagram



Note:-

1) Decoders include one or more enable i/ps to control the circuit operation. A 2 to 4 line decoder with an enable input constructed with NAND gate is shown below.

N.B:- (i) A decoder identifies or recognizes or detects a particular code.

(ii) If we have N i/p lines & M o/p lines, then one of the M o/ps will be active (High), all other o/p will be inactive (Low).

→ (iii) Some decoders are designed to produce active LOW o/p, while all other o/ps remains ~~High~~ High. (Ex:- Shown below)

Ex: 2 to 4 line decoder with NAND gate.

E	A	B	D_0	D_1	D_2	D_3
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(Enable)

When E=1, the i/p to D_0, D_1, D_2, D_3 are 0. for a NAND gate if one i/p is '0' o/p is 1. i.e shown in truth table. If E = 1, ~~O/P~~ all the

o/p of D_0, D_1, D_2, D_3 are 1. They are independent of A, B. So A, B are don't cares.

→ When E = 0, i/p to D_0, D_1, D_2, D_3 are 1. Then it behaves as decoder (Active Low decoder) that means. When

A = 0, B = 0, $D_0 = 0$

A = 0, B = 1, $D_1 = 0$

A = 1, B = 0, $D_3 = 0$

A = 1, B = 1, $D_4 = 0$

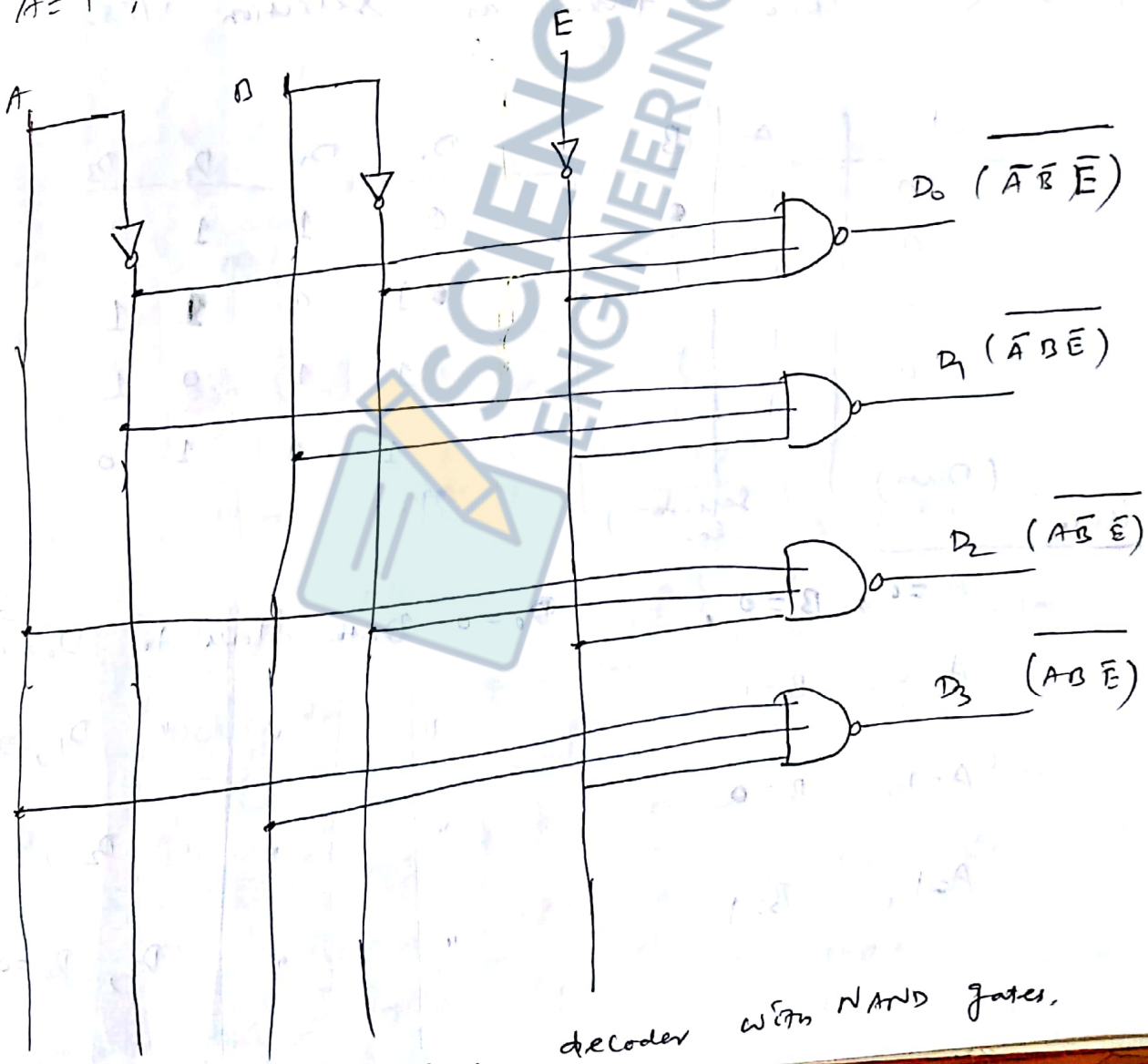


Fig 2:- 2 to 4 line decoder with NAND gates.

O/P of D_0, D_1, D_2, D_3 are 1. They are independent of A, B. So A, B are do n't Cares.

→ When E = 0, i/p to D_0, D_1, D_2, D_3 are 1. Then it behaves as decoder (Active LOW decoder) that means. when

A = 0, B = 0, $D_0 = 0$

A = 0, B = 1, $D_1 = 0$

A = 1, B = 0, $D_3 = 0$

A = 1, B = 1, $D_4 = 0$

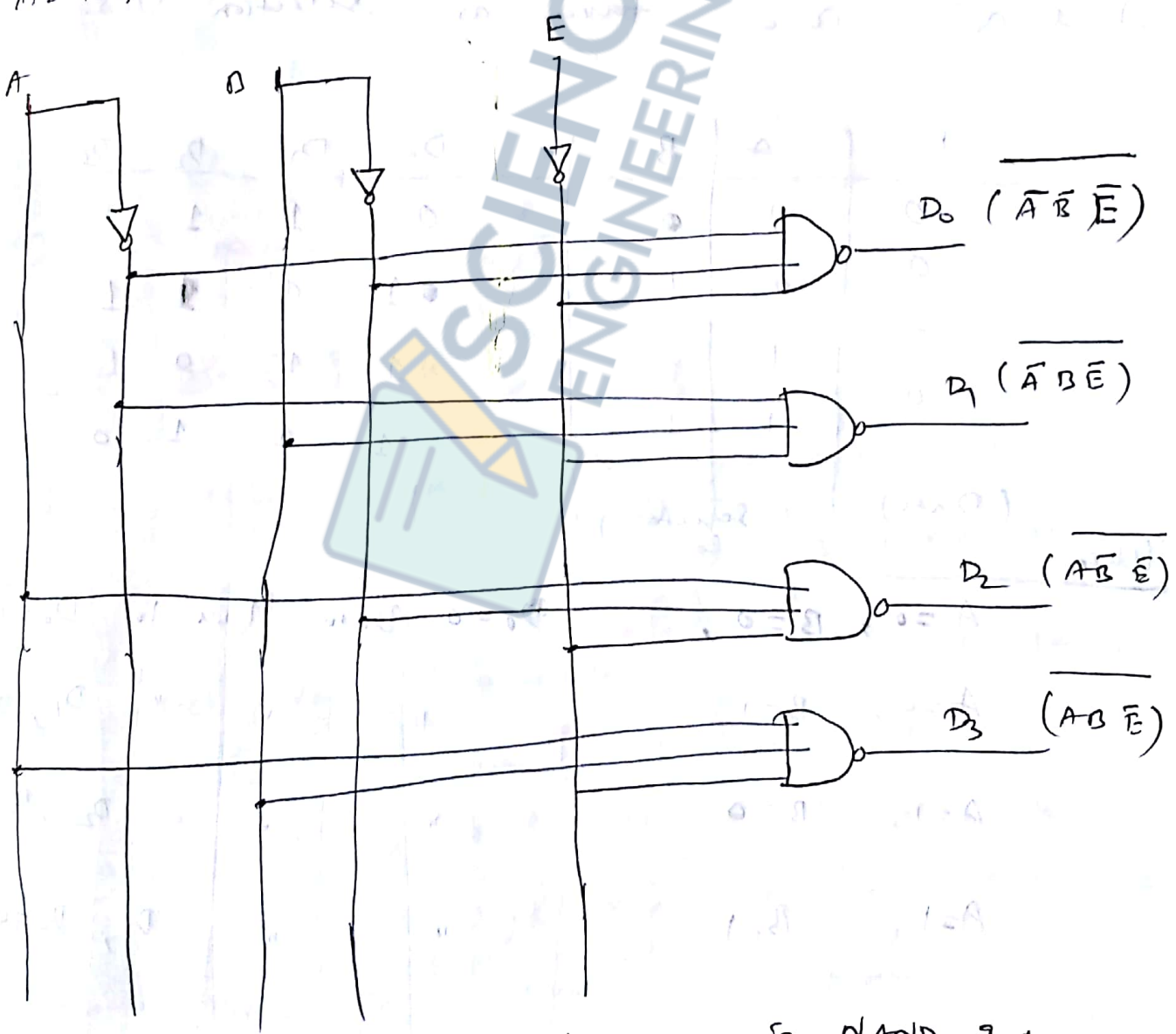


Fig 2:- 2 to 4 line decoder with NAND gates.

O/p of D_0, D_1, D_2, D_3 are 1. They are independent of A, B. So A & B are do n't Cares.

