GATE CSE NOTES

by

UseMyNotes

* **Signal**: A function that represents the variation of a physical quantity with respect to any parameter.

In Electronics, usually signal is variation of electrical quantity (current or voltage) with time.

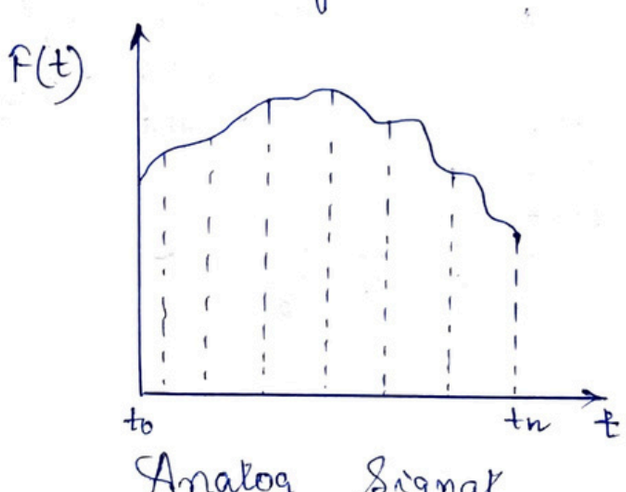
If $dI = 0$, then I is not a signal; it's DC.

* **Transducer**: Used to convert non-electrical signal to electrical signal.

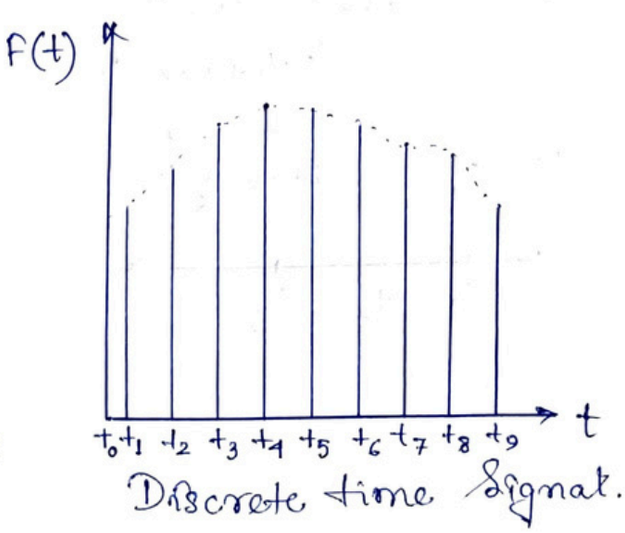
Reverse transducer: Used to convert electrical signal to non-electrical signal.

* **Analog Signal**: A continuous function that is existent for any value of independent variable.

* **Discrete Time Signal**: Signal defined for discrete intervals of time.



Analog Signal



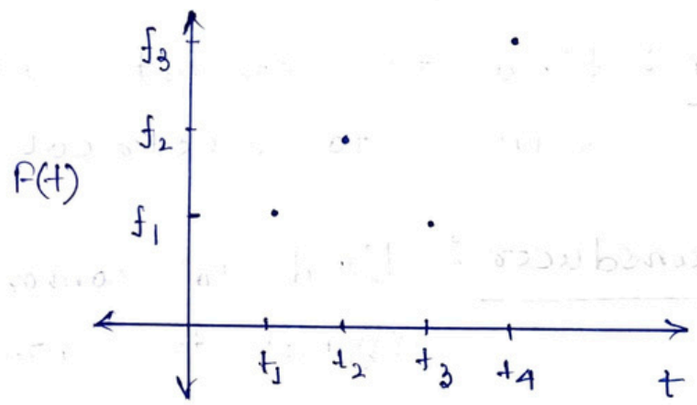
Discrete time Signal.

* Signal types :

i) Discrete time, continuous amplitude
ii) Discrete time, discrete amplitude
iii) Continuous time, continuous amplitude
iv) Continuous time, discrete amplitude.

* **Digital Signal :** Discrete time, discrete amplitude signal.

Signal can take certain values due to discretisation of amplitude.

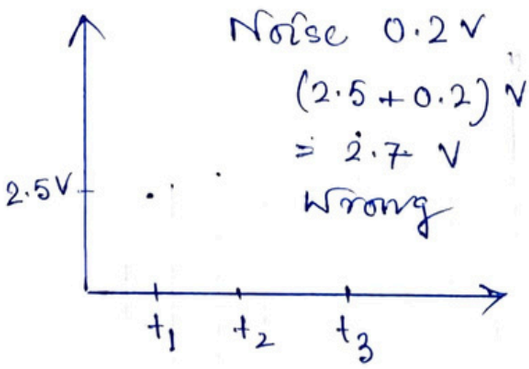


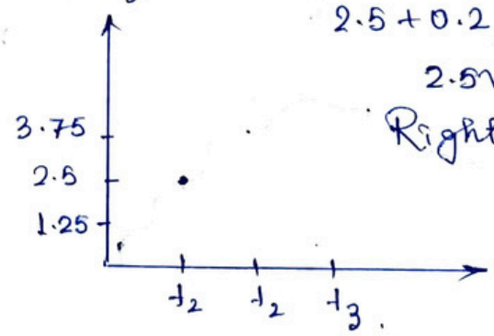
More the number of levels, lesser the error.

* **Need of Digital Signal :** Digital signal is used to minimise the effect of noise.

Transmitter ⟶ Receiver.

(Noise addition)





* Digital System → Sub System → Modules
↓
Basic units (Gates) → Circuits (Transistor, circuitry)

* Advantages of Digital System:

i) Noise Immunity
ii) Uses less bandwidth
iii) Encryption.
iv) Efficiency for long distance transmission

* Switch & Bits:

1 switch 2 Level
2 switches 4 Levels

- m switches 2^m levels. $[l = 2^m]$.
- bits - 0 & 1. used to represent switch.

* Boolean Algebra: Set of rules, used to simplify the given logic expression without changing functionality.

Used when number of variables are less.
For more variables, we use K-Map.

The final simplified result in Boolean Algebra is not always same (drawback).

* Rules:

i) Complement : $(A')' = A$

ii) AND :
$$A \cdot A = A$$
$$A \cdot 0 = 0$$
$$A \cdot 1 = A$$
$$A \cdot A' = 0$$

iii) OR :
$$A + A = A$$
$$A + 0 = A$$
$$A + 1 = 1$$
$$A + A' = 1$$

iv) Distributive :
$$A \cdot (B + C) = A \cdot B + A \cdot C$$
$$* \; A + (B \cdot C) = (A + B) \cdot (A + C).$$

$$\begin{cases} A + A'B = A + B \\ \bar{A} + AB = \bar{A} + B. \end{cases}$$

v) Commutative :
$$A + B = B + A$$
$$A \cdot B = B \cdot A$$

vi) Associative :
$$(A \cdot B) \cdot C = A \cdot (B \cdot C).$$

vii) Priority : NOT
AND
OR
↑ More

viii) De Morgan's Law :
$$\overline{(A + B)} = A' \cdot B'$$
$$\overline{(A \cdot B)} = A' + B'$$

Eg. 1. $Y = BAC' + B'AC' + BC'$

$= AC'(B + B') + BC'$

$= AC' \cdot 1 + BC'$

$= AC' + BC'$

$= (A + B)C'$

2. $AB + AB'$

$= A(B + B')$

$= A \cdot 1.$

$= A$

3. $AB + AB'C + AB'C'$

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

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

$= AB + AC + AB'C'$

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

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

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

$= A(B + 1)$

$= A \cdot 1 = A.$

4. $(A + B + C)(A + B' + C)(A + B + C').$

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

$= (A + B + 0)(A + B' + C)$

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

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

$= A + BB' + BC$

$= A + BC.$

5. $(A + B)(A + B')(A' + B)(A' + B')$

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

$= A \cdot A'$

$= 0$

* Redundancy Theorem: Consensus theorem

1. Three Variables
2. Each variable repeated twice.
3. One variable is complemented.
4. Take the complemented Variable.

Eg. $Y = AB + A'C + \underline{BC}$ redundant.

$= AB + A'C.$

Proof: $(AB + A'C + BC \cdot 1)$

$= AB + A'C + BC \cdot (A + A')$

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

$= AB(1 + C) + A'C(1 + B)$

$= AB + A'C.$

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

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

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

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

* Sum of Product:

Expression of product when the function is high (1).

Minterms	A	B	C	F
m_0	0	0	0	0
m_1	0	0	1	0
m_2	0	1	0	1
m_3	0	1	1	0
m_4	1	0	0	1
m_5	1	0	1	1
m_6	1	1	0	1
m_7	1	1	1	1

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

[SOP Form].

Canonical/Standard SOP Form (derived from Truth table).

- Minterm – $A\bar{B}C$, $\bar{A}\bar{B}C$ etc.

$$F(A,B,C) = m_2 + m_4 + m_5 + m_6 + m_7$$

$$= \Sigma m\,(2,4,5,6,7)$$

eg: $F = \bar{A}B\bar{C} + A\bar{B}[C+\bar{C}] + AB[\bar{C}+C]$

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

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

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

$F = A + B\bar{C}.$ [Minimal SOP Form].

- Canonical form: Each minterm is having all the variables in normal or complemented form.
- Minimal form: Each minterm does not have all the variables in normal or complemented form.

Eg.

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

minimize SOP expression.

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

$\boxed{Y = B}$

Eg. $Y(A,B) = \Sigma m\,(0,2,3).$

$Y = m_0 + m_2 + m_3$

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

$= \bar{B} + AB$

$= A + \bar{B}$

* Product of Sum (POS Form):

- Used when the output is low (0).

maxterms	A	B	C	Y
M_0	0	0	0	0
M_1	0	0	1	0
M_2	0	1	0	1
M_3	0	1	1	0
M_4	1	0	0	1
M_5	1	0	1	1
M_6	1	1	0	1
M_7	1	1	1	1

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

$0 \rightarrow A$	$0 \rightarrow \bar{A}$
$1 \rightarrow \bar{A}$	$1 \rightarrow A$
POS	SOP

- maxterm – $A+B+C$, $A+B+\bar{C}$ etc.

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

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

By de Morgan's Law,

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

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

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

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

$= A + B\bar{C}$

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

Eg. Minimise POS expression.

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

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

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

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

$= \bar{B}.$

$Y = \Pi (M_1, M_2)$.

$= \Pi M(1,3)$ .. Maxterm

$Y = \Sigma m(0,2)$ Minterm.

- SOP Form — AND

  POS Form — OR

Eg.

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

$Y(A,B,C) = \Sigma m(0,2,3,6,7)$ SOP

$Y(A,B,C) = \Pi M(1,4,5)$ POS.

SOP

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

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

POS

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

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

- Minimal to Canonical:

$Y = A + B'C$ SOP

$= A \cdot 1 \cdot 1 + B'C \cdot 1$

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

$= (AB + AB')(C+C') + B'CA + B'CA'$

$= ABC + ABC' + AB'C + AB'C' + B'CA + B'CA'$.

Step 1: 3 Variables A, B, C.

Step 2: m_1 A ✓ B ✗ C ✗ m_2 A ✗ B ✓ C ✓

Step 3: $B + \bar{B} = 1$ replacement

$F = (A+B+C')(A'+C)$ — POS

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

$= (A+B+C')$

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

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

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

i) 3 variables

ii)

Term 1	Term 2
A ✓	A ✓
B ✓	B ✗
C ✓	C ✓

iii) $CC' = 0$ replacement

Eg. $f = A + B'C$.

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

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

$= ABC + ABC' + AB'C + AB'C' + AB'C + A'B'C$.

- Number of logical expressions $= 2^{2^n}$

  $n \rightarrow$ no. of variables.

* Positive & Negative logic.

- **Positive logic.**

  Higher voltage corresponds to 1.

  Lower voltage corresponds to 0.

