# INDEX

NAME: Aditya Jain STD.: _____ SEC.: _____ ROLL NO.: _____ SUB.: _____

# Basic properties of switching Algebra :

- **Idempotency:**

$x \cdot x = x$ $\quad$ $x+1 = 1$ $\quad$ $x+0 = x$

$x+x = x$ $\quad$ $x \cdot 0 = 0$ $\quad$ $x \cdot 1 = x$

- **Commutativity:**

$x+y = y+x$

$x \cdot y = y \cdot x$

- **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:**

$x(y+z) = xy + xz$

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

**Note:-**

$\cdot$ $+$ then, $0$ $1$ } principle of duality

$\downarrow$ $\downarrow$ $\quad$ $\downarrow$ $\downarrow$

$+$ $\cdot$ $\quad$ $1$ $0$

# Switching Expression & simplifications :

Switching expression is a finite number of combinations of switching variables & constants $\{0,1\}$ by means of switching operations $(+, \cdot,$ Not$)$.

$+$ : And, $\cdot$ : OR

## * properties for simplifying SE:

**Absorption:** $x + xy = x$

$$= x \cdot 1 + x \cdot y$$
$$= x(1+y)$$
$$= x \cdot 1$$
$$= x$$

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

$$= x \cdot x + x \cdot y$$
$$= x + xy$$
$$= x \cdot (1+y)$$
$$= x \cdot 1$$
$$= x$$

$x + x'y = x + y$

$$= (x + x')(x+y)$$
$$= 1 \cdot (x+y)$$
$$= (x+y)$$

$x \cdot (x' + y) = xy$

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

**Consensus theorm:**

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

Note:- If the value of an expression does not depend on any term then it is called Redundant Expression.

9) **Minimize** $\bar{x}\bar{y}z + yz + xz$

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

# Demorgan's Law & simplification

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

② $(\overline{x+y}) = \bar{x} \cdot \bar{y}$

$f(a, b, c, \cdots, z, 0, 1, \cdot, +)$

(compliment) $\bar{f} = f(\bar{a}, \bar{b}, \bar{c}, \cdots \bar{z}, 1, 0, +, \cdot)$

(Duality) $f_d = (a, b, c, \cdots z, 1, 0, +, \cdot)$

$f = \bar{x} + \bar{y}$

$\bar{f}$ (compliment) $= x \cdot y$

$f_d$ (Duality) $= \bar{x} \cdot \bar{y}$

only difference is that literals is in compliment form.

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

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

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

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

$x + xy + yz + \bar{x}\bar{y} + \bar{x}\bar{z}$

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

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

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

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

$x + y' + yz + \bar{x}\bar{z}$

$x + y' + yz + z'$

$x + z' + y' + z$

$x + y' + 1 \Rightarrow 1$

# Switching functions

$f(a,b,c) = (a+bc)$

| a | b | c | f | f' |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 |

$f = 011 + 100 + 101 + 110 + 111$

$= \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c} + abc$

# Canonical sum of products:

① A product term which contains each of 'n'-variables as factors either in complemented or uncomplemented form is called a min term.

② A minterm given the value '1' for exactly one combination of the variables

③ The sum of all minterms of 'f' for which 'f' assumes '1' is called canonical sum of products or disjunctive normal form.

↓

or

$F(a,b,c)$ then the number of minterms = 8

$F(a,b,c,\ldots,n)$ then the number of minterms = $2^n$

$$f(a,b,c) = \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + a\bar{b}\bar{c} + a\bar{b}c + ab\bar{c} + abc$$

| a | b | c | f |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Canonical sum of products

$$f \Rightarrow \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + ab\bar{c}$$

$$\Rightarrow \sum(1,2,4,6)$$

# Canonical product of sums :

(1) A sum term which contains each of 'n' variables as factors either in complemented or uncomplemented form is called maxterm.

(2) A maxterm gives the value '0' for exactly one combination of the variables.

(3) The product of all maxterms of 'f' for which 'f' assumes '0' is called canonical product of sums or conjuctive Normal form.

$f(a,b,c) = (a+b+c)\ (a+b+\bar{c})\ (a+\bar{b}+c)\ (a+\bar{b}+\bar{c})$
$(\bar{a}+b+c)\ (\bar{a}+b+\bar{c})\ (\bar{a}+\bar{b}+c)\ (\bar{a}+\bar{b}+\bar{c})$

| a | b | c | f |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

Canonical product of sums

$f = (a+b+c)\ (a+\bar{b}+\bar{c})\ (\bar{a}+b+\bar{c})\ (\bar{a}+\bar{b}+\bar{c})$

$= \Pi(0,3,5,7)$

| | a | b | c | $f_1$ | $f_2$ | $f_3$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 | 1 | 1 | 0 |
| 3 | 0 | 1 | 1 | 1 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 1 | 0 |
| 5 | 1 | 0 | 1 | 1 | 0 | 0 |
| 6 | 1 | 1 | 0 | 1 | 1 | 1 |
| 7 | 1 | 1 | 1 | 1 | 0 | 1 |

$f_1 = \Sigma(2,3,5,6,7)$
$= \Pi(0,1,4)$

$f_2 = \Sigma(0,2,4,6)$
$= \Pi(1,3,5,7)$

$f_3 = \Sigma(0,1,6,7)$
$= \Pi(2,3,4,5)$

# Convert the following into the Canonical SoP & PoS:

$f(x, y, z) = x'y + z' + xyz$

$= x'y.1 + \bar{z}.1.1 + xyz$

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

$= \underset{(3)}{\bar{x}yz} + \underset{(2)}{\bar{x}\bar{y}\bar{z}} + \underset{(1)}{xy\bar{z}} + \underset{(4)}{x\bar{y}\bar{z}} + \underset{(5)}{\bar{x}y\bar{z}} + \underset{(0)}{\bar{x}\bar{y}\bar{z}} + \underset{(7)}{xyz}$

Only one should be taken from this two as they both are exactly same.

$= \Sigma(0,2,3,4,6,7)$ } Sum of products

$= \Pi(1,5)$ } Product of sums

# Functional properties:

① The canonical SoP or PoS form of a switching function is unique.

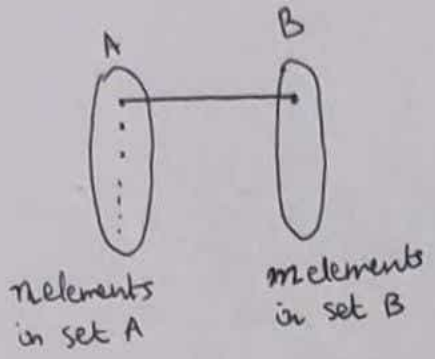
② Two switching functions $f_1(x_1, x_2, \ldots x_n)$ & $f_2(x_1, x_2, \ldots x_n)$ are said to be logically equivalent if & only if both functions have same value for each & every combination of $(x_1, x_2, \ldots, x_n)$.

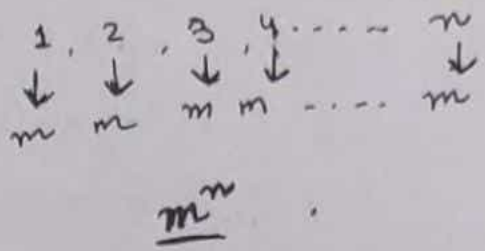
③ Two switching functions are equivalent if their canonical POS & SOP are identical.

| a | b | $f_1$ | $f_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |

$f_1$ & $f_2$ are equivalent

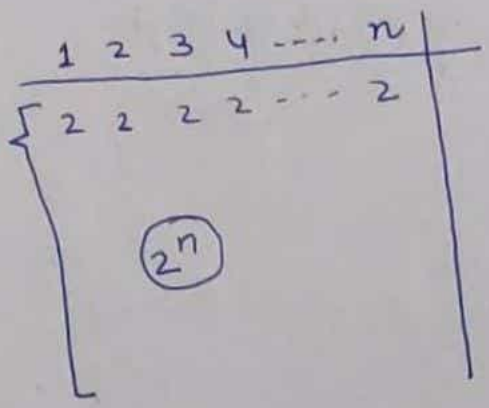
## Number of functions:

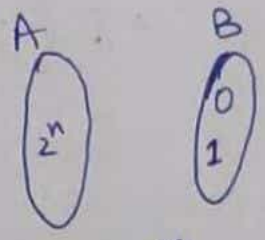



$m^n$

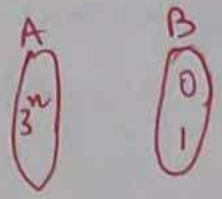
If we have n-booleans variables, then how many boolean functions are possible?



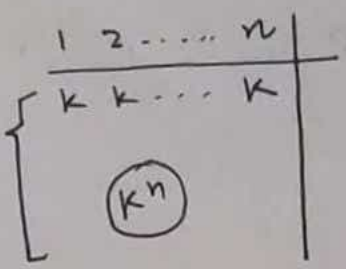

$2^{2^n}$

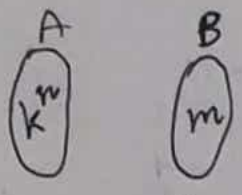
||ly n-ternary variables




$2^{3^n}$

Hence, If we have n kary variables, then how many m-ary functions are possible?




$m^{k^n}$

# Counting the number of functions & Neutral functions:

How many boolean functions are possible with 3-variables such that there are exactly 3 minterms.

| | a | b | c | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 1 1 |
| 1 | 0 | 0 | 1 | 1 0 0 |
| 2 | 0 | 1 | 0 | 1 1 1 |
| 3 | 0 | 1 | 1 | 0 1 0 |
| 4 | 1 | 0 | 0 | 0 0 1 |
| 5 | 1 | 0 | 1 | 0 0 0 |
| 6 | 1 | 1 | 0 | 0 0 0 |
| 7 | 1 | 1 | 1 | 0 0 0 |

$^8C_3 \Rightarrow 56$

How many boolean functions are possible with 3 variables such that there are almost **3** minterms

$\Rightarrow$ 0 minterm + 1 minterm + 2 minterm + 3 minterm

$\Rightarrow {}^8C_0 + {}^8C_1 + {}^8C_2 + {}^8C_3$

$\Rightarrow 1 + 8 + 28 + 112/2$

$\Rightarrow 93$

'k'-variables 'm'- minterms

$^{2^k}C_m$

binary variable

* Neutral functions:

↳ A function in which number of minterm is equal to number of maxterm ie. number of one is equal to number of zero.

For 2 variable:

| A | B | $\bar{A}$ | $\bar{B}$ | B | ⊙ (EXNOR) | ⊕ (EXOR) |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 |

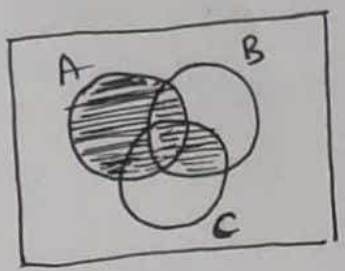
$^4C_2$

$\Rightarrow 6$

∴ for n variables

$\Rightarrow {}^{2^n}C_{(2^n/2)} = {}^{2^n}C_{2^{n-1}}$

Q) Find the boolean function that the following venn diagram represents.

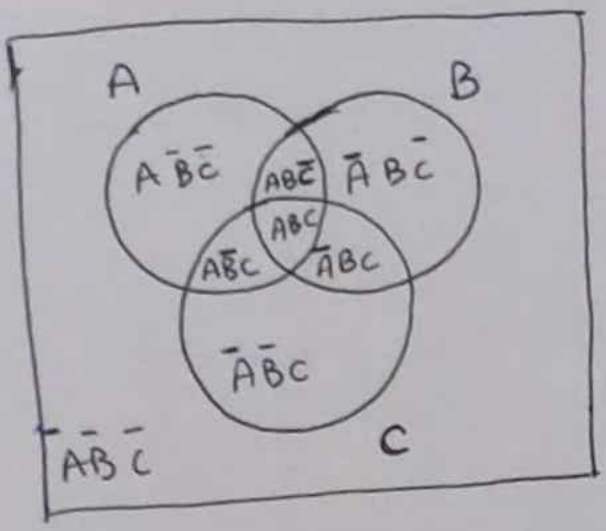


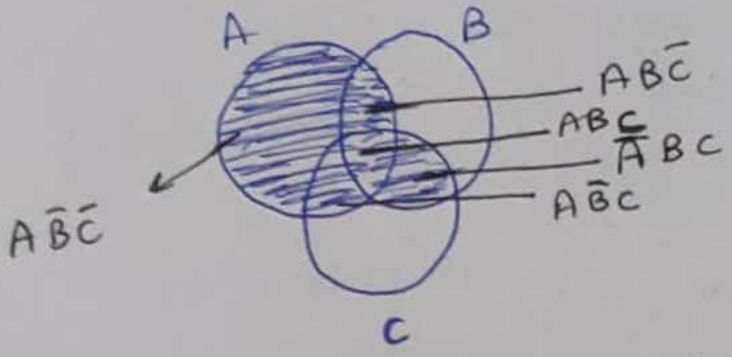

(a) $A + \bar{A}\bar{B}C$

(b) $A + BC$

(c) $A + \bar{A}BC$

(d) $AB + C$




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

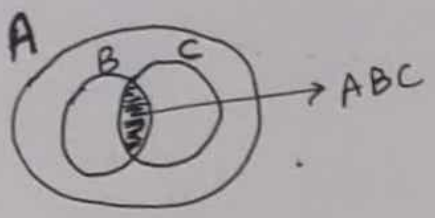
$\Rightarrow A\bar{B}(C+\bar{C}) + AB(C+\bar{C}) + \bar{A}BC$

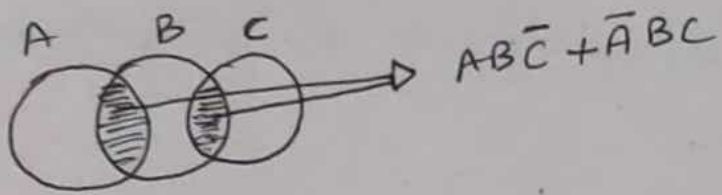
$\Rightarrow A\bar{B} + AB + \bar{A}BC$

$\Rightarrow A(B+\bar{B}) + \bar{A}BC$

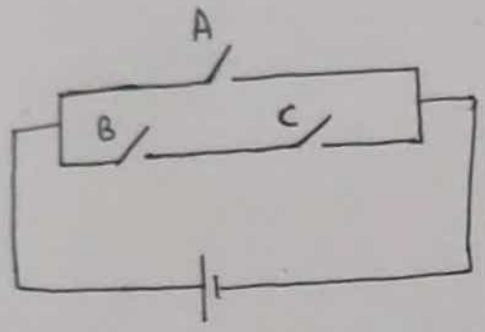
$\Rightarrow A + \bar{A}BC$ ✓

$\Rightarrow A + BC$ ✓



# Contact representation:




To pass the current through the circuit either A should be closed or B & C should be closed

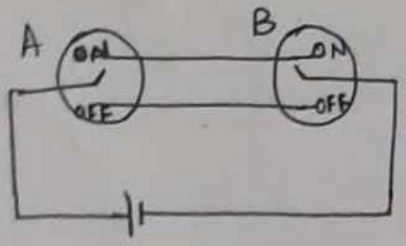
ie A+BC (To pass the current)

(1) Every boolean function can be represented with the help of serial & parallel contact.

(2) Serial contacts are performing 'AND' operation.

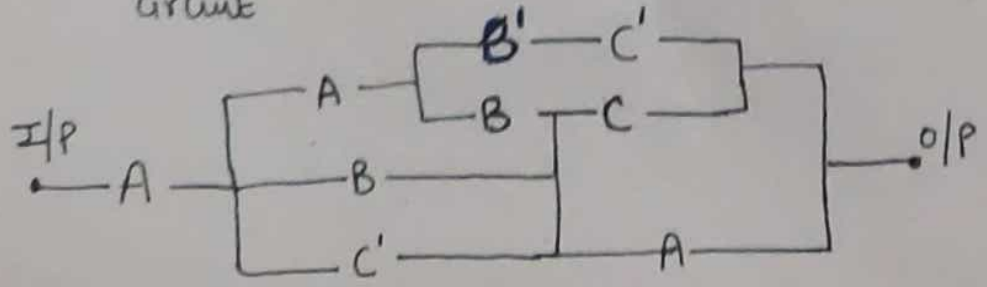
(3) Parallel contacts are performing 'OR' operation.

Which function does this following circuit represent.




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

Identify the boolean expression given by the following circuit



(1) find the valid forward path.
(2) perform OR among them.

Forward path:
Any path starting from i/p & ending at o/p without forming a cycle.

Validity:
No path should contain a variable in both true & complemented form.

$\Rightarrow A A B' C' + A A B C + A A B A + A B C + A B A$

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

Invalid as it contain both C & $\bar{C}$

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

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

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

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

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

In the following simultaneous boolean expression, what are the values of W, X, Y, Z.

$x + y + z = 1$
$xy + \bar{w} \bar{z} = 0$
$x \bar{w} + y \bar{z} = 1$

| | w | x | y | z |
|---|---|---|---|---|
| (a) | 0 | 0 | 0 | 1 |
| (b) | 1 | 1 | 0 | 1 |
| (c) | 0 | 1 | 0 | 1 |
| (d) | 1 | 0 | 0 | 0 |

$f(A,B) = A' + B$ then find

$f(f(x+y, y), z))$

↑

Nested function

↳ Hence, first implement the innermost function & then next level function.

$f(\underset{A}{x+y}, \underset{B}{y})$

$= (x+y)' + y$

$= x'y' + y$

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

$\longrightarrow$ $f(\underset{A}{x'+y}, \underset{B}{z})$

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

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

# NAND gate & properties:

| A | B | A ↑ B (NAND) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

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

**Identity**

$A \uparrow A \neq A$

| A | $\overline{A \cdot A}$ or $A \uparrow A$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

**Commutative**

$A \uparrow B = B \uparrow A$

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

$= B \uparrow A$

**Associative**

$A \uparrow (B \uparrow C) \neq (A \uparrow B) \uparrow C$

Let A be 0

LHS $\overline{0 \cdot (B \uparrow C)} = \bar{0} = 1$

RHS $(\overline{A \cdot B}) \uparrow C$

$= \overline{1 \cdot C} = \bar{C}$

# #NoR gate & properties:

| A | B | A ↓ B (NOR) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

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

**Identity:**

$A \downarrow A \neq A$

$= \overline{A + A}$

$= \bar{A}$

**Commutative:**

$A \downarrow B = B \downarrow A$

$= \overline{A + B}$

$= \overline{B + A}$

$= B \downarrow A$

**Associative:**

$A \downarrow (B \downarrow C) \neq (A \downarrow B) \downarrow C$

let A be 1

LHS $\overline{1 + (B \downarrow C)} = \bar{1} = 0$

RHS $(\overline{1 + B}) \downarrow C = \overline{0 + C} = \bar{C}$

# # Ex-OR gate & properties:

| A | B | A ⊕ B (EXOR) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

If we add 2 number & divided by 2 then the remainder what we get is modulo 2 sum & represented by EXOR [⊕] gate

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

$1 \oplus 1 \oplus 1 = 1$

$1 \oplus 0 \oplus 1 = 0$

**Identity:**

$A \oplus A \neq A$

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

$= 0$

**Commutative:**

$A \oplus B = B \oplus A$

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

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

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

**Associative:**

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

Let A = 0

LHS $0 \oplus (B \oplus C) = 0 \cdot B \oplus C + \bar{0} (B \odot C)$

$= B \oplus C$

RHS $(0 \oplus B) \oplus C = (0\bar{B} + \bar{0} B) \oplus C = B \oplus C$

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

Let $A \Rightarrow 1$

LHS $= 1 \oplus (B \oplus C)$

$= 1 \cdot \overline{(B \oplus C)} + 0 \cdot (B \oplus C)$

$= \overline{(B \oplus C)}$

RHS $= (1 \oplus B) \oplus C$

$= (1 \cdot \bar{B} + 0 \cdot B) \oplus C$

$= \bar{B} \oplus C$

equal

| B | C | $B \oplus C$ | $\overline{B \oplus C}$ | $\bar{B} \oplus C$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |

Hence,

$\overline{B \oplus C} = \bar{B} \oplus C = B \oplus \bar{C}$

# EX-NOR gate & properties:

| A | B | $A \odot B$ (Ex-NOR) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

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

Identity:

$A \odot A \neq A$

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

$= \bar{A} + A$

$= 1$

Commutative:

$A \odot B = B \odot A$

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

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

Associative:

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

Let $A \Rightarrow 0$

LHS $\Rightarrow 0 \odot (B \odot C)$

$\Rightarrow \bar{0} \cdot \overline{(B \odot C)} + 0 \cdot (B \odot C)$

$= \overline{B \odot C}$

RHS $\Rightarrow (0 \odot B) \odot C$

$\Rightarrow (\bar{0}\bar{B} + 0B) \odot C$

$\Rightarrow B \odot C$

Let $A \Rightarrow 1$

LHS $\Rightarrow 1 \odot (B \odot C) = \bar{1}\overline{(B \odot C)} + 1(B \odot C)$

$= B \odot C$

RHS $\Rightarrow (1 \odot B) \odot C = (1B + \bar{1}\bar{B}) \odot C$

$= B \odot C$

Equal

| B | C | $\overline{B \odot C}$ | $\bar{B} \odot C$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

Hence,

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

# #properties of Ex-OR & Ex-NOR:

Ex-OR

| A | B | $A \oplus B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Ex-NOR

| A | B | $A \odot B$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$\oplus \rightarrow 1$, for odd number of 1's

$\odot \rightarrow 1$, for even number of 0's

| A | B | C | $A \oplus B \oplus C$ | $A \odot B \odot C$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

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

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

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

$A \oplus B \oplus C \oplus D \oplus E = A \odot B \odot C \odot D \odot E$

If number of inputs are even, then XOR & XNOR are complemented to each other & if number of inputs are odd, then XOR & XNOR are exactly same (identical)

$n$ is even where, $n$ is the number of inputs

if 1's → odd
⊕ = 1

if 1's → even
⊕ = 0

0's → odd
⊙ = 0

0's → even
⊙ = 1

[ odd + odd = even
even + even = even ]

∴ If $n$ is even XOR & XNOR are compliment to each other

$n$ is odd where, $n$ is the number of inputs

if 1's → odd
⊕ = 1

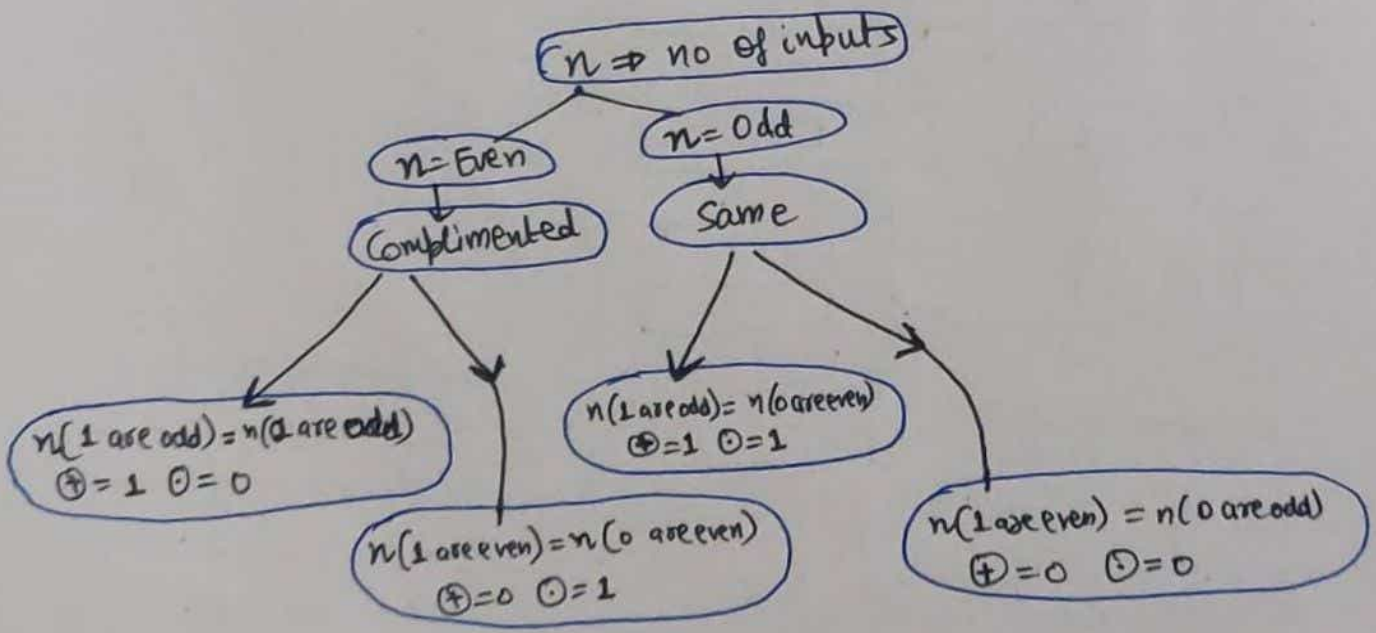
if 1's → even
⊕ = 0

0's → even
⊙ = 1

0's → odd
⊙ = 0

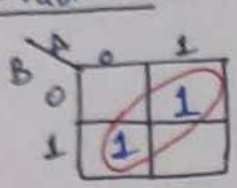
[ odd + even = odd
even + odd = odd ]

∴ If $n$ is odd XOR & XNOR are equal to each other

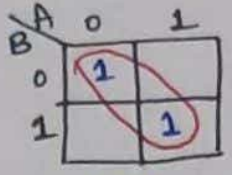


# Now Drawing K-Map

## 2 variables

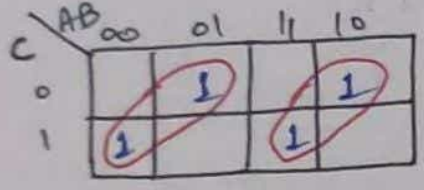



$A \oplus B$

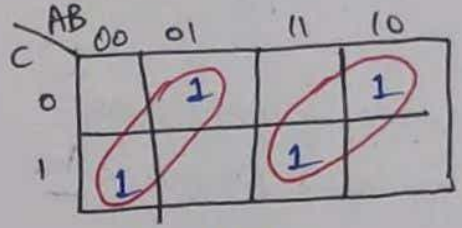



$A \odot B$

## 3 variables

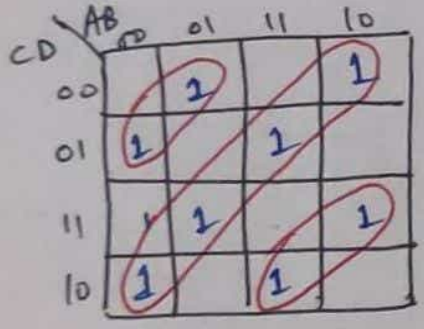



$A \oplus B \oplus C$



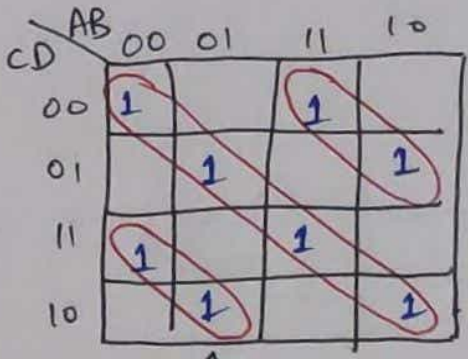

$A \odot B \odot C$

## 4 variables




$A \oplus B \oplus C \oplus D$

one's are filled at positions where no of 0's are odd
∴ It is Ex-OR & we can't minimize it further

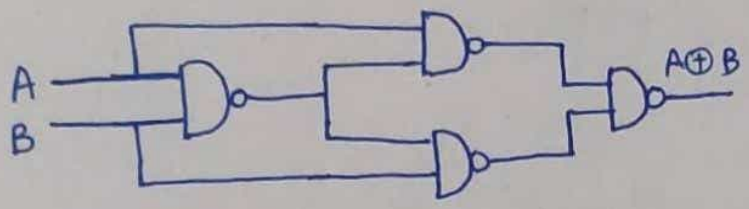



$A \odot B \odot C \odot D$

one's are filled at positions where no of 1's are even
∴ It is Ex-NOR & we can't minimize it further

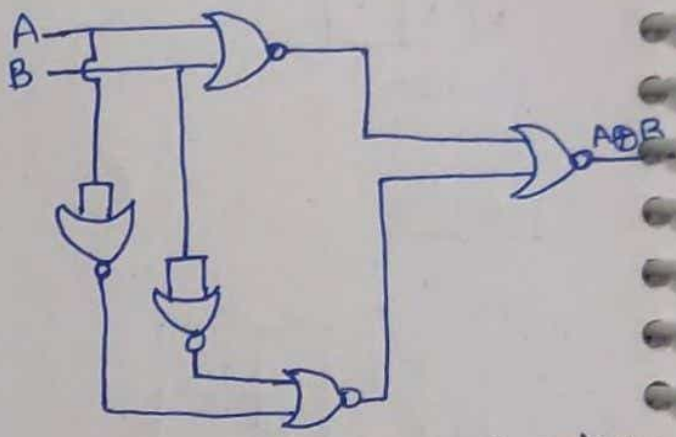
# Minimum Number of gates required for XOR & XNOR using Nand & Nor gates only :

## * Ex-OR (using NAND)



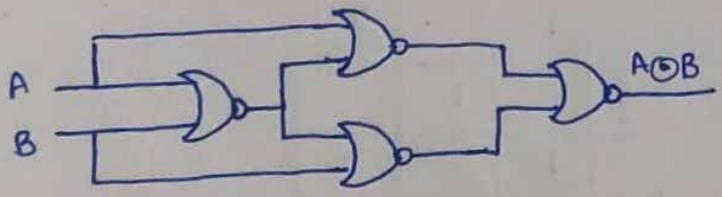

Minimum number of NAND gates required to build XOR gate is 4.

## * Ex-OR (using NOR)



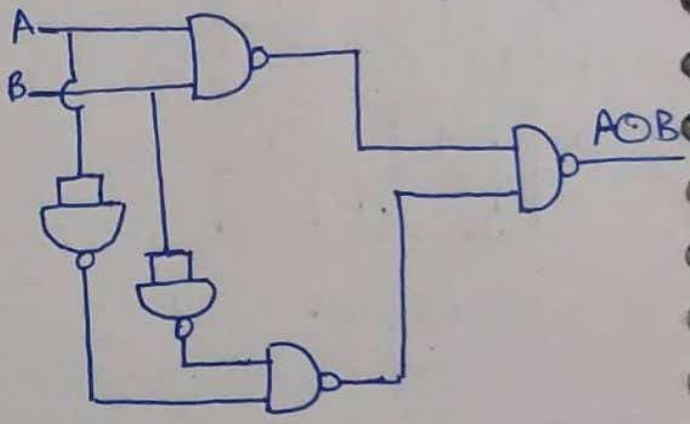

Minimum number of NOR gates required to build X-OR gate is 5.

## * Ex-NOR (using NOR)




Minimum number of NOR gates required to build X-NOR gate is 4.

## * Ex-NOR (using NAND)




Minimum number of NAND gates required to build X-NOR gate is 5.

**Note:-**

| | NAND | NOR |
|---|---|---|
| EXOR | 4 | 5 |
| EXNOR | 5 | 4 |

Q consider numbers represented in 4 bit gray code. Let $h_3h_2h_1h_0$ be the gray code representation of a number 'n' & let $g_3g_2g_1g_0$ be the gray code of n+1 (modulo 16) value of the number. which one of the following function is correct?

(a) $g_0(h_3h_2h_1h_0) = \Sigma(1,2,3,6,10,13,14,15)$

(b) $g_1(h_3h_2h_1h_0) = \Sigma(4,9,10,11,12,13,14,15)$

(c) $g_2(h_3h_2h_1h_0) = \Sigma(2,4,5,6,7,12,13,15)$

(d) $g_3(h_3h_2h_1h_0) = \Sigma(0,1,6,7,10,11,12,13)$

**Note:-**

Mirror concept

→ 0 0
0 1
1 1
1 0
→ 0 0 0
0 0 1
0 1 1
0 1 0
1 1 0
1 1 1
1 0 1
1 0 0

| S.NO | $h_3h_2h_1h_0$ | $g_3g_2g_1g_0$ | Value of $h_3h_2h_1h_0$ |
|---|---|---|---|
| 0 | 0 0 0 0 | 0 0 0 1 | 0 |
| 1 | 0 0 0 1 | 0 0 1 1 | 1 |
| 2 | 0 0 1 1 | 0 0 1 0 | 3 |
| 3 | 0 0 1 0 | 0 1 1 0 | 2 |
| 4 | 0 1 1 0 | 0 1 1 1 | 6 |
| 5 | 0 1 1 1 | 0 1 0 1 | 7 |
| 6 | 0 1 0 1 | 0 1 0 0 | 5 |
| 7 | 0 1 0 0 | 1 1 0 0 | 4 |
| 8 | 1 1 0 0 | 1 1 0 1 | 12 |
| 9 | 1 1 0 1 | 1 1 1 1 | 13 |
| 10 | 1 1 1 1 | 1 1 1 0 | 15 |
| 11 | 1 1 1 0 | 1 0 1 0 | 14 |
| 12 | 1 0 1 0 | 1 0 1 1 | 10 |
| 13 | 1 0 1 1 | 1 0 0 1 | 11 |
| 14 | 1 0 0 1 | 1 0 0 0 | 9 |
| 15 | 1 0 0 0 | 0 0 0 0 | 8 |

$g_3(h_3h_2h_1h_0) = \Sigma(4,12,13,15,14,10,11,9) \Rightarrow \Sigma(4,9,10,11,12,13,14,15)$

$g_2(h_3h_2h_1h_0) = \Sigma(2,6,7,5,4,12,13,15) \Rightarrow \Sigma(2,4,5,6,7,12,13,15)$

$g_1(h_3h_2h_1h_0) = \Sigma(1,3,2,6,13,15,14,10) \Rightarrow \Sigma(1,2,3,6,10,13,14,15)$

$g_0(h_3h_2h_1h_0) = \Sigma(0,1,6,7,12,13,10,11) \Rightarrow \Sigma(0,1,6,7,10,11,12,13)$

Idea is whenever $g_3$ value is 1 put $h_3h_2h_1h_0$ value in sigma.
whenever $g_2$ value is 1 put $h_3h_2h_1h_0$ value in sigma.
whenever $g_1$ value is 1 put $h_3h_2h_1h_0$ value in sigma.
whenever $g_0$ value is 1 put $h_3h_2h_1h_0$ value in sigma.