→ When E = 0, i/p to D_0, D_1, D_2, D_3 are 1. Then it behaves as decoder (Active LOW decoder) that means. When

A = 0, B = 0, $D_0 = 0$

A = 0, B = 1, $D_1 = 0$

A = 1, B = 0, $D_3 = 0$

A = 1, B = 1, $D_4 = 0$

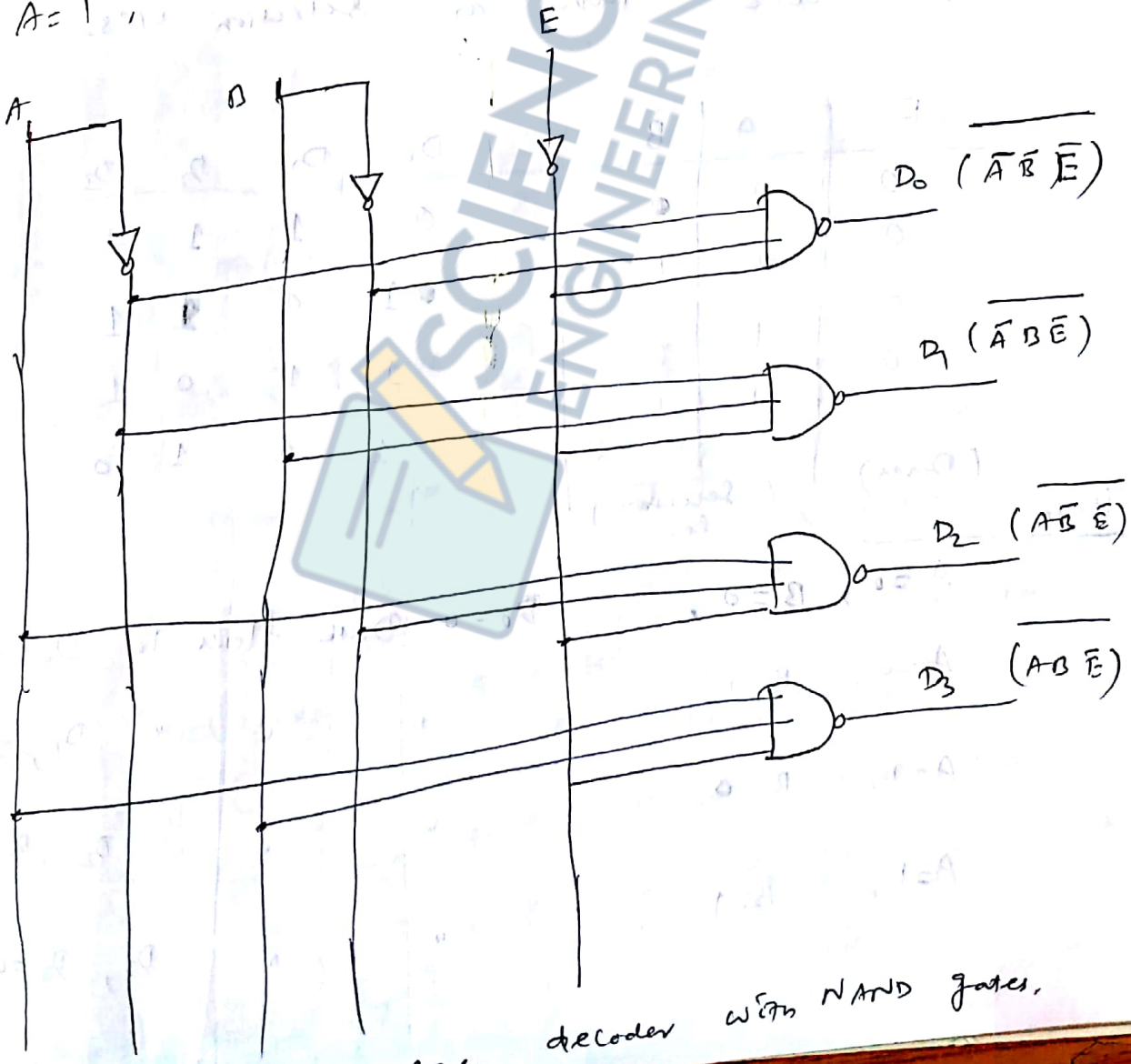


Fig 2:- 2 to 4 line decoder with NAND gates.

2) A decoder with enable i/p function can be as demultiplexer. A demultiplexer is a circuit that receives information from a single line and directs it to one of 2^n possible o/p lines.

The selection of specific o/p is controlled by the bit combination of n select lines. The decoder in figure 2 can function as 1 to 4 line demultiplexer. When E is taken as data i/p line and A & B are taken as selection i/ps.

E	A	B	D_0	D_1	D_2	D_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
(Data)	(Selection line)					

When

A = 0, B = 0, $D_0 = 0$ Data flows to D_0, $D_0 = 0$ (S0)

A = 0, B = 1 " " " D_1, $D_1 = 0$

A = 1, B = 0 " " " D_2, $D_2 = 0$

A = 1, B = 1 " " D_3, $D_3 = 0$

Combinational Logic Implementation :-

1) Ex :- Implement Full adder using 3:8 decoder.

Ans :- Full Adder

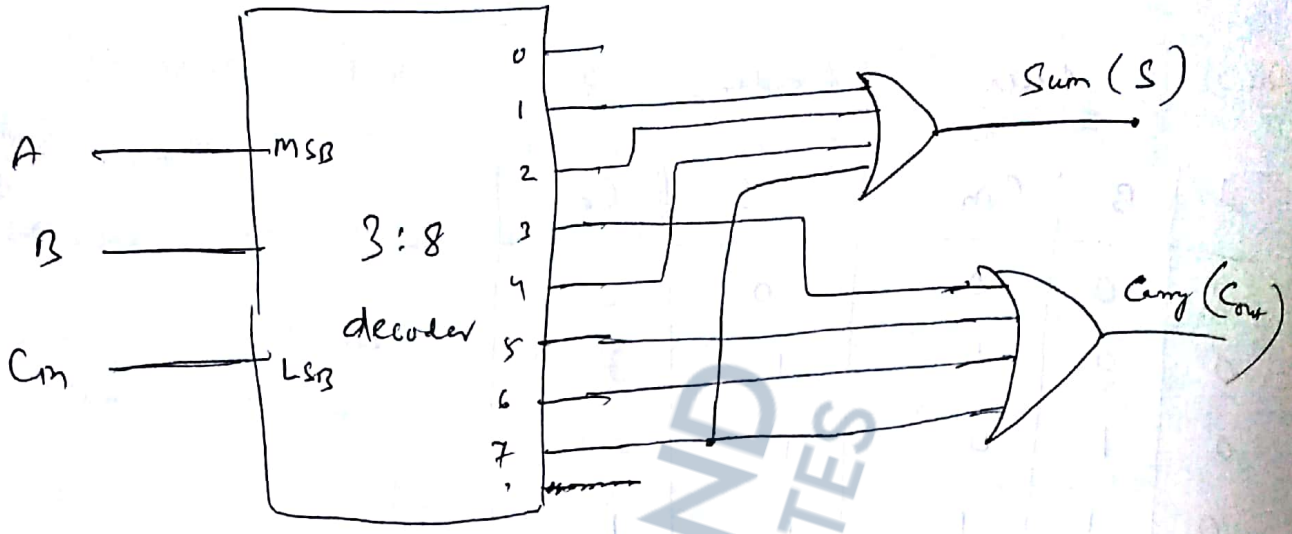
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

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

$C_{out} = \sum m(3, 5, 6, 7)$

Since there are 3 i/ps and total of 8 minterms, we need a 3-to-8 line decoder.

The decoder generates the 8 minterms for A, B, Cin. The OR gate for o/p Sum (S) forms the logical sum of the minterm (1, 2, 4, 7) and OR gate for Cout forms the logical sum of the minterms (4, 5, 6, 7).



Ex:- A B C
1 0 1 ,

At 5 → o/p is 1.

Rest 1 to 7 → o/p is zero.

Since Sum does not contain '5', S = 0

Carry contain '5' , Cout = 1 .

A B C
1 0 1 for

∴ S = 0 , C_{out} = 1

2) Full Adder Using 4:1 MUX

A	B	C_{in}	S	S	S (Refer both Cin & S)	Cout	Cout	Cout (Refer both Cin & Cout)
0	0	0	0	I_0	C_{in}	0	I_0	0
0	0	1	1			0		
0	1	0	1	I_1	$\bar{C}_{in}$	0	I_1	C_{in}
0	1	1	0			1		
1	0	0	1	I_2	$\bar{C}_{in}$	0	I_2	C_{in}
1	0	1	0			1		
1	1	0	0	I_3	C_{in}	1	I_3	1
1	1	1	1			1		

For Sum

Let's take A & B as

~~A, B~~ ~~A = 0~~,

select line.

when

A=0, B=0, I_0 is selected, Relating Sum(S)

with C_{in}, $S = C_{in} = I_0$ (when A=0, B=0 I_0 is selected)

A=0, B=1, $S = \overline{C_{in}} = I_1$ "

A=1, B=0, $S = \overline{C_{in}} = I_2$ "

A=1, B=1, $S = C_{in} = I_3$ "

For Carry =

~~Rest~~ A & B are select lines.

Carry

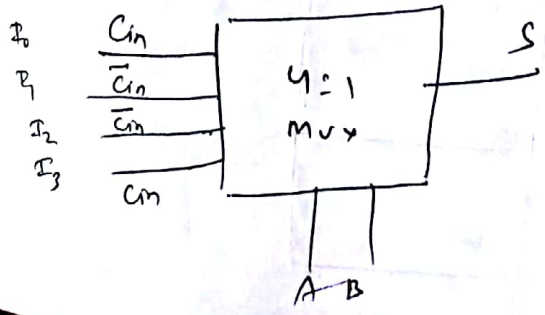
When both C_{in}

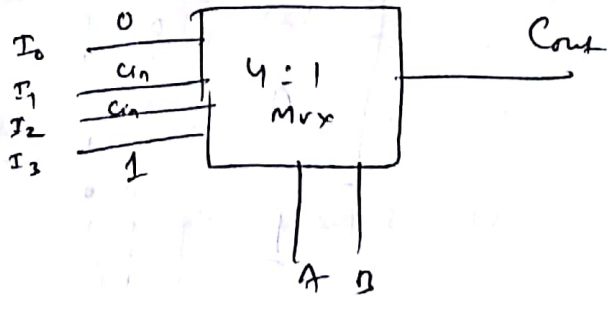
A=0, B=0, $C_{out} = 0 = I_0$ (∵ when A=0, B=0 I_0 is selected)

A=0, B=1, $C_{out} = C_{in} = I_1$ "