- **Negative logic.**

  Higher voltage corresponds to 0.

Eg. logic 0 $\rightarrow -5$ V

logic 1 $\rightarrow$ 0 V } Positive logic.

* Dual form :

- Positive & Negative Logic AND Gate :

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

+ve logic

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

-ve logic

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

0 ⟶ Low
1 ⟶ High
} +ve logic.

1 ⟶ low
0 ⟶ high
} -ve logic

- Positive & Negative Logic OR Gate :

+ve logic.

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

-ve logic

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

{ +ve logic AND ≡ -ve logic OR
{ +ve logic OR ≡ -ve logic AND.

- Dual form is used to convert between +ve & -ve logic.

Eg. $A \cdot B \xrightarrow{\text{Dual}} A + B.$

$C + D \xrightarrow{\text{Dual}} C \cdot D.$

Acquire :: { https / http / ftp } :: proxy "http://<proxy>:<port>/"
/etc/apt/apt.conf

* Self Dual :

- for any logical expression, two times dual gives the same expression. In self dual expression, one time dual gives the same expression.

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

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

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

$F \equiv (F')'$

Eg. $G = A \cdot B + B \cdot C + A \cdot C$ Self dual expression.

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

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

$= AB + AAC + BC + ACC$.

$= AB + BC + AC$.

$G = G'$ (Self Dual)

- For n variables, there are $2^{2^{n-1}}$ number self duals.

* Complementing :

AND $\leftrightarrow$ OR

$\cdot \leftrightarrow +$

$0 \leftrightarrow 1$

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

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

} Using de Morgan's.

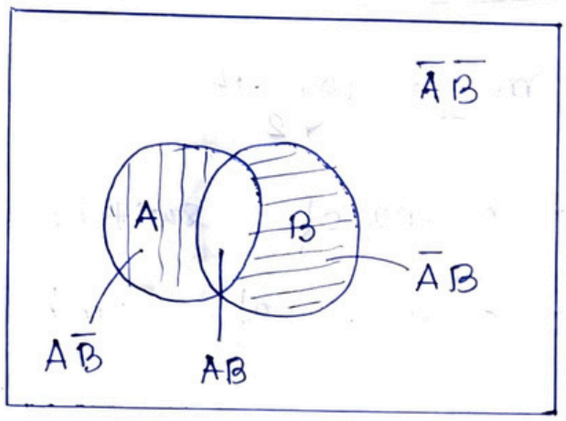
Eg. $G = A\bar{B}C + \bar{A}BC + A\bar{B}C$.

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

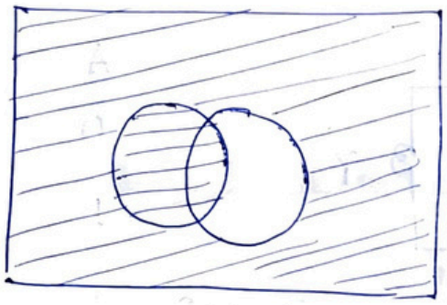
$$= \overline{(A\bar{B}C + \bar{A}BC)} \cdot \overline{(A\bar{B}\bar{C})}$$

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

* Venn Diagram.



Eg.



Minimise the SOP expression for shaded region.

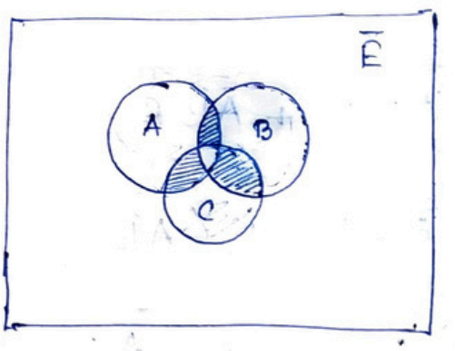
$$Y = \bar{A}\bar{B} + A\bar{B} + AB$$

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

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

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

Eg.



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

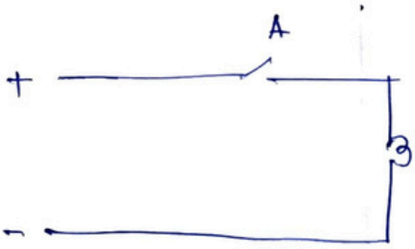
* Switching Circuits. :

- $m = \log_2 n$

$m \longrightarrow$ no. of switches required

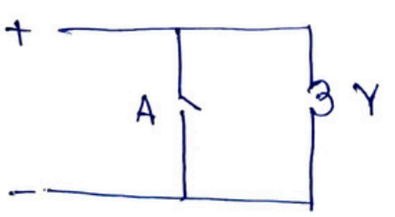
$n \longrightarrow$ no. of combinations.

- Series Switch. (A)



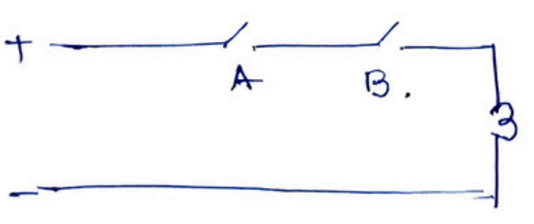
A	Y
0	0
1	1

- Parallel Switch. (NOT)

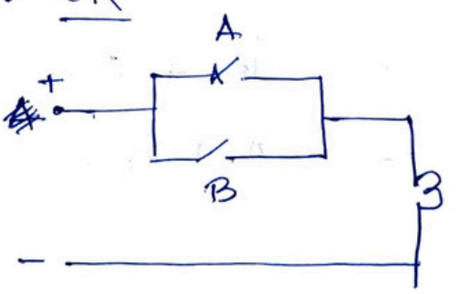


A	Y
0	1
1	0

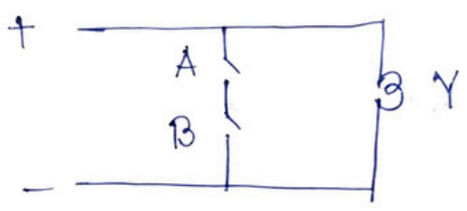
- AND



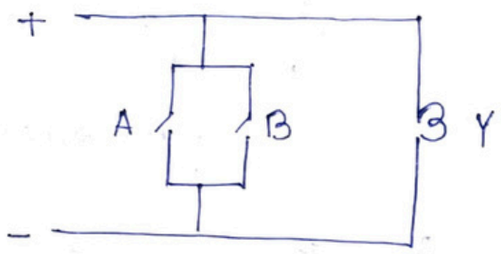
- OR



- **NAND** – $Y = \overline{AB}$



- **NOR** $Y = \overline{A+B}$

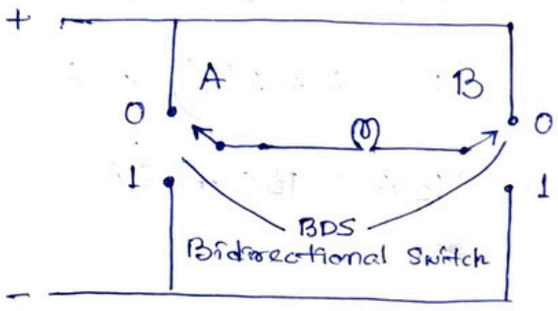


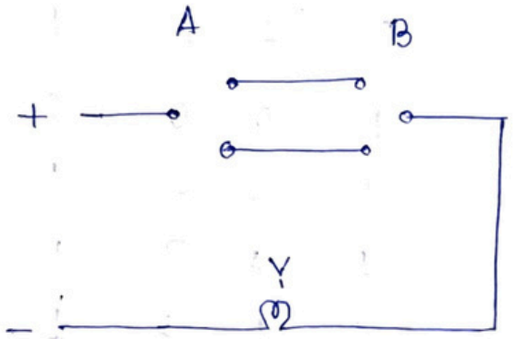
- **XOR** $Y = AB' + A'B$

$= A \oplus B$

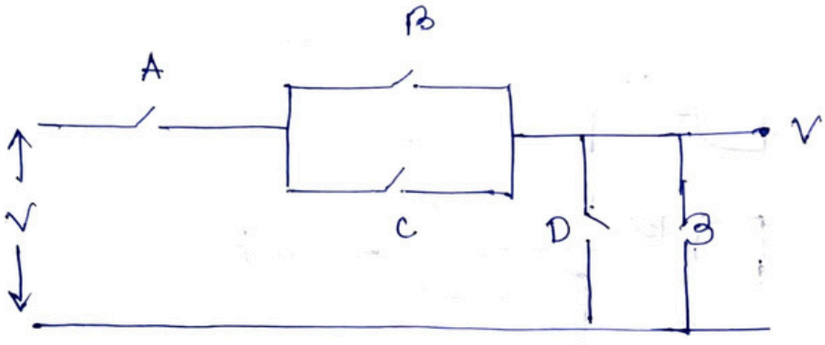
- **XNOR** $Y = AB + A'B'$

$= A \odot B$





Eg.



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

(Intuitional).

$Y = AB\bar{D} + AC\bar{D}$

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

Find ways to glow the bulb & OR them, then simplify.

* Statement Problems.

Eg. High when i) B, C true ii) A, C false iii) A, B, C true iv) A, B, C false.

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

$= BC + A'C'$

Eg. A, B, C i/p. O/p high when majority of i/p are 1.

Implement circuit.

→

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

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

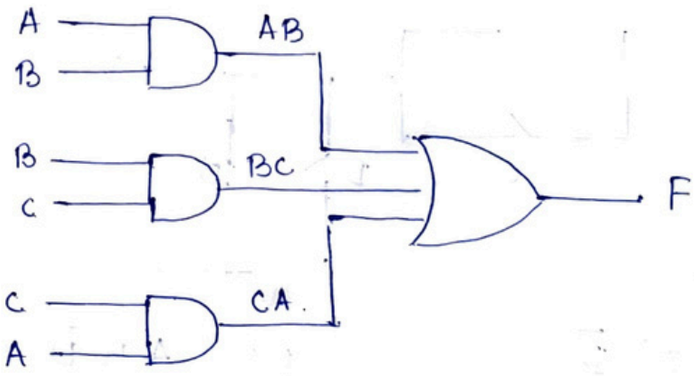
$= A'BC + AB'C + AB$

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

$= A'BC + AB + AC$

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

$= BA + BC + CA.$



* Number Systems

Defines a set of values used to represent quantity.

Name	Base/Radix (r)	(No. of distinct digits)
Binary	2	[0, 1] Bits
Octal	8	
Decimal	10	Digits —
Duodecimal	12	$(0, \ldots, r-1)$.
Hexadecimal	16	

* Binary Number System :

$(101)_2 = (5)_{10}$

- Bits – 0 & 1.
- Any number system – to decimal

ABCDE

$\rightsquigarrow A \times r^4 + B \times r^3 + C \times r^2 + D \times r^1 + E \times r$

AB.CD

$\rightsquigarrow A \times r^1 + B \times r^0 + C \times r^{-1} + D \times r^{-2}$

- MSB (Most Significant Bit)

10101

MSB

- LSB (Least Significant Bit)

10101 — LSB.

- 1 Nibble = 4 bits BCD – Hex

1 Byte = 8 bits

1 word = 16 bits = 2 bytes.

1 double word = 32 bits = 4 bytes

- Decimal → Binary.

Divide integer part by r, base & multiply fractional part by r.

Eg. $(13)_{10} \equiv (1101)_2$

Divider	Dividend	Rem.
2	13	1
2	6	0 ↑
2	3	1
2	1	1
	0	

Eg. $(25.625)_{10} \equiv 11001.101$

$0.625 \times 2 = 1.25$

$0.25 \times 2 = 0.50$

$0.50 \times 2 = 1.00$

2	25	1
2	12	0
2	6	0
2	3	1
2	1	1
	0	

* Decimal ⟶ Octal.