Q Let $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$ where $x_1, x_2, x_3$ & $x_4$ are the boolean variables & $\oplus$ is XOR operator. which of the following must always be true?

(a) $x_1 x_2 x_3 x_4 = 0$

(b) $x_1 x_3 + x_2 = 0$

(c) ✓ $\overline{x}_1 \oplus \overline{x}_3 = \overline{x_2} \oplus \overline{x_4}$

(d) $x_1 + x_2 + x_3 + x_4 = 0$

Hence, possibilities are:-

Total number of one's $= 0$
Total number of one's $= 2$
Total number of one's $= 4$

} i.e. Even number of one's

$\Rightarrow \overline{x}_1 \oplus \overline{x}_3 = \overline{x}_2 \oplus \overline{x}_4$

$\Rightarrow x_1 \oplus x_3 = x_2 \oplus x_4$

**Note:-**

$$\overline{A \oplus B} = \overline{A} \oplus B = A \oplus \overline{B}$$

Note:-

$$\overline{A \odot B} = A \oplus B = \bar{A} \odot B = A \odot \bar{B}$$

$$\overline{A \oplus B} = A \odot B = \bar{A} \oplus B = A \oplus \bar{B}$$

## Functionally complete operations:

(1) A set of operations is said to be functionally complete (or) universal if & only if every switching function can be expressed by means of operations in it.

(2) The set $\{+, \cdot, -\}$ is clearly functionally complete.

(3) The set $\{+, -\}$ is said to be functionally complete.

(4) The set $\{\cdot, -\}$ is also functionally complete.

**Note:**
A set is said to be functionally complete if we can derive a set which is already functionally complete.

$\{+, -\}$ $A \cdot B = \overline{(\bar{A} \oplus \bar{B})}$

$\{\cdot, -\}$ $A + B = \overline{(\bar{A} \odot \bar{B})}$ → And & or gate respectively

Q) $f(A,B,C) = A' + BC'$ is this Functionally complete?

$(-,\cdot)$ $(-,+)$

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

$$f\left(\underbrace{f(A,A,A)}_{A'}, \underbrace{B}_{\bar{B}}, \underbrace{f(B,B,B)}_{B'}\right)$$

$$f(A',B,B') = (A')' + B(B')'$$
$$= A + BB$$
$$= \underline{A+B}$$

Q) $F(A,B) = (\bar{A}+B)$ is this functionally complete?

$f(A,A) = \bar{A}+A = 1$
$f(B,B) = \bar{B}+B = 1$
} No way we can get rid of one variable

$$f(A,0) = \bar{A}+0$$
$$= \bar{A}$$

$$f\left(\underbrace{f(A,0)}_{\bar{A}}, \underbrace{B}_{\bar{B}}\right)$$

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

} Here, using '0' we can make it complete so when we take support from '0' & '1' then we call it Partially Complete.

Q) $F(A,B) = \bar{A}B$ is this functionally complete?

$$f(A,A) = \bar{A}A = 0$$

$$F(B,B) = \bar{B}B = 0$$

$$f(A,1) = \bar{A}$$

$$f(\underbrace{f(A,1)}_{\bar{A}}, B) = \overline{(\bar{A})}.B = A.B$$

partially functionally complete.

Q) $F(A,B,C) = AB+BC+CA$ is this functionally complete?

$f(A,A,A) = A$

If the function does not contain compliment then complementation can't be achieved

Not functionally complete

Q) $F(x,y) = \bar{x}y + x\bar{y}$ is this functionally complete?

$$f(x,x) = \bar{x}x + x\bar{x} = 0+0 = 0$$

$$f(y,y) = \bar{y}.y + y\bar{y} = 0+0 = 0$$

$$f(x,1) = \bar{x}1 + x0 = \bar{x}$$

} I am getting compliment with '1'

partially functionally complete

$f(x', y) = xy + \bar{x}\bar{y}$

$f(x, \bar{y}) = \bar{x}\bar{y} + xy$

$\boxed{f(\bar{x}, \bar{y}) = x\bar{y} + \bar{x}y}$ $\quad x \oplus y = x \odot y$

Hex, {+, −} Can't be Achieved

Not functionally complete.

Any Boolean function can be defined with which of the following operations

(a) $\oplus$, NOT → Does not lead to {+, ·} even with support of NOT ⇒ Not functionally complete

(b) $\oplus$, 1, OR, NOT → Partially complete; NOT → Functionally Complete

(c) $\oplus$, 1, NOT; NOT → Partially complete

(d) $\odot$, 1, NOT ⇒ Not functionally complete

$\boxed{f(x, y) = xy + \bar{x}\bar{y}}$

$f(x, x) = xx + \bar{x}\bar{x}$

$= x + \bar{x}$

$= 1$

Similarly $f(y, y) = 1$

$f(x, 0) = x \cdot 0 + \bar{x} \cdot 1$

$= \bar{x}$

$f(x', y) = \bar{x}y + x\bar{y}$

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

$f(\bar{x}, \bar{y}) = \bar{x}\bar{y} + xy$

Note:-

$x \oplus y = x' \odot y = x \odot y' = x' \oplus y'$

$x \odot y = x' \oplus y = x \oplus y' = x' \odot y'$

# self dual functions

$f(A,B,C) = AB + BC + CA$

$f_d(A,B,C) = (A+B)(B+C)(C+A)$

$= AB + BC + CA$

$f(A,B,C) = AB + BC + CA$

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

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

Total number of minterm possible $\Rightarrow 2^{(n)}$ → number of variable

$ABC \rightarrow A'B'C'$

$AB'C \rightarrow A'BC'$

Hence, $ABC \rightarrow A'B'C'$

$ABC' \rightarrow A'B'C$

$A'BC \rightarrow AB'C'$

$AB'C \rightarrow A'BC'$

A boolean function is self dual if

(1) It is neutral (no of minterms = no of maxterms)

(2) the function does not contain two mutually exclusive terms

# Number of self dual functions :

| # | A | B | C | Mutual exclusive |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 1 1 |
| 1 | 0 | 0 | 1 | 1 1 0 |
| 2 | 0 | 1 | 0 | 1 0 1 |
| 3 | 0 | 1 | 1 | 1 0 0 |
| 4 | 1 | 0 | 0 | 0 1 1 |
| 5 | 1 | 0 | 1 | 0 1 0 |
| 6 | 1 | 1 | 0 | 0 0 1 |
| 7 | 1 | 1 | 1 | 0 0 0 |

either select 0 or 7 ..... either select 3 or 4

(0,7) (1,6) (2,5) (3,4) ⇒ Mutual Exclusive

$F = (2) * 2 * 2 * (2) \Rightarrow 2^4 = 16$

$\Sigma(0, 1, 2, 3)$ = self dual functions

∴ Number of self dual function = $2^{(2^{n-1})}$

[ n - variables → $2^n$ terms

↓ divide into pairs (having two in each set)

$2^n/2 = 2^{n-1}$ pairs

↓

Each pair has two choices (include 1st element or second element)

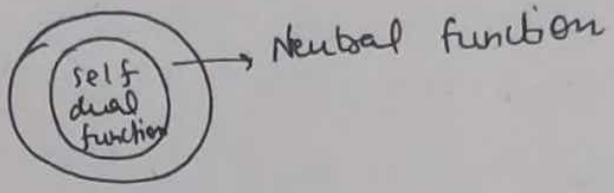
$\therefore 2^{2^{n-1}}$ ]

Number of Neutral function $\Rightarrow {}^{2^n}C_{2^{n-1}}$

Number of self dual function $\Rightarrow 2^{2^{n-1}}$

As every self dual function is Neutral function but vice versa is not true.

Hence,




Q) which of the following functions are self dual function.

(a) $f(A,B,C) = \Sigma(0,2,3)$ → No of minterms ≠ No of maxterms Not self dual

(b) $f(A,B,C) = \Sigma(0,1,6,7)$ → (1,6) ie contains mutual exclusive elements. Not self dual

(c) $f(A,B,C) = \Sigma(0,1,2,4)$ → No of minterms = No of maxterms & Mutual exclusive pairs are unique

(d) $f(A,B,C) = \Sigma(3,5,6,7)$ → No of minterms = No of maxterms & mutual exclusive pairs are unique

(0,7) (1,6) (2,5) (3,4)

$\Sigma(0,1,2,3)$

Self dual is closed under complementation (ie compliment of self dual function is self dual)

## # Introduction to electronic gates:

(1) Electronic gates generally receive voltages as inputs & produce voltages as outputs.

(2) The precise values of these voltages are not significant towards determination of logical operation of gates.

(3) The significant pt is that voltages are restricted to two ranges of values, high & low.

(4) Thus two valued variables may be used to represent these voltages.

(5) If we associate constant '1' with high voltage & '0' with low voltage, it is called positive logic system.

(6) If we associate the constant '1' with low voltage & '0' with high voltage, it is called negative logic system.

→ Consider a gate which outputs high voltage only when all inputs are high & outputs low voltage otherwise. How does this gate behave in positive logic & negative logic system.

AND gate

| A | B | A (op) B |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

| A B | A operation B |
|---|---|
| H H | H |
| H L | L |
| L H | L |
| L L | L |

+ve logic ←

→ -ve logic

OR gate

| A B | A (op) B |
|---|---|
| 0 0 | 0 |
| 0 1 | 1 |
| 1 0 | 1 |
| 1 1 | 1 |

→ Consider a gate which outputs high voltage if atleast one input is high & outputs low voltage otherwise. How does this gate behave in positive logic & negative logic system.

OR gate

| A | B | A (op) B |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

| A | B | A (op) B |
|---|---|---|
| H | H | H |
| H | L | H |
| L | H | H |
| L | L | L |

+ve logic ←

→ -ve logic

AND gate

| A B | A (op) B |
|---|---|
| 0 0 | 0 |
| 0 1 | 0 |
| 1 0 | 0 |
| 1 1 | 1 |

→ Gate take H as input & output L, L as i/p and output H

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

| A | $\bar{A}$ |
|---|---|
| H | L |
| L | H |

+ve logic ←

→ -ve logic

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

# Minimization:

## * Introduction to minimization of Boolean Expressions:-

A switching function can usually be represented by using a number of expressions.

eg $f(x,y,z) = xy + yz + zx$

$= x'yz + xy'z + xyz' + xyz$

while simplifying a switching function $f(x_1, x_2, x_3 \ldots x_n)$, our aim is to find an expression $g(x_1, x_2 \ldots x_n)$ which is equivalent to 'f', which minimizes some cost criteria

## ⇒ Criteria to determine the minimal cost:

occurrence of a variable either in true form or a complemented form

(1) Minimum number of appearances of literals

(2) Minimum number of literals in SoP or PoS expression

(3) Minimum number of terms in SOP expression, provided there is no other such expression with the same number of terms & fewer literals

## * Irredundant or Irreducable expressions:

$$f(x,y,z) = x'yz' + x'y'z' + xy'z' + x'yz + xyz + xy'z$$

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

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

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

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

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

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

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

$$= \boxed{\bar{x}\bar{z} + \bar{y}\bar{z} + yz + xz}$$

### Irredundant or Irreducable expression

An SOP expression from which no term or literal can be deleted without altering its logical value is called an irredundant or ireducable expression.

> **Note:-** An irredundant expression is not necessarily minimal, nor the minimal expression always unique

# K-Map

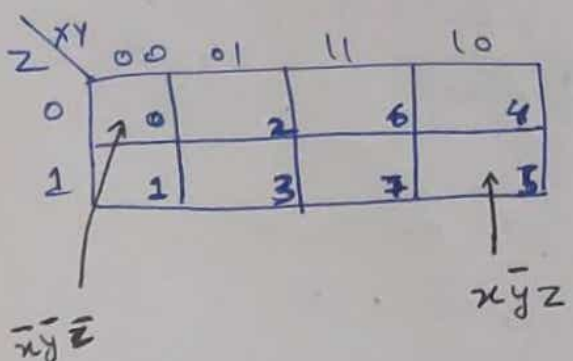
① The algebraic procedure of combining various terms & applying to them the rules becomes very tedious as the number of variable increases

② The map method provides a systematic method for combining terms & deriving minimal expression.

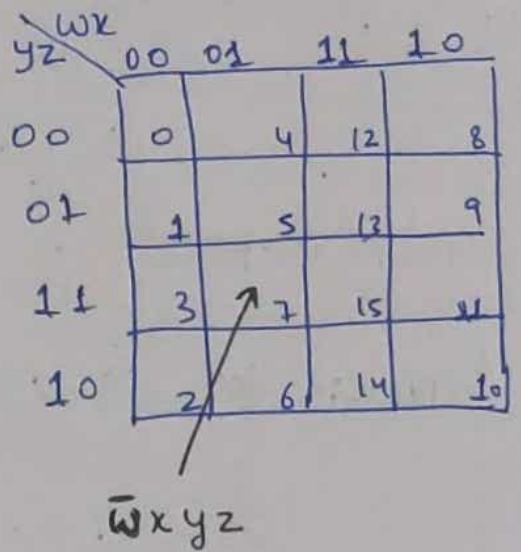
③ A k-map is the modified form of a truth table in which the arrangement of combinations is particularly convinient for minimization.

④ Every 'n' variable map consists of "$2^n$" cells (Squares), representing all possible combinations of variables

3-variable (k-Map)




4-variable (k-Map)



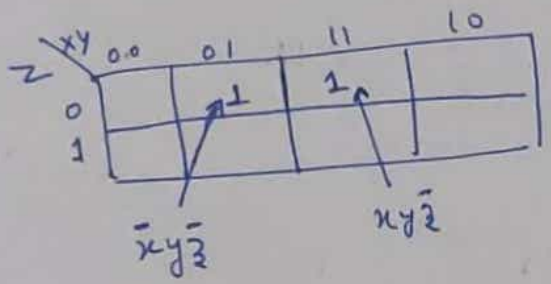
(5) The function value associated with a particular combination is entered in the corresponding cell.

(6) Cycle code is used in the combination as column & row heading.

(7) Because of these codes, two cell which have a common side correspond to combinations that differ by the value of just a single variable

(8) These 2 cells play a major role in simplification process. Because they can be combined by means of rule

$$A\bar{a} + Aa = A$$




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

$\Rightarrow y\bar{z}$

## #Simplification & minimization of functions using k-maps:

(i) A collection of $2^m$ cells, each adjacent ~~to~~ 'm' cells of collection is called a subcube, & the subcube is said to cover these cells.

(ii) Each subcube can be expressed by a product containing 'n-m' literals where 'n' is number of variables on which function depends.

iii) Any cell may be included in as many subcubes as desired.

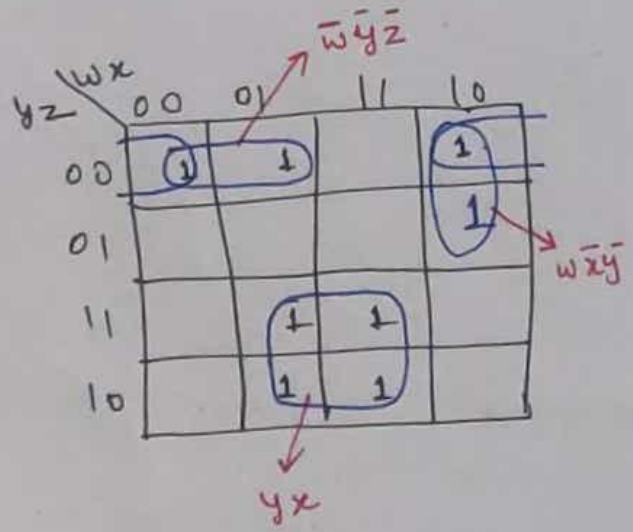
(iv) A function 'f' can be expressed as sum of those product terms which corresponds to subcubes necessary to cover all its '1' cells.

(v) The number of product terms in the expression for 'f' is equal to the number of subcubes while the number of literals in each term is determined by the size of corresponding subcubes.

(vi) Therefore to obtain a minimal expression we must cover all '1' cells with smallest number of subcube is as long as possible.

Q) Minimize

$$f(w,x,y,z) = \sum(0,4,6,7,8,9,14,15)$$

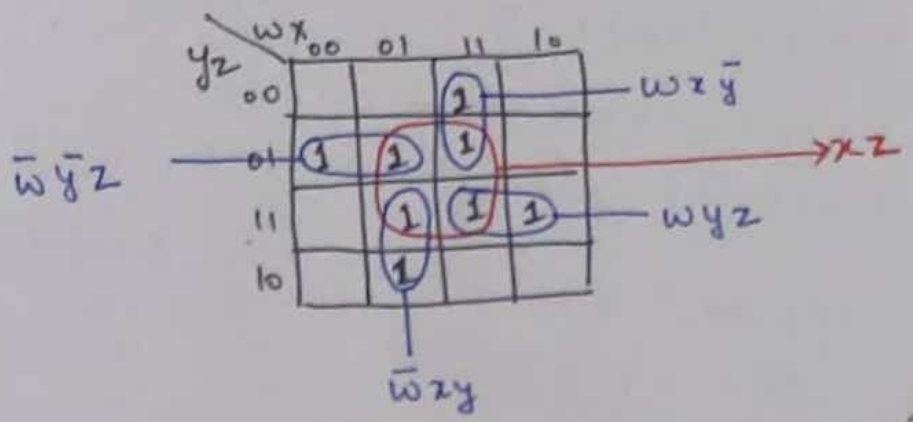



| # | w | x | y | z |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 |
| 14 | 1 | 1 | 1 | 0 |
| 15 | 1 | 1 | 1 | 1 |

$$f(w,x,y,z) = xy + w\bar{x}\bar{y} + \bar{w}\bar{y}\bar{z}$$

Q Minimize

$f(w,x,y,z) = \Sigma(1,5,6,7,11,12,13,15)$




$f(w,x,y,z) = wx\bar{y} + wyz + \bar{w}xy + \bar{w}\bar{y}z$

# Covering functions:

If the minterms present in 'g' are also present in 'f', then 'f' is said to cover 'g'.

Implicants:

A switching function $f(x_1, x_2 \ldots x_n)$ is said to cover $g(x_1, x_2 \ldots x_n)$ denoted by "'f' is superset of 'g'"

If g has 'x' min terms of g is the function of 'n' variables then the number of covering functions for $g = 2^{(2^n - x)}$

> Note:- If 'f' covers 'g' & 'g' covers 'f' then 'f & 'g' are equivalent

Q $f(w,x,y,z) = wx + yz$, then How many functions are covering 'f' ?

Number of minterms possible = $2^4 = 16$ minterms

$$f(w,x,y,z) = wx + yz$$
$$= wx(y+\bar{y})(z+\bar{z}) + yz(w+\bar{w})(x+\bar{x})$$
$$= wxyz + wxy\bar{z} + wx\bar{y}z + wx\bar{y}\bar{z} + \bar{w}xyz + wxyz + \bar{w}\bar{x}yz + w\bar{x}yz$$
$$= 1100 + 1101 + 1110 + 1111 + 0011 + 0111 + 1011 \quad \} \text{ 7 minterms}$$

Hence, f contains 7 minterms

Remaining $\Rightarrow 16 - 7$
$\Rightarrow$ 9 minterms

$\therefore$ Number of covering functions $\Rightarrow 2^9 \Rightarrow 512$

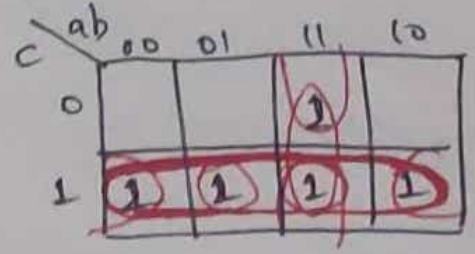
If 'f' covers 'g' then 'g' is said to imply 'f'. This is denoted by "$g \rightarrow f$" → whenever g is true, f is also true but vice-versa is not true.

eg $f(a,b,c) = ab + c$

| | | | $f_1$ | $f_2$ | $f$ |
|---|---|---|---|---|---|
| a | b | c | ab | c | ab+c |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |

f is going to cover both $f_1$ & $f_2$ which means whenever $f_1$ is one f will be one & whenever $f_2$ is one f will be one.

Hence, $f_1$ & $f_2$ ie ab & c are the implicants of 'f'



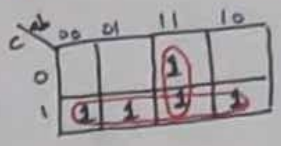

⇒ Every subcube is an Implicant

**Prime implicants:** { If the implicant is not the subset of any other implicant then it is called prime Implicant }

An implicant 'p' of a function 'f' is said to be prime implicant if

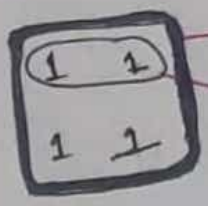
① P is a product term (i.e subcube)

② Deletion of any literal from 'p' results in a new product which is not covered by 'p' (ie A subcube can't be part of any other subcube)



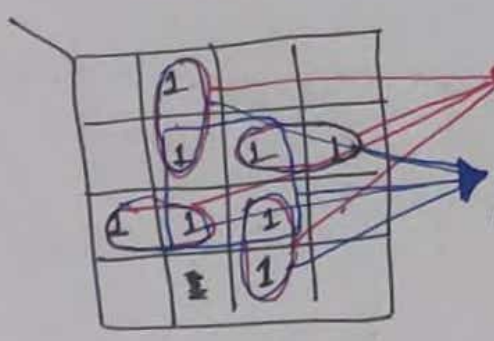

**Essential prime implicants:**

A prime implicant 'p' of a function 'f' is said to be an essential prime implicant if it covers atleast one minterm of 'f' which is not covered by any other prime implicants.



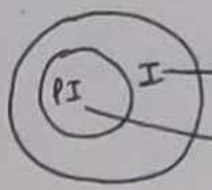
→ prime implicant

→ Implicant (Not prime implicant because it is completely covered by another implicant.



Essential prime implicant

prime implicants




Implicant

prime Implicant

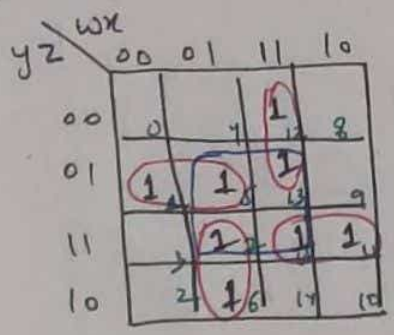
# Procedure for obtaining Minimal SOP :

(a) Determine all essential prime implicants & include them in minimal SOP.

(b) Remove from list of prime implicants, those which are covered by essential prime implicants

(c) If the set determined in step-1 covers all the minterms of 'f', then it is unique minimal expression, otherwise select the additional prime implicants so that the function 'f' is covered completely & the total number and size of prime implicants added are minimal.

eg:




$F = \sum(1,5,6,7,11,12,13,15)$

$F = \bar{w}\bar{y}z + wx\bar{y} + wyz + \bar{w}xy$ (unique minimal expression)

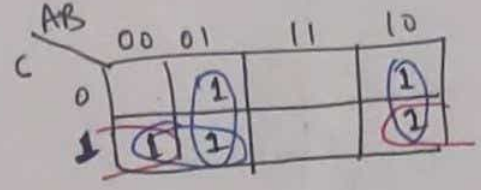
⇒ Now, check whether the above expression is minimal or not then we go for Tabulation method.

| | 1 | 5 | 6 | 7 | 11 | 12 | 13 | 15 |
|---|---|---|---|---|---|---|---|---|
| * $\bar{w}\bar{y}z$ | 1 | 1 | | | | | | |
| * $wx\bar{y}$ | | | | | | 1 | 1 | |
| * $wyz$ | | | | | 1 | | | 1 |
| * $\bar{w}xy$ | | | 1 | 1 | | | | |
| $xz$ | | 1 | | 1 | | | 1 | 1 |

- Now, $\bar{w}\bar{y}z$ covers the boxes 1,5 in k-map, so that put 1,1 in 1,5 column & $\bar{w}\bar{y}z$ row & max 1,5 (means they are covered)
- Similarly do for all implicants (Rows)
- Now check 1 minterm (1 column), It is covered only with $\bar{w}\bar{y}z$ so (↑ Column numbered 1) it is a prime implicant that is essential so put * on $\bar{w}\bar{y}z$
- similarly the min term '6' is covered by $\bar{w}xy$ so, $\bar{w}xy$ is the essential prime implicant
- Similarly analyze each column & mark the essential prime implicants & also marks the minterms that are also covered by the prime implicants, If all the minterms are covered then check *'s on the Rows & they wel form minimal Expression

# Minimal SOP example:

which of the following prime implicants are essential?




(a) $B'C$, $A'B$

(b) $A'C$, $A'B$

(c) $A'B$, $AB'$

(d) $A'B$, $AB'$, $B'C$

Now if they have asked How many minimal expressions are these, then how to find is:-

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| *$A'B$ | | (1) | 1 | | |
| *$AB'$ | | | | (1) | 1 |
| $A'C$ | (1) | · | 1 | | |
| $B'C$ | (1) | | | | 1 |
| | | | | | |

Essential prime implicants

Essential probable Candidates

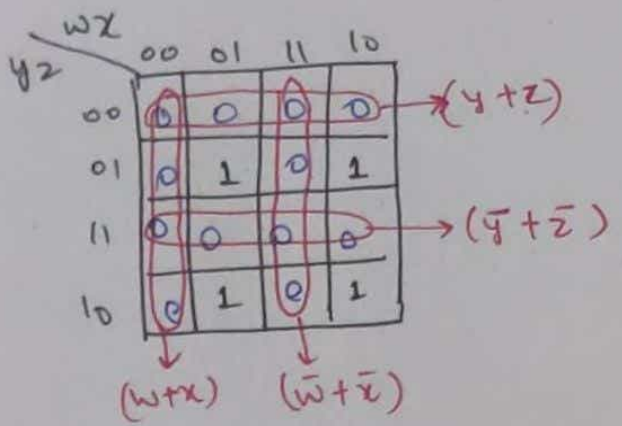
$\therefore F = \bar{A}B + A\bar{B} + \bar{A}C$

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

} 2 minimal expressions

## Minimal pos

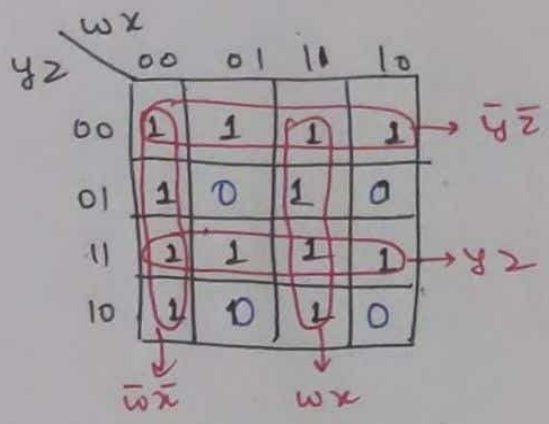
① $F(w,x,y,z) = \Sigma(5,6,9,10) = \Pi(0,1,2,3,4,7,8,11,12,13,14,15)$




$\therefore F = (y+z)(\bar{y}+\bar{z})(w+x)(\bar{w}+\bar{x})$ [POS]

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

② $F'(w,x,y,z) = \Sigma(0,1,2,3,4,7,8,11,12,13,14,15) = \Pi(5,6,9,10)$



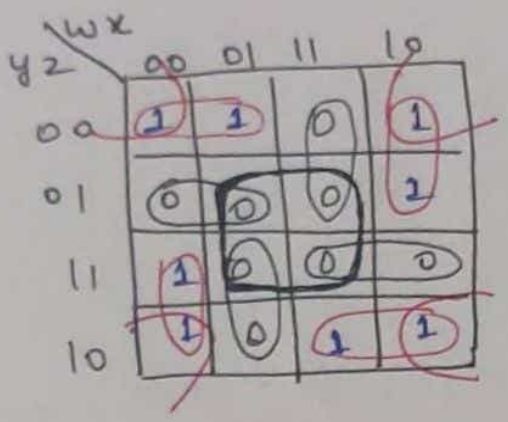

$\therefore F = \bar{y}\bar{z} + \bar{w}\bar{x} + yz + wx$ [SOP]

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

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

Find number of literals in minimum POS & SOP

for $F(w,x,y,z) = \Pi(1,5,6,7,11,12,13,15)$




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

⇒ 12 literals

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

⇒ 12 literals

# Introduction to DONT Cares :

(1) A function is said to be completely specified if it is given '0' or '1' for every combination of input variables. There are some function exists which are not completely specified.

(2) Combinations for which the value of a function is not specified are called Don't care combinations

(3) The value of a function for such combination is denoted by '$\phi$' or 'd'.

(4) since, each Dont care combination represents 2 values $\{0,1\}$ an incompletely specified function containing k-dont care combinations corresponds to a class of $2^k$ distinct functions.

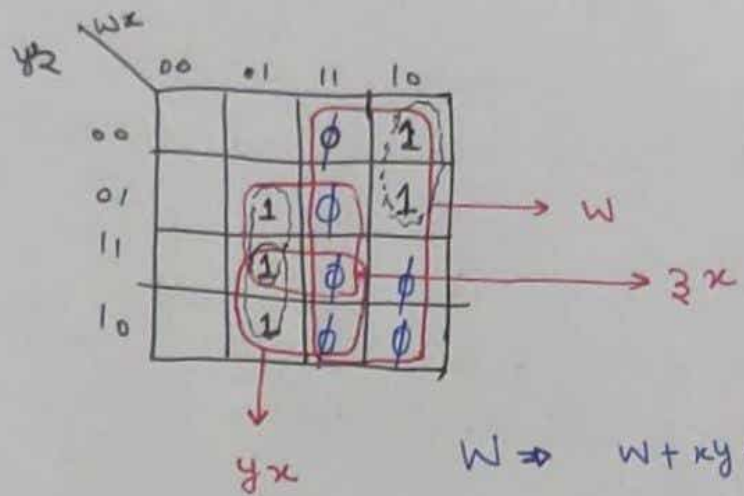
(5) Our task is to choose the function having minimal representation out of those $2^k$ functions

(6) we could assign a '0' or '1' to a '0' or '1' to a dont care combination in such a way to increase or decrease the size of a subcube.

Total 4 Possibilities

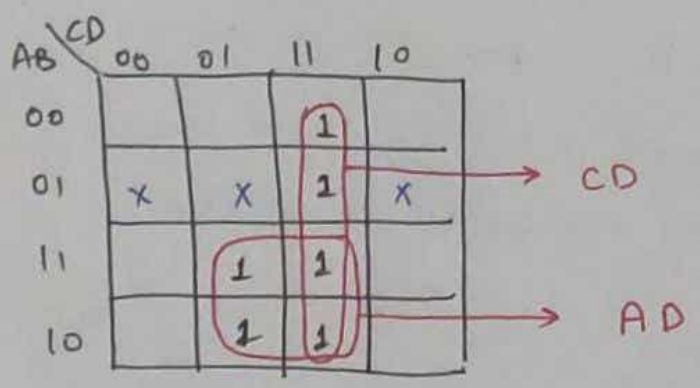
| a | b | f | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | $\phi$ | 0 | 0 | 1 | 1 |
| 1 | 1 | $\phi$ | 0 | 1 | 0 | 1 |

Q) $W = \Sigma(5,6,7,8,9) + \phi(10,11,12,13,14,15)$



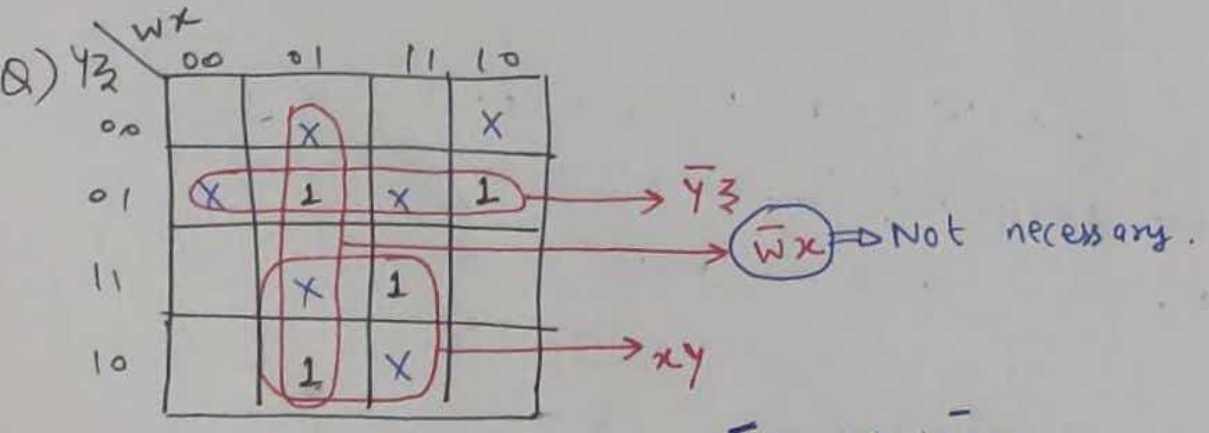

$W \Rightarrow \quad w + xy + xz \quad [SOP]$

Q)



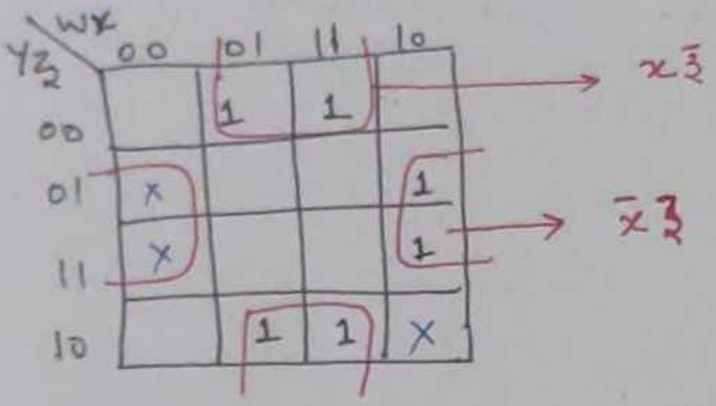

$F = AD + CD$

Q)



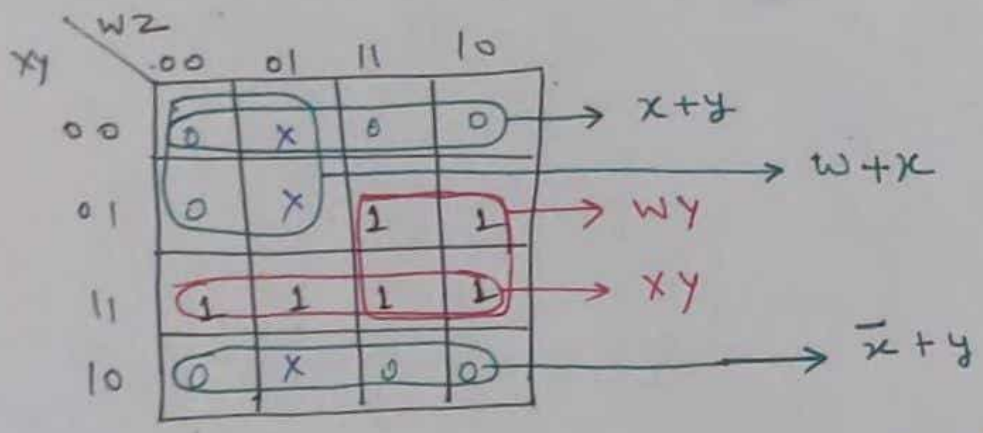

$F = xy + \bar{y}z$

Q)




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

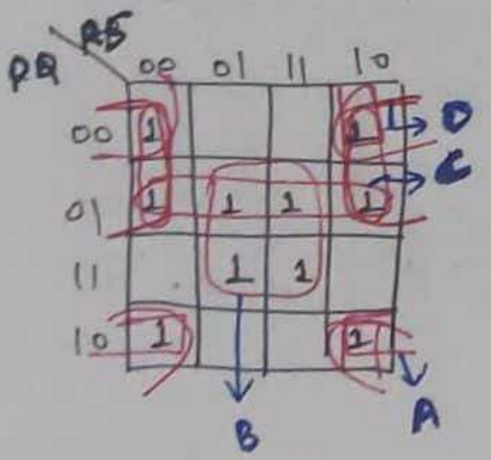
Q)




$F = WY + XY$ [SoP]

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

Q) Finding minimal expressions in the following?