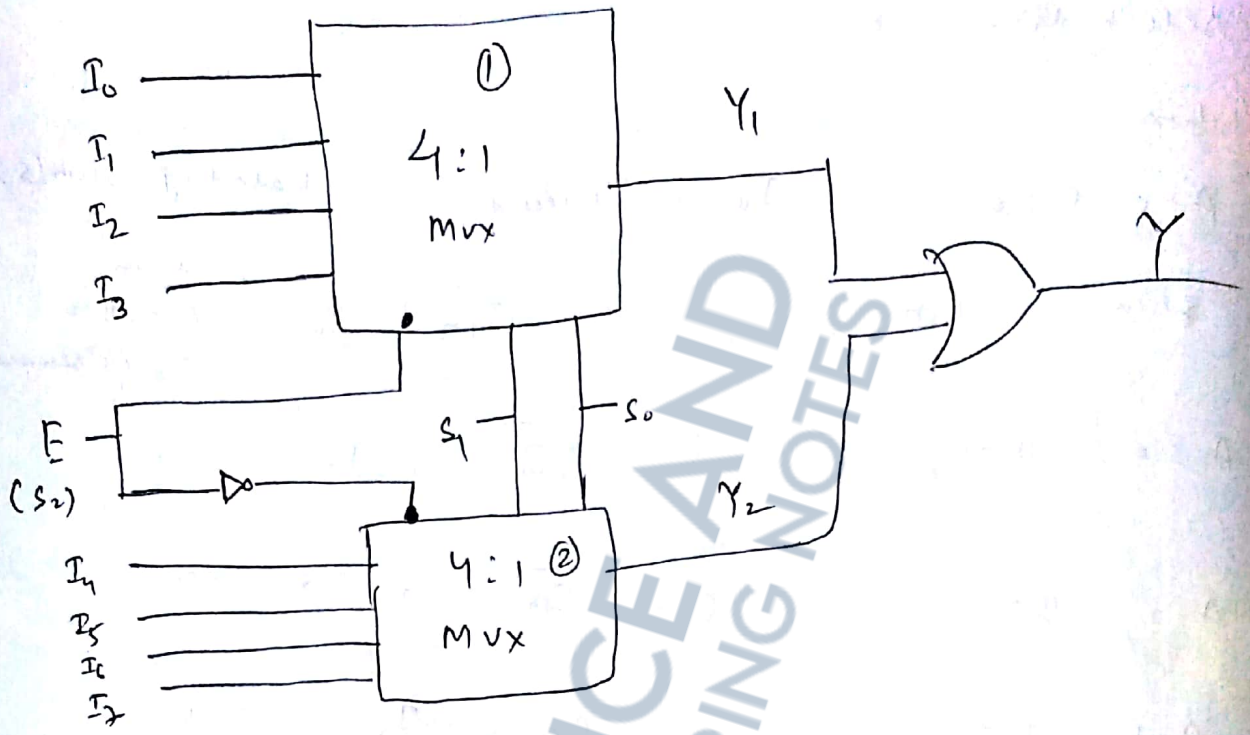
A=1, B=0, $C_{out} = C_{in} = I_2$ "

A=1, B=1, $C_{out} = 1 = I_3$ "





3) 8:1 MUX Using only 4:1 MUX



E → Enable I/P.

Assume. MUX has Active low enabler.

^

When E=0, MUX-1 works, So o/p appears at Y_1 and o/p at $Y_2 = 0$.

When E=1, MUX-2 works, So o/p appears at Y_2 and o/p at $Y_1 = 0$

E (S2)	S1	S0	Y
0	0	0	I0
	0	1	I1
	1	0	I2
	1	1	I3
1	0	0	I4
	0	1	I5
	1	0	I6
	1	1	I7

Encoder

→ An encoder is a digital circuit that performs the inverse operation of a decoder.

→ An encoder has 2^n (or fewer) i/p lines and 'n' o/p lines. The o/p lines generate the binary code corresponding to the i/p value.

→ An example of an encoder is the octal-to-binary encoder whose truth table is given below.

i/p								Output		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

→ It has eight i/ps (one for each octal digit) and three o/p that generate the corresponding binary number.

→ It is assumed that only one i/p has a value of 1 at any given time.

→ The encoder can be implemented with OR gates whose i/ps are determined directly from the truth table.

→ Referring to the truth table (Previous page) o/p Z is equal to 1 when the i/p octal digit is 1, 3, 5 or 7.

→ Similarly for y → 2, 3, 6, 7

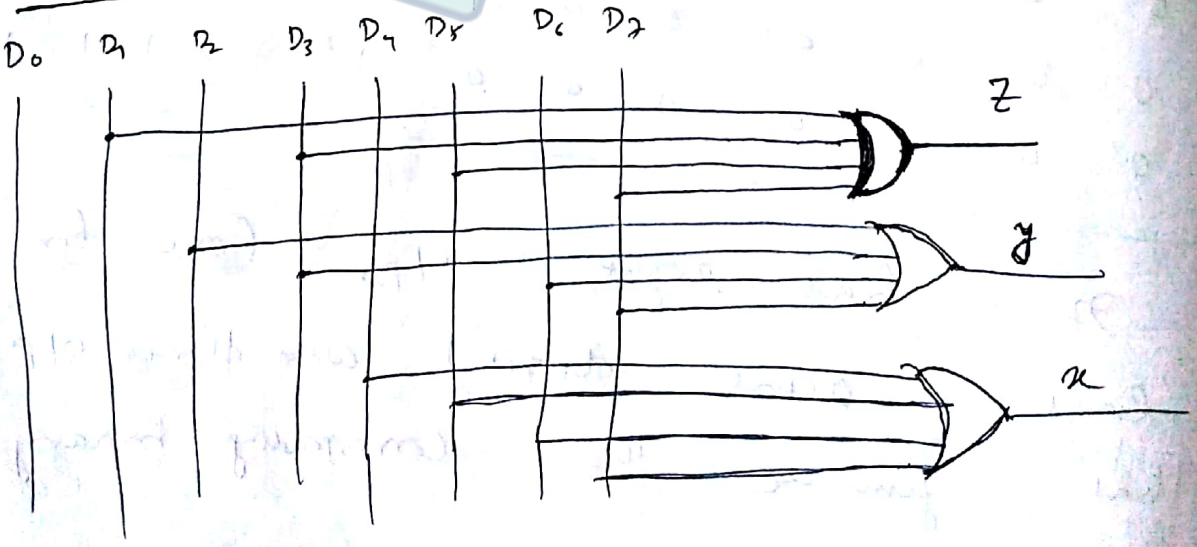
→ " " x → 4, 5, 6, 7.

∴ $Z = D_1 + D_3 + D_5 + D_7$

$y = D_2 + D_3 + D_6 + D_7$

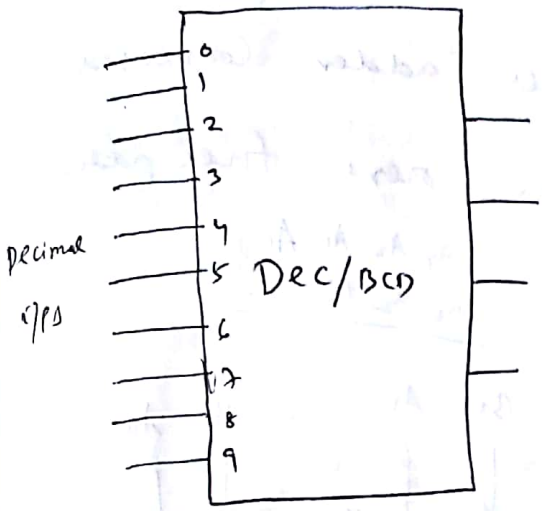
$x = D_4 + D_5 + D_6 + D_7$

Octal i/p



Decimal to BCD encoder

This type of encoder has 10 I/Ps - one for each decimal digit and 4 O/Ps corresponding to BCD code as shown in figure.



Decimal I/Ps		Binary Coded Decimal (BCD)			
		A_3	A_2	A_1	A_0
D_0	0	0	0	0	0
D_1	1	0	0	0	1
D_2	2	0	0	1	0
D_3	3	0	0	1	1
D_4	4	0	1	0	0
D_5	5	0	1	0	1
D_6	6	0	1	1	0
D_7	7	0	1	1	1
D_8	8	1	0	0	0
D_9	9	1	0	0	1

Note:- To display '5', at I/P any give '1' at '5' for rest pins give '0'.

$$A_3 = D_8 + D_9$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

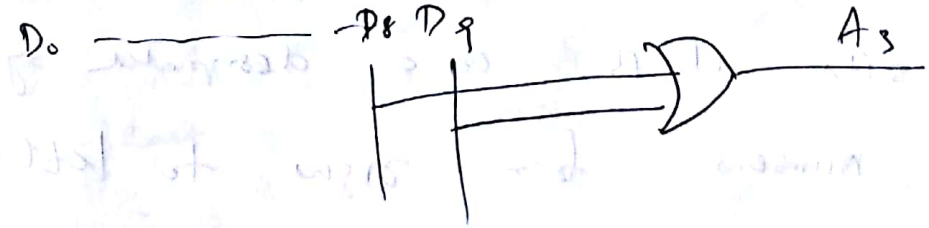
$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_0 = D_1 + D_3 + D_5 + D_7 + D_9$$

Ckt Diagram

Similarly to previous page with OR gates,

Ex:-

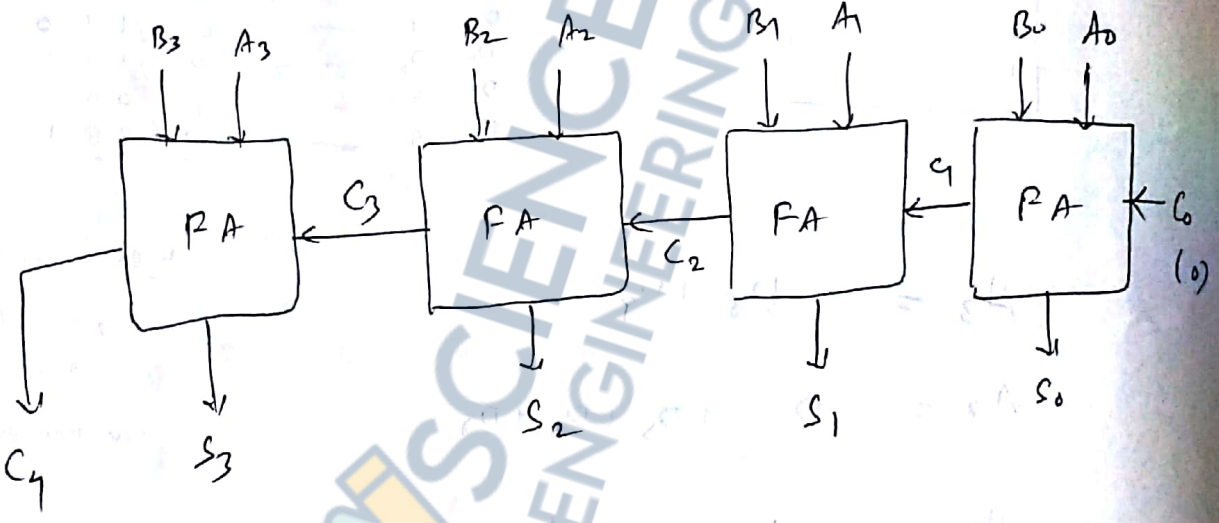


Similarly A_2, A_1, A_0.

Binary Adder / or / Binary Parallel Adder

A binary adder is a digital circuit that produces the arithmetic sum of two binary numbers. It can be constructed with full adders (FA) connected in cascade, with the o/p carry from each full adder connected to the i/p carry for the next full adder on the chain.

$A_3\ A_2\ A_1\ A_0$
$+ B_3\ B_2\ B_1\ B_0$



Ex:- 1 0 1 1
+ 0 0 1 1
0 1 1 1 0

The above figure shows interconnection of 4 full adders ckt to provide a 4 bit binary ripple carry adder.

→ The Augend bits of A and the addend bits of B are designated by subscript numbers from right to left,

With subscript 0 denoting the least significant bit.

The Carries are connected in a chain through full adders. The O/P Carry to the adder Co and it ripples through the full adder to the O/P Carry C_4. The S O/Ps generate the required Sum bits.