Eg. $(112)_{10} \longrightarrow (160)_8$

8	112	0
8	14	6
8	1	1
	0	

Eg. $(25.625)_{10} \longrightarrow (31.5)_8$

$0.625 \times 8 = 5.000$

8	25	1
8	3	3
	0	

* Decimal ⟶ Hexadecimal

Eg. $(254)_{10} \rightarrow (FE)_{16}$

16	254	14 E
16	15	15 F
	0	

Eg. $(25.625)_{10} \rightarrow 19.A$

$0.625 \times 16 = 10.00$

A

16	25	9
16	1	1
	0	

* Binary to Decimal.

Generally

$$(a_3 a_2 a_1 a_0 \cdot a_{-1} a_{-2})_r \longrightarrow$$

(r → base)

$$a_3 r^3 + a_2 r^2 + a_1 r^1 + a_0 r^0 + a_{-1} r^{-1} + a_{-2} r^{-2}$$

Eg. $(10101.11)_2 \longrightarrow$

$$1 \times 2^4 + 0 + 1 \times 2^2 + 0 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$$

$$= 16 + 4 + 1 + 0.5 + 0.25$$

$$= (21.75)_{10}$$

* Octal ⟶ Decimal.

Eg. $(57.4)_8 \longrightarrow$

$$5 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1}$$

$$= 40 + 7 + 0.5$$

$$= (47.5)_{10}.$$

* Hexadecimal ⟶ Decimal.

Eg. $(BAD)_{16} \longrightarrow$

$$11 \times 16^2 + 10 \times 16^1 + 13 \times 16^0$$

$$= 2989.$$

* Octal NS. (3 bit bin)

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111.

Octal ⟶ Bin.

$(34.75)_8 \longrightarrow$

$(011100.111101)_2$.

Bin ⟶ Octal

$(10110.11)_2 \longrightarrow (010110.110)_2$

$\longrightarrow (26.6)_8$.

Make group of 3 bits.

* Hexadecimal NS. (4 bit bin).

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A 10	1010
B 11	1011
C 12	1100
D 13	1101
E 14	1110
F 15	1111.

Hex ⟶ Bin

$(259A)_{16} \longrightarrow$

$(001001011001 1010)_2$

Bin ⟶ Hex

$(10001001.11)_2 \longrightarrow (89.C)_{16}$

Make group of 4 bits.

* Hex → Oct
  Oct → Hex

Eg. $(CAD)_{16}$ → Bin

$(110010101101)_2$

→ Oct

$(6255)_8$.

Eg. $(652)_8$ → bin → $(110101010)_2$

hex → $(1AA)_{16}$

* Binary Addition.

Eg.
$$\begin{array}{r} 1\ 1\ 0 \\ +\ \ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 1 \end{array}$$

	Sum	Carry
0+0	0	0
1+0	1	0
0+1	1	0
1+1	0	1

* Binary Subtraction.

borrow 2

Eg.
$$\begin{array}{r} 1\ 1\overset{2}{0}\ 1\ 1 \\ -\ \ 1\ 0\ 1\ 1\ 0 \\ \hline 0\ 0\ 1\ 0\ 1 \end{array} \qquad \begin{array}{r} 1\ 1\ 1\ 0 \\ -\ \ \ 1\ 1\ 1 \\ \hline 0\ 1\ 1\ 1 \end{array}$$

* Bin Multiplication.

	Product
0×0	0
1×0/0×1	0
1×1	1.

$$\begin{array}{r} 1\ 0\ 1\ 0 \\ \times\ \ \ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0 \\ 1\ 0\ 1\ 0 \\ \hline 1\ 1\ 0\ 0\ 1\ 0 \end{array}$$

```
  1 0 1 0
  x   1 1
 -------
  1 0 1 0
1 0 1 0
-------
1 1 1 1 0
```

* Binary Division.

```
          1 1 1
       -------
1 1 0 | 1 0 1 0 1 0
        1 1 0
      -------
        1 0 0 1
          1 1 0
        -------
          0 1 1 0
            1 1 0
          -------
```

* Complements.

1. r's Complement (Radix Complement).

r's complement $\rightarrow r^n - N$

r - base, n - no. of digits.

N - given number

Eg. $N = 5690$.

10's complement $= 10^4 - 5690$

$= 4310$.

Eg. $N = 1101$

2's complement $= 2^4 - 1101$

$= (16)_{10} - 1101$

$= 10000 - 1101$

$= 00011$

2. $(r-1)$'s complement (Diminished radix complement).

$(r-1)$'s comp. $= r^n - N - 1$.

$= r$'s comp. $- 1$.

No borrow operation is involved.

Eg. 7's comp. of $(5674)_8$

$= 8^4 - 5674 - 1$.

$= (4096)_{10} - 5671 - 1$

$= 10000 - 5674 - 1$

$= 7777 - 5674$

$= 2103$

$$\begin{array}{r} 7777 \\ -\ 5674 \\ \hline 2103 \end{array}$$

Eg. 8's comp. of $(5674)_8$

$= 2103 + 1$

$= 2104$.

Eg. 1's complement of 1101.

$$\begin{array}{r} 1111 \\ -\ 1101 \\ \hline 0010 \end{array} \longrightarrow \text{Ans}$$

2's comp. $= 0010 + 1$

$= 0011$.

Eg. 1's complement of $(1010)_2$.

$$\begin{array}{r} 1111 \\ -\ 1010 \\ \hline 0101 \end{array} - \text{Ans.}$$

- Shortcut for 2's complement:

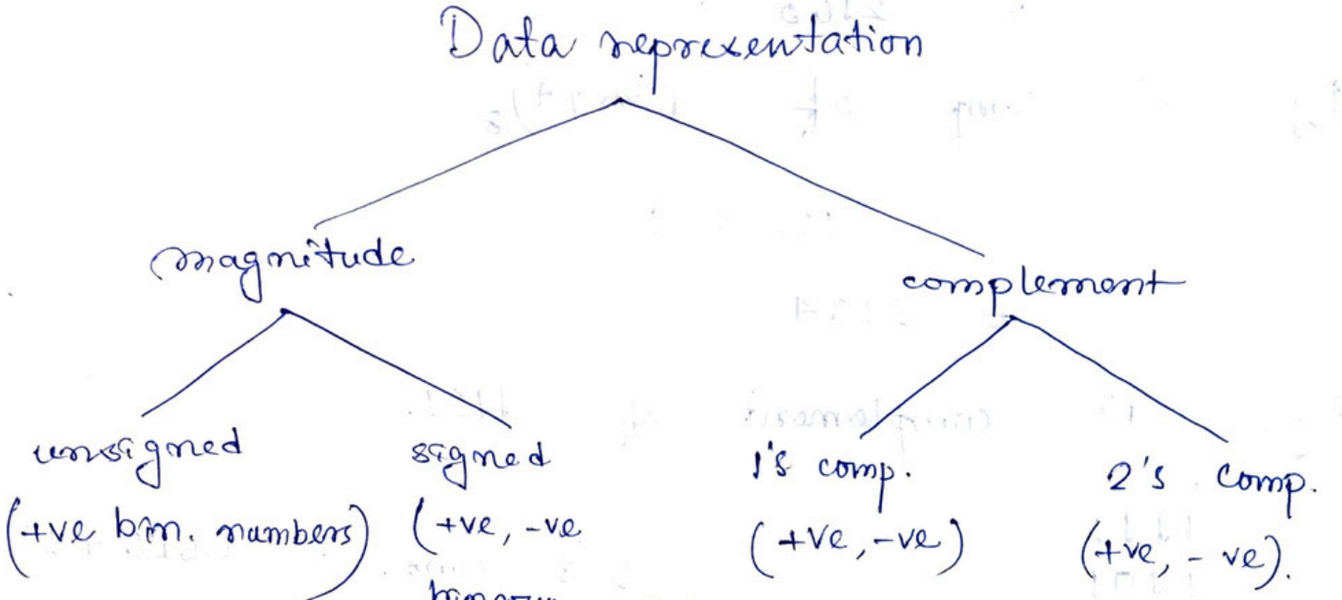
Write given number → starting from LSB copy all zeroes till first 1 → Copy first 1 → Complement all remaining bits.

Eg. 10111000 ← LSB — Step 1

01001000 — Step 2

result — Step 1.

* Data Representation using Signed Magnitude:



{ positive representation always same.

- Unsigned

110 → +6

-6 → can't represent

- Signed

+6 → 0110

-6 → 1110

Sign bit (MSB)

Sign bit	Number
0	Positive
1	Negative

} Signed Magnitude Representation.

- Range $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$ (Signed Magnitude)

$n \rightarrow$ no. of variables

- 1's complement.

$+6 \quad 0110$

$-6 \quad 1001$ ← 1's complement.

$+0 = 0000$ (+ve zero)

$-0 = 1111$ (-ve zero).

Range. $-(2^{n-1}-1)$ to $(2^{n-1}-1)$.

- 2's complement.

$+6 = 0110$

1's → 1001

$-6 = 1010$ ← +1 ← 2's

Range -2^{n-1} to $(2^{n-1}-1)$

- MSB indicates sign.

(*) Bin. Subtration using 1's comp.:

i) Convert no. to be subtracted to its 1's comp. form.

ii) Preform addition.

** iii) If final carry is 1, then add it to the result. If 0, result is negative & in the 1's complement form.

Sign bit	Number
0	Positive
1	Negative

} Signed Magnitude Representation.

- Range (Signed Magnitude) $-(2^{n-1}-1)$ to $+(2^{n-1}-1)$

$n \rightarrow$ no. of variables

- 1's complement.

$+6$ 0110

-6 1001 ← 1's complement.

$+0 = 0000$ (+ve zero)

$-0 = 1111$ (-ve zero).

Range. $-(2^{n-1}-1)$ to $(2^{n-1}-1)$.

- 2's complement.

$+6 = 0110$ → 1's → 1001 → +1 → 2's

$-6 = 1010$

Range -2^{n-1} to $(2^{n-1}-1)$

- MSB indicates sign.

(*) Bin. Subtration using 1's comp.:

i) Convert no. to be subtracted to its 1's comp. form.

ii) Preform addition.

** iii) If final carry is 1, then add it to the result. If 0, result is negative & in the 1's complement form.

Eg: $(1100)_2 - (0101)_2$.
A B

$-B = 1010$.

$A + (-B) = 1100 + 1010 = 1\,0110$

$0110 + 1 = 0111$. (Ans).

Eg. $(0101)_2 - (1100)_2$
A B

$-B = 0011$.

$A + (-B) = 0101 + 0011 = 0\,1000$

-ve.
1's comp.

$A + (-B) = 0111$.

(*) Bm. Subtraction using 2's comp.:

i) Find 2's comp. of the number to be Subtracted.

ii) Perform addition.

** iii) If final carry is generated then the result is +ve & in true form. If not generated, then is -ve & in 2's complement form.

Eg. $-(0100)_2 + (1001)_2$
B A

$-B = 1100$.

$1001 + 1100 = 1\,0101$

Result is 0101.

1011
1
1100

Eg. $(0110)_2 - (1011)_2$

A B

$-B = 0100 + 1$

$= 0101.$

$0110 + 0101 = 1011$

$1011 \xrightarrow{\text{1's}} 0100 \xrightarrow[+1]{\text{2's}} 0101$

* Condition for Overflow (2)

x & y are sign bits of two numbers

z is sign bit of result

$\bar{x}\bar{y}z + xy\bar{z} = 0$ (No overflow)

$= 1$ (Overflow).

* Codes. (group of symbols).

Classification

- Weighted (Binary, 2421, 8421)
- Non-weighted (XS-3, gray)
- Reflective code/ self-complementing (2421, XS-3)
- Sequential (8421, XS-3)
- Alpha-numeric (ASCII)
- Error detecting & correcting codes. (Hamming code).

* Binary Coded Decimal (BCD Code):

- Each decimal digit is represented by a 4-bit binary number.
- Positional weights — 8 4 2 1
- Also called 8421 code.