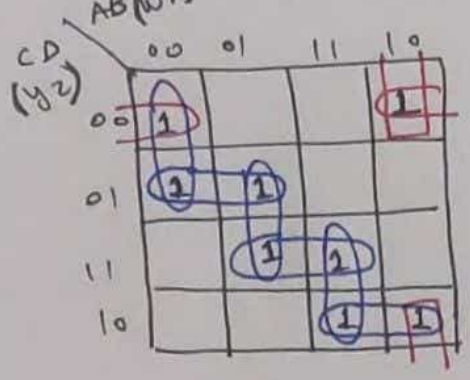



| | 0 | 1 | 2 | 5 | 7 | 8 | 9 | 10 | 13 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|
| *A | 1 | | 1 | | | 1 | | 1 | | |
| *B | | | | 1 | 1 | | | | 1 | 1 |
| C | | 1 | | 1 | | | 1 | | 1 | |
| D | 1 | 1 | | | | 1 | 1 | | | |

Minimum expression = $\frac{A+B+C}{\text{or}}$ $A+B+D$

# Branching technique for minimizing cyclic functions:

$F = \Sigma(0,1,5,7,8,10,14,15)$

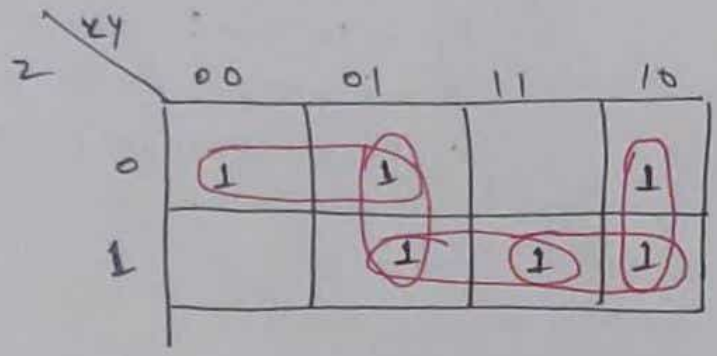



| | 0 | 1 | 5 | 7 | 8 | 10 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|
| ✓ $A = \bar{w}\bar{x}\bar{y}$ | X | X | | | | | | |
| $B = \bar{w}\bar{y}z$ | | X | X | | | | | |
| ✓ $C = \bar{w}xz$ | | | X | X | | | | |
| $D = xyz$ | | | | X | | | | X |
| ✓ $E = wxy$ | | | | | | | X | X |
| $F = wy\bar{z}$ | | | | | | X | X | |
| ✓ $G = w\bar{x}\bar{z}$ | | | | | X | X | | |
| $H = \bar{x}\bar{y}\bar{z}$ | X | | | | X | | | |

- Now, if I choose 'A' & include it in minimal expression then 'A' [included in ME]
- Now 0,1 minterms are covered by 'A'. so delete A row & 0,1 columns.
- Now, we can observe B→C (B is covered by C ➡ B is irredundant) & **H → G** (**H** is covered by **G** → **H** is irredundant)

⇒ Now, (G is covering by it ⇒ it is irredundant).
So delete, B, H rows & continue the procedure & mark the essential prime implicants

Hence, the minimal eqns will be A+C+G+E.




$F = \Sigma(0,2,3,4,5,7)$

| | 0 | 2 | 3 | 4 | 5 | 7 |
|---|---|---|---|---|---|---|
| $\bar{x}\bar{z}$ = A | X | X | | | | |
| $\bar{x}y$ = B | | X | X | | | |
| $y\bar{z}$ = C | | | X | | | X |
| $xz$ = D | | | | | X | X |
| $x\bar{y}$ = E | | | | X | X | |
| $\bar{y}\bar{z}$ = F | X | | | X | | |

F ⇒ A + C + E

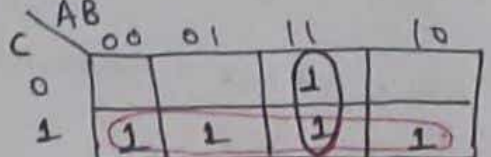
F → E { Hence, F & B is not needed }

B → C

# Difference b/w Implicants & prime implicants:

Q) which of the following can be prime implicant of a function f(a,b,c) ?

(a) ab+c
(b) bc+a
(c) ac+b
(d) ab



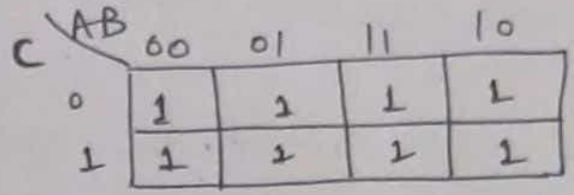
Q) which of the following can't represent prime Implicants on set of $f(A,B,C)$

(a) $\{ab, bc, ca\}$

(b) $\{\bar{a}, b, ac\}$

(c) $\{b\bar{c}a, ba, bc\}$

(d) $\{\bar{a}\bar{b}, ab\}$




$\{ab, bc, ca\}$

If ab is prime implicant then anything containing 'ab' should not be present.

$ba\bar{c}, ab, bc$

$ab\bar{c}, ab, bc$

Hence, not Prime Implicant

## Converting a function into self dual:

Q) what minterms have to be added to make the following function asself dual

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

Note:-
(0,15) (1,14) (2,13) (3,12)
(4,11) (5,10) (6,9) (7,8)

$= A\bar{B}\bar{C} + \bar{A}C\bar{D} + B\bar{D}$ terms should not be added as these are mutual exclusive to the given terms

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

$= \bar{A}BCD + \bar{A}BC\bar{D} + ABCD + A\bar{B}CD +$

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

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

7 6 15 11 13 5

8 9 0 4 2 10

The left out pairs are:
(1,14) (3,12)
= 2 choices * 2 choices
⇒ 4 minterms can be added.

{(1,3) (14,3)
{(1,12) (14,12)

## # Combining functions having Dont care:

Q) How many functions does $f_1 \cdot f_2$ & $f_1 + f_2$ represent?

$$f_1(a,b,c) = \Sigma(0,2,4) + \phi(3,5,7)$$

$$f_2(a,b,c) = \Sigma(2,3) + \phi(1,6,7)$$

| A | $f_1$ | $f_2$ | $f_1 \cdot f_2$ | $f_1 + f_2$ |
|---|---|---|---|---|
| 000 → 0 | 1 | 0 | 0 | 1 |
| 001 → 1 | 0 | $\phi$ | 0 | $\phi$ |
| 010 → 2 | 1 | 1 | 1 | 1 |
| 011 → 3 | $\phi$ | 1 | $\phi$ | 1 |
| 100 → 4 | 1 | 0 | 0 | 1 |
| 101 → 5 | $\phi$ | 0 | 0 | $\phi$ |
| 110 → 6 | 0 | $\phi$ | 0 | $\phi$ |
| 111 → 7 | $\phi$ | $\phi$ | $\phi$ | $\phi$ |

4 Dont cares
⇒ $2^4$
⇒ 16 functions

2 Dont cares ($\phi, \phi$)

Can take 2 values (0,1)

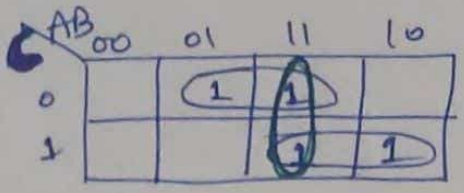
Can take 2 values (0,1)

⇒ $2^2$
⇒ 4 functions

Hence,
$f_1 \cdot f_2 \Rightarrow 4$
$f_1 + f_2 \Rightarrow 16$

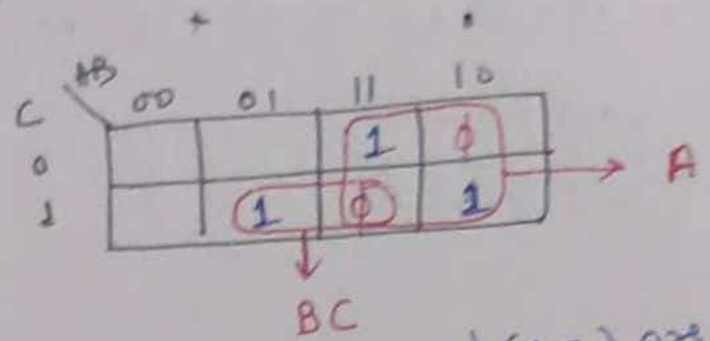
Q) What are the number of prime implicants, Essential prime implicants & Redundant prime implicants for the function

$$f(A,B,C) = \Sigma(2,5,6,7)$$




No of implicants = 3
Essential = 2
Redundant = 1

Q) A function $f(A,B,C) = \Sigma(3,5,6)$ is minimised to $(A+\bar{B})$ then what are Dont cares?




$\Rightarrow \phi(4,7)$ are the Don't cares

$\Rightarrow \Sigma_{\phi}(4,7)$

- Number of minimal expressions:

In a K-map it was found that all the essential prime implicants are covering all terms except 2 min terms These 2 minterms are in turn covered by 3-non-essential prime implicants each what is the number of minimal SoP expressions?

The Sol$^n$ is EPI + ___ + ___

↑ 3 choices ↑ 3 choices

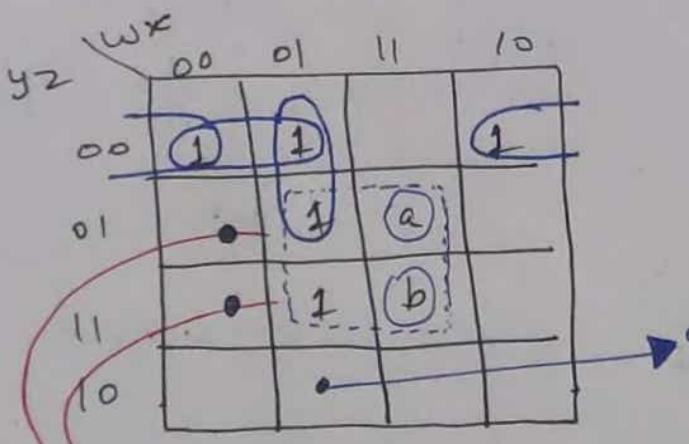
$\therefore$ 9 minimal expressions are possible

• Beautiful question on prime implicant chart:

In a prime implicant chart representing boolean expression $f(w,x,y,z)$ columns represent minterms & rows represent Prime Implicants, Identify P, Q, R, S & a, b?

| | | 0 | 4 | 5 | 7 | 8 | a | b |
|---|---|---|---|---|---|---|---|---|
| $\bar{w}\bar{y}\bar{z}$ | P | ✓ | ✓ | | | | | |
| $\bar{x}\bar{y}\bar{z}$ | Q | ✓ | | | | ✓ | | |
| $\bar{w}x\bar{y}$ | R | | ✓ | ✓ | | | | |
| $xz$ | S | | | ✓ | ✓ | | ✓ | ✓ |

$\therefore a=13, b=15$




a, b can't be present here, because if they are present then it should cover the minterm 4 but if we check 's' it does not covers the minterm 4

a, b can't be in this positions because if they were present 'p' could not become a prime implicant

# Variable entered Map (VEM)

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

| A | B | C | f |
|---|---|---|---|
| 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 | 0 |
| 1 | 1 | 1 | 1 |

So, for the given values of A, B, the function 'f' behaves exactly as 'C'.

| A | B | f |
|---|---|---|
| 0 | 0 | C |
| 0 | 1 | C |
| 1 | 0 | C |
| 1 | 1 | C |

variable for $\Sigma(1,2,5,6)$ entered Map (VEM)

A=0, B=0 ← C

A=0, B=1 ← $\bar{C}$

A=1, B=0 ← C

A=1, B=1 ← $\bar{C}$

Ily for $f = \Sigma(0,2,5,6)$ ⇒ behaves as $\Sigma(1,3,5,7)$ with complement

$f = \Sigma(1,2,3,5,6)$ ⇒ behaves as $(C, 1, C, \bar{C})$

A=0, B=0 ; A=0, B=1 ; A=1, B=0 ; A=1, B=1

## Minimization using VEM:

1. Set all the variables in a cell as '0' & obtain the SOP expression
2. (a) Make one variable in the cell as '1' & obtain SOP by making earlier minterms (1's) as don't cares.
   (b) Multiply the above SOP with the concerned variable
3. Repeat the step-2 until all the variables in the cell are covered.
4. SOP of VEM is obtained by ORing the previous SOP expression.

| C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | D' | D' |
| 1 | 0 | 1 | 0 | ϕ |

→ If a k-map is used then it contains $4 \times 4 = 16$ entries

↳ It is 4-variable map & hence the function behaves as one of the variable for a given set of other variables. Here 'f' behaves as $D/\bar{D}$ for given set of inputs ABCD.

Step 1 Replace the variables with 0's & obtain the SoP expression

| C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | ϕ |

→ $\bar{A}B$

Step-2(a) Replace any one of the variables D or D' with 1's & replace the minterms (cells having ones) with Don't cares (ϕ) [D replaced]

| C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | ϕ | 0 | 0 |
| 1 | 1 | ϕ | 0 | ϕ |

→ $\bar{A}$

Step-2(b) Now, we have replaced 'D' in step2 (a) now replace D' with 1's & 1's should be made as Don't cares & remaining cells are '0'. [D' replaced]

| C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | ϕ | 1 | 1 |
| 1 | 0 | ϕ | 0 | ϕ |

→ $A\bar{C}$

Step-3

$F \Rightarrow \bar{A}B + \bar{A}D + A\bar{C}\bar{D}$ ie [step1 + step2(a) * D + step2(b) * D' ]

Q) $f(A,B,C) = \Sigma(3,5,6,7)$ is realised by following VEM, then find P, Q, R, S.

| B \ A | 0 | 1 |
|---|---|---|
| 0 | P | S |
| 1 | R | Q |

(a) P = 0, R = S = C, Q = 1

(b) P = Q = 0, R = C, S = 1

(c) P = 0, Q = R = C, S = 1

(d) None

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

| 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 |

Now,

| B \ A | 0 | 1 |
|---|---|---|
| 0 | 0 | C |
| 1 | C | 1 |

Hence, P = 0

S = C

R = C

Q = 1

Step-1

| b \ a | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

ab

Step-2

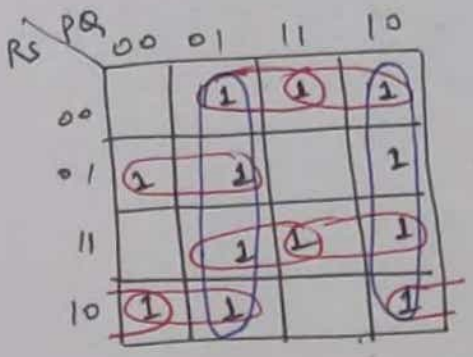
| b \ a | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 1 |

$\Rightarrow (a+b) * c$

$= ac + bc$

Step-3 $ab + bc + ca$

# Problem on k-map




10 prime implicants
& 12 Minterms

How many minimal functions are possible?

| | 1 | 2 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\bar{P}Q$ | | | ✓ | ✓ | ✓ | ✓ | | | | | | |
| $P\bar{Q}$ | | | | | | | ✓ | ✓ | ✓ | ✓ | | |
| $Q\bar{R}\bar{S}$ | | | ✓ | | | | | | | | ✓ | |
| $P\bar{R}\bar{S}$ | | | | | | | ✓ | | | | ✓ | |
| $\bar{P}RS$ | ✓ | | | ✓ | | | | | | | | |
| $\bar{Q}RS$ | ✓ | | | | | | | ✓ | | | | · |
| $QRS$ | | | | | | ✓ | | | | | | ✓ |
| $PRS$ | | · | | | | | | | | ✓ | | ✓ |
| $\bar{Q}R\bar{S}$ | | ✓ | | | | | | | ✓ | | | |
| $\bar{P}R\bar{S}$ | | ✓ | | · | ✓ | | | | | | | |

Now, if include $\Rightarrow \bar{P}Q$ then (4, 5, 6, 7) will be covered
if include $\Rightarrow P\bar{Q}$ then (8, 9, 10, 11) will be covered

Now, $\bar{Q}\bar{R}\bar{S}$, $\bar{P}\bar{R}\bar{S}$ → 2 ways

$\bar{P}RS$, $\bar{Q}RS$ → 2 ways

$QRS$, $PRS$ → 2 ways

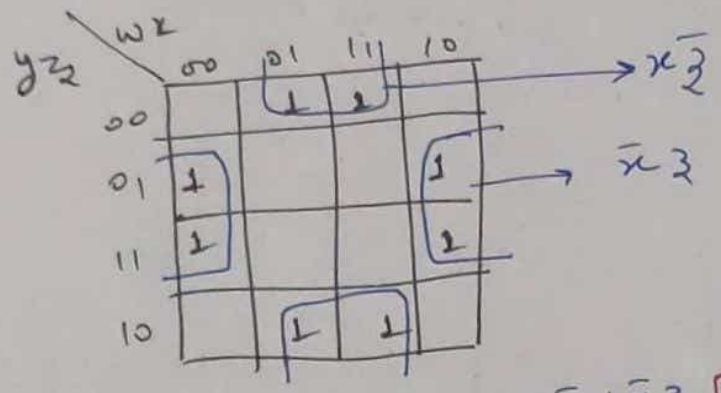
$\bar{Q}RS$, $\bar{P}R\bar{S}$ → 2 ways

$2^4 \Rightarrow 16$ functions are possible

Hence, $\bar{P}Q + P\bar{Q} + (Q\bar{R}\bar{S}, P\bar{R}\bar{S}) + (\bar{P}RS, \bar{Q}RS)$
(4,5,6,7) (8,9,10,11) (12) (1)
$+ (QRS, PRS) + (\bar{Q}\bar{R}\bar{S}, \bar{P}\bar{R}\bar{S})$
(15) (2)

## # finding free variables:

Q) How many variables are free in the expression denoted by the following k-map?

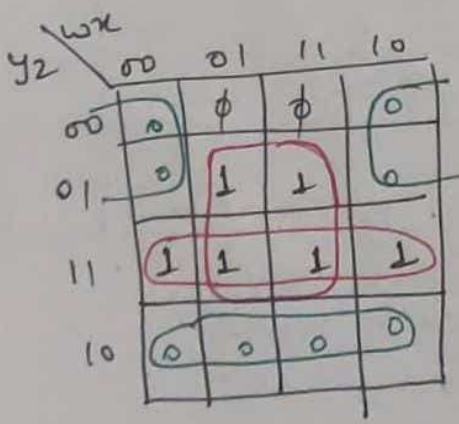



$F \Rightarrow (x+\bar{z})(\bar{x}+z)$ [POS]

$F = x\bar{z} + \bar{x}z$ [SoP]

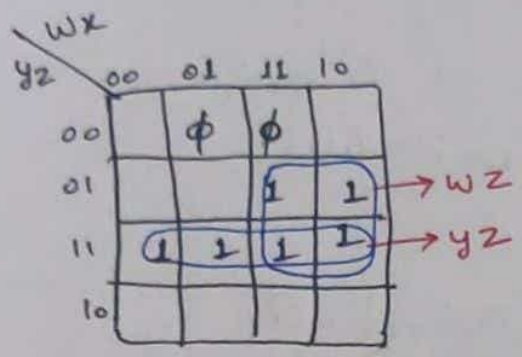
∴ (w, y) are not required

## Relationship in b/w Minimal POS, SOP in case of Dont Cares:




$F = yz + xz$ [SoP]

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

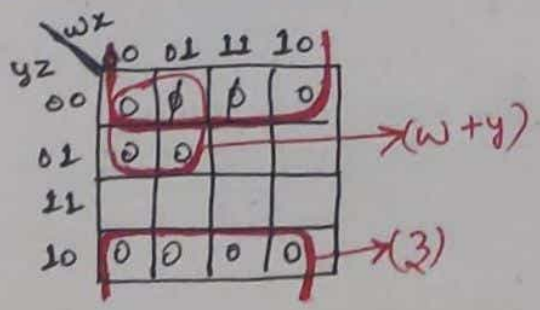



$\Rightarrow wz + yz = f1$

| S.No | w | x | y | z | f | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 0 | 0 | $\phi$ | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 10 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 12 | 1 | 0 | 1 | 1 | $\phi$ | 1 | 1 | 1 | 1 |
| 13 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 14 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 15 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 16 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

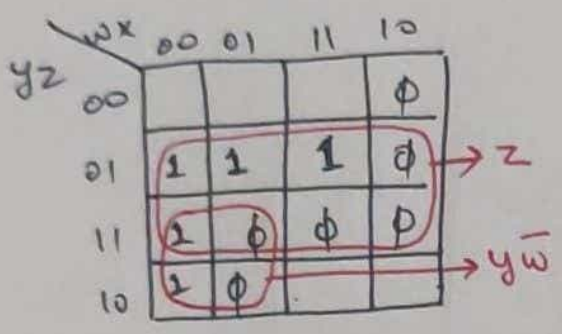
~~f~~

$f_1$ $f_2$
$f_3$ $f_4$



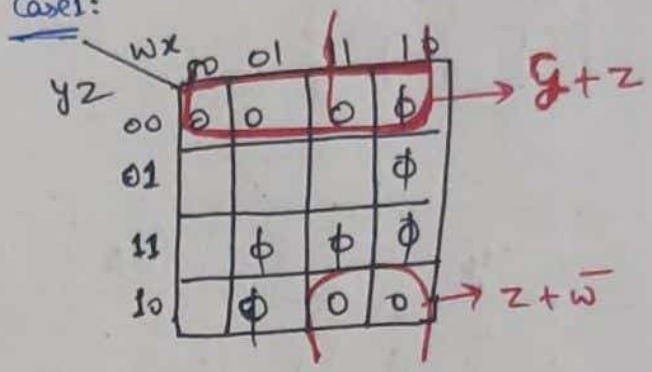

$\Rightarrow (w+y)(z) = f1$

Let $t(w,x,y,z) = \Sigma(1,2,3,5,13) + \Sigma\phi(6,7,8,9,11,15)$ & let $f(w,x,y,z)$ be the minimal SOP & $g(w,x,y,z)$ be the minimal POS. Are f & g are identical?




$f = \Sigma(1,2,3,5,6,7,9,11,13,15)$

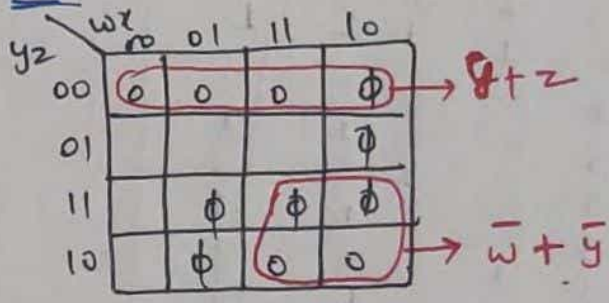
Case1:




$g = \Pi(0,4,10,8,12,14)$

$= \Sigma(1,2,3,5,6,7,9,11,13,15)$

Case2:




$g = \Pi(0,4,8,10,11,12,14,15)$

$= \Sigma(1,2,3,5,6,7,9,13)$

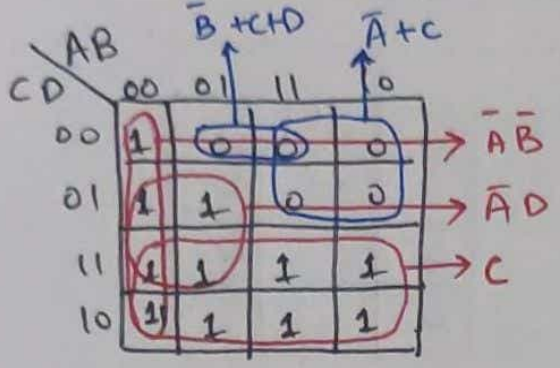
∴ F & G are not identical in case of inclusion of Don't cares

Here, There are 6 Don't Cases, so $2^6$ functions are possible [As each Don't Cases has two choices either 0 or 1]

→ $2^6$ functions Are possible

# Comparing independent variables in minimal SOP & POS :

Q Minimal expression represented by the k-map is free from?



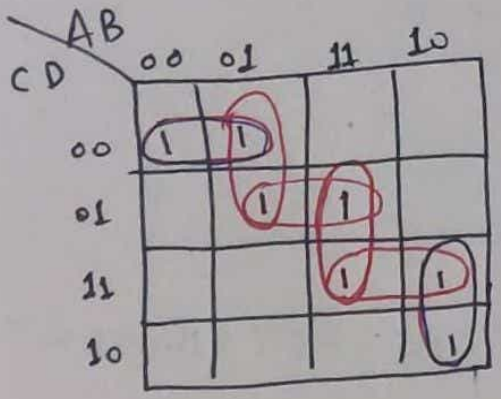

$SOP = \bar{A}\bar{B} + \bar{A}D + C$

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

(a) 1-variable
(b) 2-variable
(c) 3-variable
(d) Dependent on all

Both SOP as well as POS are dependent on all variables ie A, B, C, D. Hence, Minimal expression is dependent on all the variables

# Number of irredundant & minimal Expression :-

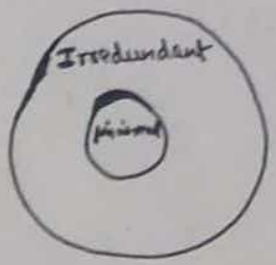



(1) Number of prime Implicants ? 6 (red + blue)
(2) Number of Essential prime Implicants ? 2 (blue)
(3) Number of Redundant prime Implicants ? 4 (red)
(4) Minimal SOP ?
(5) How many minimal Expressions are there?

| | 0 ✓ | 4 ✓ | 5 | 10 ✓ | 11 ✓ | 13 | 15 |
|---|---|---|---|---|---|---|---|
| P = $\bar{A}\bar{C}\bar{D}$ | ✓ | ✓ | | | | | |
| Q = $\bar{A}B\bar{C}$ | | ✓ | ✓ | | | | |
| R = $B\bar{C}\bar{D}$ | | | ✓ | | | ✓ | |
| S = ABD | | | | | | ✓ | ✓ |
| T = ACD | | | | | ✓ | | ✓ |
| V = $A\bar{B}\bar{C}$ | | | | ✓ | ✓ | | |

Q is dominated by R [Q → R]
T is dominated by S [T → S]

F = P + V + R + S (minimal SOP)




i.e. All minimal expressions are Irredundant but all Irredundant expressions are not minimal.

5 (Q+R) 0 1 — 13 (R+S) 0 1 — 15 (S+T) 0 1

= QRS + QRT + QSS + QST + RRS + RRT + RSS + RST

= QRS + QRT + QS + QST + RS + RT + RS + RST

= QRS + RS + QS + QST + RT + RST + QRT

= RS + QS + RT + QRT

= RS + QS + RT

∴ P + V + Q + S
P + V + R + T
P + V + R + S
} 3 Irredundant minimal Expression

$$f(V,W,X,Y,Z) = \Sigma(13,15,17,18,19,20,21,23,25,27,29,31) + \Sigma_{\phi}(1,2,12,24)$$

| | 13 | 15 | 17 | 18 | 19 | 20 | 21 | 23 | 25 | 27 | 29 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A = VZ | | | ✓ | | ✓ | . | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| B = WXZ | ✓ | ✓ | | | | | | | | | ✓ | ✓ |
| C = $VW\bar{X}\bar{Y}$ | | | | | | ✓ | ✓ | | ✓ | | | |
| D = $V\bar{W}XY$ | | | | ✓ | ✓ | ✓ | ✓ | | | | | |
| E = $V\bar{W}\bar{X}Y$ | | | | ✓ | ✓ | | | | | | | |
| F = $\bar{V}WX\bar{Y}$ | ✓ | | | | | | | | | | | |
| G = $\bar{W}\bar{X}Y\bar{Z}$ | | | | ✓ | | | | | | | | |
| H = $\bar{W}\bar{X}\bar{Y}Z$ | | | ✓ | | | | | | | | | |

A + B + D + E
A + B + D + G
} Minimal as well as Irredundant

Don't cares are not included in the prime Implicant chart

# f(vwxyz)

| | 0 | 1 | 3 | 4 | 7 | 13 | 15 | 19 | 20 | 22 | 23 | 29 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A = wxz | | | | | | ✓ | ✓ | | | | | ✓ | ✓ |
| B = xyz | | | | | ✓ | | ✓ | | | | ✓ | | ✓ |
| C = $\bar{w}yz$ | | | ✓ | | ✓ | | | ✓ | | | ✓ | | |
| D = $v\bar{w}xy$ | | | | | | | | | | ✓ | ✓ | | |
| E = $w\bar{x}\bar{z}$ | | | | | | | | | ✓ | ✓ | | | |
| F = $\bar{w}x\bar{y}\bar{z}$ | | | | ✓ | | | | | ✓ | | | | |
| G = $\bar{v}\bar{w}\bar{x}z$ | | ✓ | ✓ | | | | | | | | | | |
| H = $\bar{v}\bar{w}\bar{y}\bar{z}$ | ✓ | | | ✓ | | | | | | | | | |
| I = vwxy | ✓ | ✓ | | | | | | | | | | | |

0 1 4 20 22

(I+H) (G+I) (F+H) (E+F) (D+E)

⇒ (I+GH) (F+H) (E+FD)

0 1 0 1 0 1

000 001 010 011 100 101 110 111

⇒ IFE + IFFD + IHE + IHFD + GHFE + GHFFD + GHHE + GHHFD

⇒ IFE + IFD + IHE + IHFD + GHFE + GHFD + GHE + GHFD

⇒ IFE + IFD + IHFD + IHE + GHEF + GHE + GHFD

⇒ IFE + IFD + IHE + GHE + GHFD

A + c + I + F + E
A + c + I + F + D
A + c + I + H + E
A + c + GH + E

} 4 minimal expression which are Irredundant

A + c + G + H + F + D

# Consider 3 variable functions

$f_1(A,B,C) = \Sigma(0,7) + \Sigma\phi(1,2,5)$

$f_2(A,B,C) = \Sigma(0,1,3) + \Sigma\phi(4,7)$

Find $f_1 \cdot f_2$ and $f_1 + f_2$ when minimized?