Subscript i: 3 2 1 0

i.e like $\begin{pmatrix} C_3 & C_2 & C_1 & C_0 \\ A_3 & A_2 & A_1 & A_0 \\ B_3 & B_2 & B_1 & B_0 \end{pmatrix}$

I/P Carry → 0 1 1 0 ← C_i

Augend → 1 0 1 1 ← A_i

Addened → + 0 0 1 1 ← B_i

Sum → 1 1 1 0 ← S_i

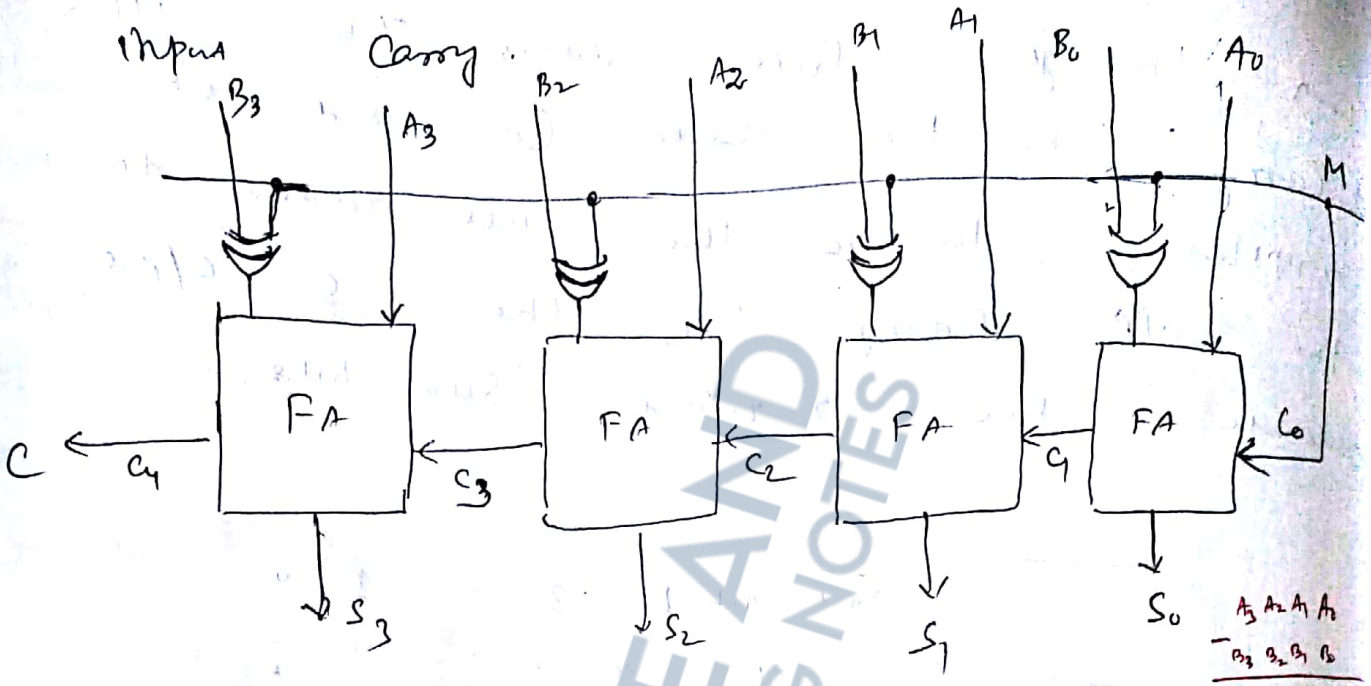
O/P Carry → 0 1 1 1 ← C_{i+1}

Binary Subtractor:-

The Subtraction A−B can be done by taking 2's Complement of B and adding it to A.

The 2's Complement can be obtained by taking 1's Complement and adding '1' to the least significant pair of bits.

→ The 1's complement can be implemented with inverters and a '1' can be added to the sum through the input Carry.



$A_3\ A_2\ A_1\ A_0$
$B_3\ B_2\ B_1\ B_0$

⇒ Prg:- 4 bit Adder Subtractor.

→ The circuit for subtracting A − B consists of an adder with inverters placed between data i/p B and corresponding input of the full adder.

∴ when M = 1, i/p to all the Ex-OR gates is 1.

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

Complement

We observe that if one i/p is '1' in ex-or gate the o/p Y is complement of the other i/p.

i.e. if B = 1, $Y = \bar{A}$

→ So here ex-or gate behave as NOT gate when one I/P is maintained 1 (M = 1) i/ps

→ The I/P Carry Co must be equal to '1' while performing subtraction, because 2's Compliment is 1's Complement + 1.

→ Thus the operation performed be comes, A, plus 1's Complement of B, plus 1. This is equal to A plus the 2's Complement of B.

→ The addition & Subtraction operations can be Combined into one ckt with one Common binary adder.

→ This is done by including an ex-or gate with each full adder.

→ when M = 0, the ckt is an adder.

✓ when M = 1, " " " Subtractor.

→ ✓ when M = 0, if one I/P = 0 for one of ex-or gate i/p same as other i/p. ex-or gate the O/P (Y = A). (if B = 0)

A	B	Y	
0	0	0	→ Y = A
0	1	1	
1	0	1	→ Y = A
1	1	0	

→ When M = 0, B ⊕ 0 = B. And also if M = 0, $C_0 = 0$.

→ This act as a binary parallel adder.

→ When M = 1, B ⊕ 1 = $\bar{B}$ and $C_0 = 1$. The complemented B i/p inputs are and a '1' is added through i/p Carry (C_0).

→ Thus the CKt performs the operation A plus 2's Complement of B.

Ex:- (i) 7 → Minuend, −5 → Subtrahend, 2

2's Complement of 5: 5 → 0101, 1's Complement → 1010, 2's " → 1010 + 1 = 1011

Add → 7 + 2's Complement of 5

0111 + 1011 = 10010 → discard → (+2)

(ii) 5 − 7 = −2

7 → 0111, 1's Complement → 1000, 2's " → 1001

Add +5 with 2's Complement of 7

0101 + 1001 = 1110 → (−2)

Check whether it is (−2)? → +2 → 0010, 1's → 1101, 2's → 1101 + 1 = 1110

For Computer it is (−2) → in 2's Complement.

For Human being, we again find 2's Complement of the addition & assign a −ve sign. [1110 → 1's → 0001 → 2's → 0010 → −(010) = −2]

Carry Lookahead Adder (CLA)

Parallel Binary Adder

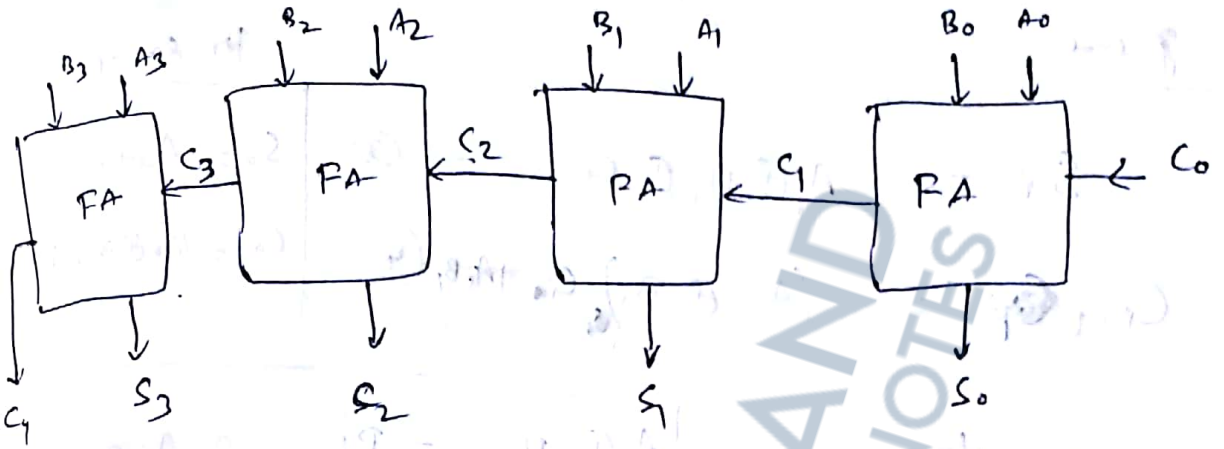


Fig 1:- 4 bit binary adder FA → Full Adder.

In a 4 bit binary adder, Consider output S_3 in fig 1. Inputs A_3 & B_3 are available as soon as i/p signals are applied to the adder.

But, input Carry C_3 does n't settle to its final value until C_2 is available from the previous stage. Similarly, C_2 has to wait for C_1 & so on down to C_0.

Thus, only after the Carry propagates and ripples through all stages will the last o/p S_3 and Carry C_4 settle to their final correct value.

Consider a full adder which adds three bits. (A, B, Cin).

The expression for

$$Sum = A \oplus B \oplus C_{in} \quad \text{—— (1)}$$

$$Carry = (A \oplus B) C_{in} + AB \quad \text{—— (2)}$$

In general,

$$S_i = A_i \oplus B_i \oplus C_i \quad \text{—— (3)}$$

$$C_{i+1} = (A_i \oplus B_i) C_{in} + A_i B_i \quad \text{—— (4)}$$

For example

$$S_0 = A_0 \oplus B_0 \oplus C_0$$

$$C_1 = (A_0 \oplus B_0) C_0 + A_0 B_0$$

If we take

$$A_i \oplus B_i = P_i \;, \quad A_i B_i = G_i$$

Eqn (3) becomes,

$$S_i = P_i \oplus C_i \quad \text{—— (5)}$$

$$C_{i+1} = P_i C_i + G_i \quad \text{—— (6)}$$

Note:- From eqn (3) & (4) S_i & C_{i+1} can be found.