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

Decimal No. → BCD.

i) $(17)_{10} \longrightarrow (00010111)$

BCD → Decimal No.

i) $(10100) \longrightarrow (14)_{10}$

(00010100)

- Packed BCD – BCD representation for dec. No. > 9.
- BCD is less efficient than Binary as it uses more no. of bits.
- BCD Addition.

1. Sum ≤ 9, Final carry = 0 — Correct ans.
2. Sum ≤ 9, " " = 1 — Add (0110) to ans.
3. Sum > 9, " " = 0 — Add (0110) to ans.

Eg. $(2)_{10} + (6)_{10}$

0010 0110

= 1000 ✓ Sum < 9, FC = 0.

(Ans)

Eg. $(3)_{10} + (7)_{10}$

$\downarrow$ $\downarrow$

0011 0111

$=$ 1010 (Sum) > 9.

$\downarrow + 0110$

000 10000 BCD

$\downarrow$

$(10)_{10}$

Eg. $(8)_{10} + (9)_{10}$

$\downarrow$ $\downarrow$

1000 1001

$=$ 1 0001

FC Sum < 9

$\downarrow + 0110$

10111 BCD

$\downarrow$

$(17)_{10}$

Eg. $(57)_{10} + (26)_{10}$

$\downarrow$ $\downarrow$

01010111 0010 0110

$=$ 0 1 1 1 1 0 1 1

7 $<$ 9 13 $>$ 9

0111 1101 $\longrightarrow$ Check every 4 bits individually & check conditions & perform it

$+$ 0110

1000 0011 BCD

$\downarrow$

$(83)_{10}$.

* Shift ADD-3 Method. (Bin. $\longrightarrow$ BCD).

$(15)_{10}$

Operations	Tens	Ones	Decimal
Original No.			1111
Shift		1	111
Shift		11	11
Shift		111	1
Add-3		$+\ 011$	
		1010	1
Shift		10101	

BCD for 15.

* 2421 Code (BCD.) Self-Complementing

Decimal Digit	Set I 2*4 2 1	Set II 2*4 2 1 (Self-Complement)
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 1 1
6	0 1 1 0	1 1 0 0
7	0 1 1 1	1 1 0 1
→ 8	1 1 1 0	1 1 1 0
→ 9	1 1 1 1	1 1 1 1

* Excess-3 Code. (X3) Self-Complementing.

- Decimal → 8-4-2-1 (BCD) —Add 0011 (3)→ Excess-3

Eg. 5 → 0101 —+0011→ 1000

- Unweighted Code., 4-bit Code
- Decimal

Decimal	BCD		XS3
0	0000	+0011	0011
1	0001		0100
2	0010		0101
3	0011		0110
4	0100		0111
5	0101		1000
6	0110		1001
7	0111		1010
8	1000		1011
9	1001		1100

Eg. $(24)_{10} \longrightarrow$ (BCD) (00100100) $\xrightarrow{\text{Add } 0011}$ $00100100 + 00110011$

$\boxed{01010111}$ Ans.

Eg. $(658)_{10} \longrightarrow$ (BCD) $(0110\ 0101\ 1000)$

$\downarrow +0011$

$\boxed{1001\ 1000\ 1011}$

- XS-3 is only unweighted code that is self-complementing.

* Whether self complementing or not —

Weights. — w_1, w_2, w_3, w_4. [For weighted Codes]

$w_1 + w_2 + w_3 + w_4 = 9$ ~ Self Complementing

$\neq 9$ ~ Not "

* Excess-3 Code Addition:

Eg. $(2)_{10} + (5)_{10}$.

BCD: 0010 0101.

XS3 $+0011$: 0101 1000

0101 — XS3

$+\ 1000$ — XS3

1101 — XS6.

$-\ 0011$ — Subtract 3.

1010 — (Ans)

** Eg. $(27)_{10} + (39)_{10}$

BCD: $0010\ 0111$ $0011\ 1001$

$\downarrow 0011$ $\downarrow$

XS3 $+0011$: $0101\ 1010$ $0110\ 1100$

$0101\ 1010 + 0110\ 1100$

$1100\ 0110$

$-0011 + 0011$

$\boxed{1001\ 1001}$ Ans.

* { FC = 1, add 3; FC = 0, sub. 3

* Gray Code (Frank Gray, AT&T).

- Reflected binary code (RBC).
  Unweighted, cyclic code
- Unit distance code & minimum error code.
- Two successive values differ in only one bit
- Binary no. converted to GC to reduce switching operation.

Decimal	Binary	GC.
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000
16	10000	0000

- Binary → GC. (MSB remains same)

Step 1. Record MSB as it is.

Step 2. Add MSB to the next bit, record the sum, neglect carry (XOR)

Step 3. Repeat the process.

Eg. $(1011)_{Bin}$

1011
↘+↘+↘
1110 — Ans.

Eg. $(1110)_{Bin}$

1110
1001 Ans.

- GC → Binary.

Step 1. Record MSB as it is.

Step 2. Add MSB to next bit of Gray code, record sum & neglect carry (X-OR)

Step 3. Repeat.

Eg. $(1110)_{gc}$

1110
+↗+↗+↗
1011. (Ans)

Eg. $(1001)_{gc}$

1001

1110 (Ans).

* Parity:

→ Single bit error is detected by it.

→ i) Even parity – No. of 1s even

ii) Odd parity – No. of 1s odd.

0100 : 1 0100 : 0

1100 : 0 1100 : 1

even odd.

→ Transmit Receive.

0100 : 1 ⟶ (Due to noise) 0101 : 1

Even parity — Odd parity ✗

Error detected.

* Hamming Code – Error Detection.

→ Given by R.W. Hamming

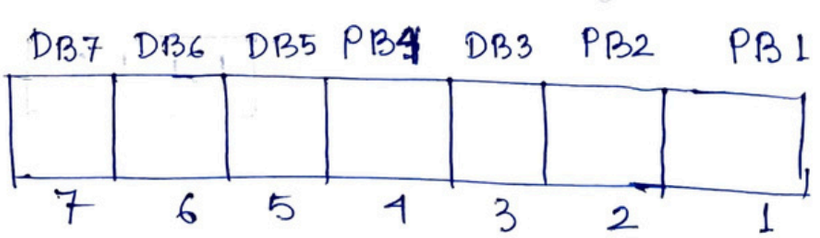
→ Easy to implement

→ 7-bit hamming code is used commonly.

→ Data bits – 4 } for 7-bit

Parity bits – 3 }

→ Position of parity bits 2^n.

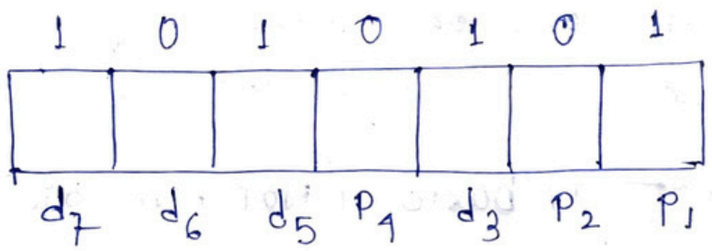


PB1 associated with DB3, DB5, DB7

PB2 " " DB3, DB6, DB7

PB4 " " DB5, DB6, DB7

Eg. (1011) Transmitter side



$P_1 \rightarrow d_3, d_5, d_7 \rightarrow [1]\ 1\ 1\ 1$ even parity

$P_2 \rightarrow [0]\ 1\ 0\ 1$ $\quad P_4 \rightarrow [0]\ 1\ 0\ 1$

Error detection –

due to noise →

$d_7 d_6 d_5\ d_3$
1 1 1 0 1 0 1.
$P_4\ P_2\ P_1$

Check $P_1 \rightarrow d_3, d_5, d_7$
even parity ✓

Check $P_2 \rightarrow d_3, d_6, d_7$
✗ error

Check $P_4 \rightarrow d_5, d_6, d_7$
✗ error.

* Hamming Code – Error Correction

Eg. If the 7 bit hamming code word received by a receiver is 1011011, assuming the even parity state whether the received code is correct or wrong ? Locate error bit. [Correct – 1001011]

$d_7\ d_6\ d_5\ P_4\ d_3\ P_2\ P_1$

1	0	1	1	0	1	1

→ $P_1 \rightarrow 1\ (1\ 1\ 0) \times P_1 = 1$

$P_2 \rightarrow 1\ (1\ 0\ 0)\ \checkmark\ P_2 = 0$

$P_4 \rightarrow 1\ (1\ 0\ 1) \times P_4 = 1$

$(P_4 P_2 P_1) \equiv (101)_2 \equiv (5)_{10}$ — Error

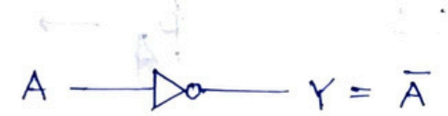
Error bit

Whenever parity unmatches
$P_i = 1$, otherwise
$P_i = 0$.

* Logic Gates.

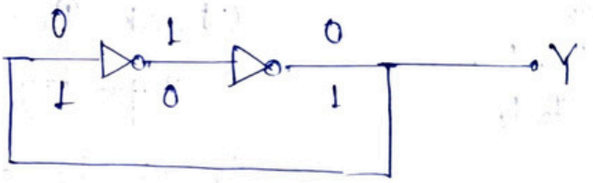
- A physical device that performs logic operation on one or more logical inputs, and produces a single logical output.
- Gates — i) basic (NOT, AND, OR)
  ii) universal (NAND, NOR)
  iii) arithmetic (X-OR, X-NOR)

- **NOT Gate**
  IC 7404



A	Y
1	0
0	1

Eg. Q.



a) buffer
b) astable multivibrator
c) bistable multivibrator
d) square wave generator

buffer

i/p	o/p
0 →	0
1 →	1

Y = 1 if i/p = 1
Y = 0 if i/p = 0.

a) Not a buffer because of the feedback.

astable — not stable on either state, continually changes from one state to another.

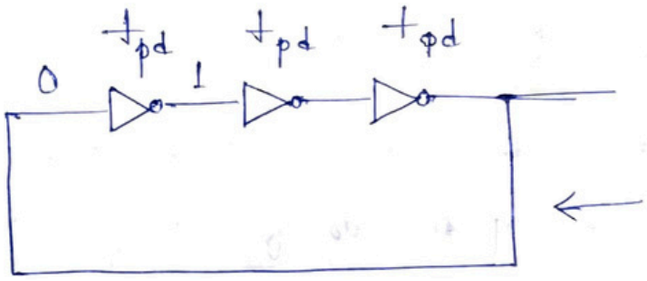
b) Not a stable.

c) bistable — stable on either state. Flipped by an external trigger pulse.

d) Not square wave generator.

Clock generator

Eg.

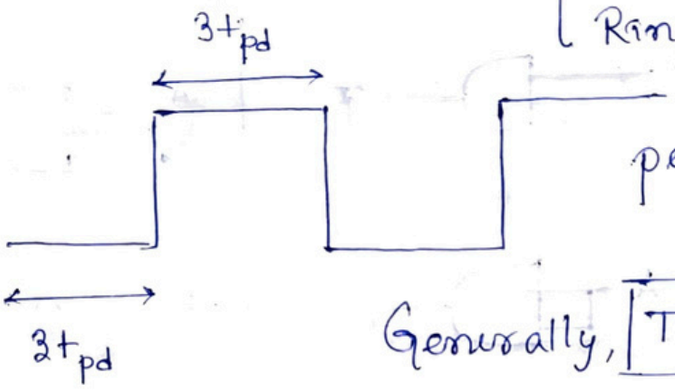


← Square wave generator

t_{pd} → propagation delay for each not gate

{ Astable MV.
{ Ring Oscillator

graph –



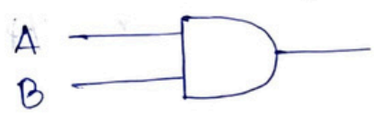
period, $T = 6t_{pd}$.