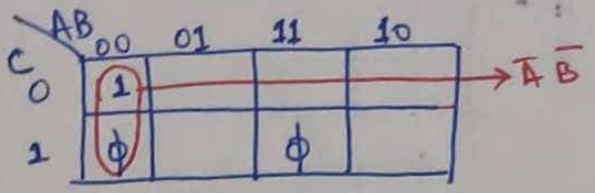
Method-1:

| | $f_1$ | $f_2$ | $f_1 \cdot f_2$ | $f_1 + f_2$ |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 |
| 1 | $\phi$ | 1 | $\phi$ | 1 |
| 2 | $\phi$ | 0 | 0 | $\phi$ |
| 3 | 0 | 1 | 0 | 1 |
| 4 | 0 | $\phi$ | 0 | $\phi$ |
| 5 | $\phi$ | 0 | 0 | $\phi$ |
| 6 | 0 | 0 | 0 | 0 |
| 7 | 1 | $\phi$ | $\phi$ | 1 |

$1 + \phi = 1 \quad 1 \cdot \phi = \phi$

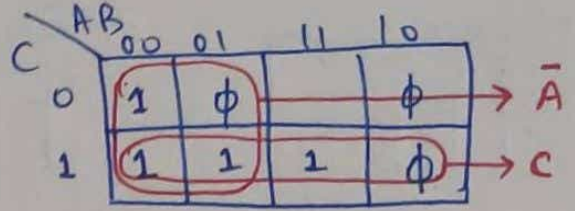
$0 + \phi = \phi \quad 0 \cdot \phi = 0$

$f_1 \cdot f_2$




$\Rightarrow \bar{A}\bar{B}$

$f_1 + f_2$



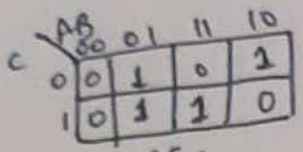

$\Rightarrow \bar{A} + C$

Method-2:

$f_1 \cdot f_2 = \Sigma(0) + \Sigma\phi(1,7)$

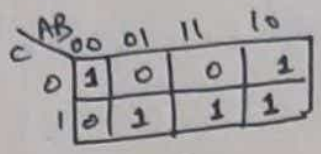
$f_1 + f_2 = \Sigma(0,1,3,7) + \Sigma\phi(2,5,4)$

Q Consider a new boolean operation '$' defined as $A \$ B = \bar{A} + B$, then find $f_1 \$ f_2$



| C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |

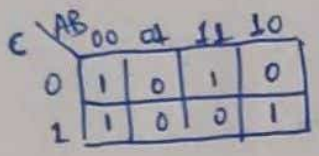
($f_1$)



| C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |

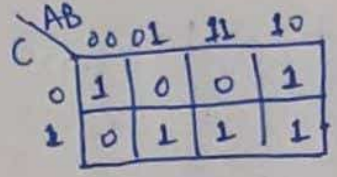
($f_2$)

$f_1 \$ f_2 = \bar{f_1} + f_2$



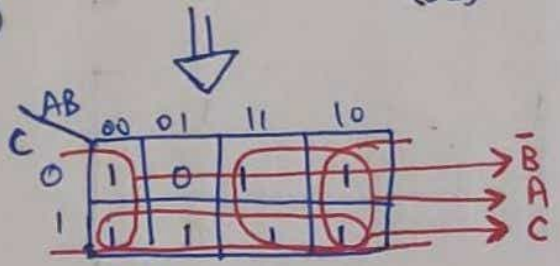
| C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

($\bar{f_1}$)



| C \ AB | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |

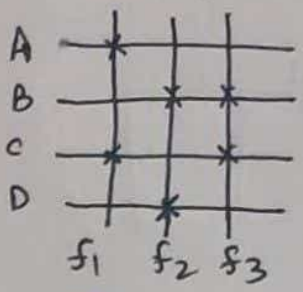
($f_2$)



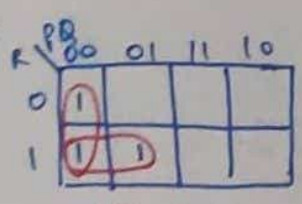

($\bar{f_1} + f_2$)

$\Rightarrow \underline{A + \bar{B} + C}$

Q Consider three-three variable functions $f_1(P,Q,R) = \Sigma(0,1,3)$, $f_2(P,Q,R) = \Sigma(3,5,7)$, $f_3(P,Q,R) = \Sigma(1,3,7)$. These functions are sharing 4 prime implicants A, B, C, D (all of them are pairs) as shown

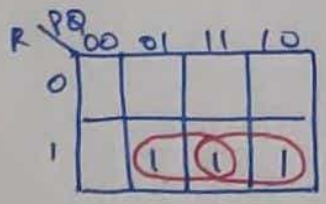



Method-1:



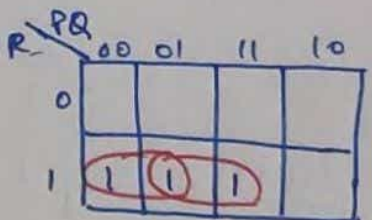
| R \ PQ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | | | |
| 1 | 1 | 1 | | |

$f_1 = \bar{P}\bar{Q} + \bar{P}R$



| R \ PQ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | 1 | 1 |

$f_2 = QR + PR$

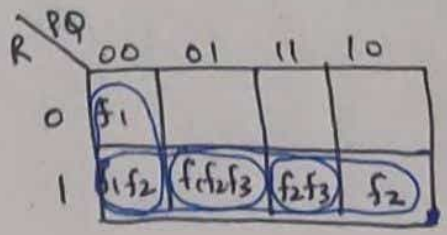


| R \ PQ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | 1 | 1 | 1 | |

$f_3 = \bar{P}R + QR$

Hence, $C = \bar{P}R$, $B = QR$, $D = PR$, $A = \bar{P}\bar{Q}$

Method-2:

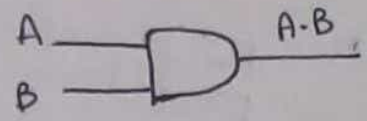


# Introduction to Logic Design

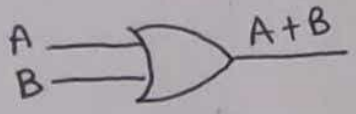
## Logic Design:

- The main application of switching theory is in the design of digital circuits. It is called logic design

- These circuits are designed using basic elements called GATES

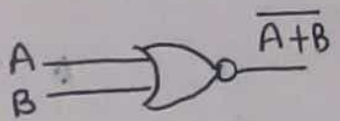



[AND] → $A \cdot B$

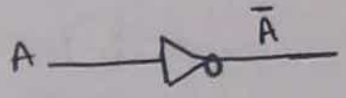



[OR] → $A+B$

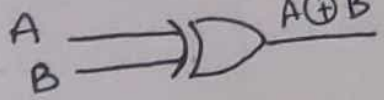

[NAND] → $\overline{A \cdot B}$



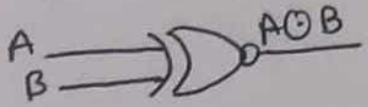

[NOR] → $\overline{A+B}$




[NOT] → $\bar{A}$

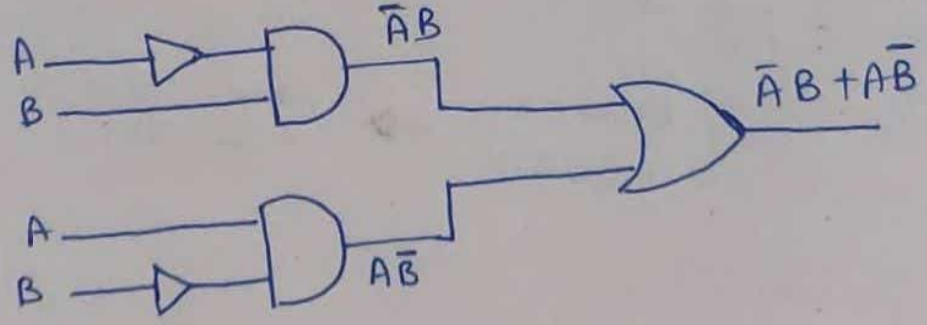



[EX-OR] → $\bar{A}B + A\bar{B}$




[EX-NOR] → $\bar{A}\bar{B} + AB$

Design a circuit for $\bar{A}B + A\bar{B}$ (ie Ex-OR) with NOT, AND & OR Gate only.



If we are using less than 10 gates ⇒ Small scale Integration (SSI)

10-100 gates ⇒ Medium scale Integration (MSI)

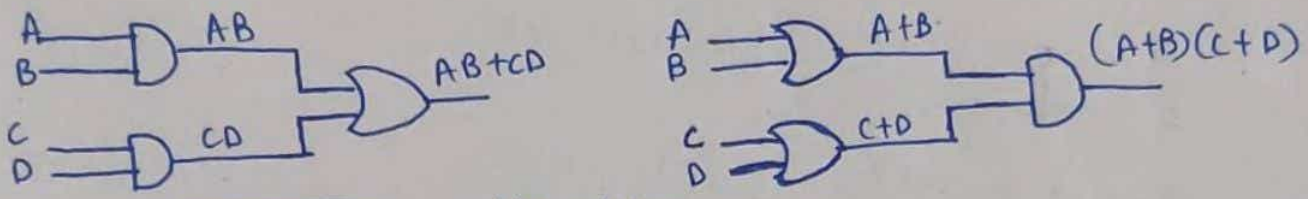
100-1000 gates ⇒ large scale Integration (LSI)

>1000 gates ⇒ very large scale Integration (VLSI)

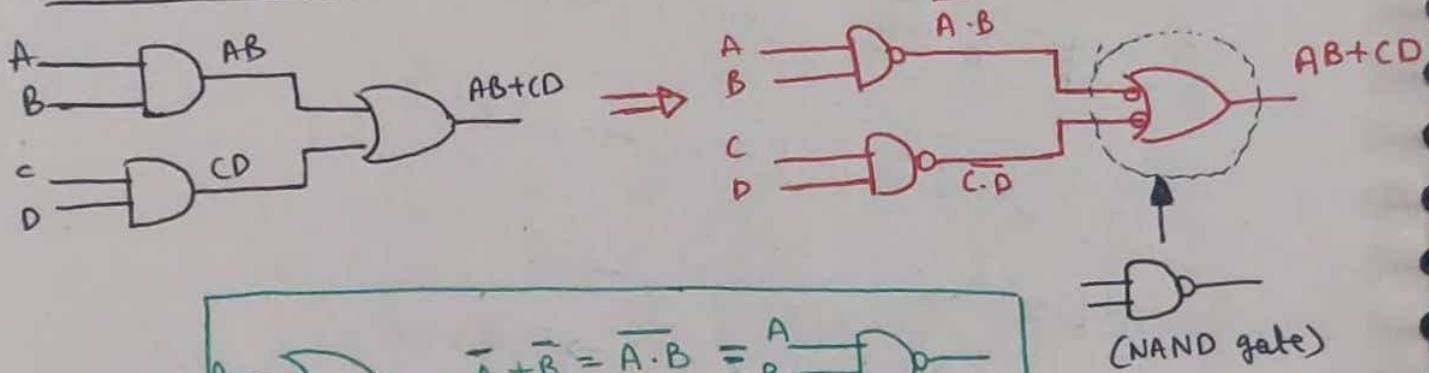
✱ AND-OR Realization ← Implementation.

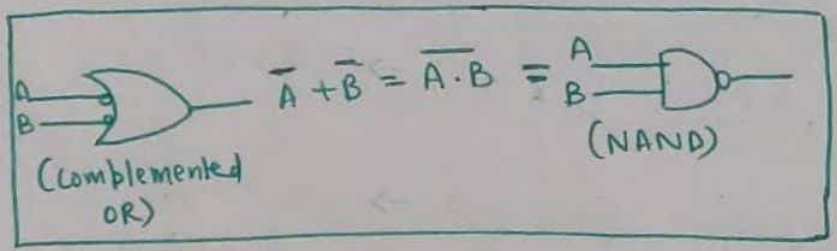
OR-AND Realization ← Implementation.

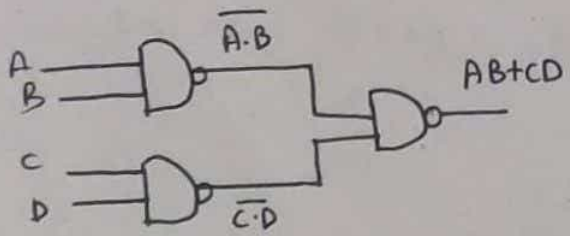
$f(A,B,C,D) = AB+CD$ [SOP] $\quad f(A,B,C,D) = (A+B)(C+D)$ [POS]




① AND-OR Realization ← Implementation

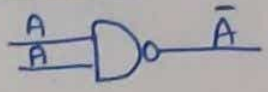


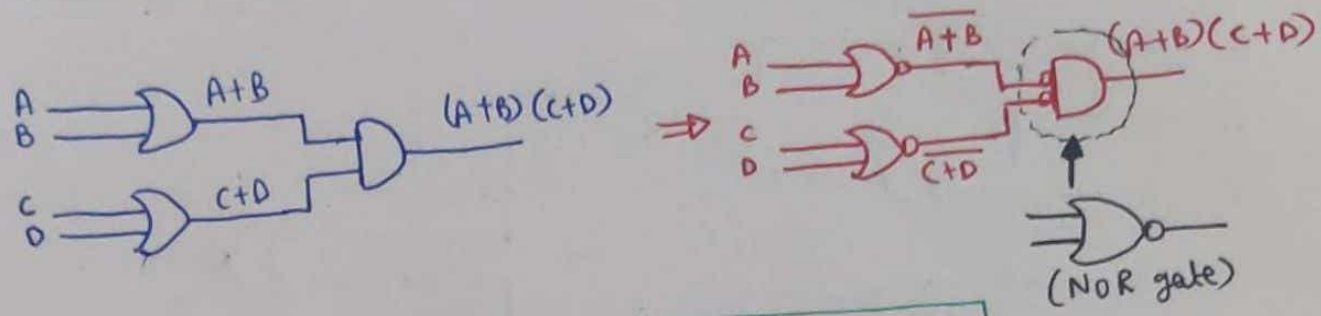



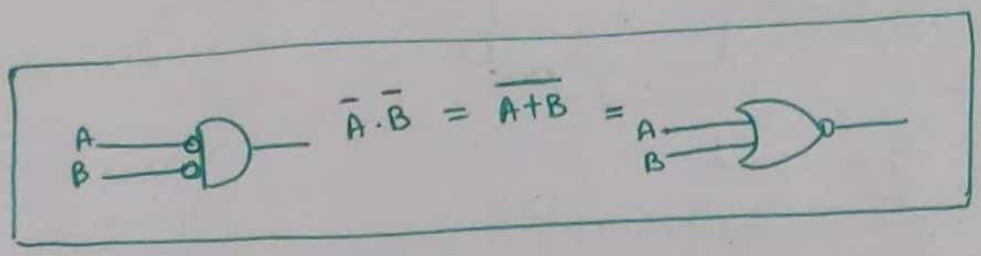

AND-OR Realization ⇒ NAND-NAND Realization

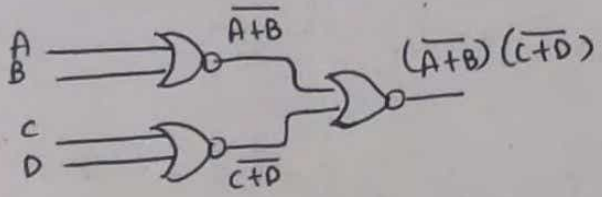
NOT using NAND gate



(2) OR-AND Realization ← Implementation

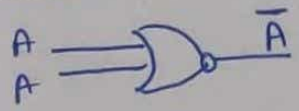


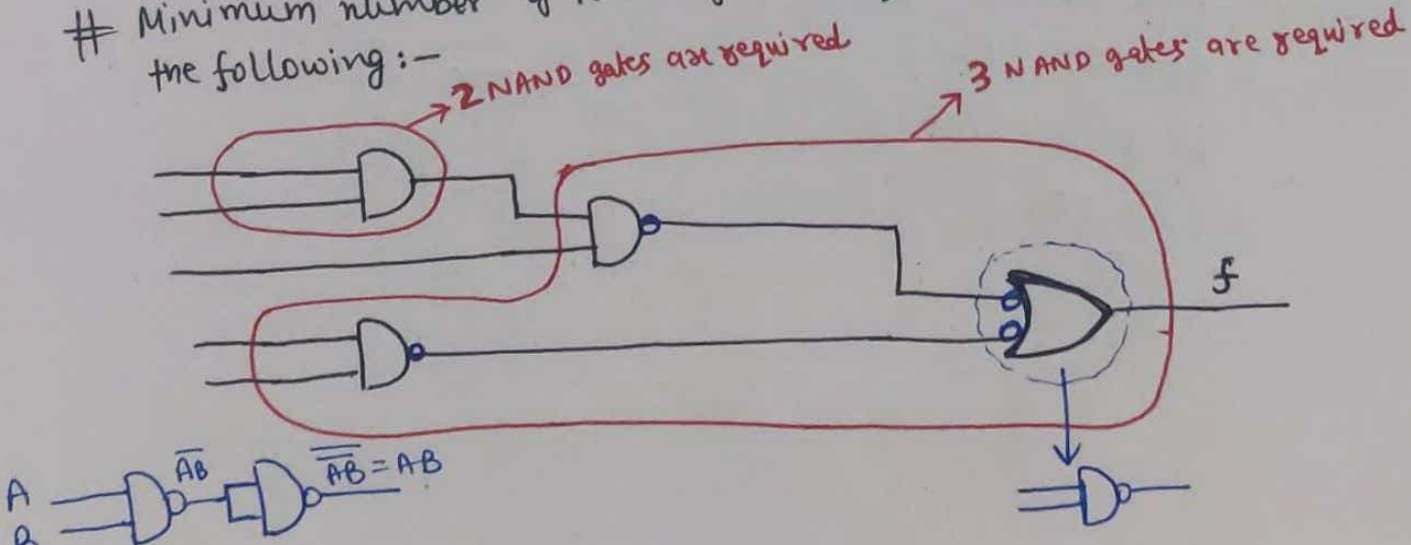




OR-AND Realization ⇒ NOR-NOR Realization

NOT using NOR gate

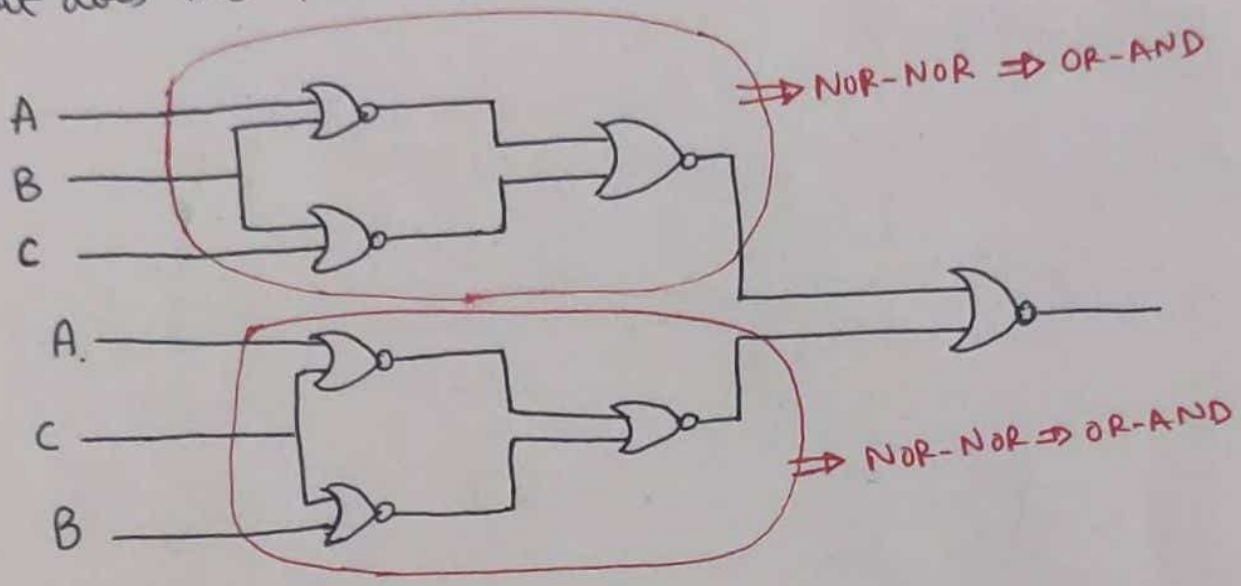


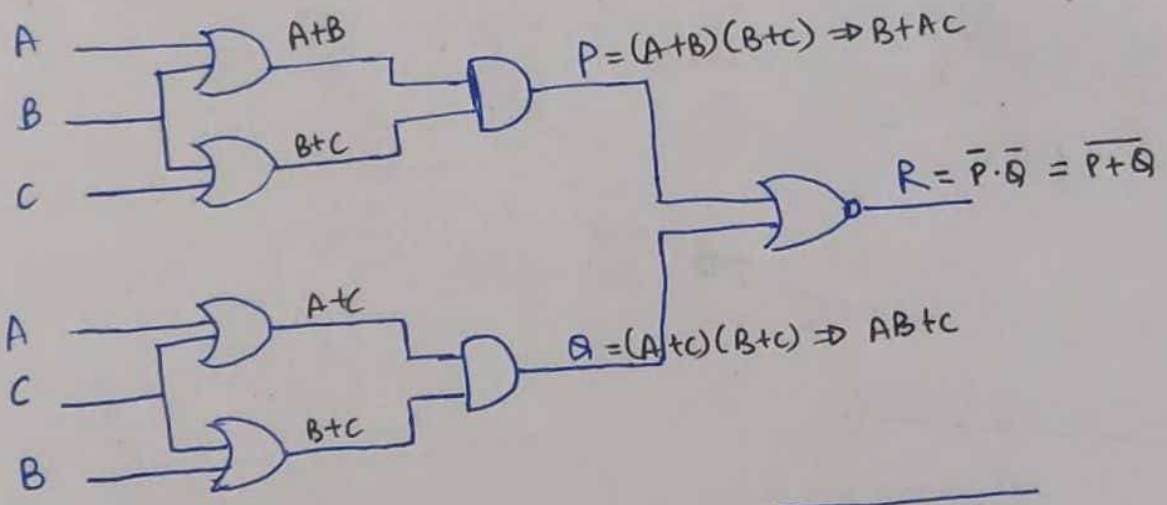

# Minimum number of NAND gates required to represent the following :-




Total NAND gates required = 2+3
⇒ 5

Q what does the following function represent?




$R \Rightarrow (\overline{B+AC})(\overline{C+AB})$

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

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

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

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

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

$= \overline{B+C}$

<u>or</u>

$R = \overline{B+AC+AB+C}$

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

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

$= \overline{B+C}$

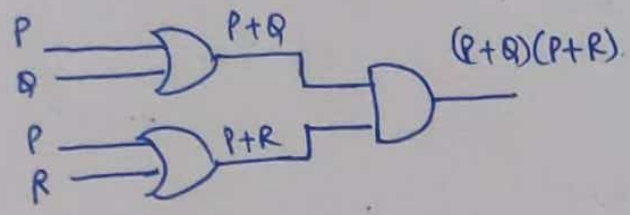
# Minimum number of NOR gates required to represent the following :-

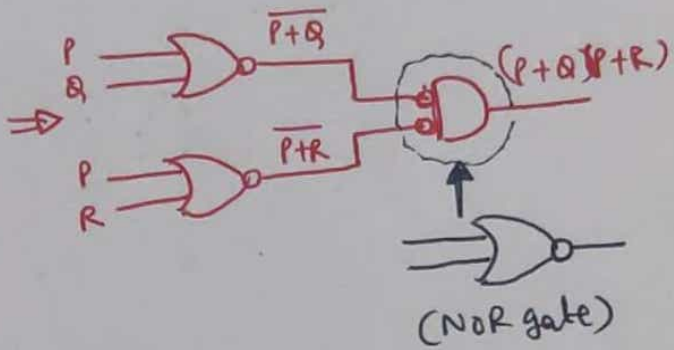
$$F(P,Q,R) = P+QR$$

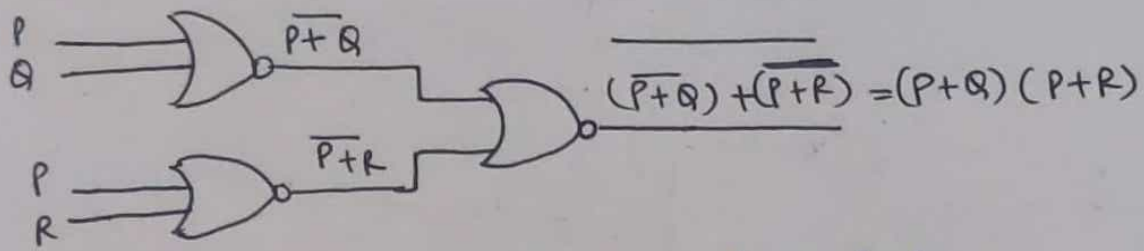
NOR-NOR ⇒ OR-AND ⇒ POS

$F \Rightarrow P+QR$

$= (P+Q)(P+R)$




Hence, 3 NOR gates are required to design the given function.

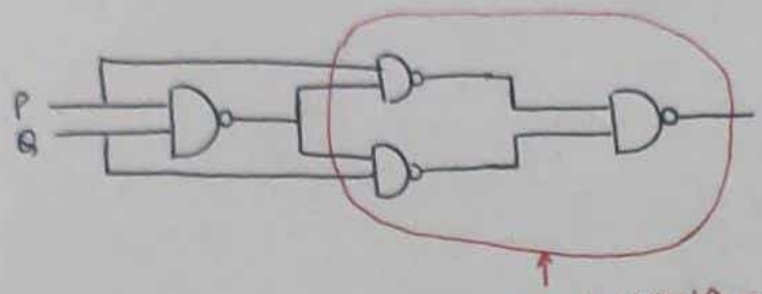
# Minimum number of NOR gates required to represent the following

$$F(X,Y,Z) = X + X\bar{Y} + X\bar{Y}Z$$
$$= X(1 + \bar{Y} + \bar{Y}Z)$$
$$= X(1)$$
$$= X$$

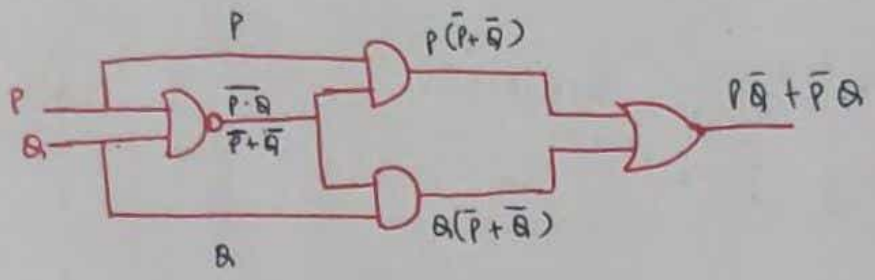
First Minimize the given function & then try to design the circuit

Hence, 0 NOR gates are required to design the given function.

# Ex-OR & Ex-NOR implementation with NOR :-

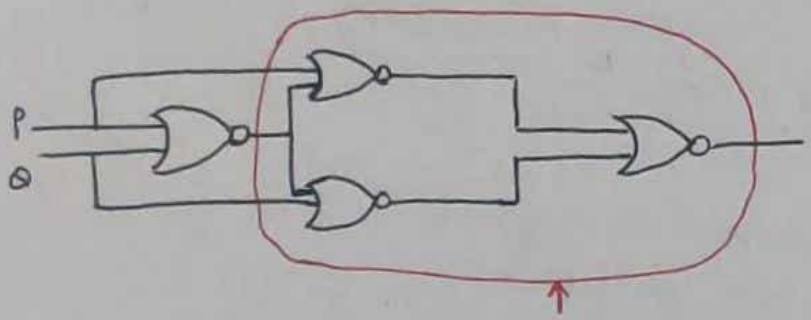
- 


NAND-NAND = AND-OR realization realization

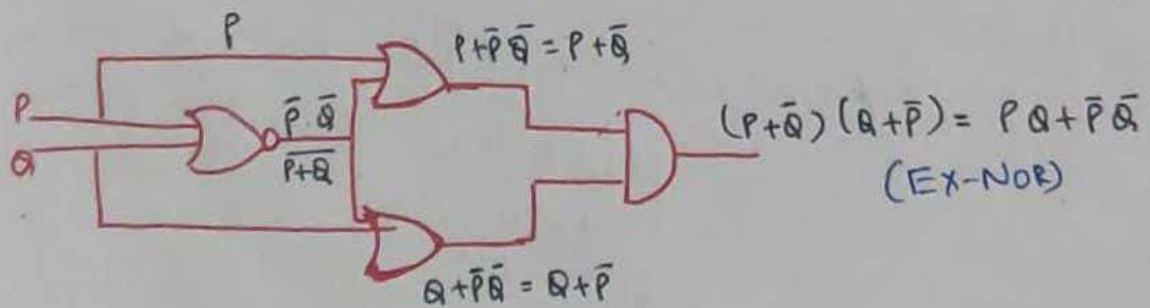



(Ex-OR)

Ex-OR $\leftrightarrow$ NAND $|_4$

- 


NOR-NOR = OR-AND realization realization




(Ex-NOR)

Ex-NOR $\leftrightarrow$ NOR $|_4$

$$= \overline{P\bar{Q} + \bar{P}Q}$$

$$= (\overline{P\bar{Q}}) * (\overline{\bar{P}.\bar{Q}})$$

$$= (\bar{P} + Q)(P + \bar{Q})$$

$$= P\bar{Q} + \bar{P}Q$$

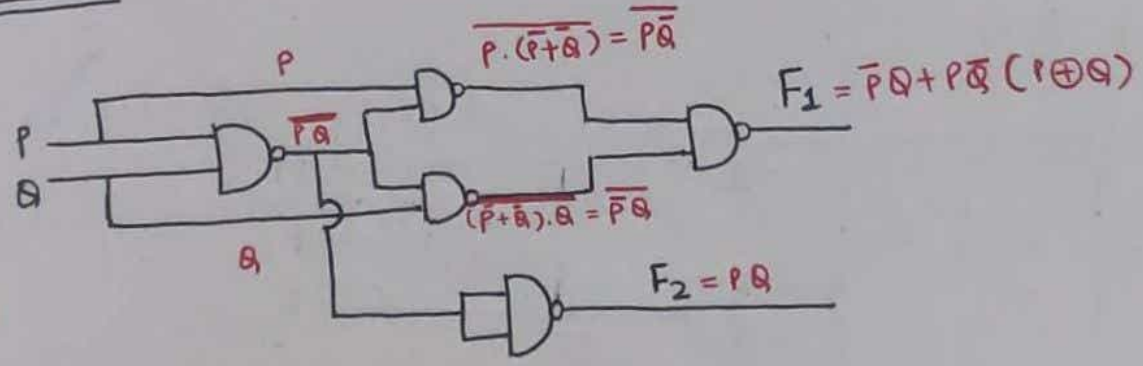
} Negation of EXOR is EXNOR

For designing the EXNOR gate using NAND gate we can first design the EXOR gate using NAND gate & lastly put one more NAND gate to complement the final result i.e For converting EXOR to EXNOR

Hence, EXNOR → NAND $|_{4+1=5}$

II$^{ly}$ For EXOR → NOR $|_{4+1=5}$

Half adder:




$F_1 = P \oplus Q$

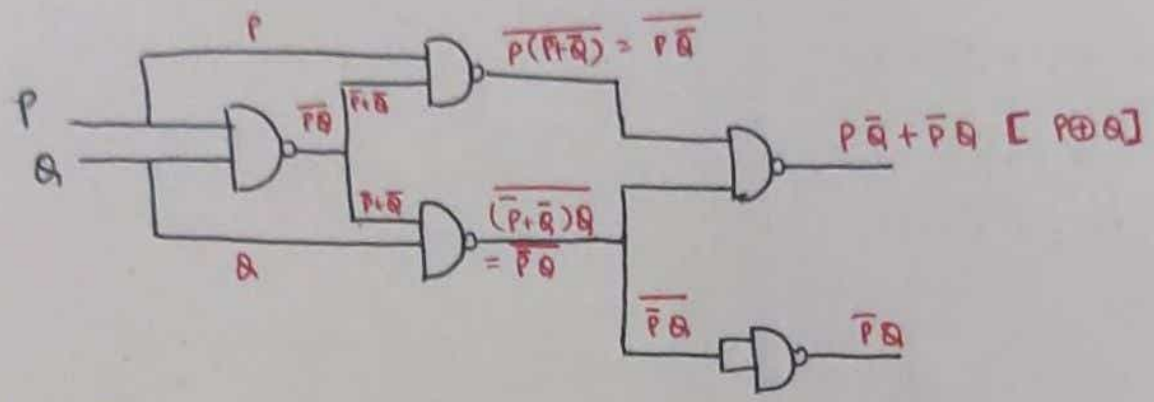
$F_2 = PQ$

| P | Q | Sum ($F_1$) | Carry ($F_2$) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Half adder : (A, B)

Full adder : (A, B, C)

# Half Substractor:



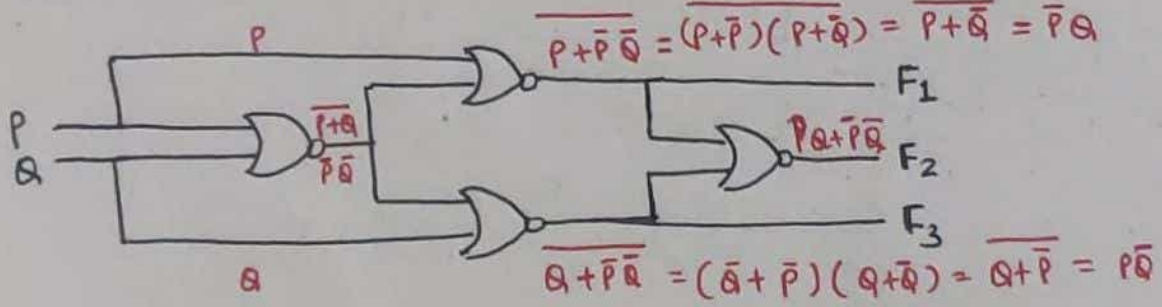

$F_1 = P \oplus Q$

$F_2 = \overline{P}Q$

| P | Q | Substract $F_1$ | Borrow $F_2$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

Half substractor : (A,B)

# Comparator:




$F_1 = \overline{P}Q$

$F_2 = PQ + \overline{P}\overline{Q}$

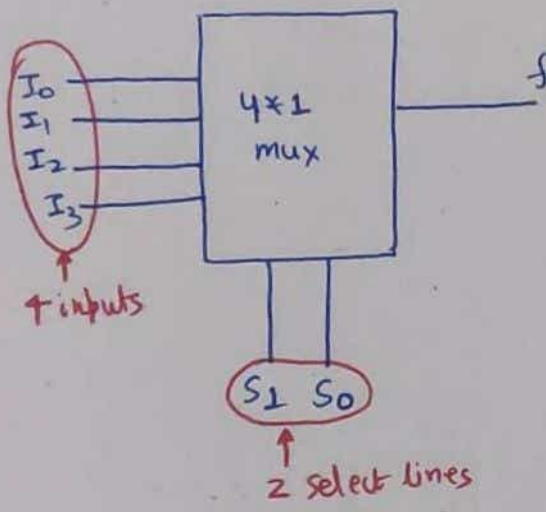
$F_3 = P\overline{Q}$

| P | Q | $F_1$ (P<Q) | $F_2$ (P=Q) | $F_3$ (P>Q) |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |

# Multiplexers:

- A multiplexer is an electronic switch that can connect one out of n inputs to output
- It can't change the logical level of the input, it only provides the connection b/w input & output
- It is functionally complete, ie all boolean functions can be realized using only multiplexer without any other gates.