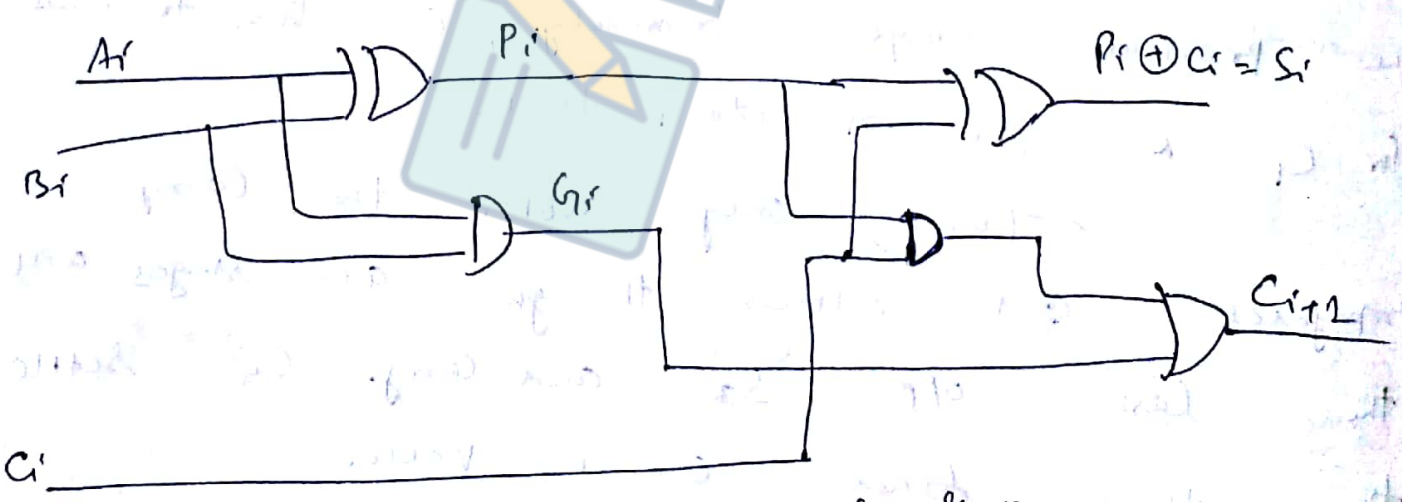


Fig:-2: Full adder with P & G shown

If there are four full adders in the adder, the O/P carry C_4 would have $2 \times 4 = 8$ gate levels from C_0 to C_4

because the signal from the i/p carry 'Ci' to o/p carry Ci+1, propagates through 'AND' gate & 'OR' gates which constitutes two gate levels. For an n-bit adder, there are '2n' gate level for carry to propagate from i/p to o/p.

→ An obvious solution for reducing the carry propagation delay time is to employ faster gates with reduced delays. However, physical circuits have a limit to their capability.

→ Another solution is to increase the equipment complexity in such a way that the carry delay time is reduced. One of this type cut is Carry Look ahead Adder.

From eqn ⑤ & ⑥.

$$S_i = P_i' \oplus C_i$$

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

Where G_i is called a carry generate

and it produces a Carry of 1 when both A_i & B_i are 1, regardless of i/p Carry C_i.

'P_i' is called Carry Propagate because it is the term associated with propagation of the Carry from C_i to C_{i+1}.

Now we can write Boolean function for Carry outputs of each stage and substitute for each 'C_i' its value from previous equations.

Let C_0 = i/p Carry [It is generally taken '0']

From eqn ⑥

$C_1 = G_0 + P_0 C_0$ — ⑦ [$i = 0$ in eqn (6)]

$C_2 = P_1 C_1 + G_1 = G_1 + P_1 C_1$ [$i = 1$ in eqn ⑥]

$= G_1 + P_1 (G_0 + P_0 C_0)$ [using eqn (7)]

$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$ — ⑧

$C_3 = G_2 + P_2 C_2$ [$i = 2$ in eqn ⑥]

$C_3 = G_2 + P_2 [G_1 + P_1 G_0 + P_1 P_0 C_0]$ → [using eqn 8]

$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$ — ⑨

$C_4 = G_3 + P_3 C_3$ [$i = 3$ in eqn (6)]

$$C_4 = G_3 + P_3(G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0) \quad (8)$$

(using eqn 9)

$$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0 \quad (10)$$

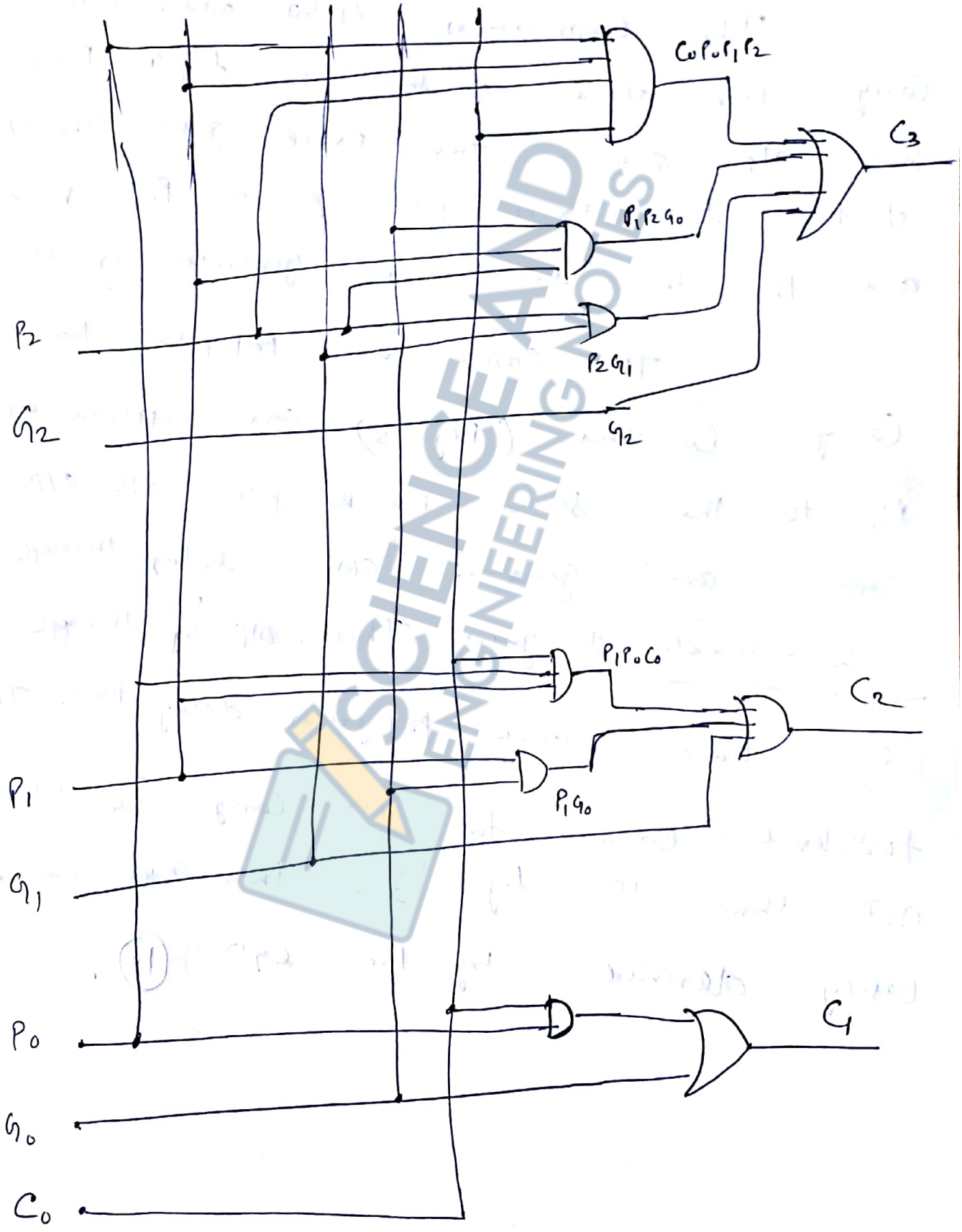


Fig 3: Logic diagram of Carry Lookahead Generator.

Note that C_3 does not have to wait for C_2 and C_1 to propagate; in fact C_3 is propagated at the same time as C_1 & C_2.

The Construction of 4 bit adder with Carry lookahead is shown in fig 4. Each Sum o/p requires two Ex-OR gates. The o/p of the first Ex-OR gate generates P_i variable, and the the AND gate generates G_i variable.

The Carries are propagated through Carry lookahead (Fig 3) and applied as i/p to the second Ex-OR gate. All o/p Carries are generated after delay through 2-level gates. Thus, o/p 'S_1' through 'S_3' have equal propagation delay times. The two-level circuit for o/p Carry 'C_4' is not shown in fig '3'. This exp can be easily derived by the eqn (10),

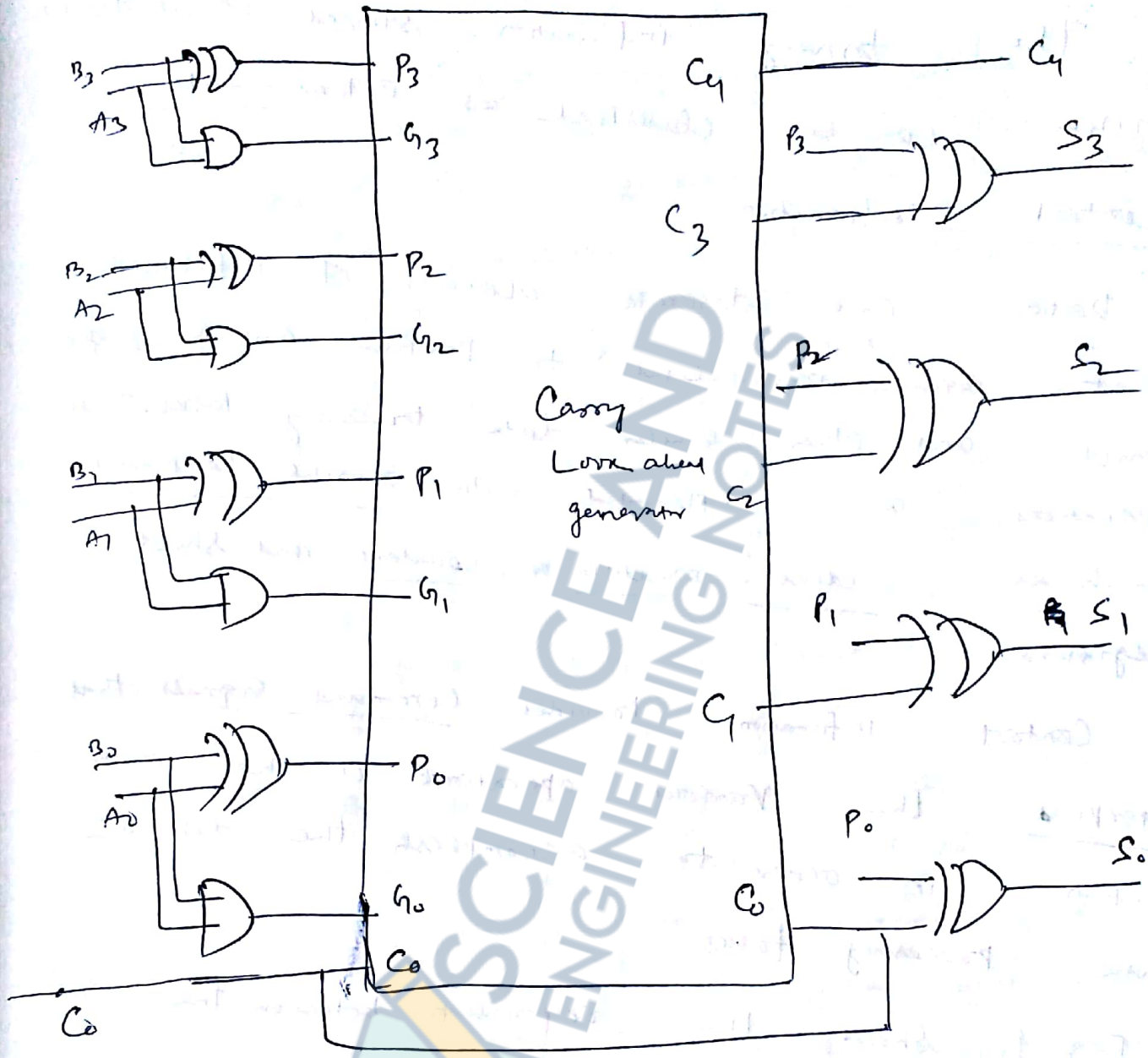


Fig 4:- 4 bit Adder with carry lookahead.