Generally, $\boxed{T = 2nt_{pd}}$

n → no. of NOT gates.

- **AND Gate** o/p high when all i/p high.

IC 7408



$Y = A \cdot B$.

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

→ $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

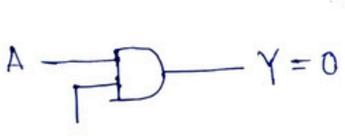
[Associativity]

→ $A \cdot B = B \cdot A$

[Commutativity]

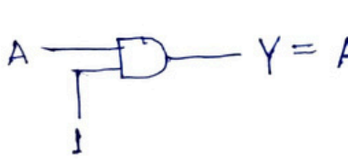
→ Enable & Disable.

→ When A = 0, Y = 0.



0 is disable for AND Gate.

→ When A = 1, Y = 0/1



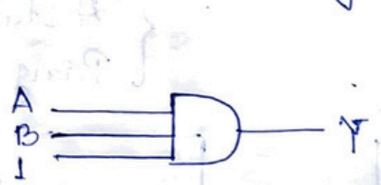
1 is enable (buffer).

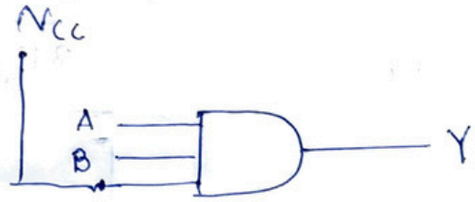
→ Unused I/P.

↝ In TTL logic (Transistor – Transistor) if any i/p is open or floating it will act as 1.

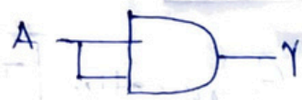
↝ In ECL logic if open, act as 0.

↝ i)

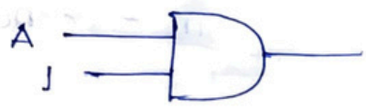




ii)

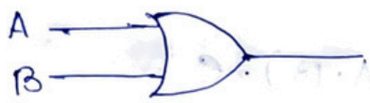


iii)



TTL logic

• OR Gate IC 7432 One of all or all are high. ⟶ O/p high



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

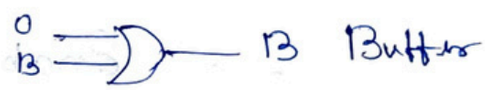
↝ (A+B)+C = A+(B+C)

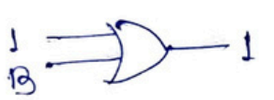
A+B = B+A.

↝ Enable & Disable.

0 ⟶ Enable

1 ⟶ Disable.



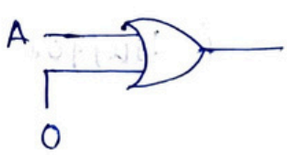


↝ Unused I/P

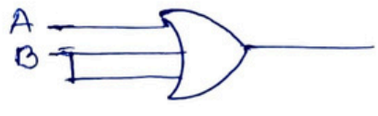
TTL – 1

ECL – 0.

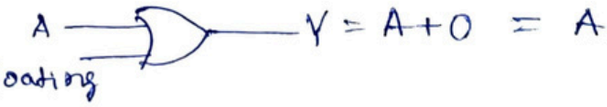
i) Connect to 0



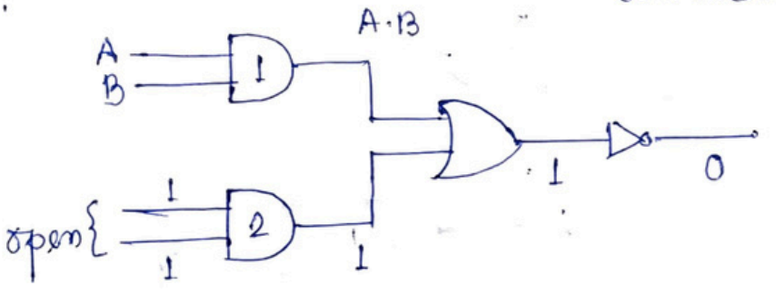
ii)



iii) ECL logic



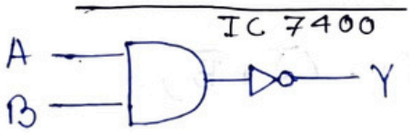
Eg. Consider TTL logic.

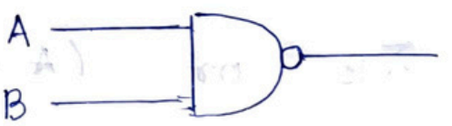


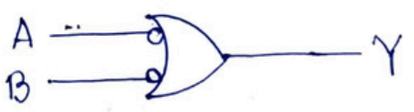
Y = 0. (Ans)

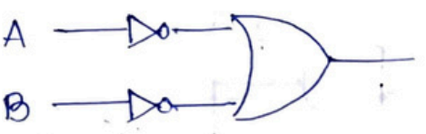
* NAND Gate. $(\bar{A}+\bar{B})$ Bubbled OR

IC 7400









0 → disable

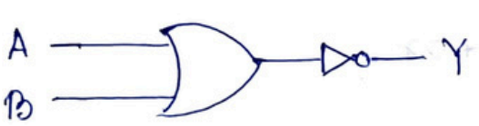
1 → enable

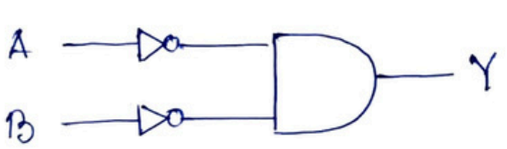
Unused i/p – same as AND.

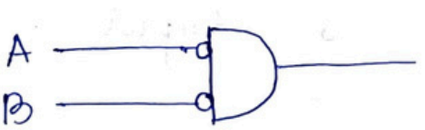
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

* NOR Gate $(\bar{A}\cdot\bar{B})$ Bubbled AND.

IC 7402







0 → enable

1 → disable.

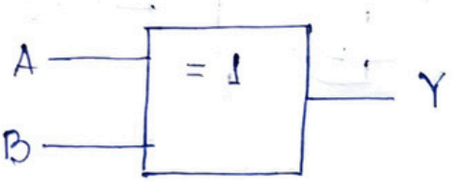
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Unused i/p – same as OR

* XOR Gate.

IC 7486

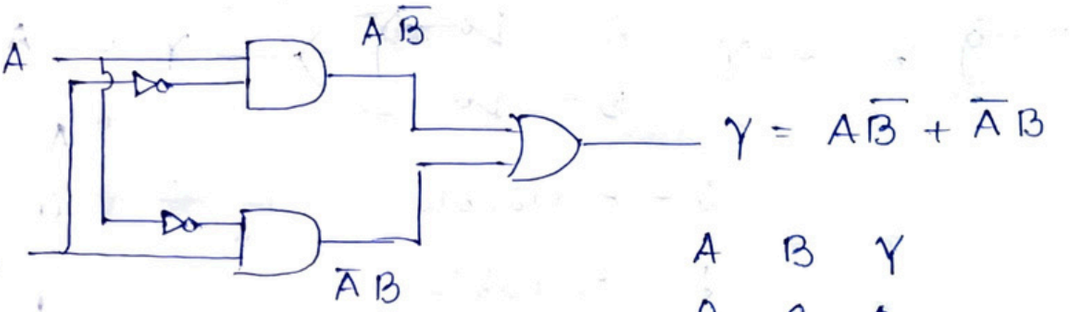
IEEE Symbol -



$Y = A \oplus B$

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

CKt

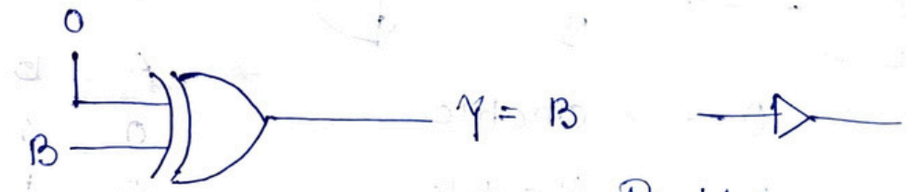


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Enable - 0 buffer
Enable - 1 inverter

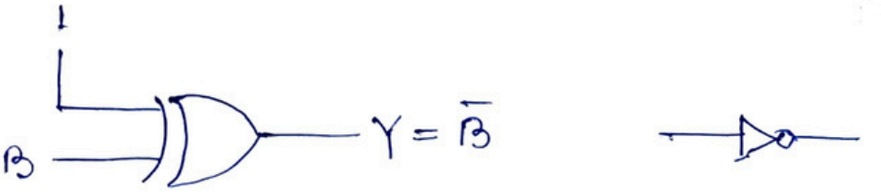
- XOR is odd one's detector.
- XOR - controlled inverter

We can make it buffer or inverter according to input.



Buffer

B	Y
0	0
1	1



Inverter.

B	Y
0	1
1	0

• Properties of XOR :

1. $A \oplus A = 0$

   $A \oplus \bar{A} = 1$

   $A \oplus 0 = A$

   $A \oplus 1 = \bar{A}$

2. If $A \oplus B = C$ then

   $B \oplus C = A$

   $A \oplus C = B$

   $A \oplus B \oplus C = 0$.

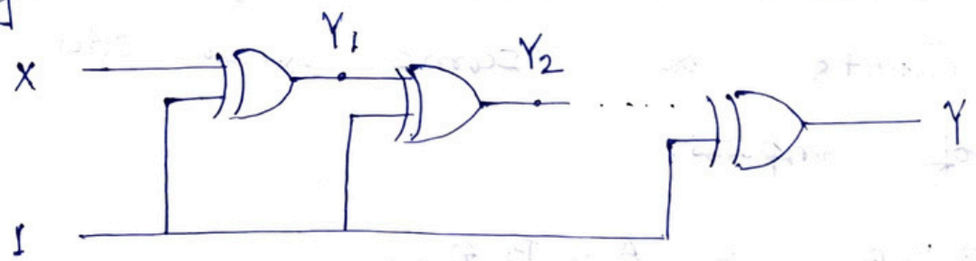
3. $A \oplus A \oplus A = A$

   $A \oplus A \oplus \cdots \oplus \cdots n$ times $=$

   A, $n \rightarrow$ odd

   0, $n \rightarrow$ even.

Eg.



Cascading of 20 XOR Gates.

$Y_1 = X \oplus 1 = \bar{X}$

$Y_2 = \bar{X} \oplus 1 = X$

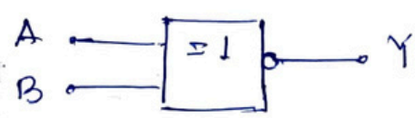
$\vdots$

$Y_{20} \Rightarrow Y = X$.

* XNOR Gate. $(A \odot B)$

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

IEEE Symbol



- $A = B$ o/p $= 1$

  $A \neq B$ o/p $= 0$

- XNOR follows commutative, associative law.
- Enable – 0, inverter

  Enable – 1, buffer.

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

- Properties.

i) $A \odot A = 1$

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

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

$A \odot 1 = A$.

ii) $A \odot A \odot A = A$.

$A \odot A \odot \ldots$ n times

$= A$, n is odd

$= 1$, n is even

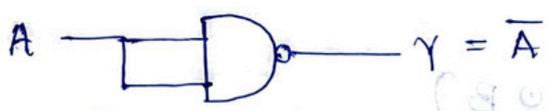
iii) XOR & XNOR are complementary when even inputs & same when odd no. of inputs.

Eg. $A \odot B \odot C = A \oplus B \oplus C$.