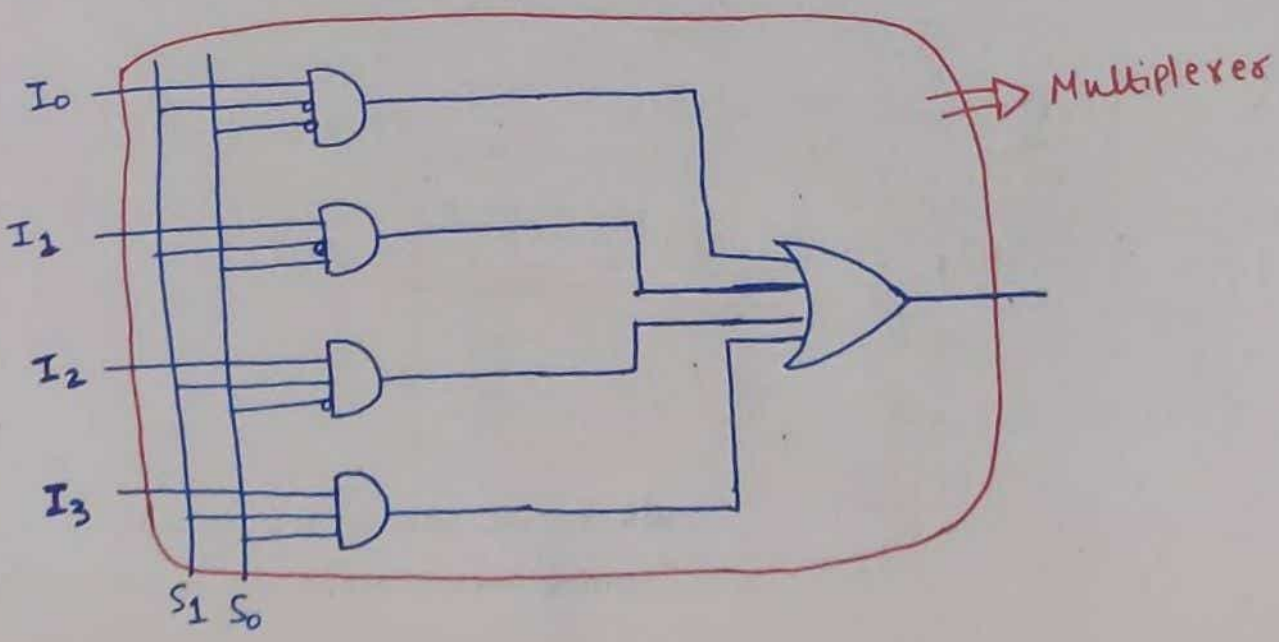
If $2^n \times 1$ mux,
n bits are required i.e.
n select lines are required




| $S_1$ | $S_0$ | function (f) |
|---|---|---|
| 0 | 0 | $I_0$ |
| 0 | 1 | $I_1$ |
| 1 | 0 | $I_2$ |
| 1 | 1 | $I_3$ |

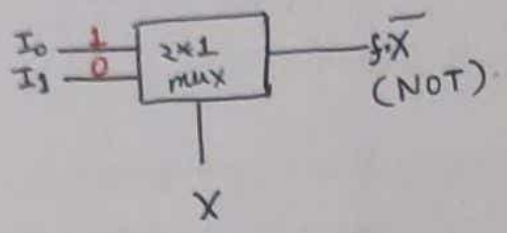
(characteristics Table)

function (f) $= \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$ } AND-OR realization



# Proving that mux is functionally complete:

To say functionally complete, we should derive
$(+,\cdot,-)$ or $(+,-)$ or $(\cdot,-)$



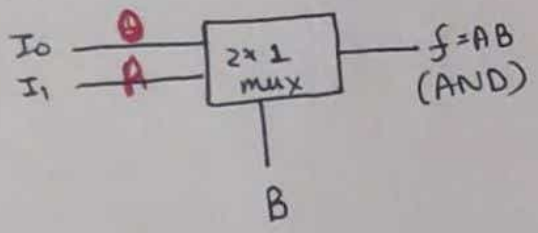

| X | X' |
|---|---|
| 0 | 1 |
| 1 | 0 |

$f = \bar{X} I_0 + X I_1$

$\bar{X} = \bar{X}\cdot(1) + X(0)$

$I_0 = 1$ $\quad$ $I_1 = 0$

when $X=0$, $I_0$ selected
when $X=1$, $I_1$ selected

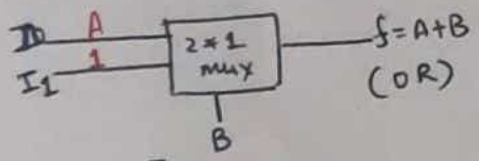



| A | B | AB |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$f = \bar{B} I_0 + B I_1$

$AB = \bar{B}(0) + B(A)$

$I_0 = 0$ $\quad$ $I_1 = A$




| A | B | A+B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

$f = \bar{B} I_0 + B I_1$

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

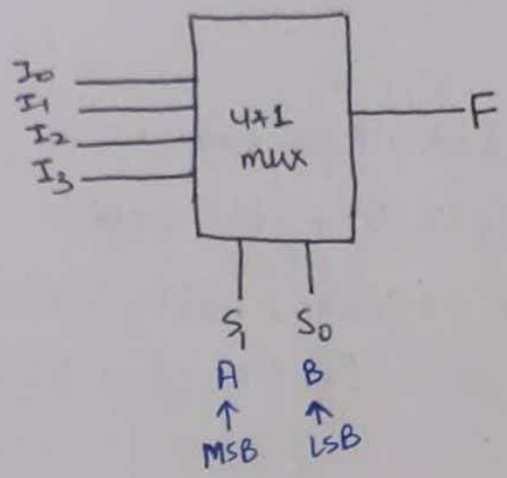
$= \bar{B}(A) + B(A+\bar{A})$

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

$I_0 = A$ $\quad$ $I_1 = 1$

We got NOT, AND, OR using the mux
Therefore mux is functionally complete device.

# Implementing functions with mux :




$$F = \bar{A}\bar{B}I_0 + \bar{A}BI_1 + A\bar{B}I_2 + ABI_3$$

Now, If we want to Implement function 'g' say -

| A | B | g |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

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

$$= \bar{A}\bar{B}(1) + \bar{A}B(0) + A\bar{B}(0) + AB(1)$$

($\bar{A}\bar{B}$ → $I_0$, $\bar{A}B$ → $I_1$, $A\bar{B}$ → $I_2$, $AB$ → $I_3$)

∴ Mux can be Implemented

Now, using 4×1 multiplexer we can implement a function of 2 variables

⇒ $(2^2 * 1)$ Mux — 2 variable function can be Implemented.

⇒ $(2^3 * 1)$ Mux — 3 variable function can be Implemented.

⋮

⇒ $(2^n * 1)$ Mux — $n$ variable function can be Implemented.

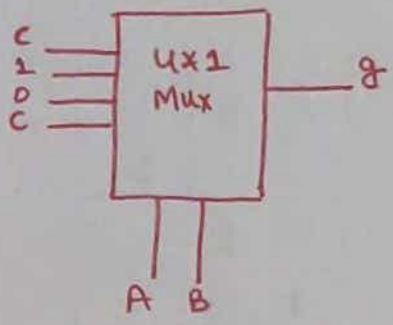
Can we Implement a 3 variable function using (4*1) mux?

Yes, It is possible but we might have to use more gates (in some cases)

| A | B | C | g |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

$$g = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + ABC$$
$$= \bar{A}\bar{B}(C) + \bar{A}B(\bar{C}) + \bar{A}B(C) + AB(C)$$
$$= \bar{A}\bar{B}(C) + \bar{A}B(C+\bar{C}) + A\bar{B}(0) + AB(C)$$
$$= \bar{A}\bar{B}(C) + \bar{A}B(1) + A\bar{B}(0) + AB(C)$$

($C$ → $I_0$, $1$ → $I_1$, $0$ → $I_2$, $C$ → $I_3$)




∴ Now, Just by using $(2^n * 1)$ mux, we may implement a function having more than $n$-variables but It may require some more gates.

Now, we can't implement a function of 4 variables using 4*1 mux because out of the 4 variables, 2 variables must be given to select lines & the remaining 2 variables are not independent i.e. they are the function of two variables & we require more gates.

The function that is formed with 4 variables is :-

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

($CD$ → $I_0$, $I_1$, $I_2$, $I_3$)

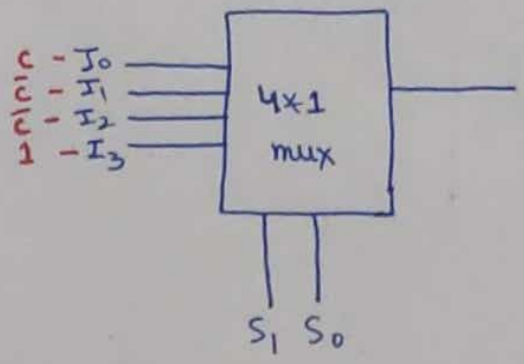
$f(A,B,C) = \Sigma(1,2,4,6,7)$

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

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

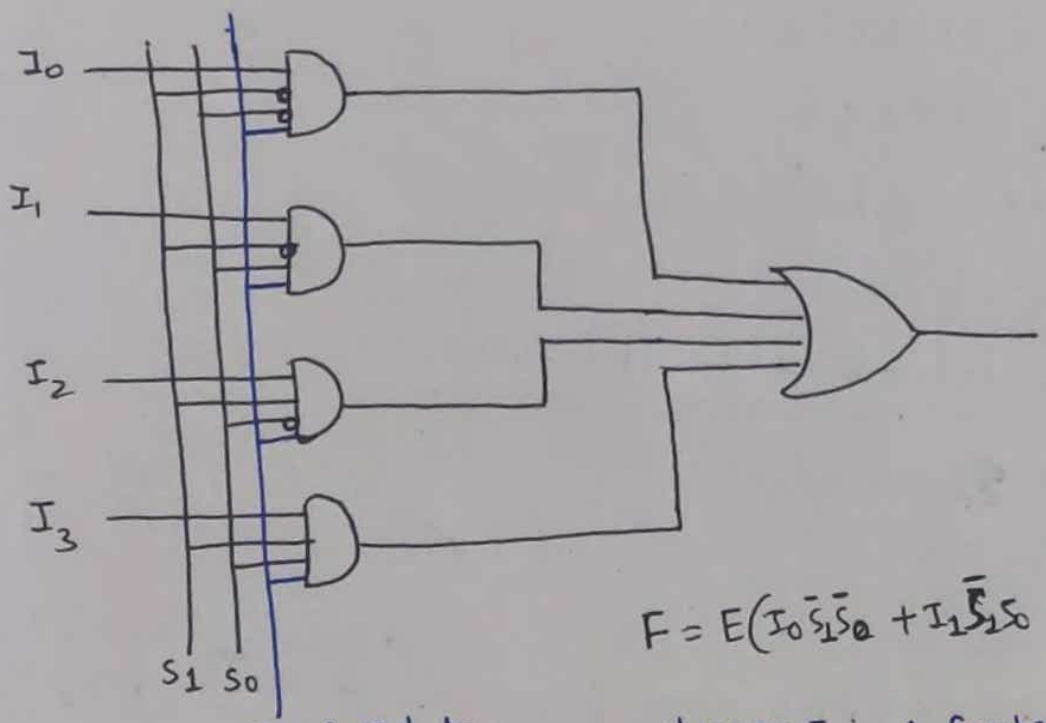
$I_0 \quad I_1 \quad I_2 \quad I_3$

| A | B | output |
|---|---|---|
| 0 | 0 | C |
| 0 | 1 | $\bar{C}$ |
| 1 | 0 | $\bar{C}$ |
| 1 | 1 | 1 |




# Multiplexer with Enable Input:

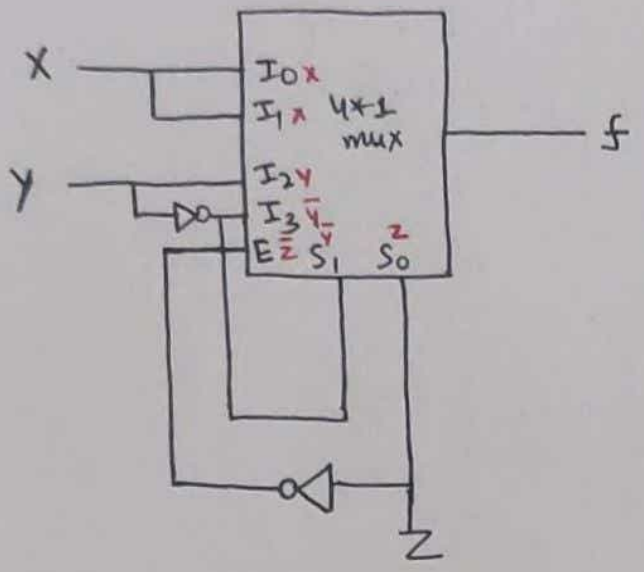
(I want to put ON-OFF switch to the mux Device)




$F = E(I_0\bar{S}_1\bar{S}_0 + I_1\bar{S}_1S_0 + I_2S_1\bar{S}_0 + I_3S_1S_0)$

- whenever E is 1, function will be acting as mux & gives output 1
- whenever E is 0, function will be acting as switch off & gives output 0

# Minimize the function represented by the following mux :




$$f = E(I_0 \bar{S}_1 \bar{S}_0 + I_1 \bar{S}_1 S_0 + I_2 S_1 \bar{S}_0 + I_3 S_1 S_0)$$

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

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

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

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

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

$$= xy\bar{z}$$

# Relation b/w select lines & input lines :

$f(A,B,C) = \sum(2,3,5,6,7)$

(a) $S_1 = B$, $S_0 = C$

(b) $S_1 = C$, $S_0 = B$

$S_1 = B$, $S_0 = C$ ⇒ $I_0 = 0$, $I_2 = 1$, $I_1 = A$, $I_3 = 1$

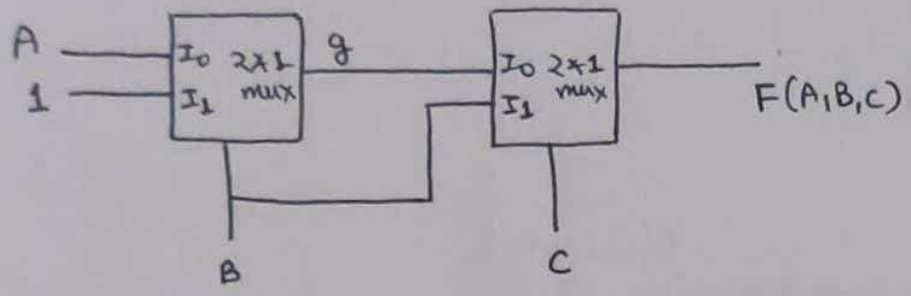
$S_1 = C$, $S_0 = B$ ⇒ $I_0 = 0$, $I_2 = A$, $I_1 = 1$, $I_3 = 1$

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

$$= \bar{B}\bar{C}(0) + \bar{B}C(A) + B\bar{C}(1) + BC(1)$$

# Cascading multiplexers :-

what is the function in canonical SOP?




$g = \bar{B} I_0 + B I_1$

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

$= A\bar{B} + B$

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

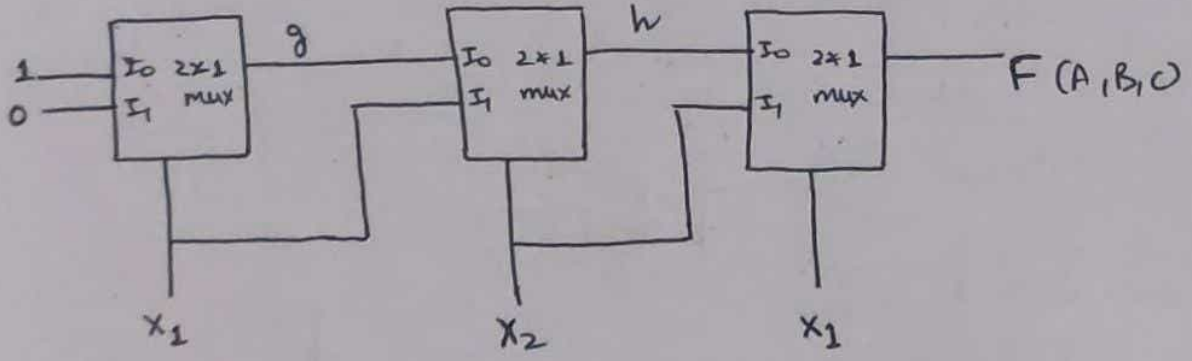
$= \bar{C}(A\bar{B} + B) + CB$

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

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

(4) (6) (2) (3) (7)

$F \Rightarrow \Sigma(2,3,4,6,7)$




$g = \bar{X}_1(1) + X_1(0)$

$= \bar{X}_1$

$h = \bar{X}_2(\bar{X}_1) + X_2(X_1)$

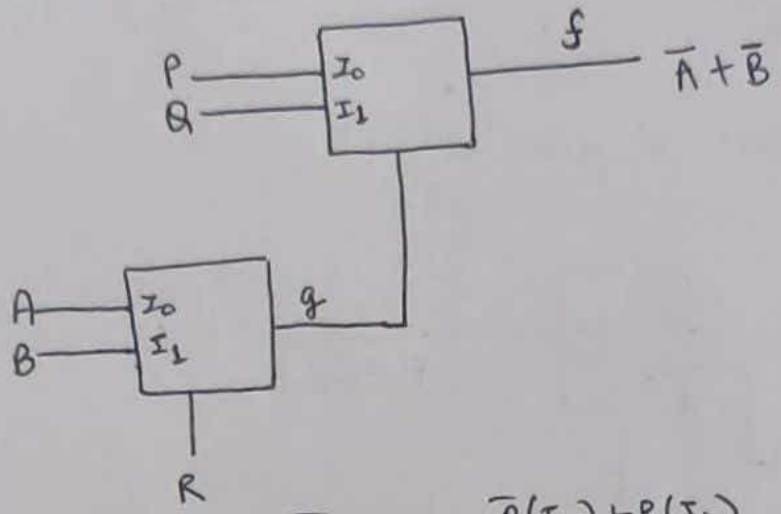
$= X_1 \odot X_2$ [EX-NOR]

$F = \bar{X}_1(\bar{X}_1\bar{X}_2 + X_1X_2) + X_1X_2$

$= \bar{X}_1\bar{X}_1\bar{X}_2 + X_1\bar{X}_1X_2 + X_1X_2$ ($\bar{X}_1X_1 = 0$)

$= \bar{X}_1\bar{X}_2 + X_1X_2$

$= X_1 \odot X_2$ [EX-NOR]




$g = \bar{R}A + RB \rightarrow \bar{R}(I_0) + R(I_1)$

$f = \overline{(\bar{R}A + RB)}\,P + (\bar{R}A + RB)\,Q$

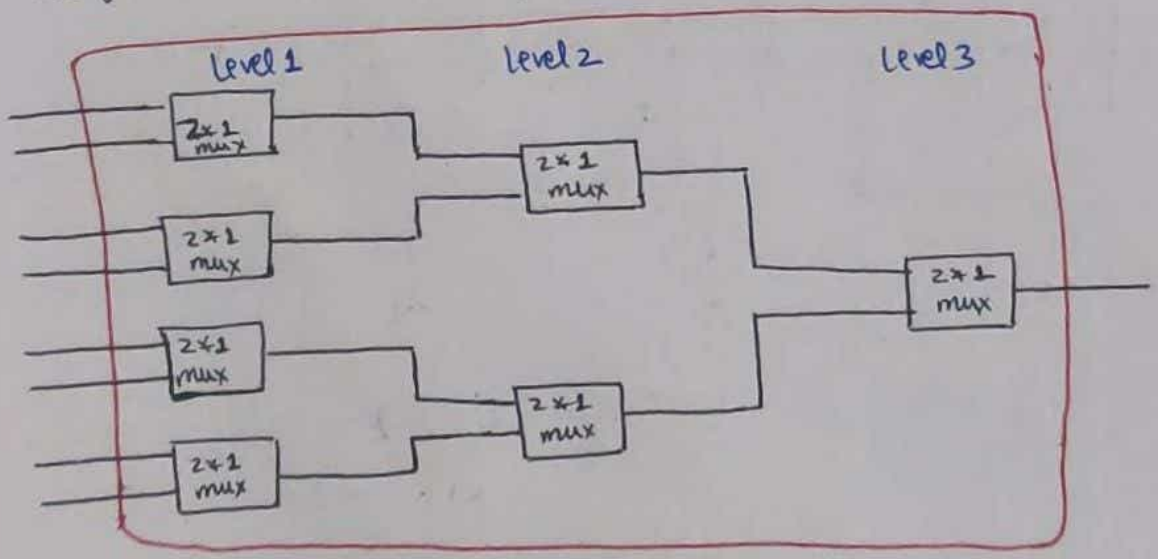
$= (R+\bar{A})(\bar{R}+\bar{B})P + A\bar{R}Q + BRQ$

$= R\bar{B}P + \bar{A}\bar{B}P + \bar{A}\bar{R}P + A\bar{R}Q + BRQ = \bar{A}+\bar{B}$

$= RP(\bar{B}) + P(\bar{A}\bar{B}) + P\bar{R}(\bar{A}) + \bar{R}Q(A) + RQ(B)$

# Expansion of Multiplexers:

Design 8×1 mux using 2×1 mux only.




8×1 mux

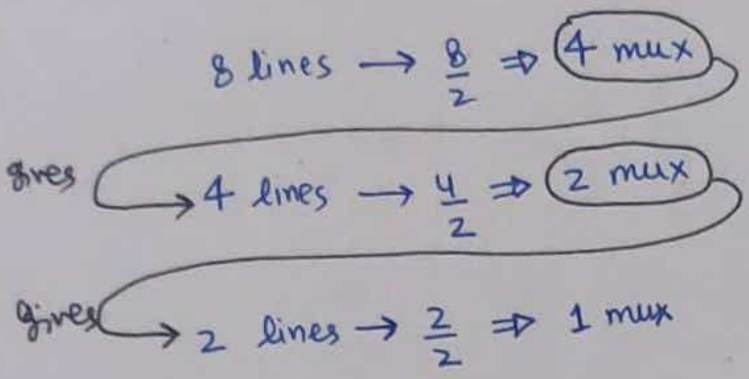
Level 1 → 4
Level 2 → 2
Level 3 → 1

} Total 7 [2×1 mux] are needed to construct 8×1 mux.

Construct 8×1 mux using 2×1 mux

1 mux → 2 lines

1 line → $\frac{1}{2}$ mux



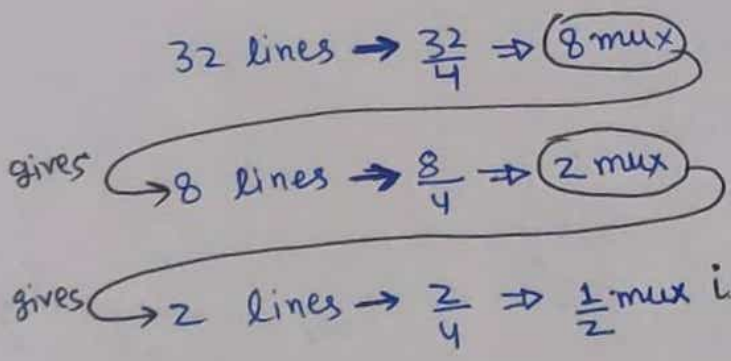

Total ⇒ 4+2+1
⇒ 7

Hence, To construct 8×1 mux with the help of 2×1 mux we need 7 (2×1 mux)

Construct 32×1 mux using 4×1 mux

1 mux → 4 lines

1 line → $\frac{1}{4}$ mux




Total ⇒ 8+2+1
⇒ 11

Last mux is not used with full capacity.

Hence, To construct 32×1 mux with the help of 4×1 mux we need 11 (2×1 mux)

Construct 64*1 mux using 4*1 mux.

1 mux → 4 lines

1 line → $\frac{1}{4}$ mux

64 lines → $\frac{64}{4}$ ⇒ 16 mux

gives → 16 lines → $\frac{16}{4}$ ⇒ 4 mux

gives → 4 lines → $\frac{4}{4}$ ⇒ 1 mux

Total ⇒ 16 + 4 + 1
⇒ 21

Hence, To construct 64*1 mux with the help of 4*1 mux we need 21 (4*1 mux)

Whenever we are trying to design M*1 mux using N*1 mux then how many Total mux are required & Total number of levels are required.

1 Device → N lines

1 line → $\frac{1}{N}$ Devices

M lines → $M * \frac{1}{N}$ Devices

gives → $\frac{M}{N}$ lines → $\frac{M}{N} * \frac{1}{N}$ Devices

gives → ⋮

$\frac{M}{N^{k-1}}$ lines → $\frac{M}{N^k}$ Devices

So, Number of Devices reqd = $\sum_{k=1}^{\lceil \log_2 M \rceil} \frac{M}{N^k}$

Number of Devices reqd at level k = $\frac{M}{N^k}$

$\frac{M}{N^k} \leq 1$

$M \leq N^k$

$k \geq \log_N M$ or $k = \lceil \log_N M \rceil$ → ceil

At least - ceil
At most - floor

# 8×1 using 2×1

M = 8, N = 2

$\lceil \log_N M \rceil = \lceil \log_2 8 \rceil = 3$ levels

$\Rightarrow \frac{8}{2^1} + \frac{8}{2^2} + \frac{8}{2^3}$

$\Rightarrow 4 + 2 + 1$

$\Rightarrow 7$ mux

# 32×1 using 4×1

M = 32, N = 4

$\lceil \log_N M \rceil = \lceil \log_2 5 \rceil = 3$ levels

$\Rightarrow \frac{32}{4^1} + \frac{32}{4^2} + \frac{32}{4^3}$

$\Rightarrow 8 + 2 + 1$

$\Rightarrow 11$ mux

# 64×1 using 4×1

M = 64, N = 4

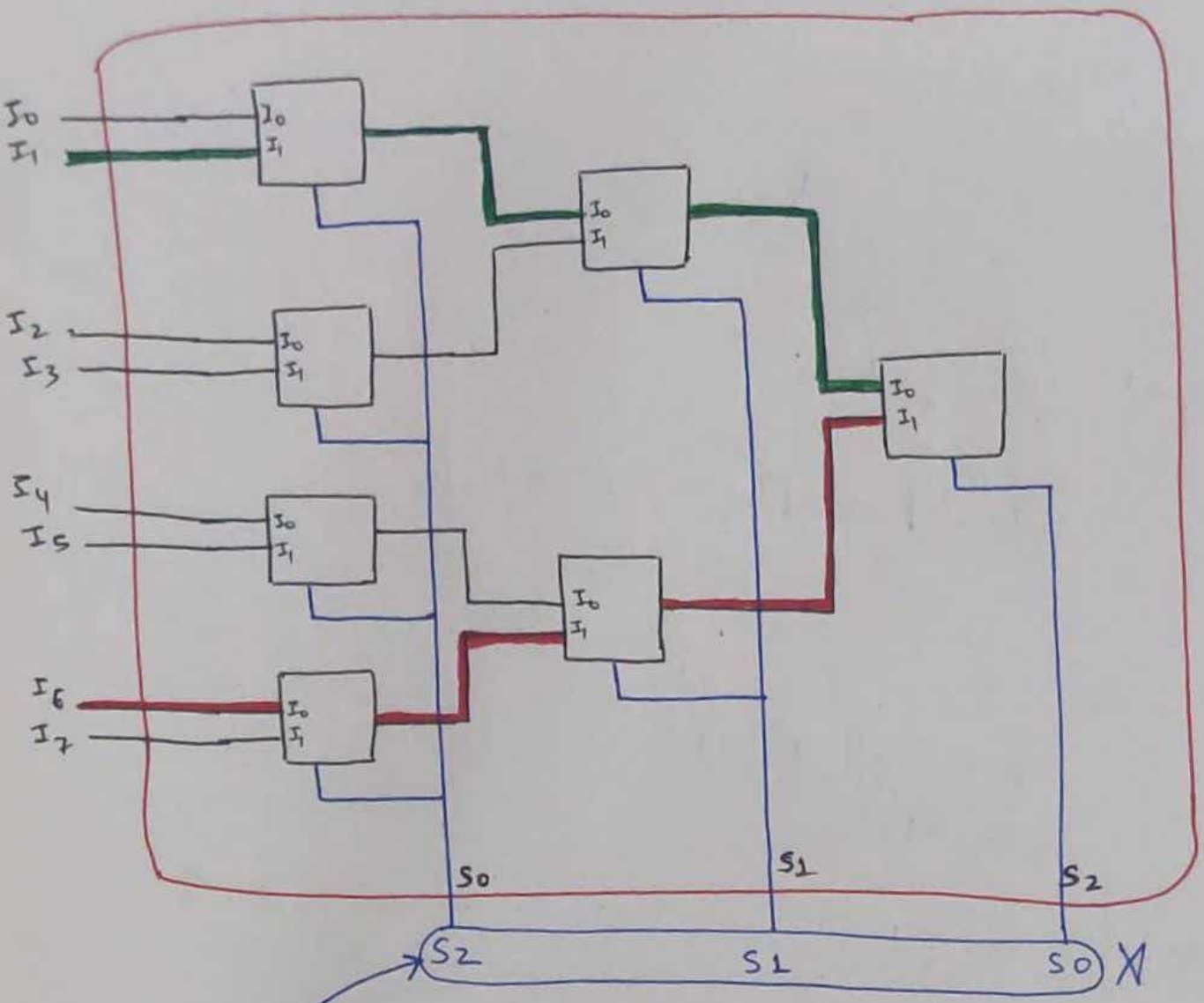
$\lceil \log_N M \rceil = \lceil \log_4 64 \rceil = 3$ levels

$\Rightarrow \frac{64}{4^1} + \frac{64}{4^2} + \frac{64}{4^3}$

$\Rightarrow 16 + 4 + 1$

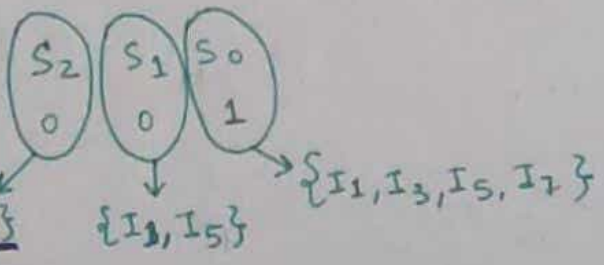
$\Rightarrow 21$ mux

# Assigning Select lines while Expanding:

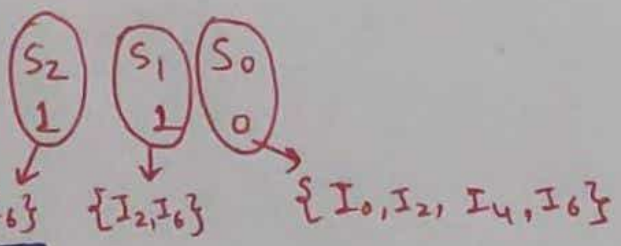



This assignment of Select lines is wrong because if we give $S_2S_1S_0$ = 0 0 1, then the o/p should be $I_1$, but if $S_2$ is zero, then it will select $I_0$ not $I_1$

If:

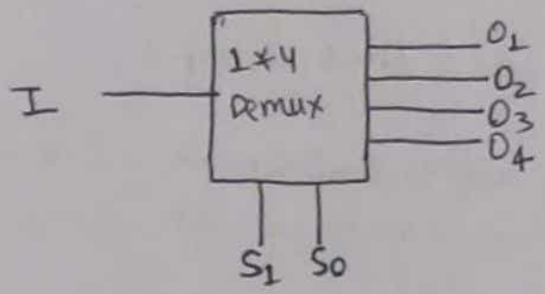


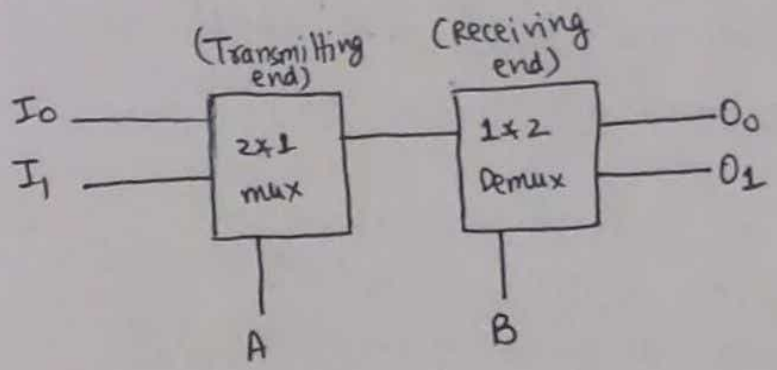

If:

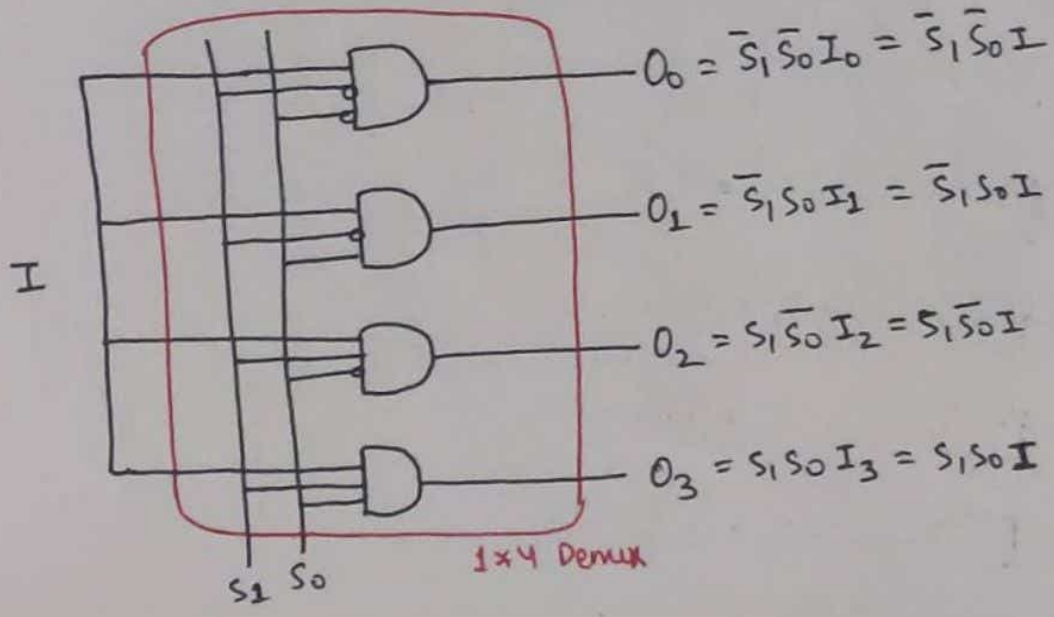


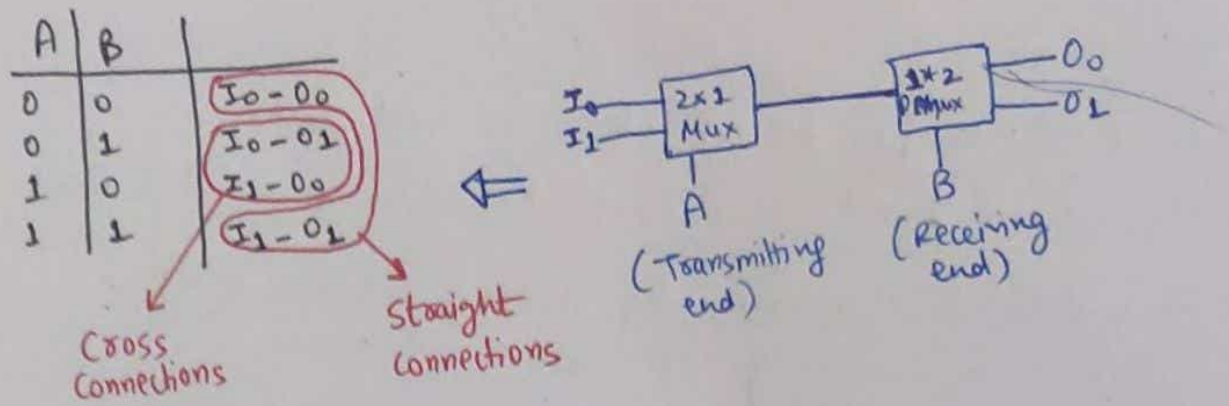
# Introduction to Demultiplexer:

- It performs opposite operation of multiplexer
- It has one Input & $2^n$ outputs where n is select lines
- It is derived from mux by joining all the inputs together & removing 'OR' gate
- It is mainly used in the construction of switches
- Demux is used at receiving end & mux is used at transmitting end.