The eqn of S_1 can be found from eqn (5).

$$S_0 = P_0 \oplus C_0$$

$$S_1 = P_1 \oplus C_1$$

$$S_2 = P_2 \oplus C_2$$

$$S_3 = P_3 \oplus C_3$$

Binary Multiplier

→ Multiplication of binary numbers is performed in the same way as in decimal numbers.

→ The multiplicand is multiplied by each bit of the multiplier starting from the least significant bit.

→ Each such multiplication forms a partial product. Successive partial products are shifted one position left. The final product is obtained from the sum of the partial products.

Ex: 2 bit binary multiplier.

$B \rightarrow B_1\ B_0$ ← (Multiplicand)

$\times A \rightarrow \times A_1\ A_0$ ← (Multiplier)

$A_0B_1 \quad A_0B_0$

$A_1B_1 \quad A_1B_0$

$C_3 \quad C_2 \quad C_1 \quad C_0$ ← (Product)

→ The first partial product is formed by multiplying A_0 by B_1B_0. The multiplication of 2 bits such as A_0 and B_0 produces a 1 if both bits are 1; otherwise it produces a 0. This is identical to an AND operation.

→ Therefore, the partial product is performed by ~~multiplying~~ the AND gates as shown in the diagram.

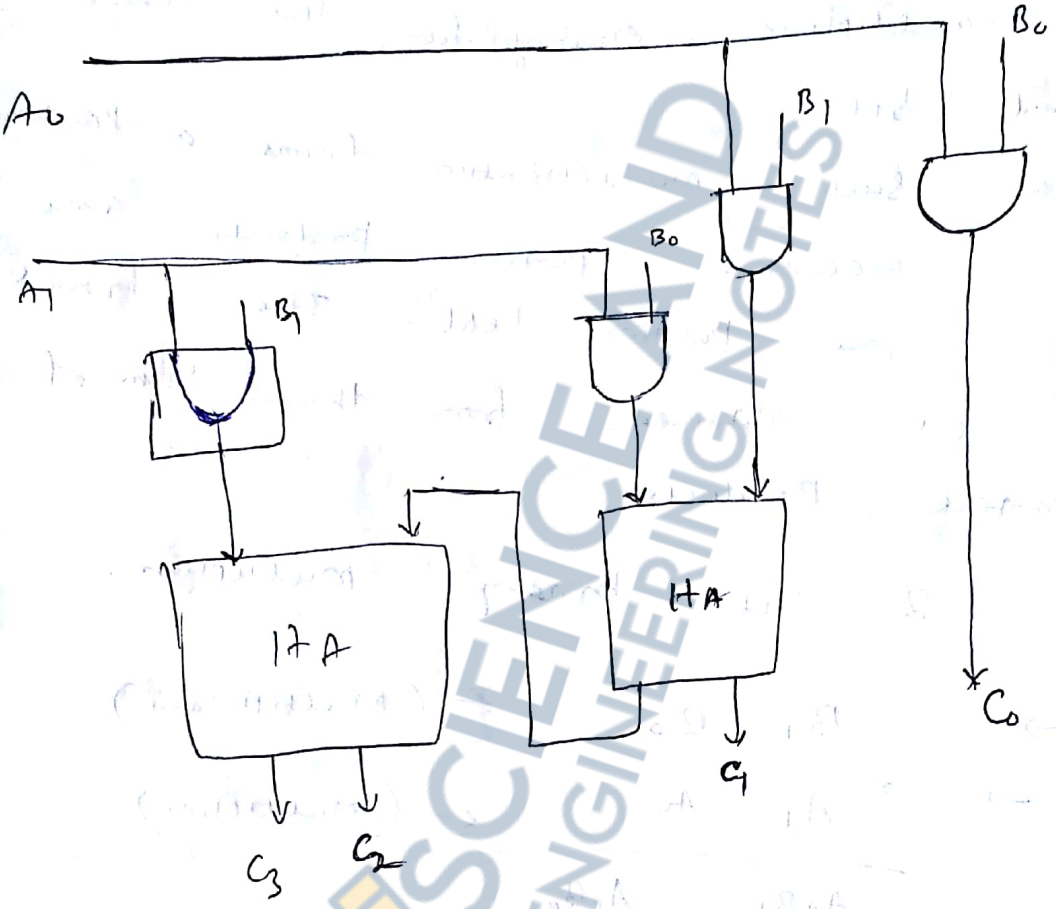


Fig:- 2bit by 2-bit Binary multiplier.

→ The second partial product is formed by multiplying A1 by B1 & B0 and shifted one position to the left.

→ The two partial products are added with two Half Adders (HA) circuits.

→ Note that the least significant bit of the product does not have to go through

an adder since it is formed by the 113 o/p of the first AND gate.

→ A combinational circuit binary multiplier with more bits can be constructed in a similar fashion.

Ex:-

$B_3\ B_2\ B_1\ B_0$ ← multiplicand

× $A_2\ A_1\ A_0$ ← multiplier

0	0					
0	1					
1	1	1	1			
1	1	0	1	0	0	1

B_3A_0 B_2A_0 B_1A_0 B_0A_0

0 0 B_3A_1 B_2A_1 A_1B_1 B_0A_1

0 B_3A_2 B_2A_2 B_1A_2 B_0A_2

C_1 C_0

So we have $C_0 \rightarrow A_0B_0$

$C_1 \rightarrow A_0B_1 + A_1B_0$

- - - - - - - and so on.

- - - - - - -

→ A bit of the multiplier is ANDed with each bit of the multiplicand in as many levels as there are bits in the multiplier.

→ The binary o/p of the AND gates is added with the partial product in each level of a new partial product. The last level

Produces the product.

Ex:- Consider a multiplier circuit that multiplies a binary number of 4 bits by a number of 3 bits.

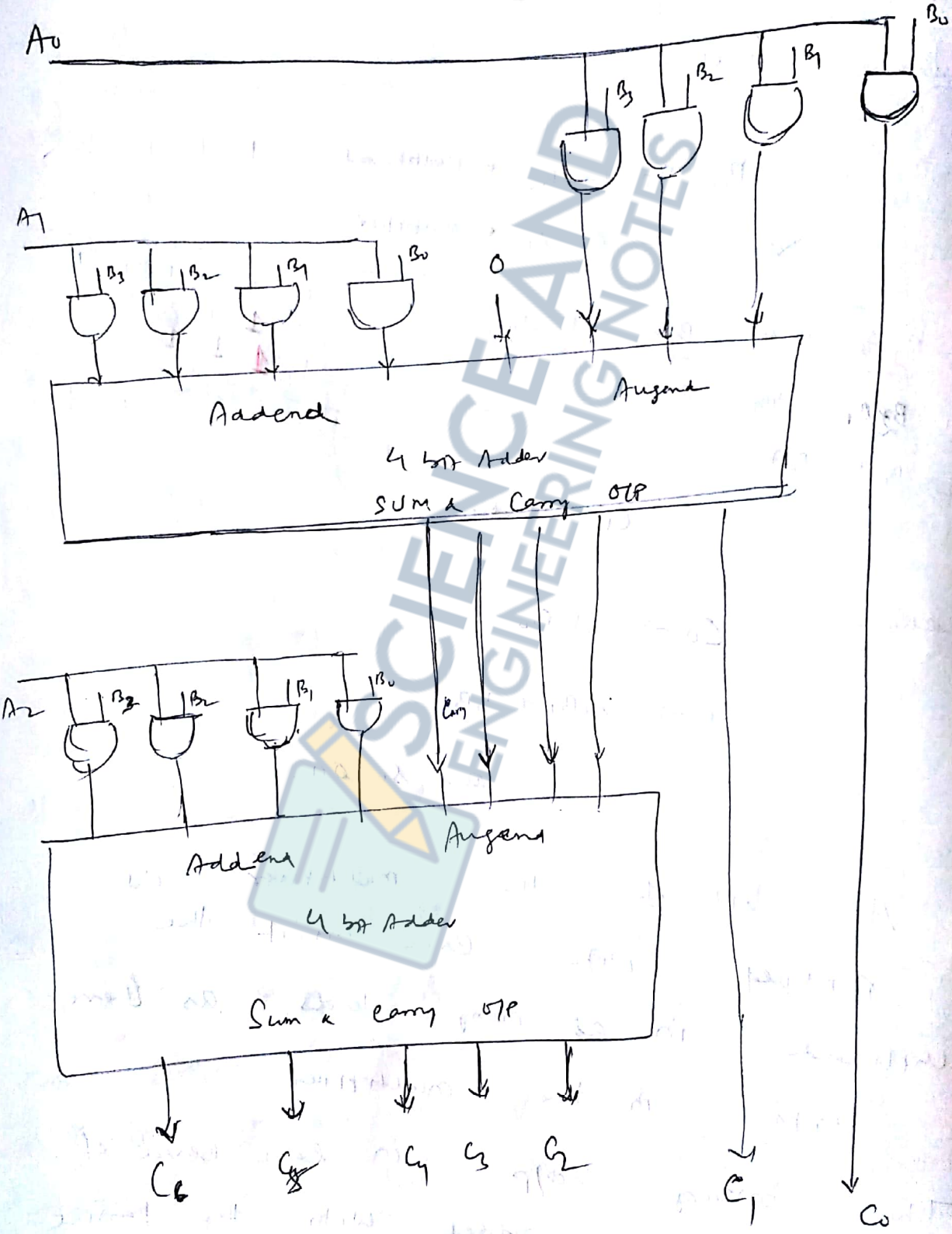
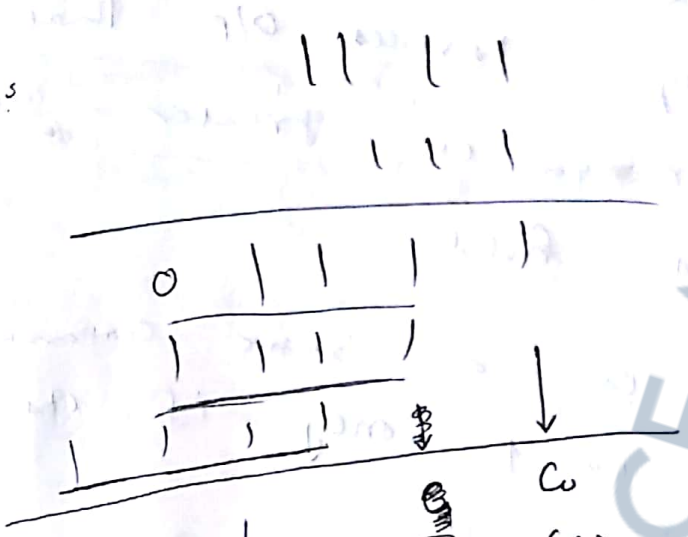


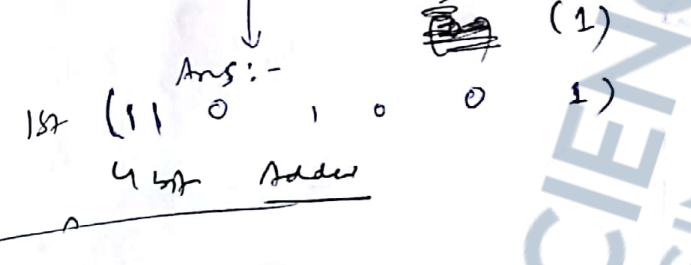
fig: 4 bit by 3 bit Binary multiplier.