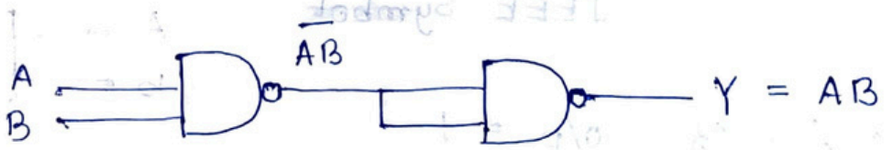
$A \odot B \odot C \odot D = \overline{A \oplus B \oplus C \oplus D}$

* NAND as Universal Gate.

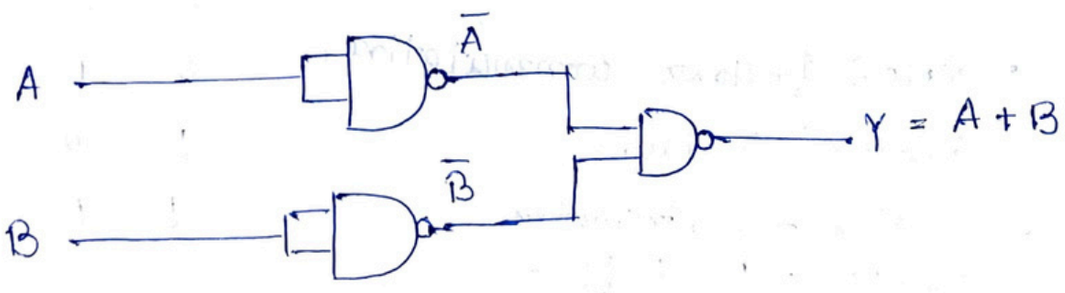
- NAND as NOT. $Y = \overline{AB}$, $Y = \overline{AA} = \bar{A}$



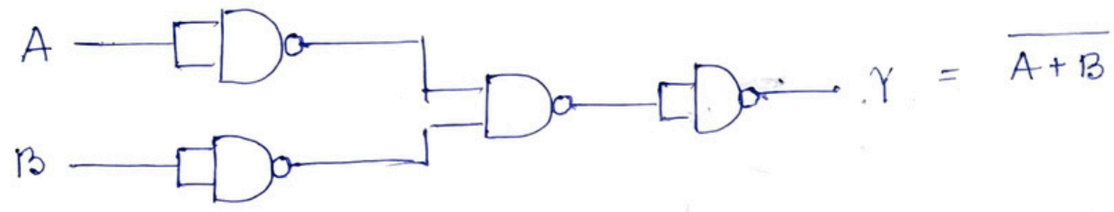
- NAND as AND.



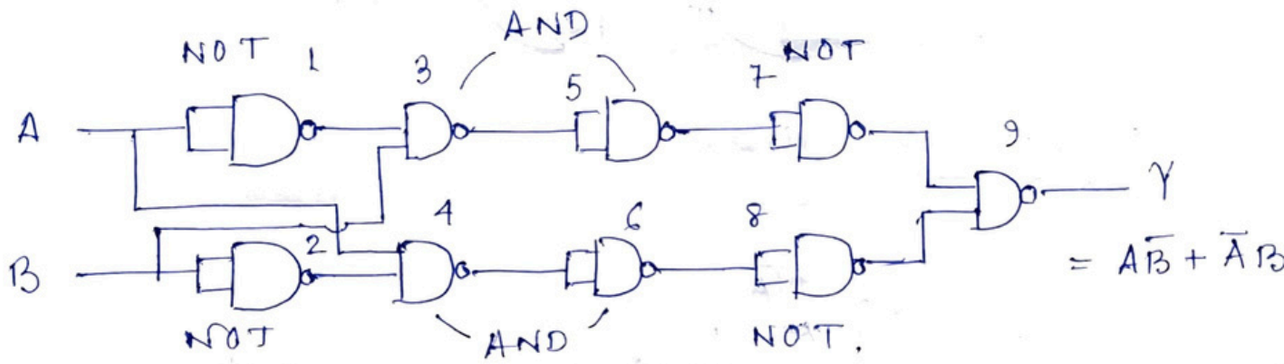
- NAND as OR. $Y = \overline{AB} = \bar{A} + \bar{B}$



• NAND as NOR.



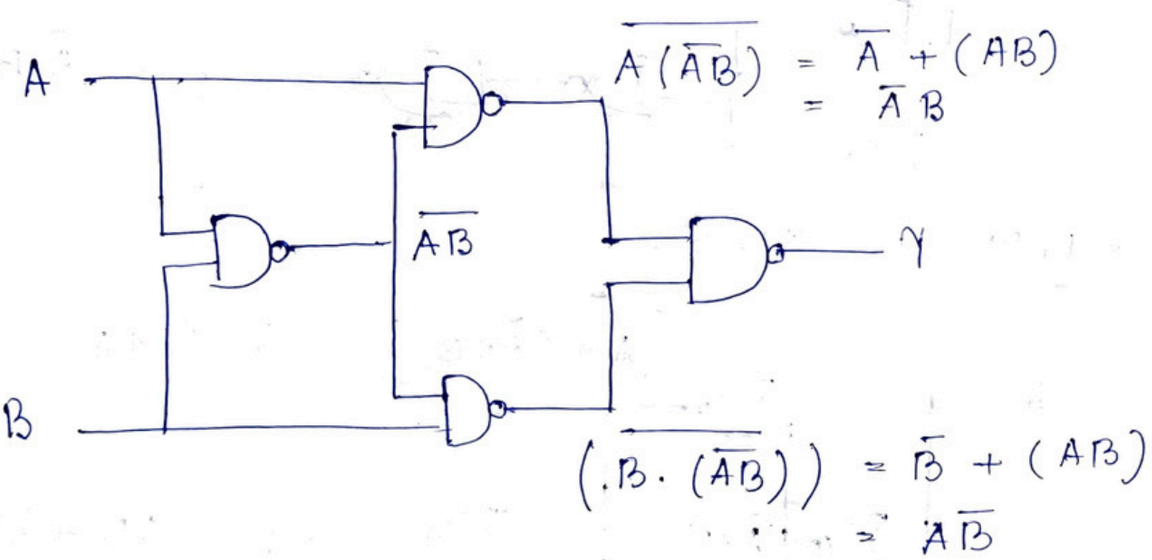
* *
• NAND as XOR. $Y = A\bar{B} + \bar{A}B$.



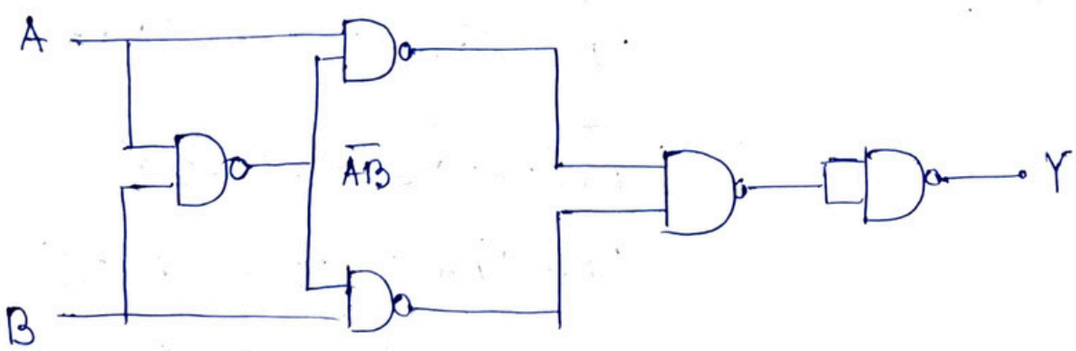
$Y = \left(\overline{(\bar{A}B) \cdot (A\bar{B})} \right)$ | 5, 7 & 6, 8 not required

III

* * *

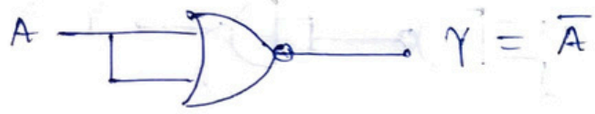


• NAND as XNOR.

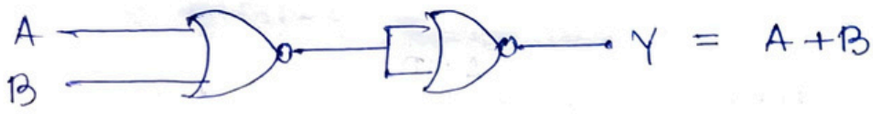


* NOR as Universal Gate.

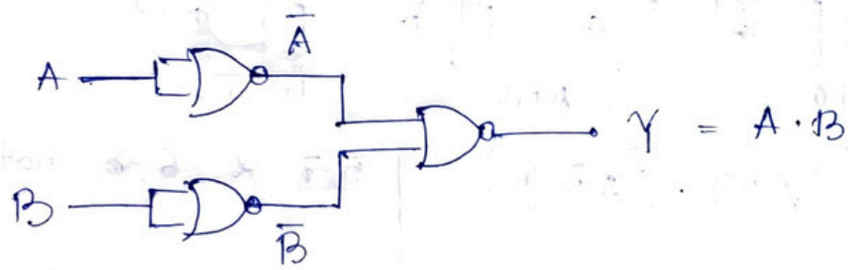
- **NOR as NOT.** $Y = \overline{A+B}$ $\quad Y = \overline{A+A} = \overline{A}$



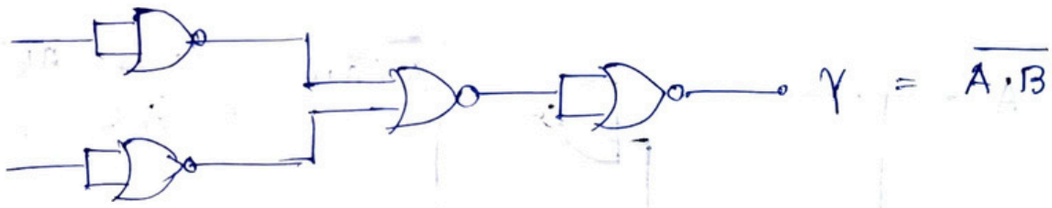
- **NOR as OR.**



- **NOR as AND.**

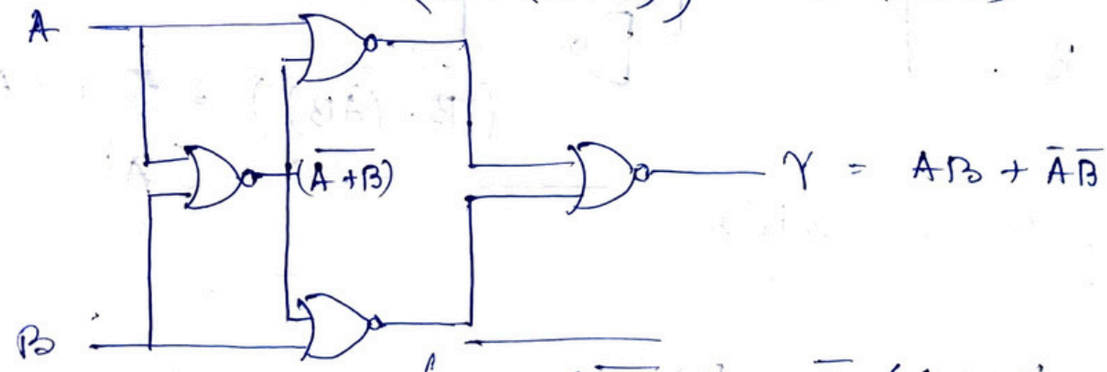


- **NOR as NAND.**



- **NOR as XNOR.**

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



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

$$\therefore \overline{[\overline{A} \cdot (A+B) + \overline{B} \cdot (A+B)]}$$

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

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

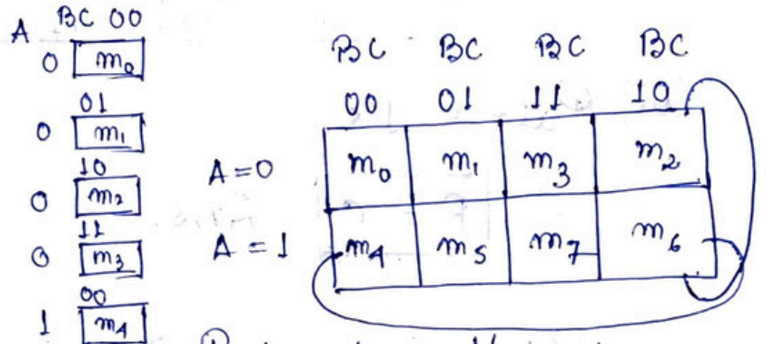
• NOR as XOR.