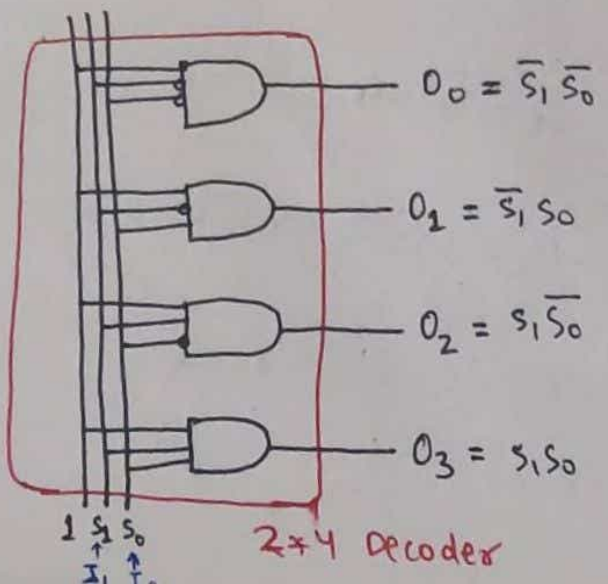



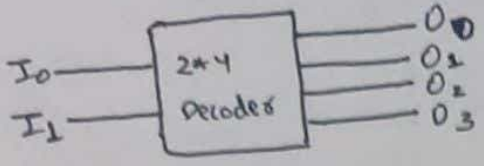



# Introduction to Decoders:

- A demux can be converted to a decoder by always setting I = 1 & making select lines as input
- Since a Decoder provides all the minterms, we could implement any function in canonical SoP using 'OR' gate.
- If all the 'AND' gates are replaced with 'NAND' gates, then the Decoder becomes active low Decoder.

⇒ Multiplexer is a 2 level realization ie 1st level is having AND gate & 2nd level is having OR gate (AND-OR Realization), but Decoder is AND realization only ie 1 level realization

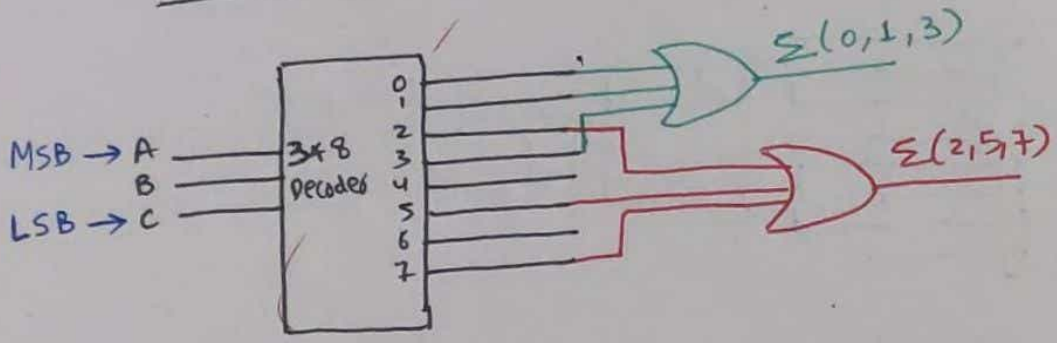




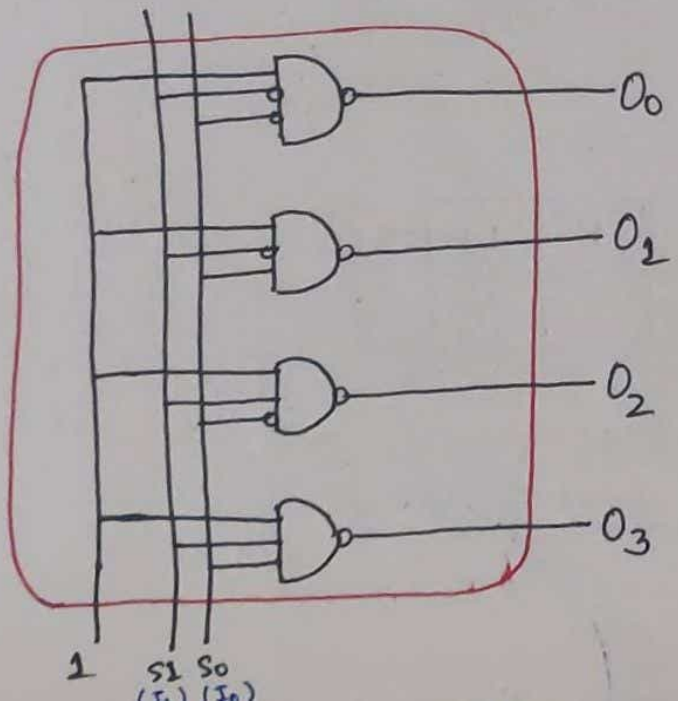

Mux $= 2^n * 1$
Demux $= 1 * 2^n$
Decoder $= n * 2^n$

| $I_1$ ↑ $S_1$ | $I_0$ ↓ $S_0$ | $O_0$ | $O_1$ | $O_2$ | $O_3$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |

# 3*8 Decoder:

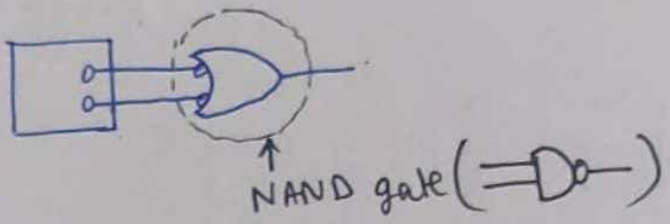



## If All AND gates are replaced by NAND gates:-



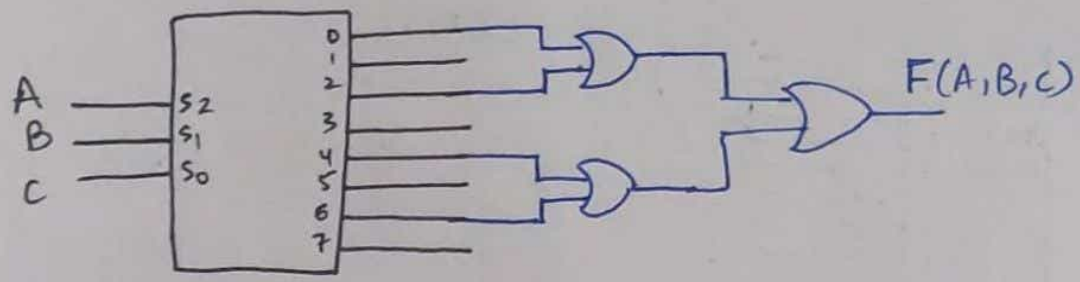

| $S_0$ | $S_1$ | $O_0$ | $O_1$ | $O_2$ | $O_3$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

In order to get the minterms we have to complemented the output produced by the Decoder




ie AND-OR realization ⇒ NAND-NAND realization.

## # Implementing functions with Decoder:




F is free from?

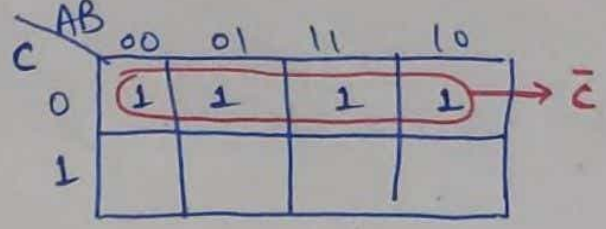
(a) 1 variable

(b) 2-variable

(c) 3-variable

(d) None

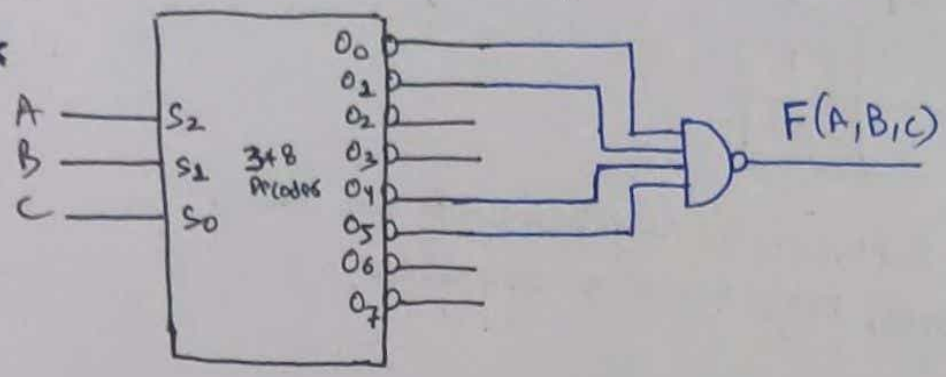
$F(A,B,C) \Rightarrow \Sigma(0,2,4,6)$




So, $F(A,B,C) = \Sigma(0,2,4,6)$ is dependent on one Variable

Hence, Free from 2 variables

*



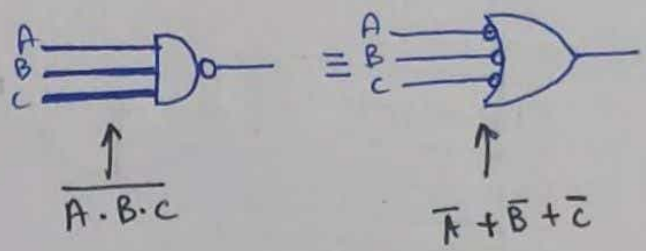

F is free from?

(a) 1 variable

~~(b)~~ 2 variable ✓

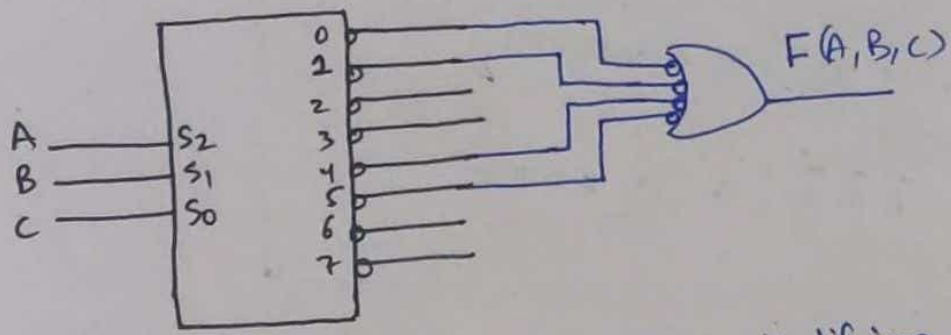
(c) 3 variable

(d) None

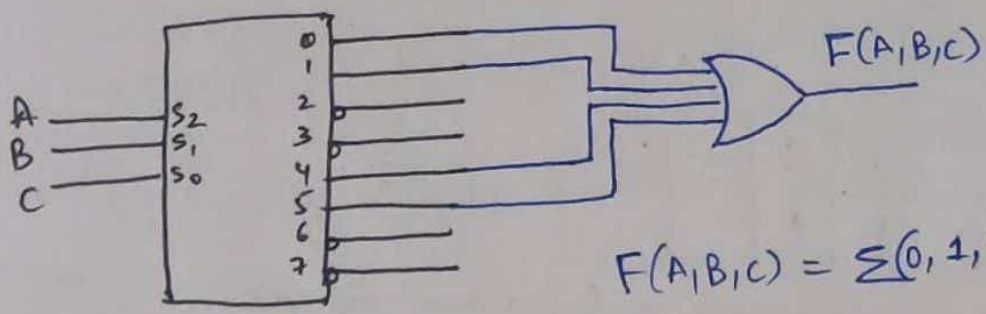



i.e. NAND gate is equal to Complemented OR gate

Hence, we can convest NAND gate into the complemented OR gate ie Modifying the given Question.

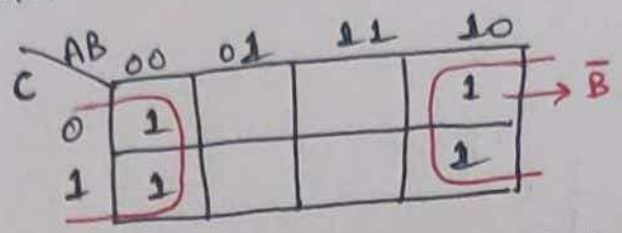



$\bar{\bar{X}} = X$ Hence, Again Modifying the Question



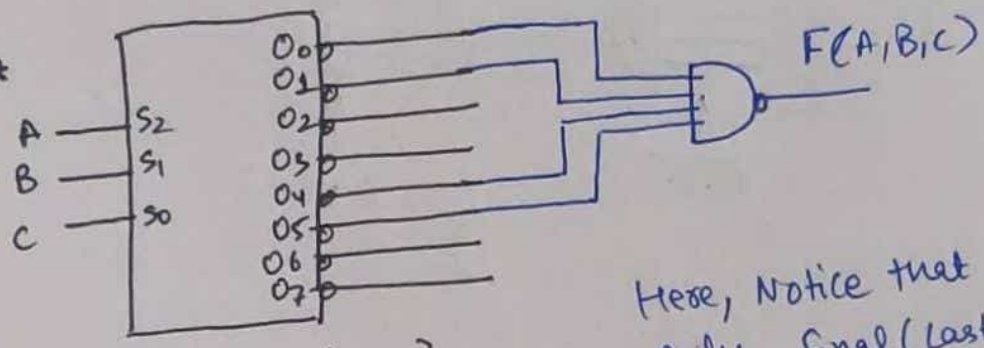

$F(A,B,C) = \Sigma(0,1,4,5)$

$F(A,B,C) = \Sigma(0,1,4,5)$




So, $F(A,B,C) = \Sigma(0,1,4,5)$ is dependent on one variable. Hence, Free from 2 variables.

*




F is Free from?

(a) 1 variable
(b) 2 variable ✓
(c) 3 variable
(d) None

Here, Notice that difference is only final (last) NAND gate is replaced by AND gate.

After the bubbling (complementing) of the minterms, we get the maxterm.

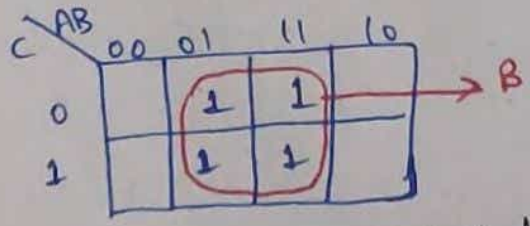
for example:

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

(Minterm) (Maxterm)

$F(A,B,C) = \Pi(0,1,4,5)$

$= \Sigma(2,3,6,7)$




So, $F(A,B,C) = \Sigma(2,3,6,7)$ is dependent on one variable. Hence, Free from 2 variables

# Converting one code to Another using Decoder

| | A | B | C | D |
|---|---|---|---|---|
| | 8 | 4 | 2 | 1 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |

| W | X | Y | Z |
|---|---|---|---|
| 2 | 4 | 2 | 1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

Self complement code

(9's complement of 5) ⇒ 4
(9's complement of 6) ⇒ 3
(9's complement of 7) ⇒ 2
(9's complement of 8) ⇒ 1
(9's complement of 9) ⇒ 0

1's complement of 1:

$1 - 1 = 0$

1's complement of 0:

$1 - 0 = 1$

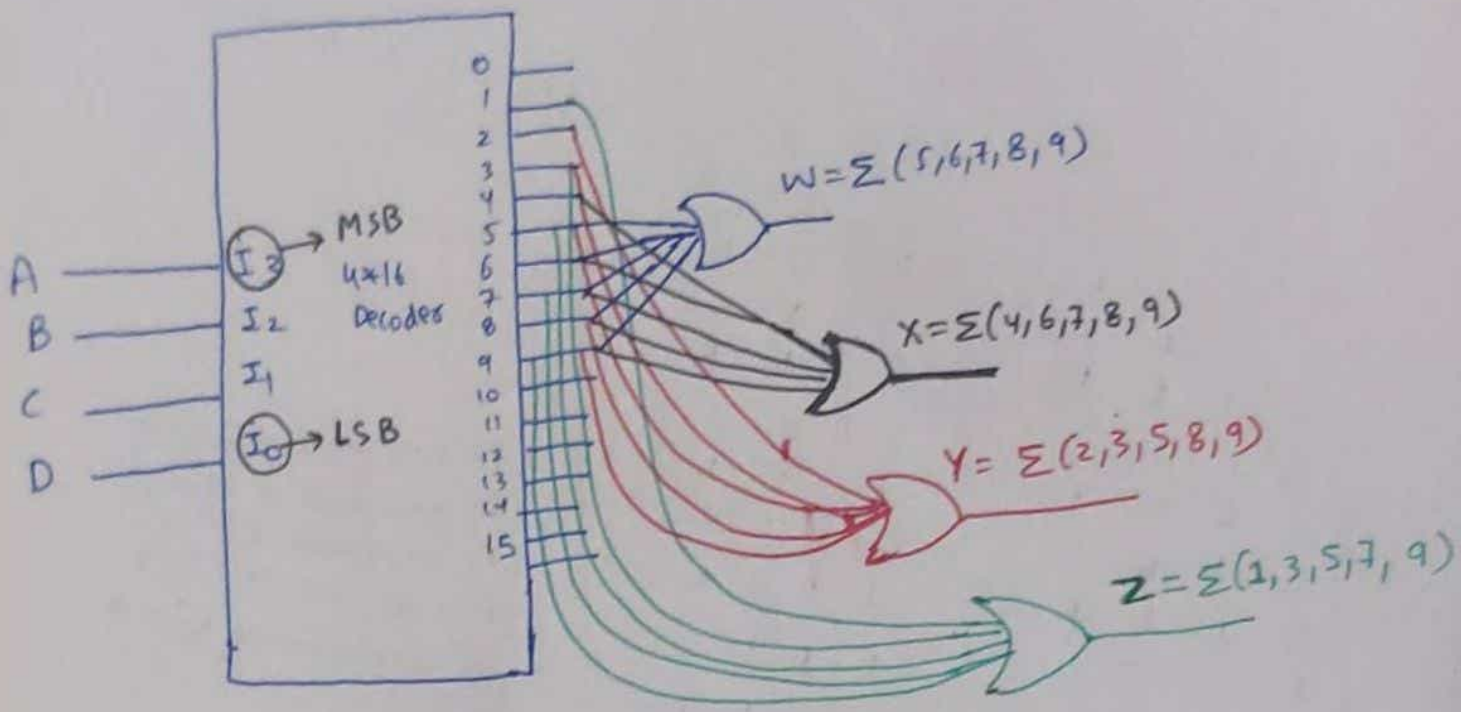
Ily 9's complement of 1:

$9 - 1 = 8$

Don't cares are only used in minimization not in realization.

$$W = \sum(5,6,7,8,9) + \sum\phi(10,11,12,13,14,15)$$
$$X = \sum(4,6,7,8,9) + \sum\phi(10,11,12,13,14,15)$$
$$Y = \sum(2,3,5,8,9) + \sum\phi(10,11,12,13,14,15)$$
$$Z = \sum(1,3,5,7,9) + \sum\phi(10,11,12,13,14,15)$$

Hence, we need 4×16 Decoder as there are 4 inputs & 16 different outputs.



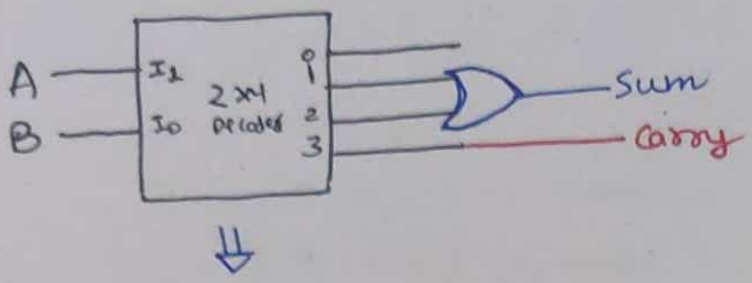

Output ⇒ WXYZ

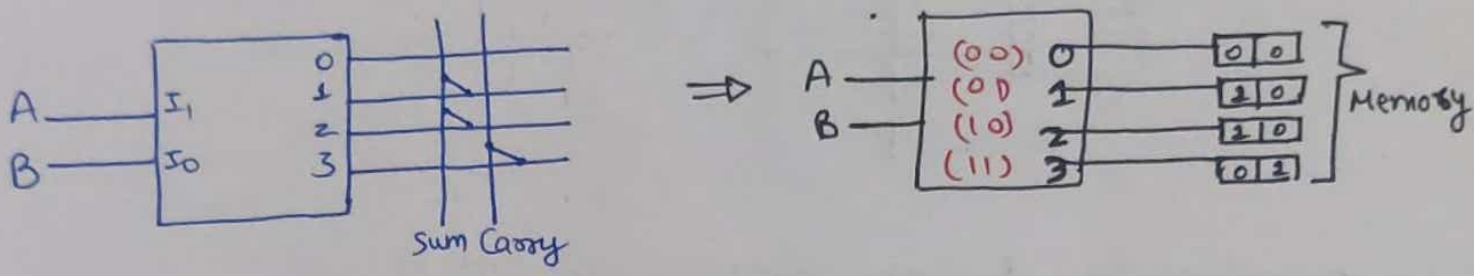
# (ROM) Implementation using Decoder: (Read only Memory)

- ROM can be used to realize combinational functions by storing appropriate values at appropriate locations
- Every ROM is expressed in terms of ROM matrix & Decoder
- ROM matrix contains set of links & connections. The lines entering the matrix & leaving it are called connections & the intersection of rows & columns are called links.

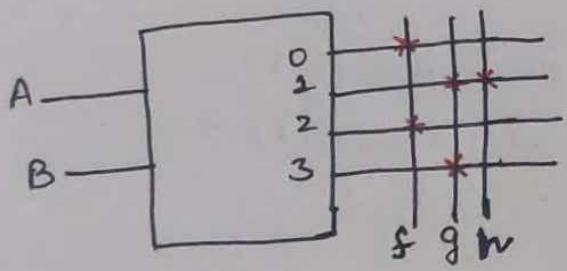
| A | B | Sum | Carry |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

Sum = Σ(1,2)
Carry = Σ(3)




## # Implementing functions using only Decoder :




$f = \Sigma(0,2)$

$g = \Sigma(1,3)$

$h = \Sigma(1)$

To implement the functions with $n$ variable & $m$ functions then, $n * 2^n$ Decoder is needed along with the ROM matrix of

Horizontal lines (Incoming) $\Rightarrow 2^n$

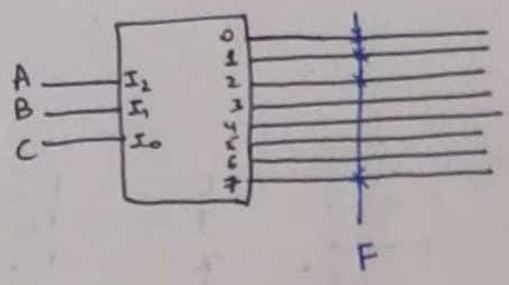
vertical lines (Outgoing) $\Rightarrow m$

↑ number of functions we want to implement

Hence, $2^n + m$ connections are required.

& $2^n * m$ links are required.

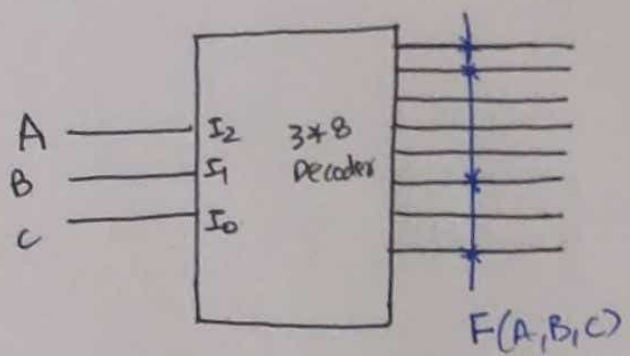
$F(A,B,C) = \Sigma(0,1,2,7)$




Connections are (8+1) ⇒ 9
Links are (8×1) ⇒ 8

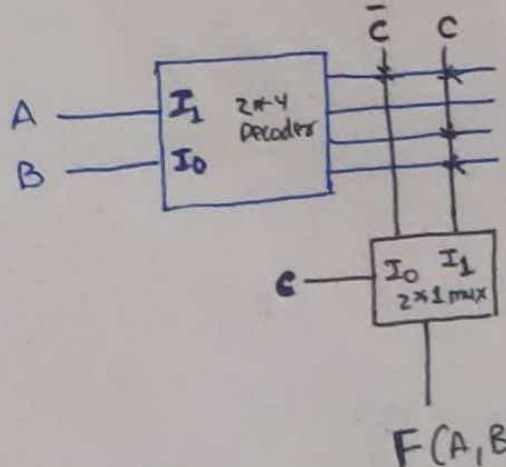
## # Implementing functions using Decoder & MUX:

$F(A,B,C) = \Sigma(0,1,5,7) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}C + ABC$




Connections are (8+1) ⇒ 9
Links are (8×1) ⇒ 8

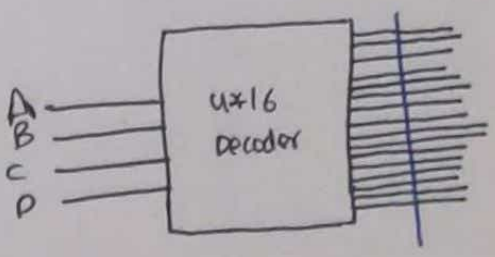
Now, If we have to implement the same function using 2×4 Decoder & 2×1 mux.



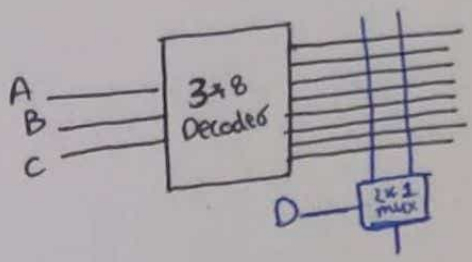

Connections are (4+2) ⇒ 6
Links are (4+2) ⇒ 8

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

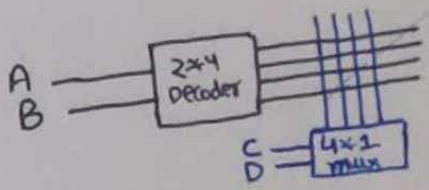
for all these inputs output should be 1 and other than these input we get output as 0.




Connections are (16+1) ⇒ 17
Links are (16 + 1) ⇒ 16




Connections are (8+2) ⇒ 10
Links are (8*2) ⇒ 16




Connections are (4+4) ⇒ 8
Links are (4*4) ⇒ 16

Let 10 variables and 1 function

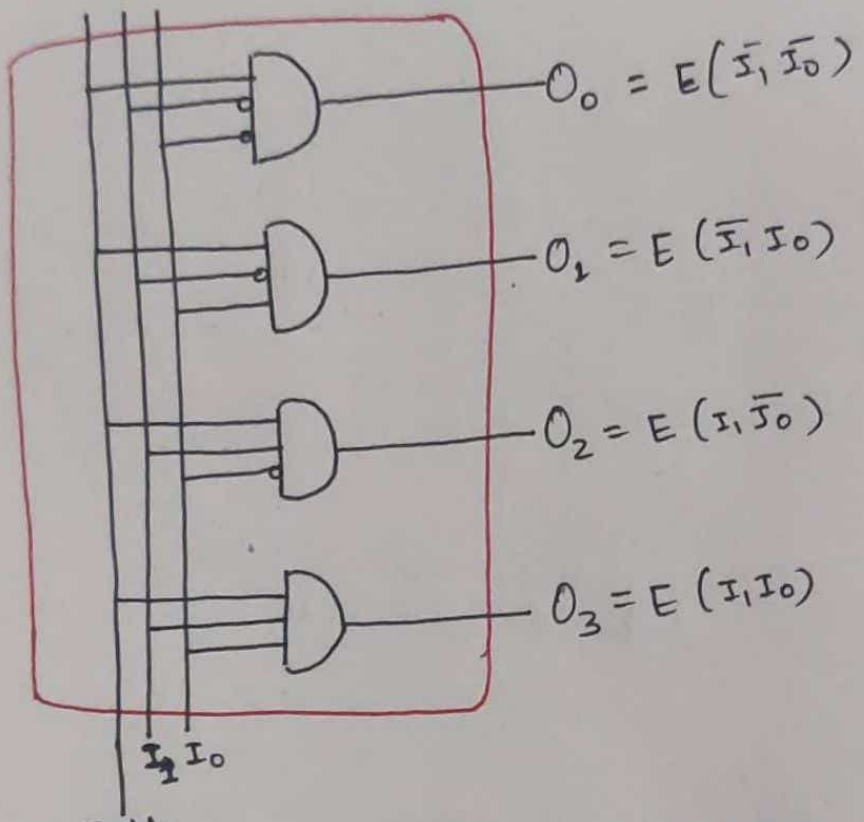
| Number of variable to Decoder | Number of variable to mux | Connections (C) | Links (L) |
|---|---|---|---|
| 10 | 0 | $1024+1=1025$ | $1024*1=1024$ |
| 5 | 5 | $2^5+2^5=64$ | $2^5*2^5=2^{10}=1024$ |
| 4 | 6 | $2^4+2^6=80$ | $2^4*2^6=2^{10}=1024$ |
| 3 | 7 | $2^3+2^7=136$ | $2^3*2^7=2^{10}=1024$ |
| 2 | 8 | $2^2+2^8=260$ | $2^2*2^8=2^{10}=1024$ |

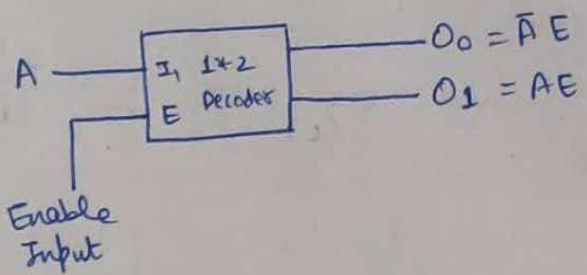
If, N variable (Decoder) M Variable (Mux)

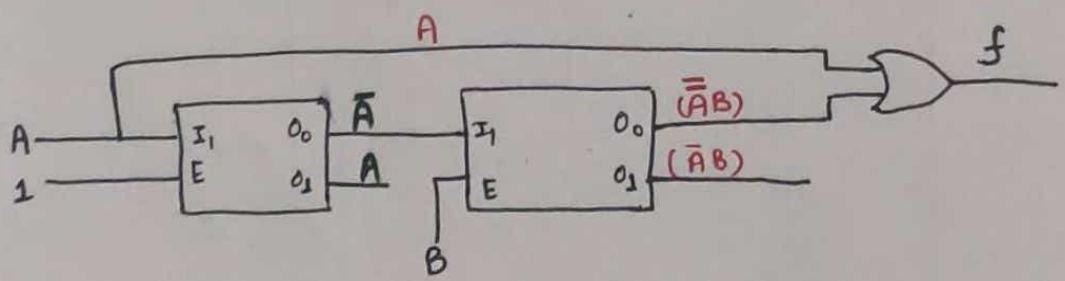
then, Connections (C) $= 2^N + 2^M$

Links (L) $= 2^{N+M}$

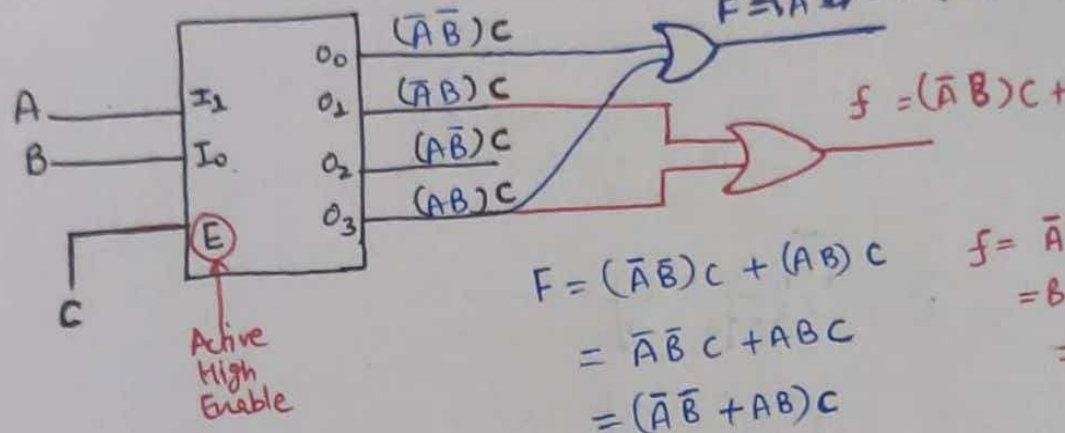
# Decoder with Enable Input:




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




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

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

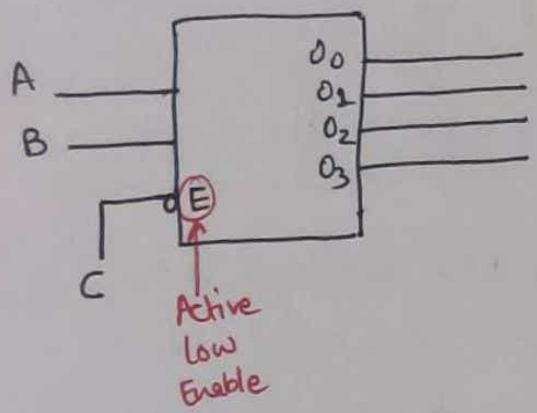
$= (\bar{A}\bar{B} + AB)C$

$= (A \odot B)C$

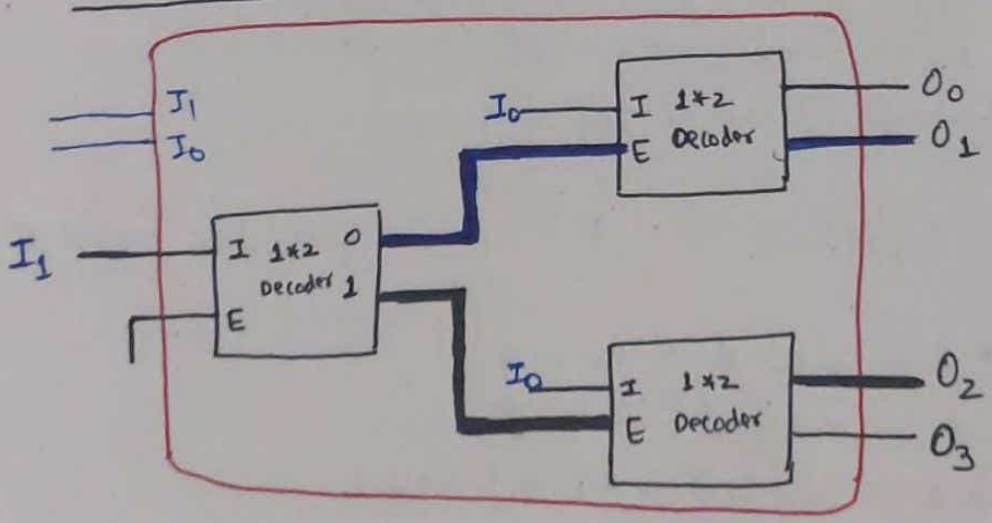
$f = \bar{A}BC + ABC$

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

$= BC$



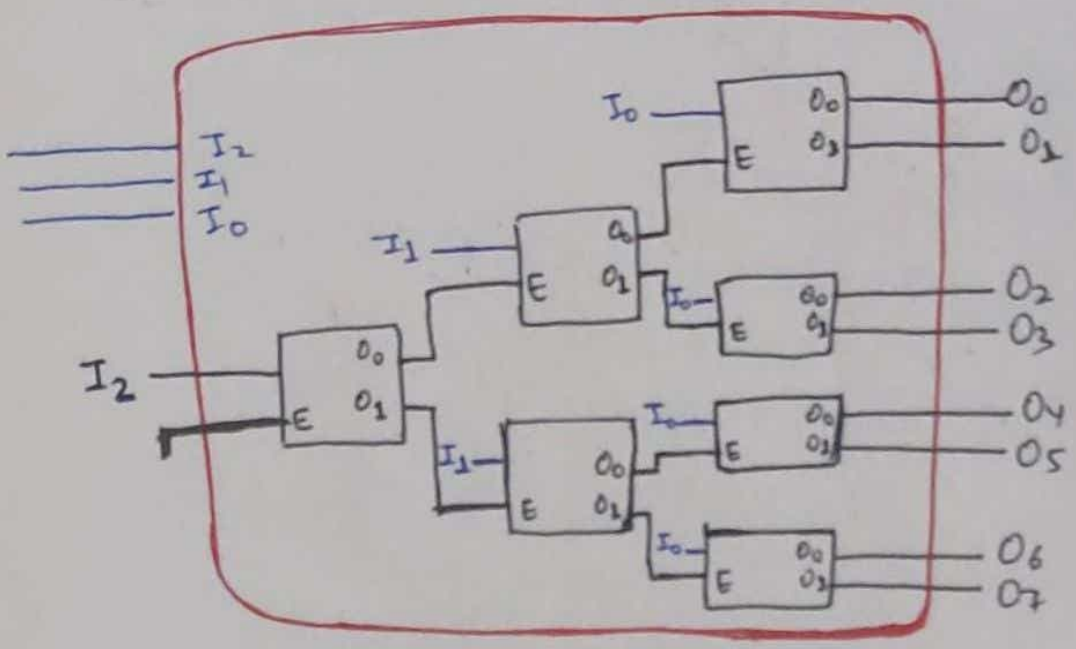

# Construct 2*4 Decoder using 1*2 Decoder:




$I_1$ $I_0$ } $O_1$ will be selected.
0 1

$I_1$ $I_0$ } $O_2$ will be selected.
1 0

# Construct 3*8 Decoder using 1*2 Decoder:




For Decoder start building from Right side Always.

- we have to covers lines ($O_0$ $O_1$ ..... $O_7$) & each 1*2 Decoder covers 2 lines, so at level 1 we will have 4 decoders. Now, join all the Enables of 4 Decoder & 4 Enables need two decoders at level 2 & one decoder at level 1

- Now, let me assign Input $I_0$ at level 1 for all decoders & $I_1$ at level 2 and $I_2$ at level 3 (we are not sure about it, we just assigned randomly to check the correctness)

- Now, If we say $I_2 I_1 I_0 = (0\ 0\ 1)$, then a should be enabled

$I_2 = 0 \rightarrow 0$ is selected & passed

$I_1 = 0 \rightarrow 0$ is selected & passed

$I_0 = 1 \rightarrow 1$ is selected & $O_1$ is enabled.

# construct 6*64 Decoder using 3*8 Decoders:-

64 Lines should come out.

1 Decoder ⇒ 8 lines

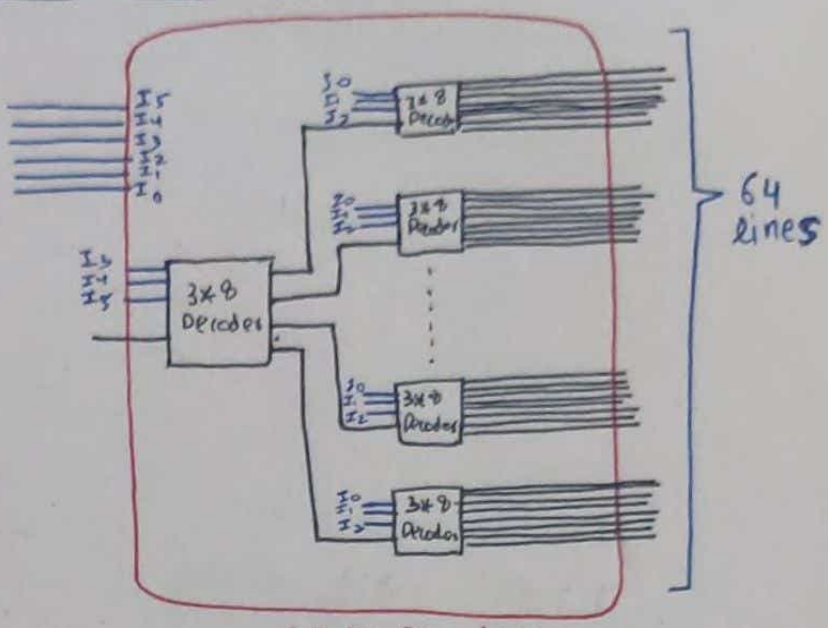
1 line ⇒ $\frac{1}{8}$ Decoder

64 Lines = $\frac{1}{8} * 64$ ⇒ 8 Decoders

8 Lines = $\frac{1}{8} * 8$ ⇒ 1 Decoder

Total ⇒ 8+1

⇒ 9 Decoders




# construct 6*64 Decoder using 2*4 Decoder:-

64 Lines should come out.

1 Decoder ⇒ 4 lines

1 line ⇒ $\frac{1}{4}$ Decoder

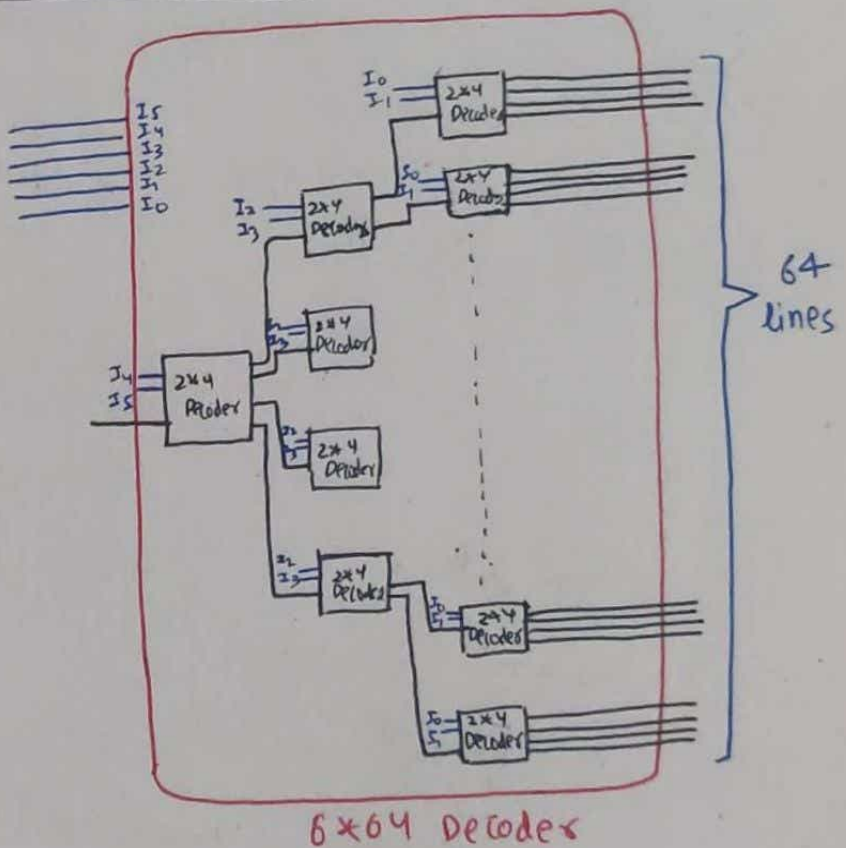
64 lines = $\frac{1}{4} * 64$ ⇒ 16 Decoders

16 lines = $\frac{1}{4} * 16$ ⇒ 4 Decoders

4 lines = $\frac{1}{4} * 4$ ⇒ 1 Decoder

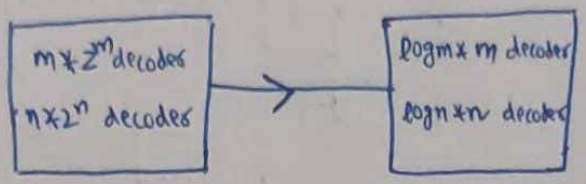
Total ⇒ 16+4+1

⇒ 21 Decoders



# Expansion of Decoder in General:

Now, I want to construct $m*2^m$ decoder using $n*2^n$ decoder




1 device $\longrightarrow$ $n$ lines

1 line $\longrightarrow$ $\frac{1}{n}$ devices

$m$ lines $\longrightarrow$ $m * \frac{1}{n} \Rightarrow \frac{m}{n}$ devices

gives

$\frac{m}{n}$ lines $\longrightarrow$ $\frac{m}{n} * 1/n \Rightarrow \frac{m}{n^2}$ devices

gives

$\frac{m}{n^2}$ lines $\longrightarrow$ $\frac{m}{n^2} * 1/n \Rightarrow \frac{m}{n^3}$ devices

$\vdots$

$\frac{m}{n^k}$ devices

$$\frac{m}{n^k} \leq 1$$

$$m \leq n^k$$

ceil $\longrightarrow$ $k \geq \lceil \log_n m \rceil$

Number of levels $\Rightarrow \lceil \log_n m \rceil$

Number of devices needed $\Rightarrow \sum_{k=1}^{\log_n m} \left(\frac{m}{n^k}\right)$

eg ① 6*64 Decoder | m = 64
4*16 Decoder | n = 16

Number of levels $= \left\lceil \frac{\log 64}{\log 16} \right\rceil = \left\lceil \frac{6}{4} \right\rceil =$ 2 levels

Number of devices $= \frac{64}{16^1} + \frac{64}{16^2}$

$= 4 + \left\lceil \frac{1}{4} \right\rceil$

$=$ 5 decoders

② 7*128 Decoder | m = 128
1*2 Decoder | n = 2

Number of levels $= \left\lceil \frac{\log 128}{\log 2} \right\rceil = \left\lceil \frac{7}{1} \right\rceil = 7$ levels

Number of Devices $= \frac{128}{2^1} + \frac{128}{2^2} + \frac{128}{2^3} + \frac{128}{2^4} + \frac{128}{2^5} + \frac{128}{2^6} + \frac{128}{2^7}$

$= 64+32+16+8+4+2+1$

$=$ 127 decoders

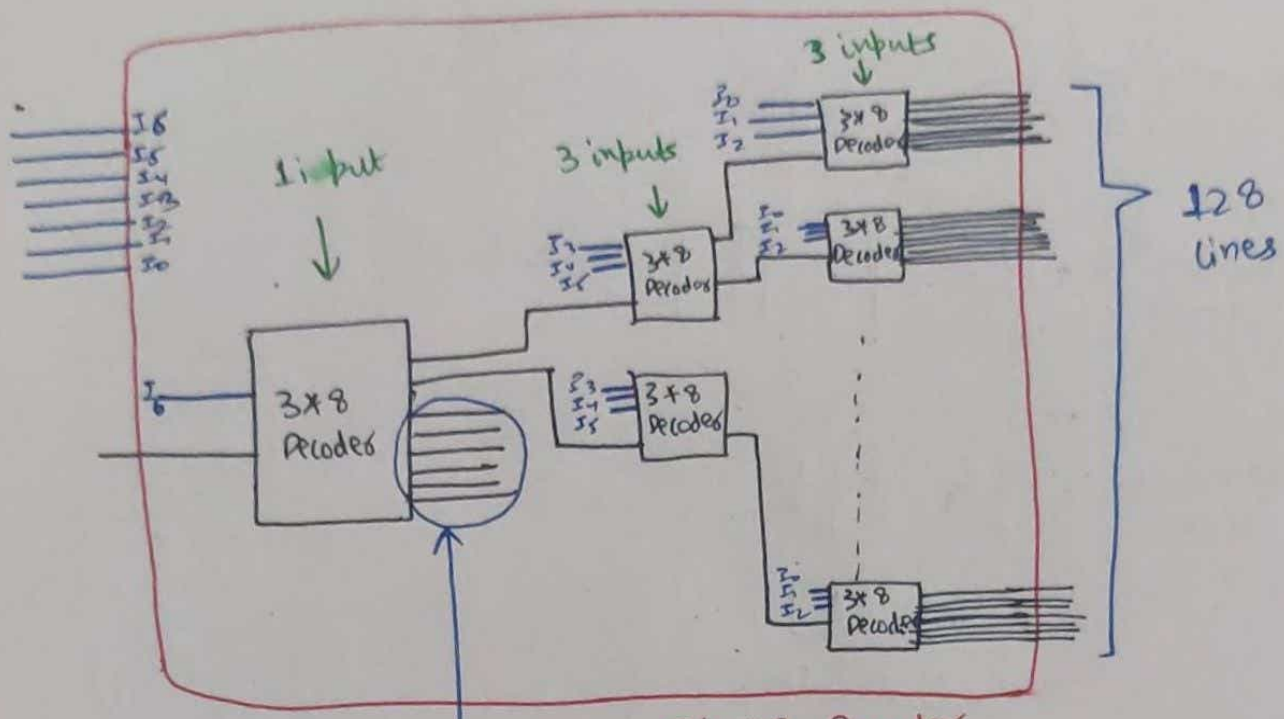
③ 7*128 Decoder | m = 128
3*8 Decoder | n = 8

Number of levels $= \left\lceil \frac{\log 128}{\log 8} \right\rceil = \left\lceil \frac{7}{3} \right\rceil =$ 3 levels

Number of Devices $= \frac{128}{8^1} + \frac{128}{8^2} + \frac{128}{8^3}$

$= 16 + 2 + \left(\frac{1}{4}\right)^{-1}$

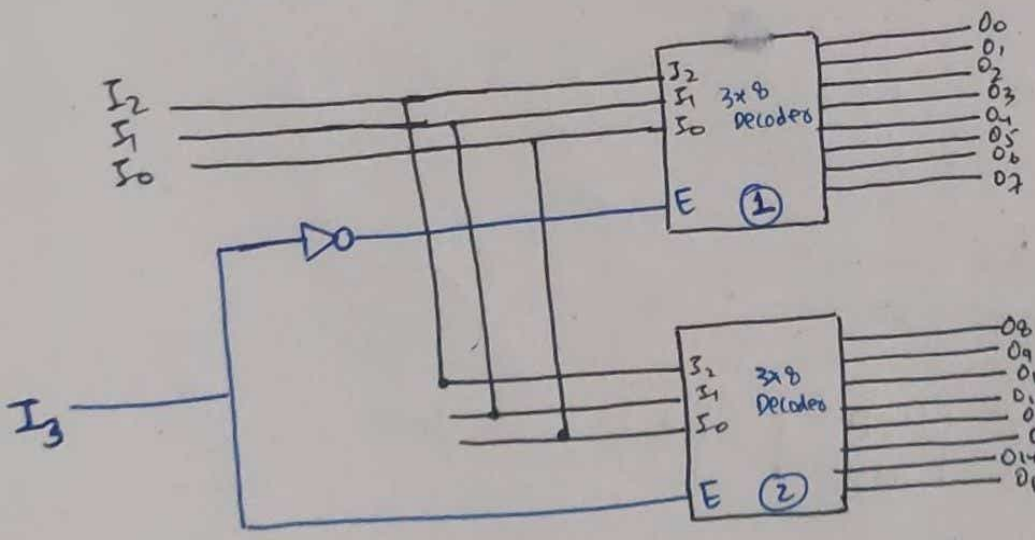
$\Rightarrow$ 19 Decoders




Only 2 lines are used out of 8 lines ie 6 lines are not used

# Expansion of Decoder in Another way:

- 4*16 Decoder using 2 (3*8 Decoder)

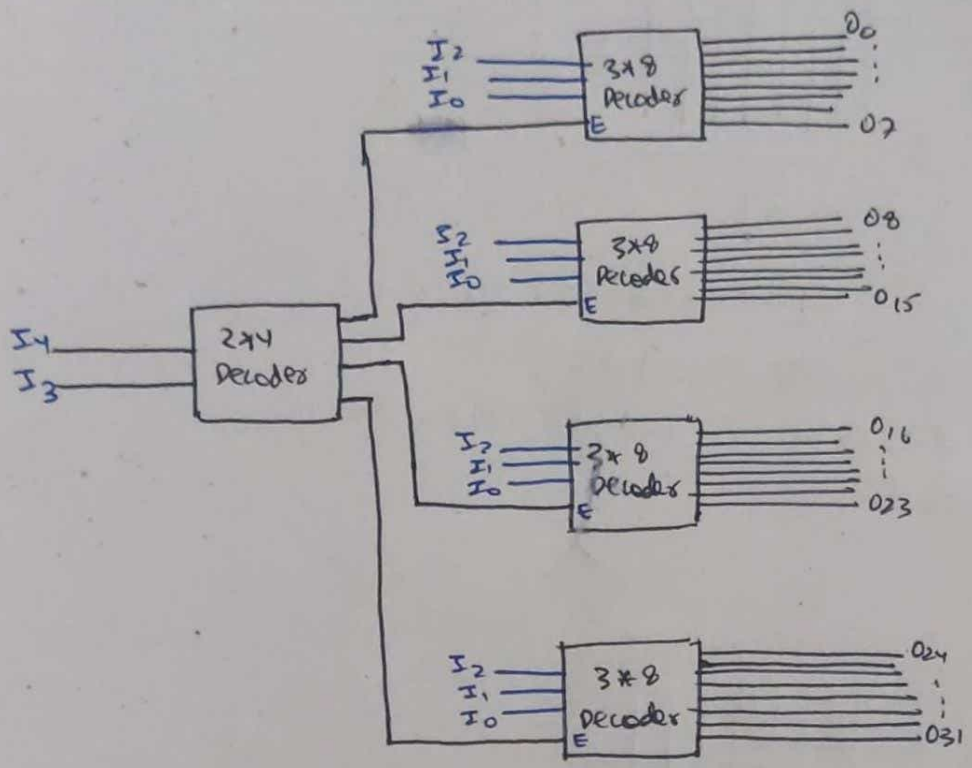



| $I_3$ | $I_2$ | $I_1$ | $I_0$ | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $\Rightarrow O_0$ |
| 1 | 0 | 0 | 0 | $\Rightarrow O_8$ |

i.e If $I_3=0$, $(O_0-O_7)$ → selected
If $I_3=1$, $(O_8-O_{15})$ → selected

If $I_3I_2I_1I_0=0000$, then the (2) decoder will be disable because E=0 (Infact the decoder (2) represent the output in which the most significant bit is 1 ($I_3$ is MSB) & Decoder 1 represents the MSB ($I_3=0$)

# 5*32 Decoder using 4 (3*8) Decoder

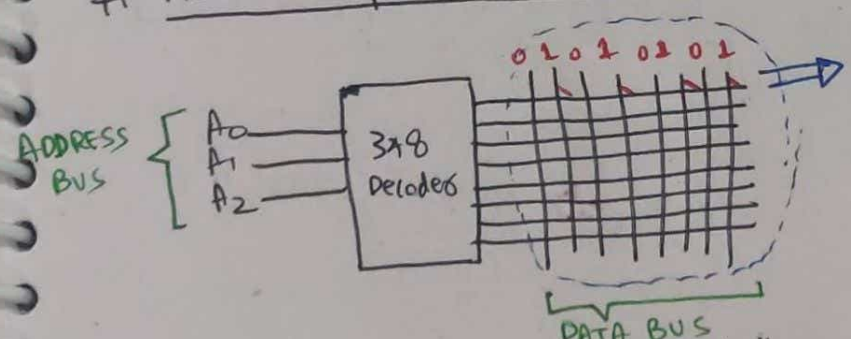



$I_4$ $I_3$ $I_2$ $I_1$ $I_0$

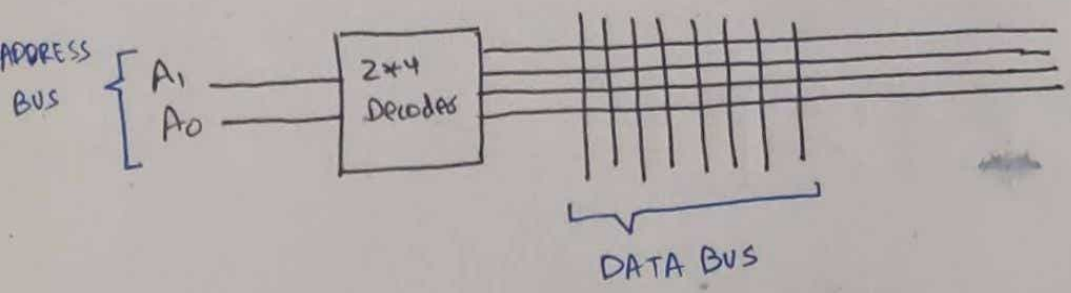
(0 0) ⇐ (D0-D7 selected)
(0 1) ⇐ (D8-D15 selected)
(1 0) ⇐ (D16-D23 selected)
(1 1) ⇐ (D24-D31 selected)

0 0 0 ⇒ $D_0$ selected
0 0 0 ⇒ $D_8$ selected
0 0 0 ⇒ $D_{16}$ selected
0 0 0 ⇒ $D_{24}$ selected

# Address Expansion in ROM:



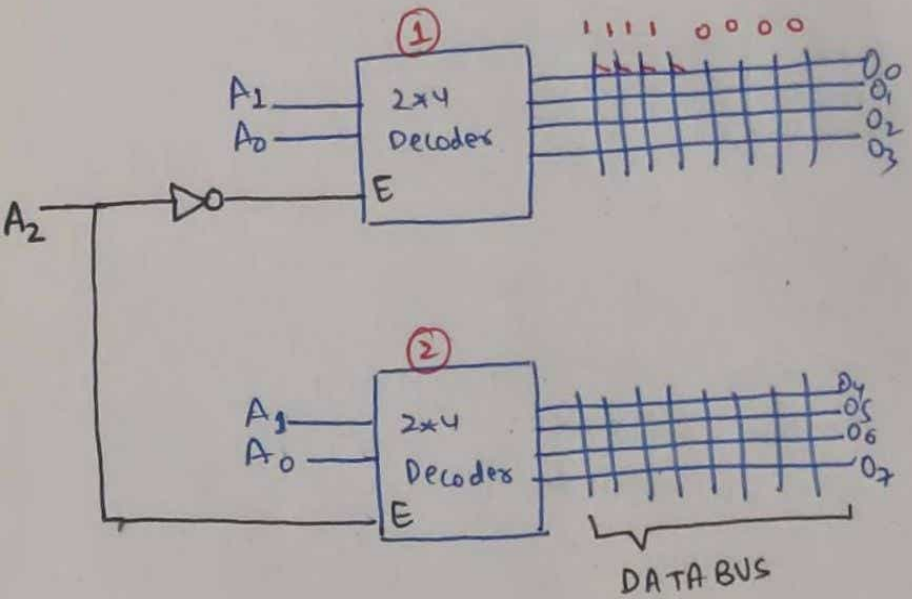

8 words * 8 bits
⇒ 64 bits




4 words * 8 bits

⇒ 32 bits.

If we have to Identify 8 words, then we need 3 Address Bus




8 words * 8 bits

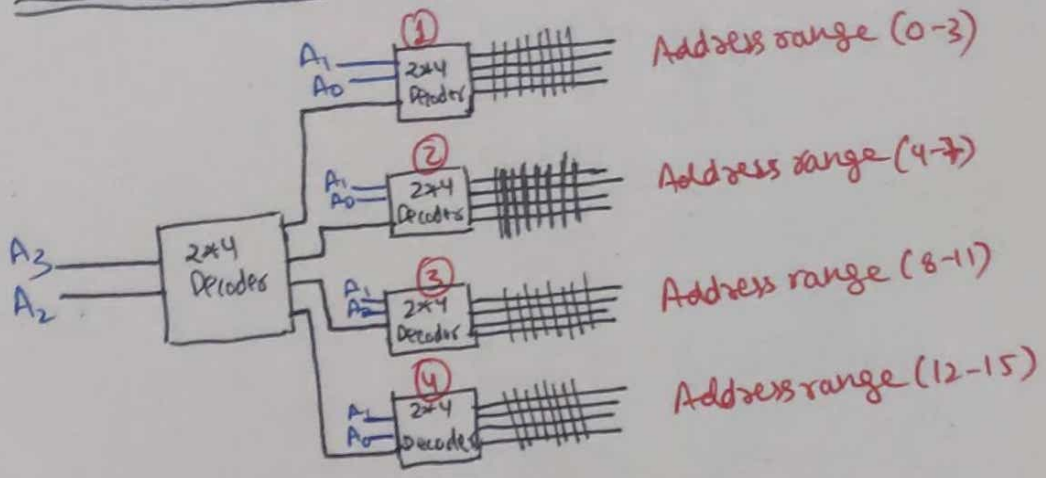
⇒ 64 bits [using 2 (2*4 decoders)]

$A_2$ $A_1$ $A_0$

0 { 0 0 → $O_0$ ; 0 1 → $O_1$ ; 1 0 → $O_2$ ; 1 1 → $O_3$ } Address range (0-3) ⇒ Device 1 is selected

1 { 0 0 → $O_4$ ; 0 1 → $O_5$ ; 1 0 → $O_6$ ; 1 1 → $O_7$ } Address range (4-7) ⇒ Device 2 is selected

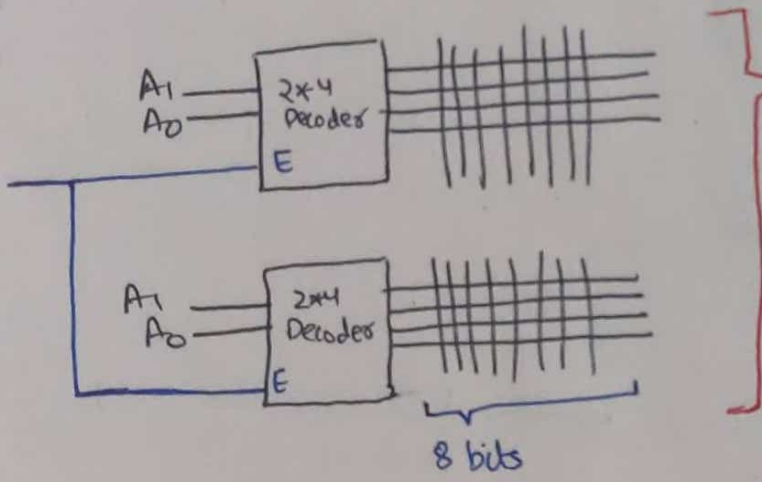
# #Word Expansion of ROM:




| $A_3$ | $A_2$ | $A_1$ | $A_0$ | |
|---|---|---|---|---|
| 0 | 0 | | | → Device 1 is selected. |
| 0 | 1 | | | → Device 2 is selected. |
| 1 | 0 | | | → Device 3 is selected. |
| 1 | 1 | | | → Device 4 is selected. |

16 words * 8 bits

⇒ 128 bits




what happens if we don't use the not gate i.e. what happen if we apply the same enable signal to both of the Devices

i.e If we give Enable as 1, then both the devices will be selected simultaneously & we have only 4 words ie 4 address bus.

| $A_1$ | $A_0$ |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

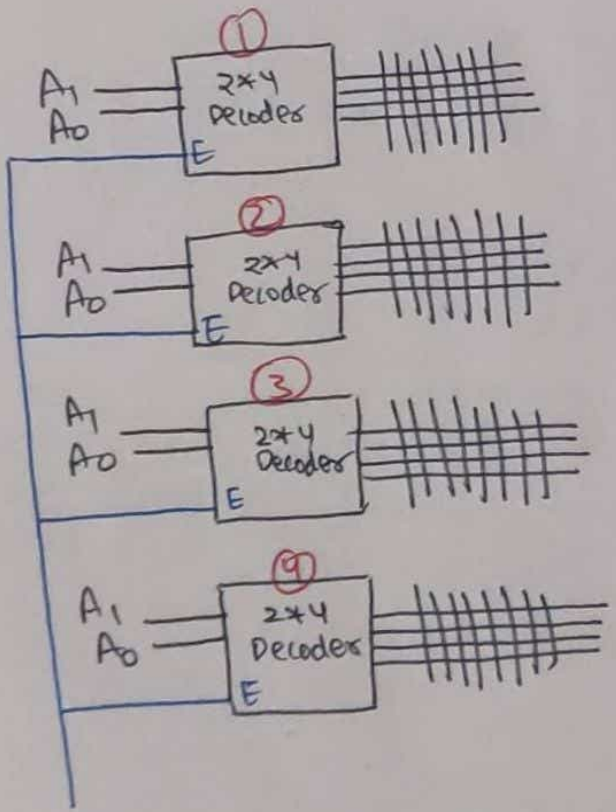
→ 8 bits are come from Device 1 &
8 bits are come from Device 2

⇓

Total 16 bits
(16 Data Bus)

4 words are possible & size of the each word is 16 bits

# 4 Words * 32 bits:

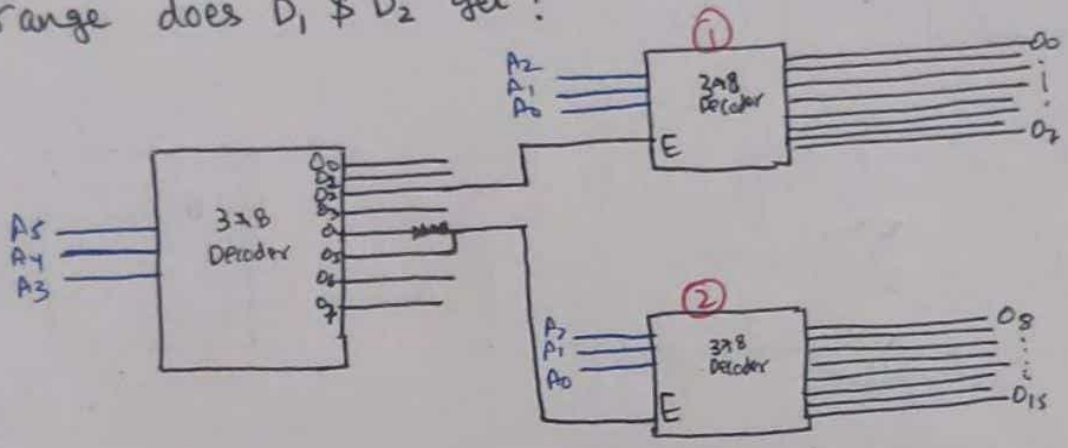



1 ⇒ 8 bits come from each Devices ①, ②, ③ & ④
ie 8+8+8+8
= 32 bits

4 words are possible & size of each word is 32 bits

# Finding the address Range of Devices:

If $A_5A_4A_3A_2A_1A_0$ are the Addresses what range does $D_1$ & $D_2$ get?