→ So the multiplicand $B_3 B_2 B_1 B_0$ and the 115
multiplier $A_2 A_1 A_0$ are multiplied. We
need 12 AND gates & two 4 bit adders
to produce a product of seven bits.

Ex:-

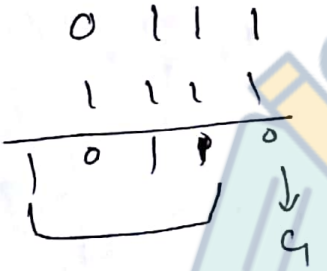


→ 15 × 7 = 105



Ans:- (1101001)

1st 4 bit adder



2nd 4 bit adder

1011
1111
10101
$C_4 C_3 C_2 C_1 C_0$

Binary $1101001 \rightarrow (105)_{decimal}$

$(64+32+8+1 = 105)$

Magnitude Comparator

→ A Comparator is a logic circuit used to compare the magnitude of 2 binary numbers. Depending on the design, it may either simply provide an o/p that is active (goes HIGH) when the two numbers are equal or additionally provides o/p that signify which number is greater & when equality does not hold.

→ X-NOR gate is a basic comparator, because its o/p is 1 only if its two bits are equal.

A	B	A ⊙ B
0	0	1
0	1	0
1	0	0
1	1	1

→ Suppose $A \rightarrow A_3\ A_2\ A_1\ A_0$

$B \rightarrow B_3\ B_2\ B_1\ B_0$

$A = B$, If $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$ & $A_0 = B_0$

i.e Equality = ~~$(A_3 \odot B_3)(A_2$~~ $(A_3 = B_3)$ && $(A_2 = B_2)$ && $(A_1 = B_1)$ && $(A_0 = B_0)$

$= (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$

1 bit Magnitude Comparator

A_0	B_0	$A<B$	$A=B$	$A>B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Let the 1 bit number $A = A_0$ &

$B = B_0$.

Mathematically

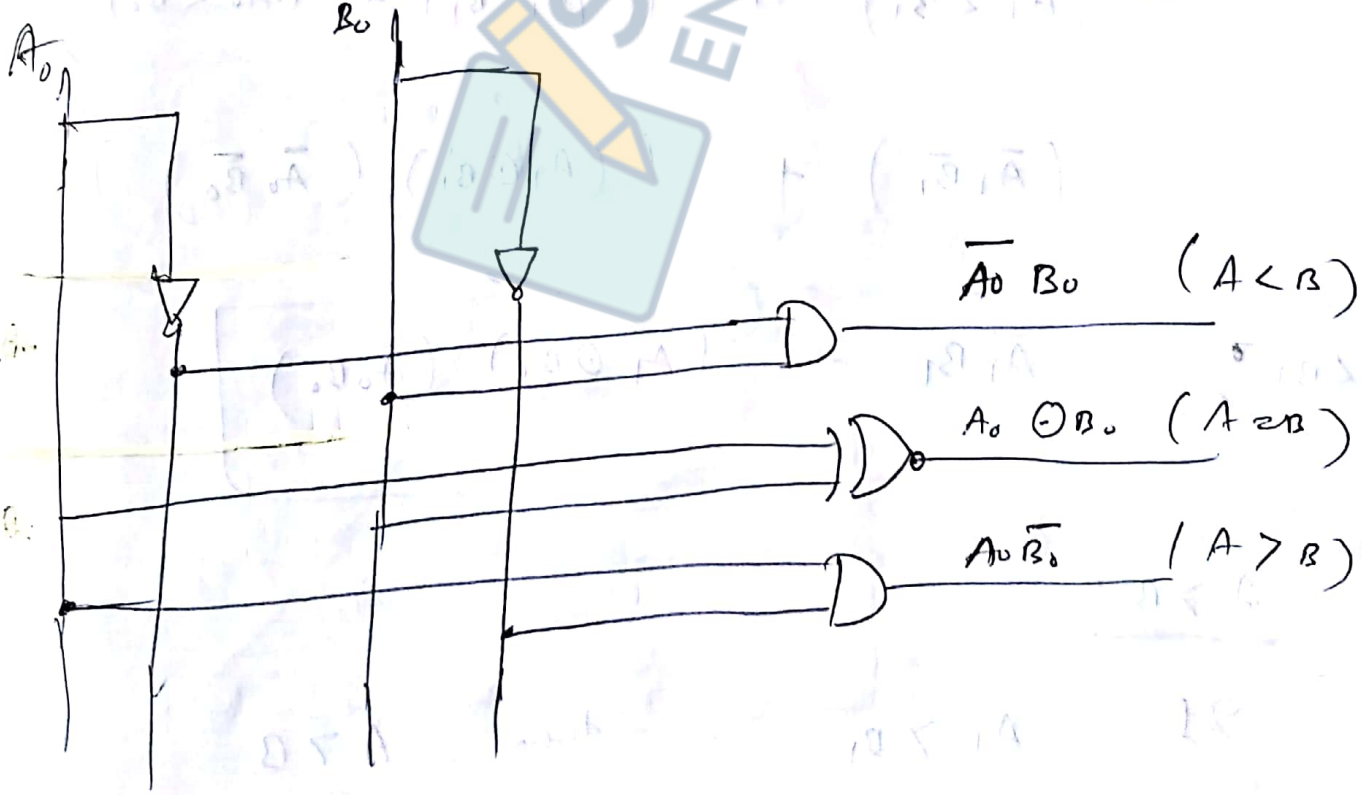
If $A_0 = 1$ && $B_0 = 0$, $A > B$, $A_0\bar{B_0}$

If $A_0 = 0$, && $B_0 = 1$, $A < B$, $\bar{A_0}B_0$

If $A_0 = 0 = B_0$ && $A_0 = 1 = B_0$, $A = B$, ~~$\bar{A_0}\bar{B_0} + A_0B_0$~~

$\bar{A_0}\bar{B_0} + A_0B_0$

$= A_0 \odot B_0$



The Logic for a 2bit Magnitude Comparator:-

Let the two 2bit numbers are A & B

$A = A_1 A_0$ & $B = B_1 B_0$.

1. If $A_1 = B_1$ & $A_0 = B_0$, $A = B$

Logically $(A_1 = B_1)$ && $(A_0 = B_0)$

$(A = B) = (A_1 \odot B_1)(A_0 \odot B_0)$

2. $A < B$

If $A_1 < B_1$, $A < B$.

or If

$(A_1 = B_1)$ then $(A_0 < B_0)$

Logically

$$(A_1 < B_1) \text{ or } \left((A_1 = B_1) \text{ && } (A_0 < B_0) \right)$$

$$= \left(\bar{A}_1 B_1 \right) + \left((A_1 \odot B_1)(\bar{A}_0 B_0) \right)$$

$$(A < B) = \bar{A}_1 B_1 + (A_1 \odot B_1)(\bar{A}_0 B_0)$$

3. $A > B$.

If $A_1 > B_1$, then $A > B$

or

If $(A_1 = B_1)$ then $(A_0 > B_0)$.

Logically

$$(A_1 > B_1) \text{ or } \left((A_1 = B_1) \text{ and } (A_0 > B_0) \right)$$

$$= (A_1 \bar{B}_1) + \left((A_1 \odot B_1)(A_0 \bar{B}_0) \right)$$

$$A > B = (A_1 \bar{B}_1) + (A_1 \odot B_1)(A_0 \bar{B}_0)$$

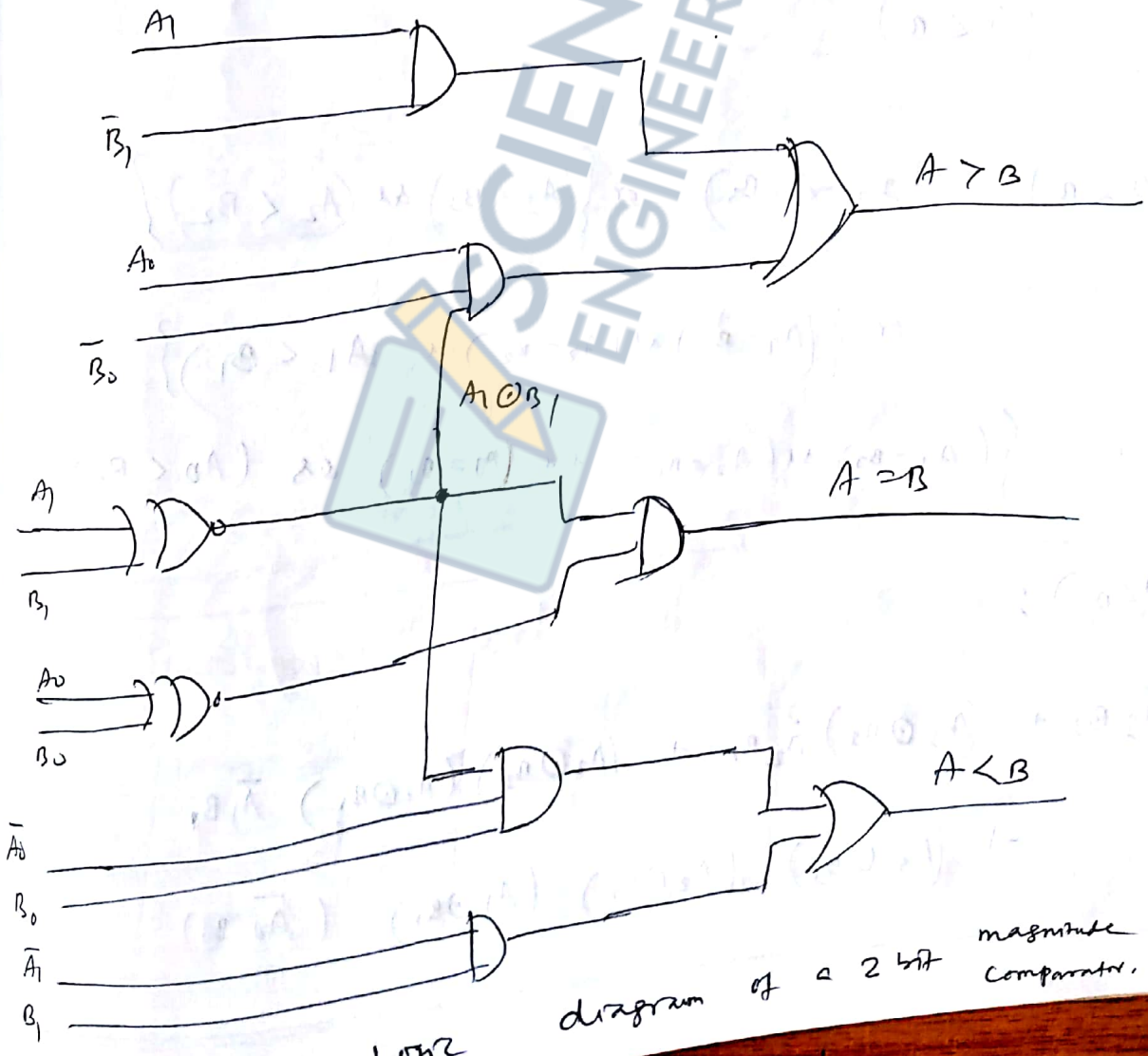


fig:- Logic diagram of a 2 bit magnitude comparator.