* Karnaugh Map [K' Map] :

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

A	B	C	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

In k' Map the adjacent cells are like only one variable should change.



Rule of adjacency.

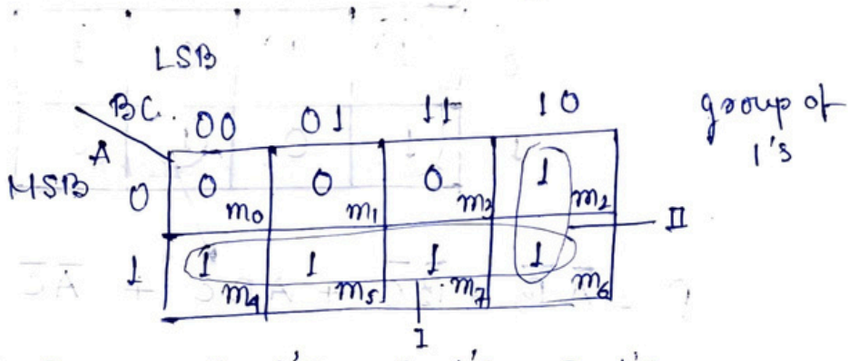
Rolling of K' Map

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

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

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

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



• pairing 2 1's, 4 1's, 8 1's, 16 1's

(No diagonal pairing, or adjacency not follows).

Same 1 can be paired more than once.

* Inside

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

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

$= B\bar{C}$.

$f = I + II$.

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

$= A + B\bar{C}$

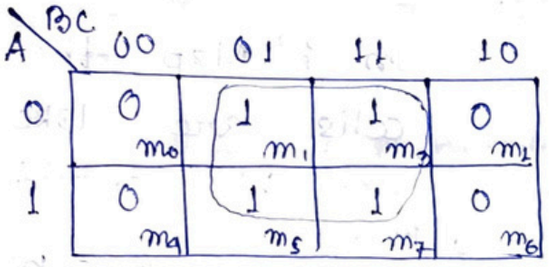
If variable changes → 1
If not → X/$\bar{X}$.

Eg. $f(A,B,C) = \Sigma m(1,3,5,7)$.

A → MSB, C → LSB, Σm → SOP.

(i) Find out no. of variables : 3.

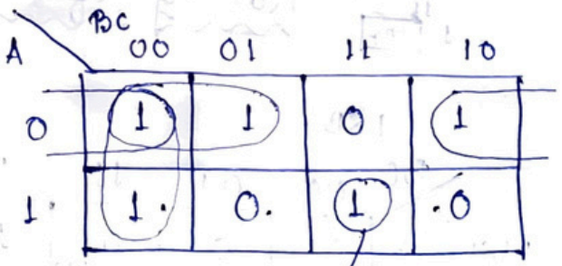
(ii) No. of cells in K' Map : $2^3 = 8$.



(iii) Pair 1's.

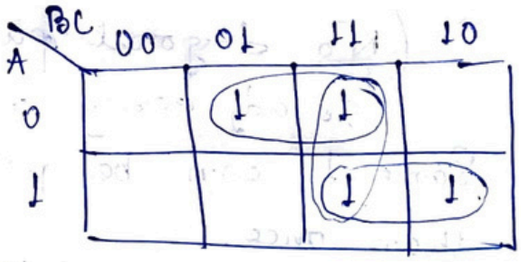
$\boxed{F = C}$ Ans.

Eg. $F(A,B,C) = \Sigma m(0,1,2,4,7)$.



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

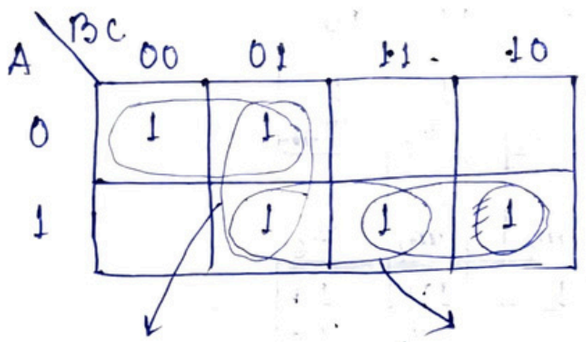
Eg. $F(A,B,C) = \Sigma m(1,3,6,7)$.



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

$= AB + \bar{A}C$ [by Redundancy Theorem]

Eg. $F(A, B, C) = \Sigma m(0, 1, 5, 6, 7)$.



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

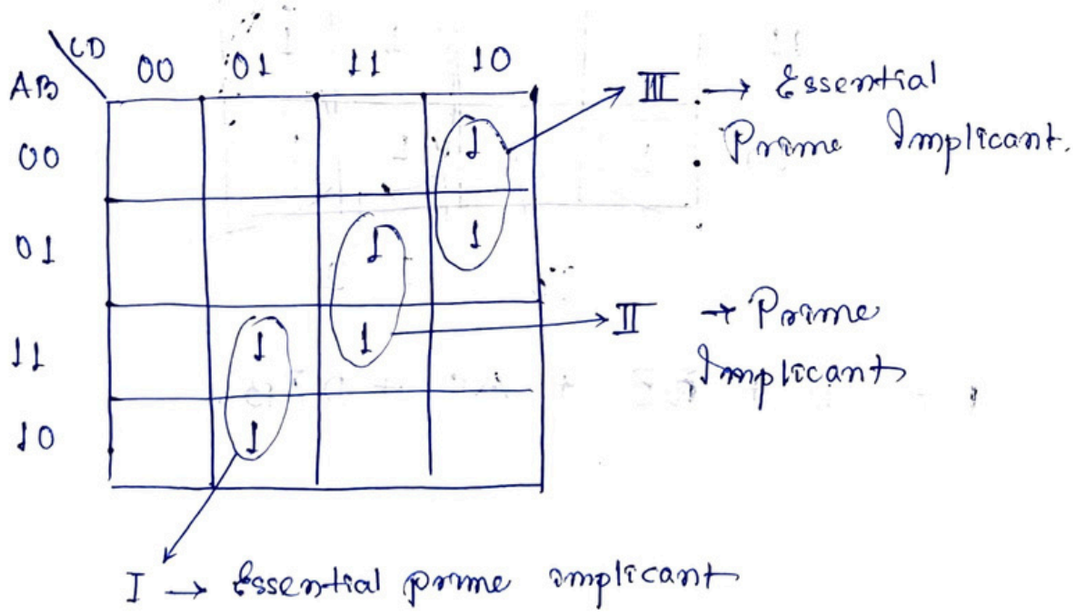
Result is minimum, but may not be unique.

- Implicants: Group of 1's in K' Map.

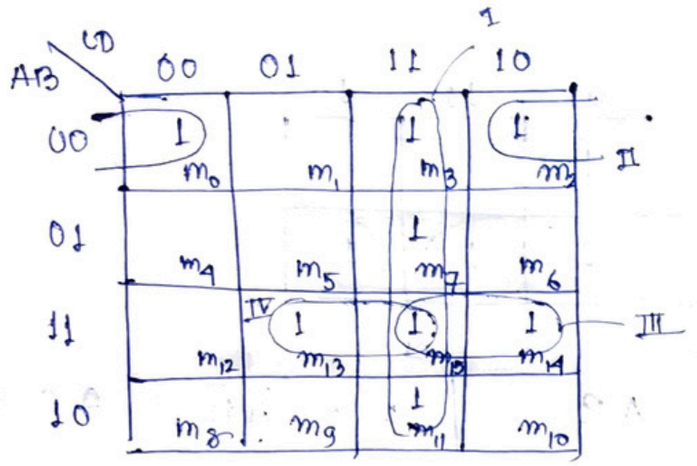
Prime Implicants: Largest possible group of 1's.

Essential Prime Implicants: At least there is a single one(1) which can not be combined in any other way.

Eg.



Eg. $f(A,B,C,D) = \Sigma m\,(0,2,3,7,11,13,14,15)$.

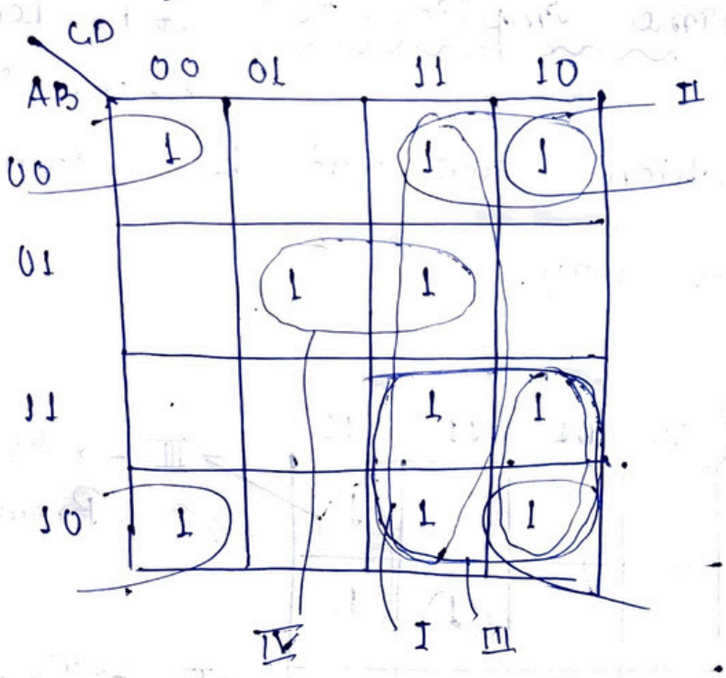


$$f = CD + \bar{A}\bar{B}\bar{D} + ABC + ABD.$$

- NB – Combining 2 1's → 1 literal is reduced
  - " 4 1's → 2 " " "
  - " 8 1's → 3 " " "
  - " 16 1's → 4 " " ".

2^n n

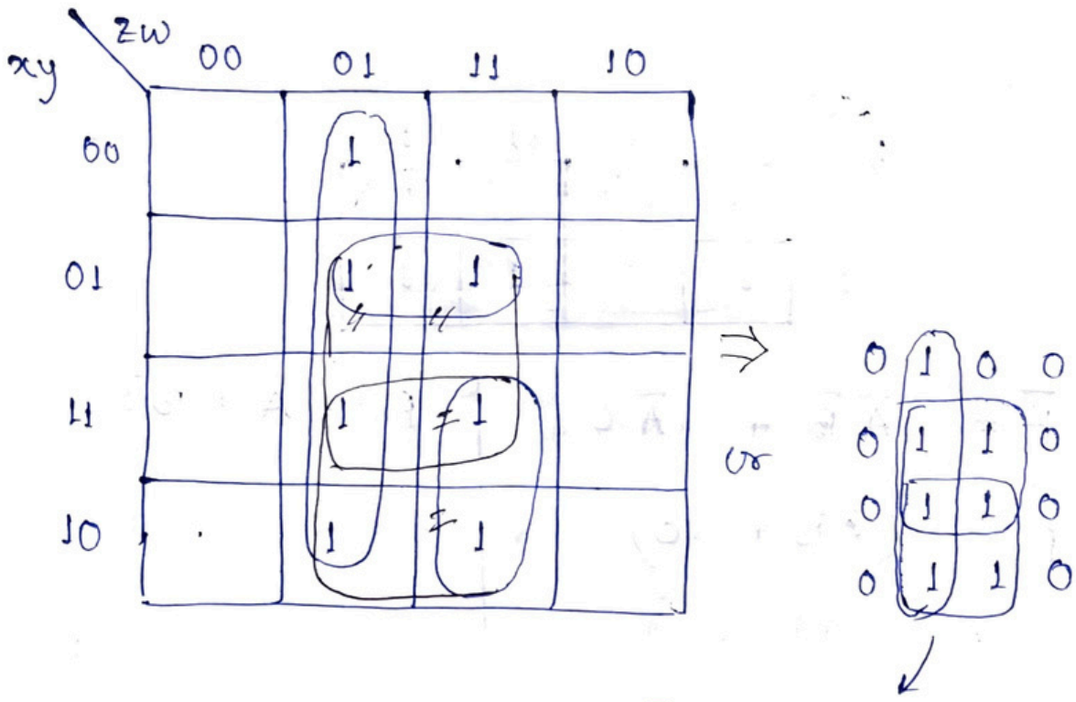
Eg. $F(A,B,C,D) = \Sigma m\,(0,2,3,5,7,8,10,11,14,15)$.



00	10
00	00
10	00
10	10

$$f = CD + \bar{B}\bar{D} + AC + D\bar{A}B.$$

Eg. $F(x,y,z,w) = \Sigma m\,(1,5,7,9,11,13,15)$.

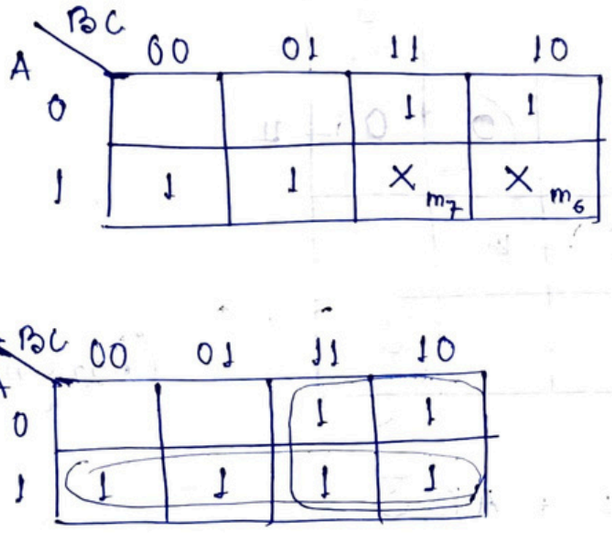


$F = xw + yw + \bar{z}w$.

* Don't Care in K' Map:

Eg. $F(A,B,C) = \Sigma m\,(2,3,4,5) + \Sigma d\,(6,7)$.

↳ don't cares



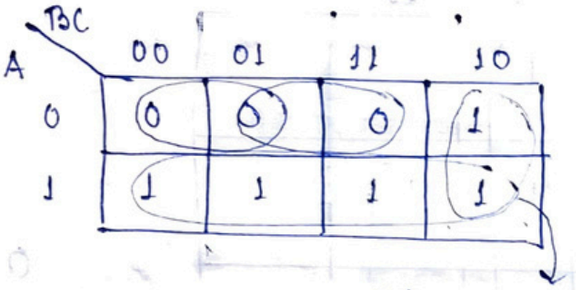
$f = A + B$

d → 0 or 1.

We take don't cares as 1, when reducing using minterms, & take as 0 when reducing using maxterms.

* K'Map using Maxterm.

Eg.



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

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

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

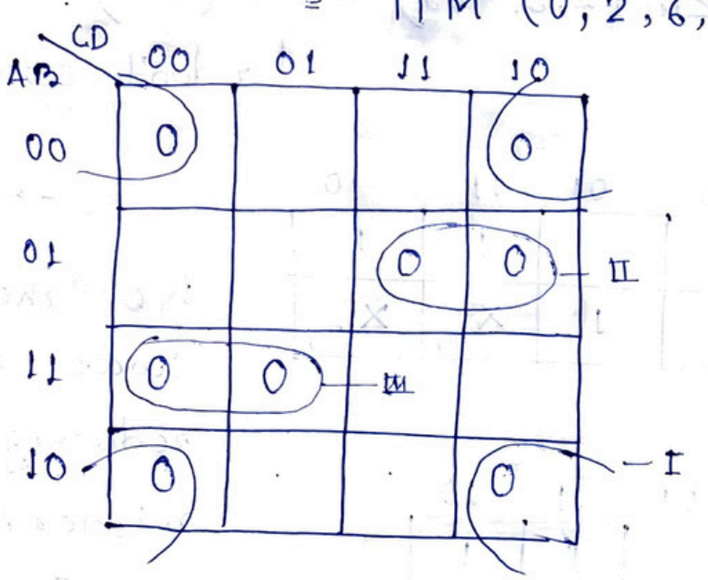
$= (A+B)\cdot(A+\bar{C})$. POS.

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

Complementing

Eg. $F(A,B,C,D) = \Sigma m\,(1,3,4,5,9,11,14,15)$.

$= \Pi M\,(0,2,6,7,8,10,12,13)$



Changing $+ \rightarrow \cdot$, $\cdot \rightarrow +$, $\bar{X} \rightarrow X$, $X \rightarrow \bar{X}$

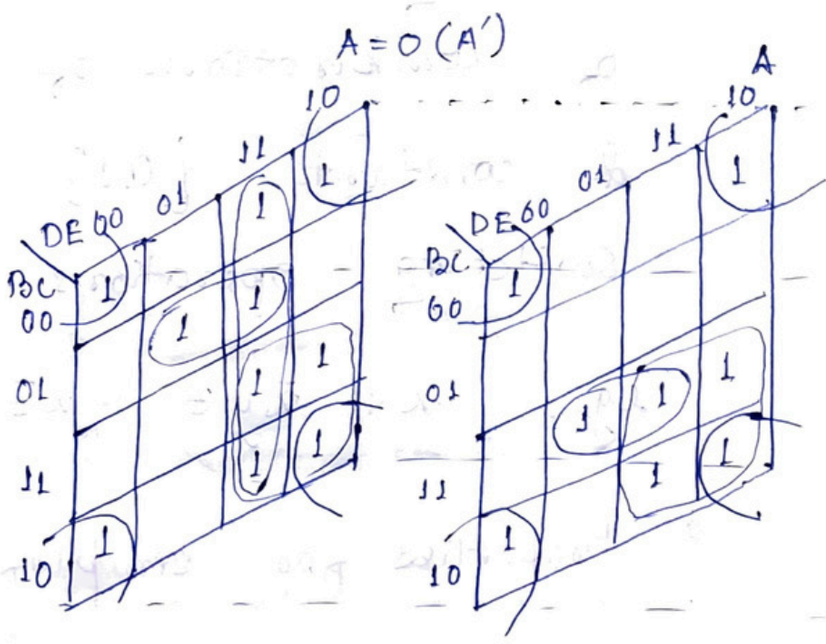
$\bar{F} = \bar{B}\bar{D} + \bar{A}BC + A\bar{B}\bar{C}$.

Complement

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

* K' Map with 5 variables:

	A	B	C	D	E
16	0	0	0	0	0
	0	0	0	0	1
	⋮				
	0	1	1	1	1
16	1	0	0	0	0
	1	0	0	0	1
	⋮				
	1	1	1	1	1



$$F = \bar{C}\bar{E} + BD + \bar{A}DE + \bar{A}BCE + ABCE$$

Switching Algebra.

* **Idempotency:**

$$x \cdot x = x$$
$$x + x = x.$$
$$x + 1 = 1 \qquad x \cdot 0 = 0$$
$$x + 0 = x \qquad x \cdot 1 = x.$$

* **Commutativity:**

$$x + y = y + x$$
$$xy = yx.$$

* **Associativity:**

$$((x+y)+z) = (x + (y+z))$$
$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

* **Complementation:**

$$x + \bar{x} = 1$$
$$x \cdot \bar{x} = 0$$

* **Distributivity:**

$$x \cdot (y+z) = x \cdot y + x \cdot z$$
$$x + y \cdot z = (x+y) \cdot (x+z)$$

* **Duality Principle**

$$\downarrow \quad + \quad \downarrow \quad 0 \quad \downarrow \quad 1 \quad \downarrow$$
$$+ \qquad \cdot \qquad 1 \qquad 0$$

* Switching expression is a finite number of combinations of switching variables & constants $\{0,1\}$ by means of switching operations $(+, \cdot, \sim)$.

eg. $x + \bar{x}yz + x\bar{z}$

* Properties for simplifying the SE:

1. Absorption: $x + x \cdot y = x$

$x + x' \cdot y = x + y$

2. Dual: $x \cdot (x' + y) = xy$

3. Consensus theorem:

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

Proof

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

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

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

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

eg. $x'y'z + yz + xz.$

$= z(x'y' + y + x).$

$= z[(x' + y)(y + y') + x]$

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

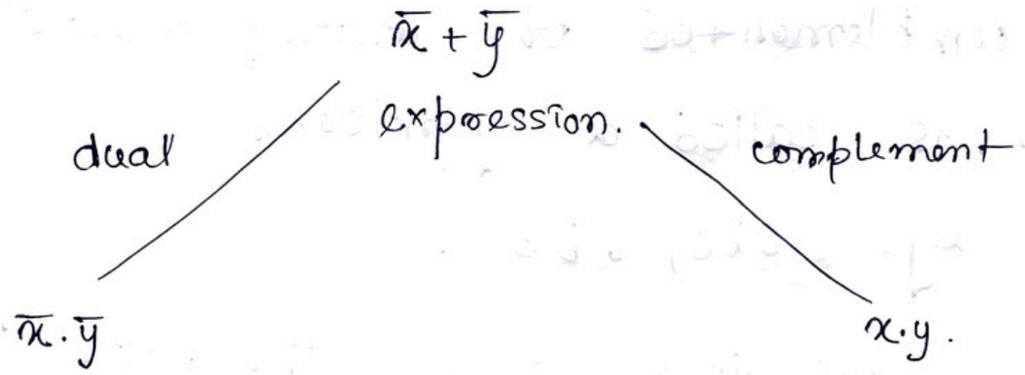
$= z.$

* De Morgan's Law :

$$\overline{xy} = \bar{x} + \bar{y}$$

$$\overline{x+y} = \bar{x} \cdot \bar{y}$$

*



* Eg.

$$(x+y)\left[x'(y'+z')\right]' + x'\bar{y} + x'z'$$

$$= (x+y)\left[x + (y'+z')'\right] + x'y' + x'z'$$

$$= (x+y)\left[x + yz\right] + x'(y'+z').$$

$x + y \cdot yz$.

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

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

$$= x + y' + yz + x'z'$$

$$= x + z' + y' + yz$$

$$= x + z' + y' + z.$$

$$= 1.$$

* Canonical forms :

→ A product term which contains each of n variables as factors either in complemented or uncomplemented form, is called a minterm.

eg. abc, ab'c .

→ A minterm given the value 1 for exactly one combination of the variables.

→ The sum of all minterms of f for which f assumes 1 is called canonical sum of products or disjunctive normal form.

$a + bc + ac$ SOP

$abc + \bar{a}bc$ Canonical SOP.

→ To find canonical SOP, take the sum of product terms of literals when the function value is 1.

→ A sum term that contains each of n variables as factors either in complemented or uncomplemented form is called maxterm.

eg. $(A + \overline{B} + C)(A + B)$

→ A maxterm gives the value 0 for exactly one combination of the variable.

→ The product of all maxterms of f for which f assumes 0 is called canonical product of sums or conjunctive normal form.

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

$(A+B+C)(A+\overline{B}+\overline{C})$ Canonical POS.

→ To find canonical POS form of f, take the maxterms of literals when value of f is 0.

* SOP minterm f is high (1) Σ

POS maxterm f is low (0) Π