4 bit Magnitude Comparator

The Logic for a 4 bit magnitude Comparator.

Let the the two 4 bit numbers be $A = A_3 A_2 A_1 A_0$ & $B = B_3 B_2 B_1 B_0$.

1. $(A = B)$

$(A = B)$: $(A_3 = B_3)$ && $(A_2 = B_2)$ && $(A_1 = B_1)$ && $(A_0 = B_0)$

$$(A=B): (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$$

2. $(A < B)$

$(A < B)$: $(A_3 < B_3)$ or $\{(A_3 = B_3) \&\& (A_2 < B_2)\}$

or $\{(A_3 = B_3) \&\& (A_2 = B_2) \&\& (A_1 < B_1)\}$

or $\{(A_3 = B_3) \&\& (A_2 = B_2) \&\& (A_1 = B_1) \&\& (A_0 < B_0)\}$

$(A < B)$:

$$\bar{A}_3 B_3 + (A_3 \odot B_3)\bar{A}_2 B_2 + (A_3 \odot B_3)(A_2 \odot B_2)\bar{A}_1 B_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(\bar{A}_0 B_0)$$

3) $A > B$

$A > B$: $(A_3 > B_3)$ or $\{(A_3 = B_3)$ && $(A_2 > B_2)\}$

or $\{(A_3 = B_3)$ && $(A_2 = B_2)$ && $(A_1 > B_1)\}$

or

$\{(A_3 = B_3)$ && $(A_2 = B_2)$ && $(A_1 = B_1)$ && $(A_0 > B_0)\}$

I.C

$(A > B)$: $A_3 \bar{B}_3 + (A_3 \odot B_3)(A_2 \bar{B}_2) + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \bar{B}_1) + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1) A_0 \bar{B}_0$

→ IC 7485 is a 4 bit Comparator.

Code Converters

→ Binary → Gray

→ Gray → Binary

} Discussed earlier } Page 15

Parity Bit Generators / Checkers

→ Ex-OR functions are very useful in systems requiring error detection & correction codes.

→ Binary data, when transmitted and processed, is susceptible to noise that can alter its 1s to 0s and 0s to 1s.

→ To detect such errors, an additional bit called the parity bit is added to the data bits and the word containing the data bits and parity bit is transmitted.

→ At the receiving end the number of 1s in the word received are counted and the error, if any, is detected.

→ This parity check, however, detects only single bit errors.

→ The circuit that generates the parity bit in the transmitter is called a parity generator.

→ The ckt that checks the parity in the receiver is called parity checker.

→ A parity bit, a 0 or a 1 is attached to the data bits such that the total number of 1s in the word is even for even parity and odd for odd parity.

→ The parity bit can be attached to the code group either at the beginning or at the end depending on the system design. A given system operates with either even or odd parity but not both. Such a word always contains either an even or an odd number of 1s.

→ At the receiving end, if the word received has an even number of 1s for the odd parity system or an odd number of 1s in the even parity system, that an error has occured.

→ In order to check or generate the proper parity bit in a given code word, the basic principle used is, "the modulo sum of an even numbers of 1s is always a 0 and any modulo sum of an odd number of 1s is always a 1". Therefore,

In order to check for an error, all the bits in the received words are added. If modulo Sum is a Zero for an odd parity system or a 1 for an even parity system, an error is detected.

→

EX-OR

A	B	A ⊕ B	
0	0	0	→ When even no. of 1's o/p is 0.
0	1	1	
1	0	1	
1	1	0	→

→ To generate an even parity bit, the 4 data bits are added using 3, X-OR gates. The Sum bit will be the parity bit

Ex:- To send 0101
P = 0
∴ 0 0101

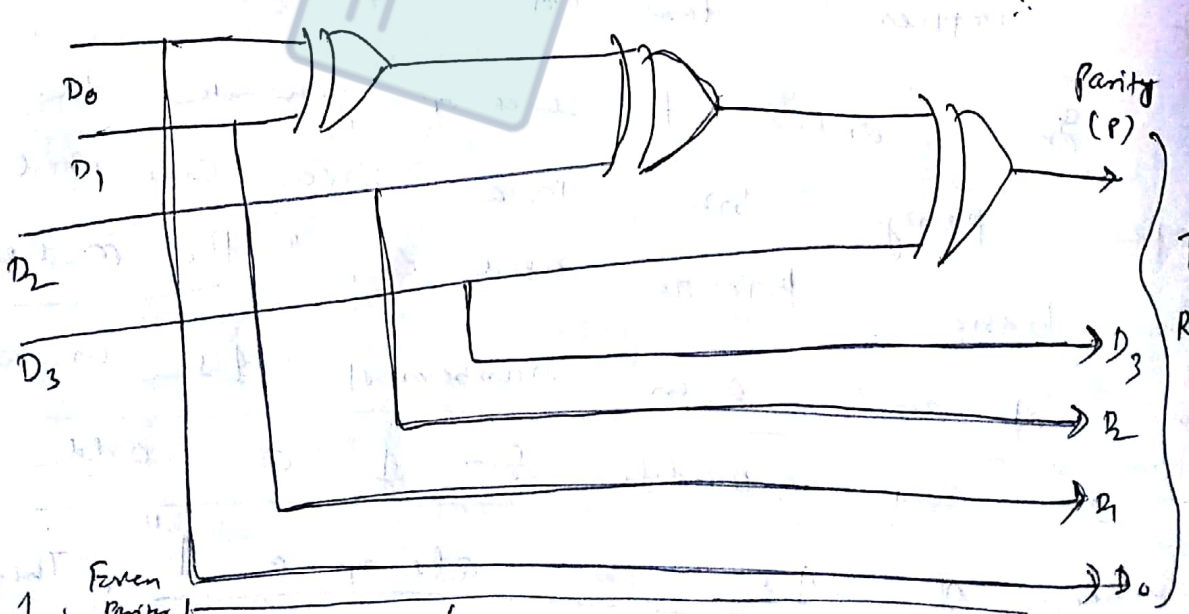
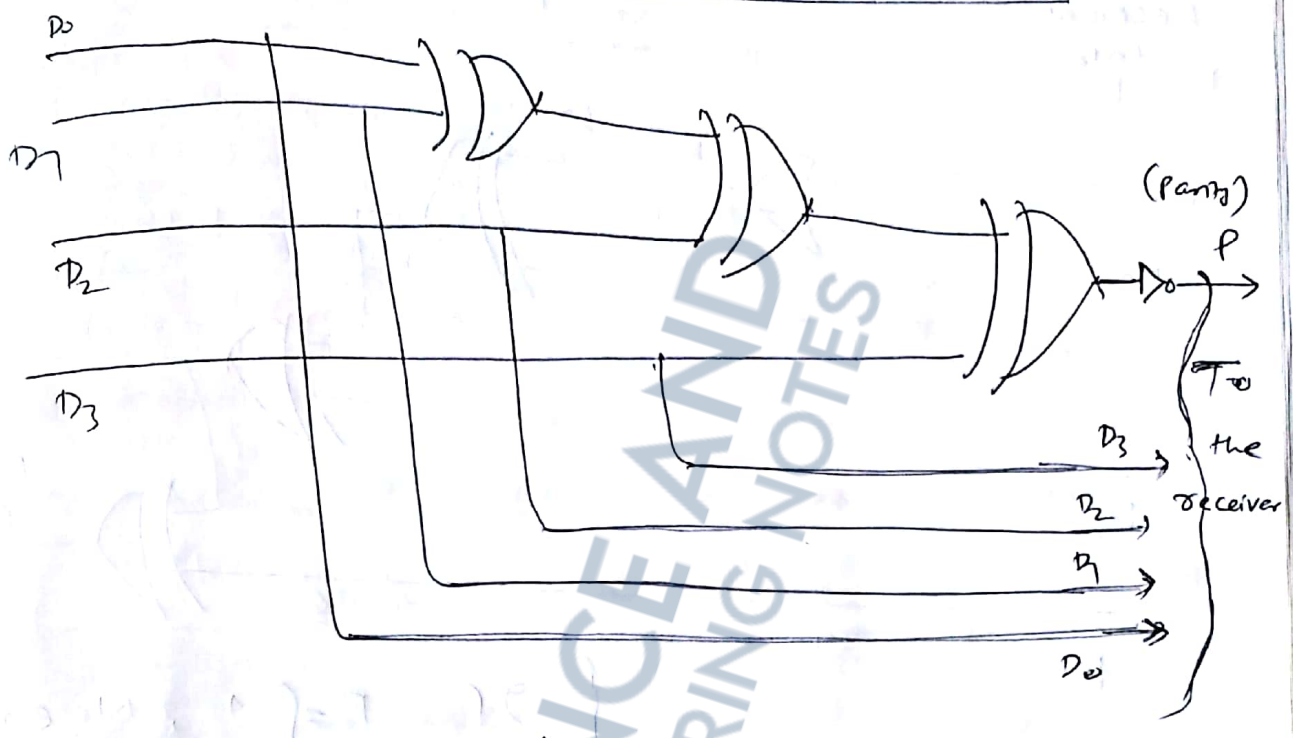


Fig 1:- Even Parity generator

Code → (P D3 D2 D1 D0) is sent.

To Generate an odd Parity bit, the four data bits are added using 3 Ex-OR gates and the Sum bit is inverted.



→ Code → (P D_3 D_2 D_1 D_0) is sent.

(Fig 2:- Odd parity generator)

Fig 3:- Shows an even parity checker.

If E = { 0, No error
1, error

Fig 3:- Even parity Checker.

Fig 4:- Shows an odd parity checker