For Decoder 1 to be Enabled, the $O_2$ should be enabled

$\Rightarrow A_5A_4A_3$ should be (0 1 0)

$\therefore$

| $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 |
| | | | 0 | 0 | 1 |
| | | | 0 | 1 | 0 |
| | | | 0 | 1 | 1 |
| | | | 1 | 0 | 0 |
| | | | 1 | 0 | 1 |
| | | | 1 | 1 | 0 |
| | | | 1 | 1 | 1 |

$(O_2)$

Illy For Decoder 2 to be Enabled, the $O_5$ should be enabled

$\Rightarrow A_5A_4A_3$ should be (1 0 1)

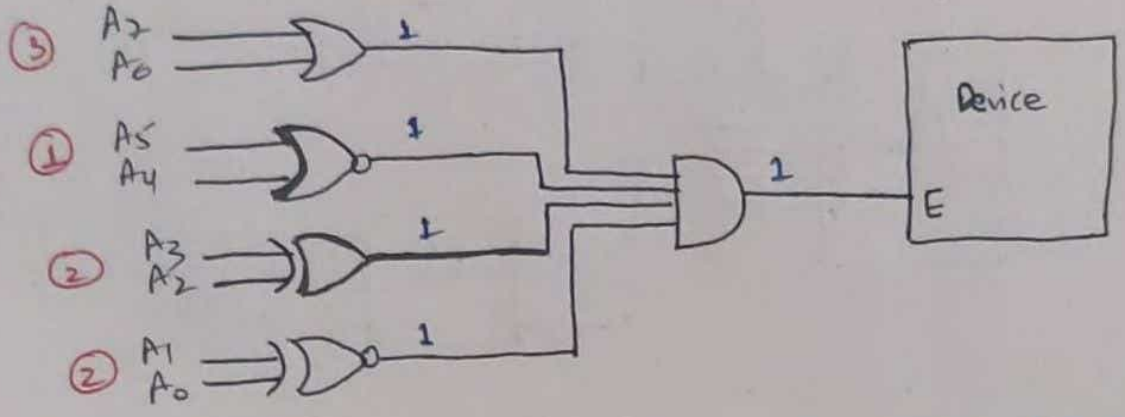
$\therefore$

| $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 0 | 0 | 0 |
| | | | 0 | 0 | 1 |
| | | | 0 | 1 | 0 |
| | | | 0 | 1 | 1 |
| | | | 1 | 0 | 0 |
| | | | 1 | 0 | 1 |
| | | | 1 | 1 | 0 |
| | | | 1 | 1 | 1 |

$(O_5)$

# Example of Enabling a Device:

Consider a device that is enabled using 8 address bits as shown below. For how many addresses, the device is enabled




(a) 1
(b) 8
(c) 12
(d) More

The device is enabled when the output of AND gate is 1, the output of AND gate is 1 iff All of its inputs should be definetely 1

$F_1 = 1$ , $F_2 = 1$ , $F_3 = 1$ , $F_4 = 1$

↑ 3 combinations ↑ 1 combination ↑ 2 combinations ↑ 2 combinations

Total
$3 * 1 * 2 * 2$
$\Rightarrow$ 12 addresses

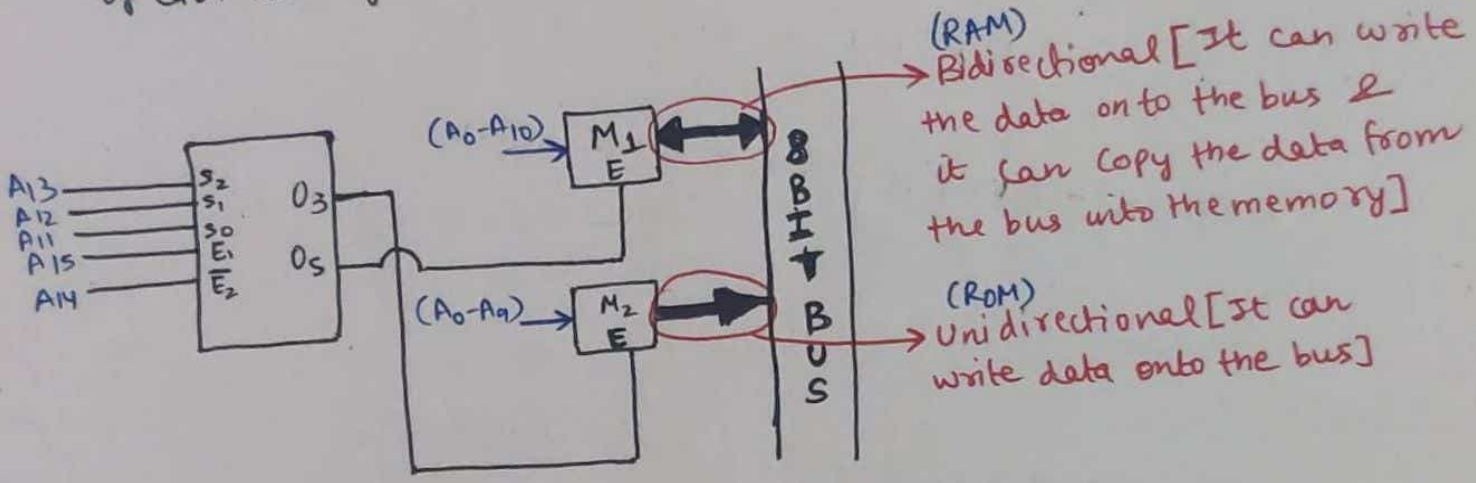
| $A_7$ | $A_6$ | $A_5$ | $A_4$ | $A_3$ | $A_2$ | $A_1$ | $A_0$ |
|---|---|---|---|---|---|---|---|
| (1, 0) | | (0, 0) | | (1, 0) | | (0, 0) | |
| (0, 1) | | | | (0, 1) | | (1, 1) | |
| (1, 1) | | | | | | | |

01001000 } 8bit Address Bus
4 8

$\Rightarrow$ 48 in Hexadecimal number system

# Finding the address ranges of the memory devices :-

Consider the following interface of two memory devices with 3*8 Decoder. Compute the nature & address range of each memory device.




**Memory 1:**

with 11 address bit

$2^{11}$ addresses are possible

} Means $2^{11}$ words can be stored in this single device & length of each word is 8 bits.

$2^{11} * 8$ bits

$\Rightarrow 2^{14}$ bits (Capacity of M1)

**Memory 2:**

with 10 address bit

$2^{10}$ addresses are possible

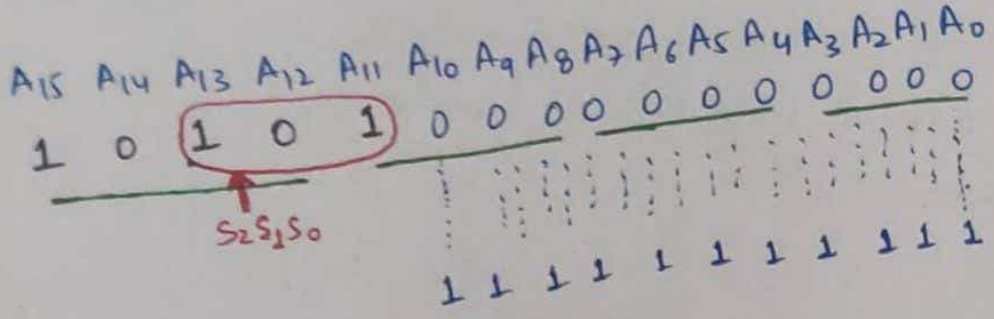
} Means $2^{10}$ words can be stored in this single device & length of each word is 8 bits

$2^{10} * 8$ bits

$\Rightarrow 2^{13}$ bits (capacity of M2)

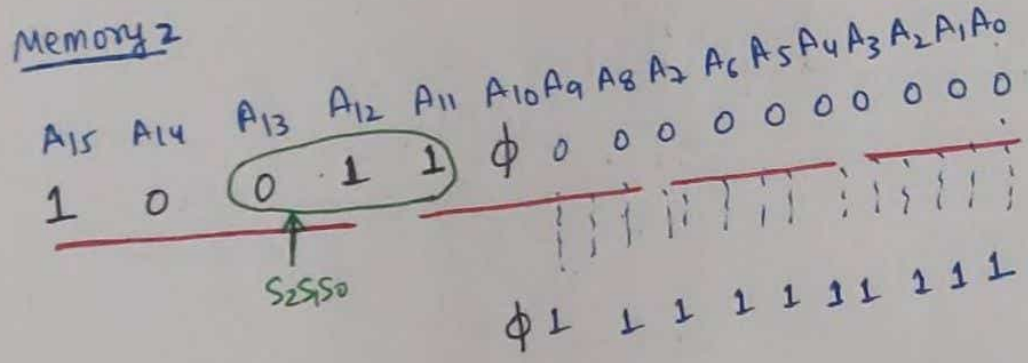
Hence, To enable M1 $\Rightarrow O_5$ should be enabled ie $S_2S_1S_0$ be 101 & the Decoder should also be enabled so that $S_2S_1S_0$ (101) will be enabled.

Memory 1




A800H - AFFFH } Address range of M1

Memory 2




9800H - 9FFFH } Addressrange of M2

# Introduction to Encoders → Encoders are actually opposite of Decoders

① They convert one code to Another.
② They perform lossless compression.
③ They are of two types.

| $n*2^n$ = Decoder |
|---|
| $2^n*n$ = Encoder |

(a) Non priority Encoder (Does not support simultaneous input activation)

(b) Priority Encoder (Supports simultaneous input activation & used for interrupt servicing)

④ Due to static priorities, the lower priority input is exposed to starvation.

eg 4*2 Encoder (Non-priority)

↑ If we give 4 inputs, we get 2 outputs

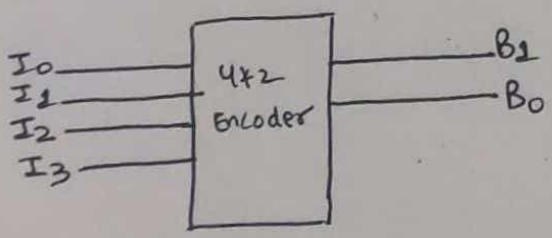
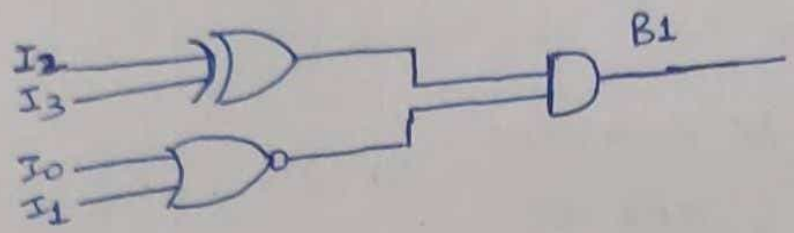


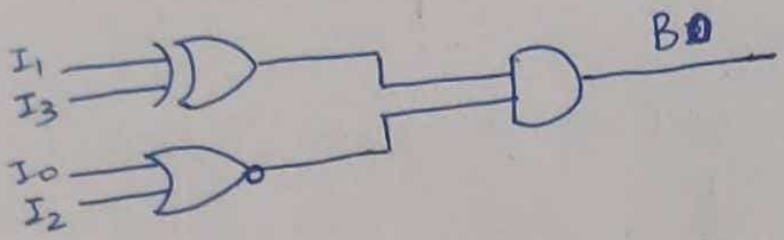

Table:-

| $I_0 I_1 I_2 I_3$ | $B_1$ | $B_0$ |
|---|---|---|
| 0 0 0 1 | 1 | 1 |
| 0 0 1 0 | 1 | 0 |
| 0 1 0 0 | 0 | 1 |
| 1 0 0 0 | 0 | 0 |

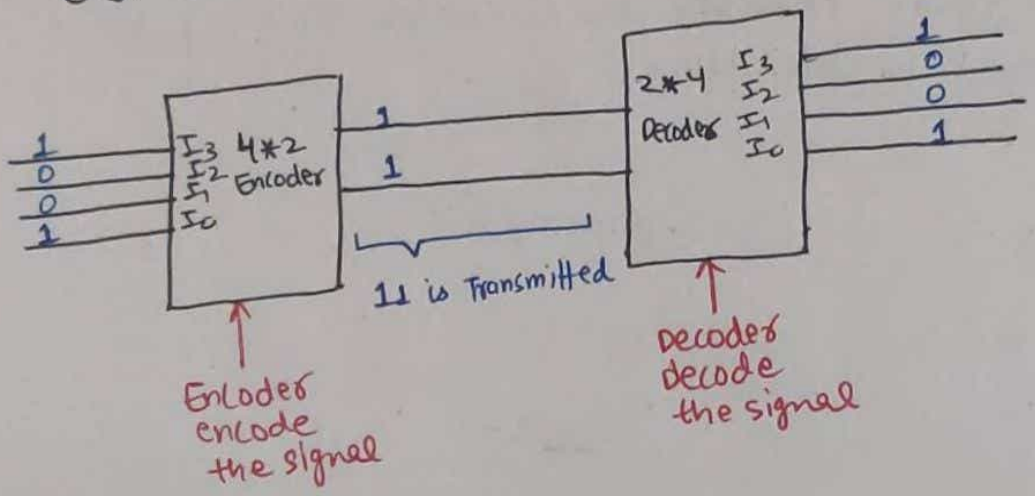
$$B1(I_3, I_2, I_1, I_0) = I_0' I_1' I_2' I_3 + I_0' I_1' I_2 I_3'$$
$$= I_0' I_1' (I_2' I_3 + I_2 I_3')$$
$$= I_0' I_1' (I_2 \oplus I_3)$$

$$Ba(I_3, I_2, I_1, I_0) = I_0' I_1' I_2' I_3 + I_0' I_1 I_2' I_3'$$
$$= I_0' I_2' (I_1' I_3 + I_3' I_1)$$
$$= I_0' I_2' (I_1 \oplus I_3)$$




⇒ Generally Encoders are used at Transmitting end & Decoders are used at Receiving end.




# Priority Encoders

$I_3 > I_2 > I_1 > I_0$ (Highest suffix input is given the highest priority)

| $I_0$ | $I_1$ | $I_2$ | $I_3$ | B1 | B0 |
|---|---|---|---|---|---|
| $\phi$ | $\phi$ | $\phi$ | 1 | 1 | 1 |
| $\phi$ | $\phi$ | 1 | 0 | 1 | 0 |
| $\phi$ | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |

$$B1 = I_3 + I_2 I_3'$$

$$= (I_3 + I_2)(I_3 + I_3')$$

$$= I_2 + I_3$$

$$B0 = I_3 + \bar{I}_3 \bar{I}_2 I_1$$

$$= (I_3 + \bar{I}_2 I_1)(I_3 + \bar{I}_3)$$

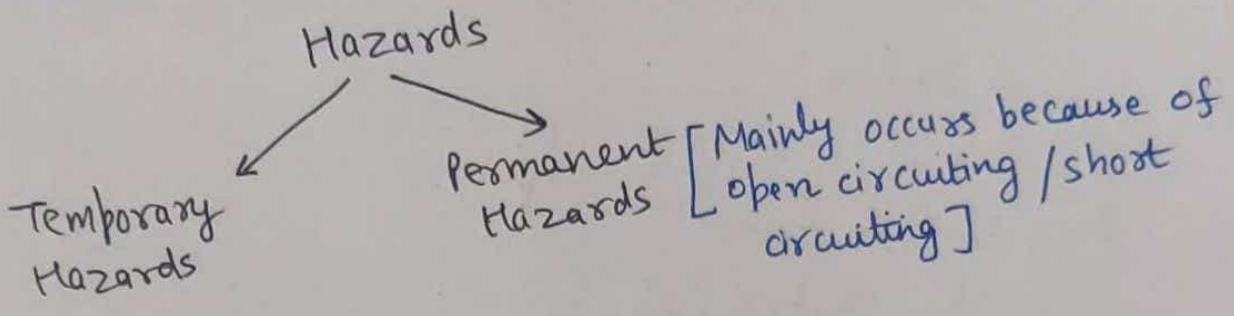
$$= I_3 + \bar{I}_2 I_1$$

* Difference b/w priority Encoder & Non-priority Encoder ?

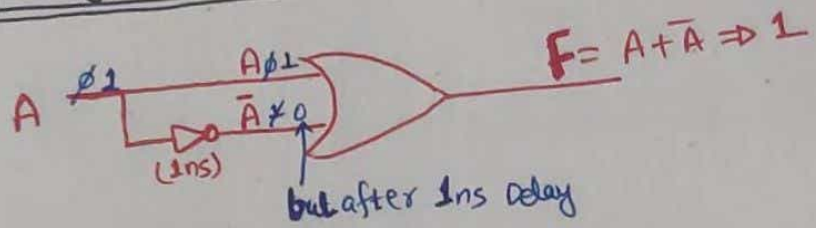
⇒ In priority Encoder, we can give many signal simultaneously.

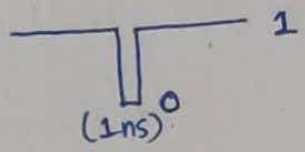
but, in Non-priority Encoder, we are supposed to give more than one input.

# Introduction to Hazards:

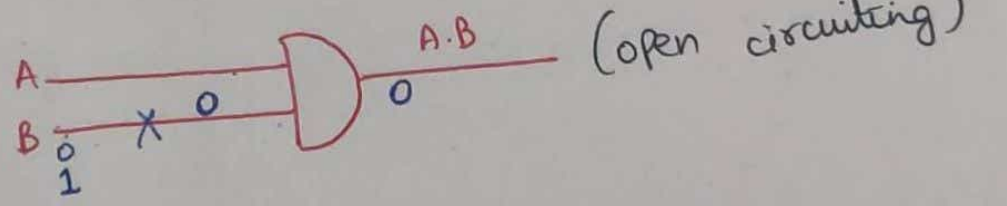


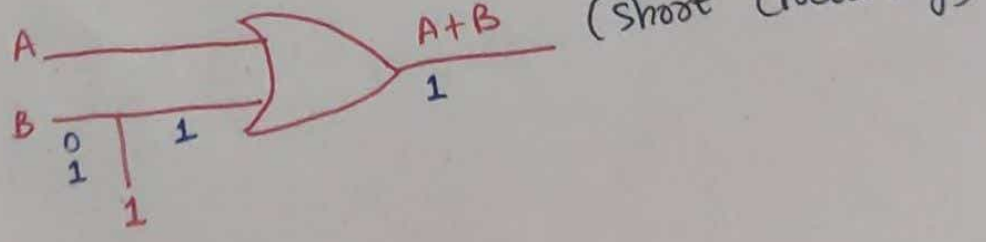

- Temporary Hazards:




- Permanent Hazards:



# Hazards in Digital circuits:

(1) The mal function of the digital circuit is called Hazard. They can be permanent or temporary.

(a) The temporary hazards are due to uneven delay of inputs.

(b) The permanent Hazards are resulted due to open circuit & short circuit of the connecting leads (Terminals)

(2) The Hazards can be Stuck at 0 (s-a-0), Always the value is 0

Stuck at 1 (s-a-1) Always the value is 1

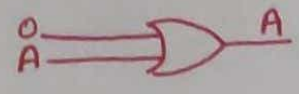
(3) The single fault Analysis is made using "path sensitization technique". This technique provides test vector.

## * Procedure of finding the test vector:

(i) Inactivate all the other path except the tested one.

- For 'AND' or 'NAND' apply the logic 1 for inactivation

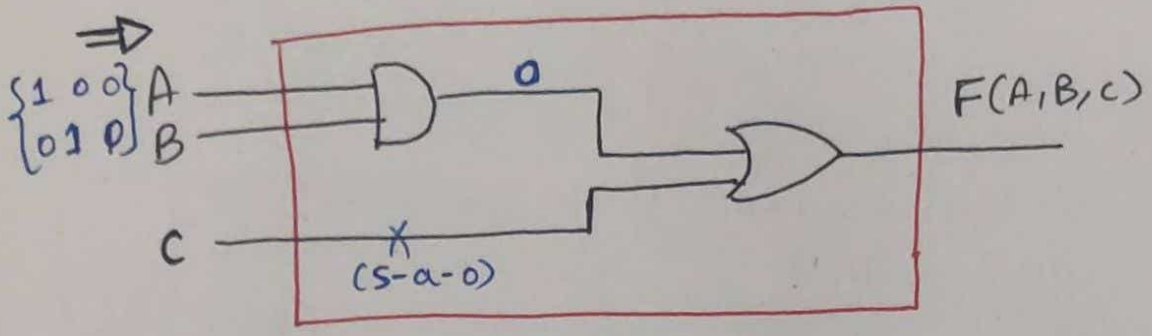

- For 'OR' or 'NOR' apply the logic 0 for inactivation




(ii) Apply the opposite logic value at the tested place (s-a-0) ← If the value appears to be 1, then we can say that it is not stuck at 0

(iii) The input combination satisfying the above requirements forms the test vector.

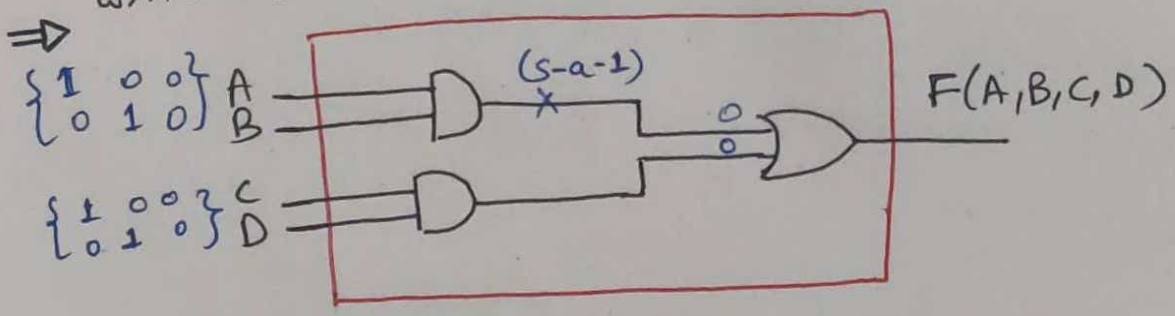
## Example of Test vectors :




| A | B | C |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | |

If the output is 1, then we can say that it work fine or it is not stuck at 1

Not allowed as it give 1 as output with AND gate




Actually 3 input combinations for AB & 3 input combinations for CD

$\therefore (3 \times 3) \Rightarrow$ 9 combinations

| A | B | C | D |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |

Total 9 input combinations

# Adders

Adders are the most frequently occuring circuits in a computer Any operation that we can take is actually implemented in terms of Addition.

## (1) Half Adder

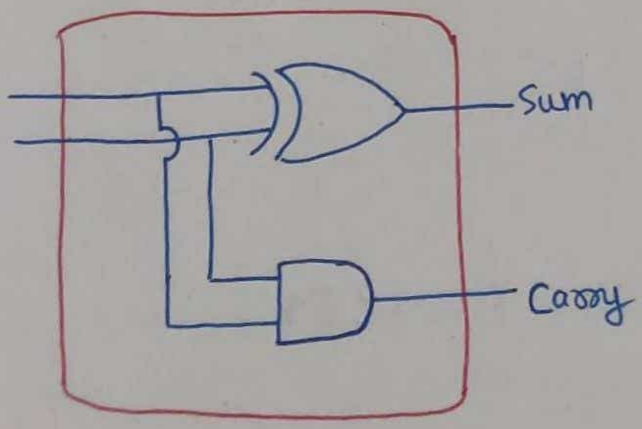
(Two-Input, Two-Output) → circuit

| X | Y | Sum | Carry |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$Sum = x\bar{y} + \bar{x}y$

$\Rightarrow x \oplus y$

$Carry \Rightarrow xy$




Half Adder

Half Adder means Adding 2 bits

## (2) Full Adder:

(3 Input, 2 output) → circuit

Used for adding 3 bits

| X | Y | Z | Carry | Sum |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

$Sum = X \oplus Y \oplus Z$

$Carry = Z(X \oplus Y) + XY$

$Sum = \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XYZ \quad [\Sigma(1,2,4,7)]$

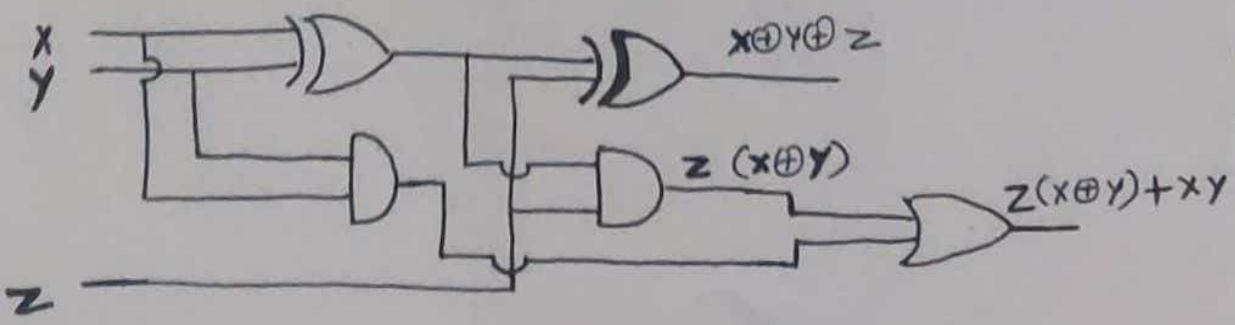
| Z \ XY | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | 1 | | 1 |
| 1 | 1 | | 1 | |

$\Rightarrow X \oplus Y \oplus Z$

$Carry = \bar{X}YZ + X\bar{Y}Z + XY\bar{Z} + XYZ \quad [\Sigma(3,6,5,7)]$

| Z \ XY | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | | | 1 | |
| 1 | | 1 | 1 | 1 |

$\Rightarrow XY + YZ + ZX = Z(X \oplus Y) + XY$

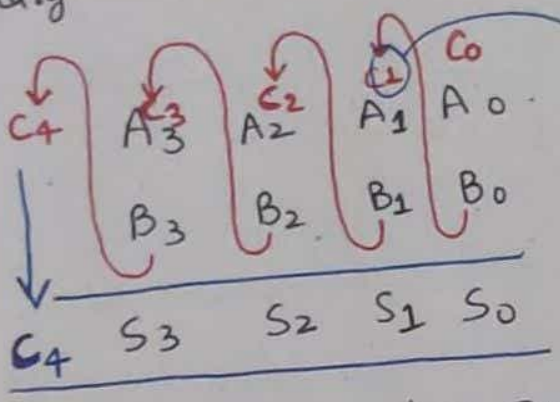


(3) Binary Adder :

#Ripple Carry Adder :

The ripple carry adder is used to add two Binary numbers

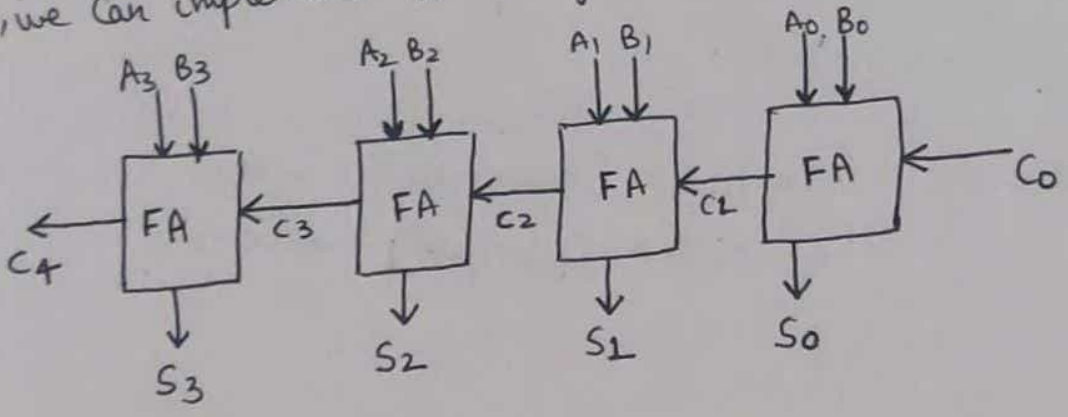
1st Binary number = $A_3 A_2 A_1 A_0$
2nd Binary number = $B_3 B_2 B_1 B_0$
} Now, carry $C_0$ is provided, then -




Carry generated by Adding $A_0$ & $B_0$

$C1 = C_0(A_0 \oplus B_0) + A_0B_0$
$C2 = C1(A_1 \oplus B_1) + A_1B_1$
$C3 = C2(A_2 \oplus B2) + A_2B_2$
$C4 = C3(A_3 \oplus B3) + A_3B_3$

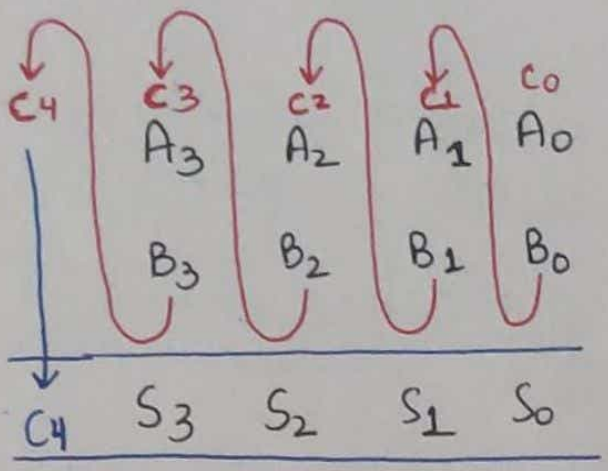
Now, we can implement it using Full Adder :




⇒ The disadvantage of the Ripple carry adder is $C_1$ is generated after $C_0$ & $C_2$ is generated only after $C_1$.

∴ Full Adders can't be executed in parallel & it takes more time to compute. So some Automated circuits are designed & placed at each Full adder & it is called carry look ahead adder

# Carry look Ahead Adder:




$C_1 = C_0(A_0 \oplus B_0) + A_0B_0$

$C_2 = C_1(A_1 \oplus B_1) + A_1B_1$

$C_3 = C_2(A_2 \oplus B_2) + A_2B_2$

$C_4 = C_3(A_3 \oplus B_3) + A_3B_3$

**Let** Assume, $G_i = A_iB_i$ [Generating Function]

$P_i = A_i \oplus B_i$ [propogating Function]

Now, $C_1 = C_0P_0 + G_0$

$C_2 = C_1P_1 + G_1$

$C_3 = C_2P_2 + G_2$

$C_4 = C_3P_3 + G_3$

$C_1 = C_0P_0 + G_0$

$C_2 = C_1P_1 + G_1$

$= (C_0P_0 + G_0)P_1 + G_1$

$= C_0P_0P_1 + G_0P_1 + G_1$

$C_3 = C_2P_2 + G_2$

$= (C_0P_0P_1 + G_0P_1 + G_1)P_2 + G_2$

$= C_0P_0P_1P_2 + G_0P_1P_2 + G_1P_2 + G_2$

$C_4 = C_3P_3 + G_3$ $\quad + G_2$

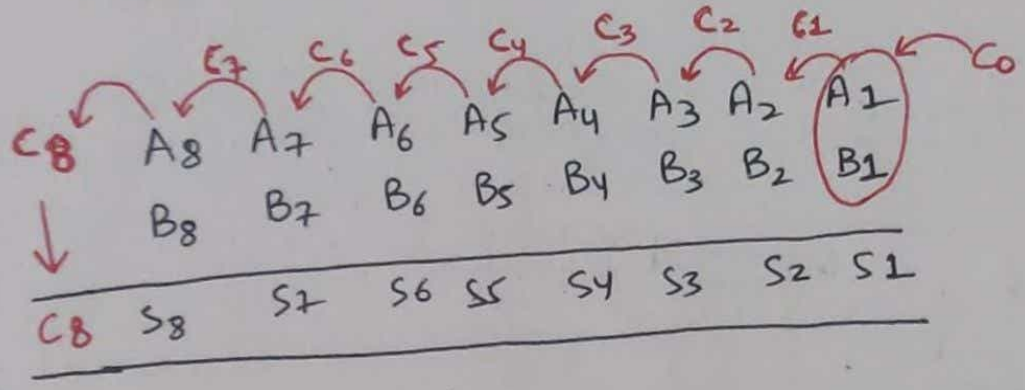
$= (C_0P_0P_1P_2 + G_0P_1P_2 + G_1P_2)P_3 + G_3$

$= C_0P_0P_1P_2P_3 + G_0P_1P_2P_3 + G_1P_2P_3 + G_2P_3 + G_3$

# Hybrid adder

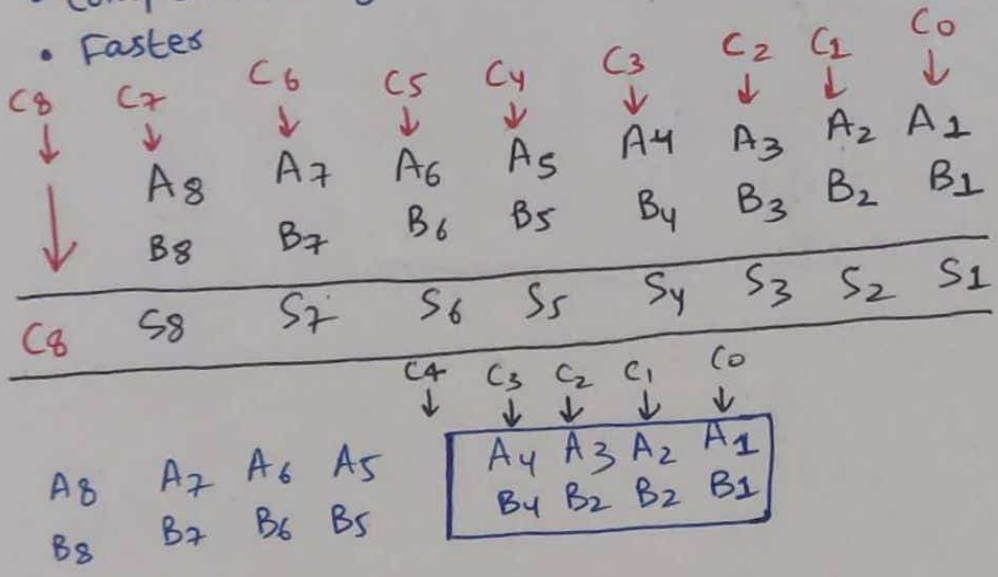
## ⇒ RCA (Ripple carry Adder)

- simple, cheaper
- slower




## ⇒ CLA (Carry look Ahead Adder)

- Complex, costly
- Faster



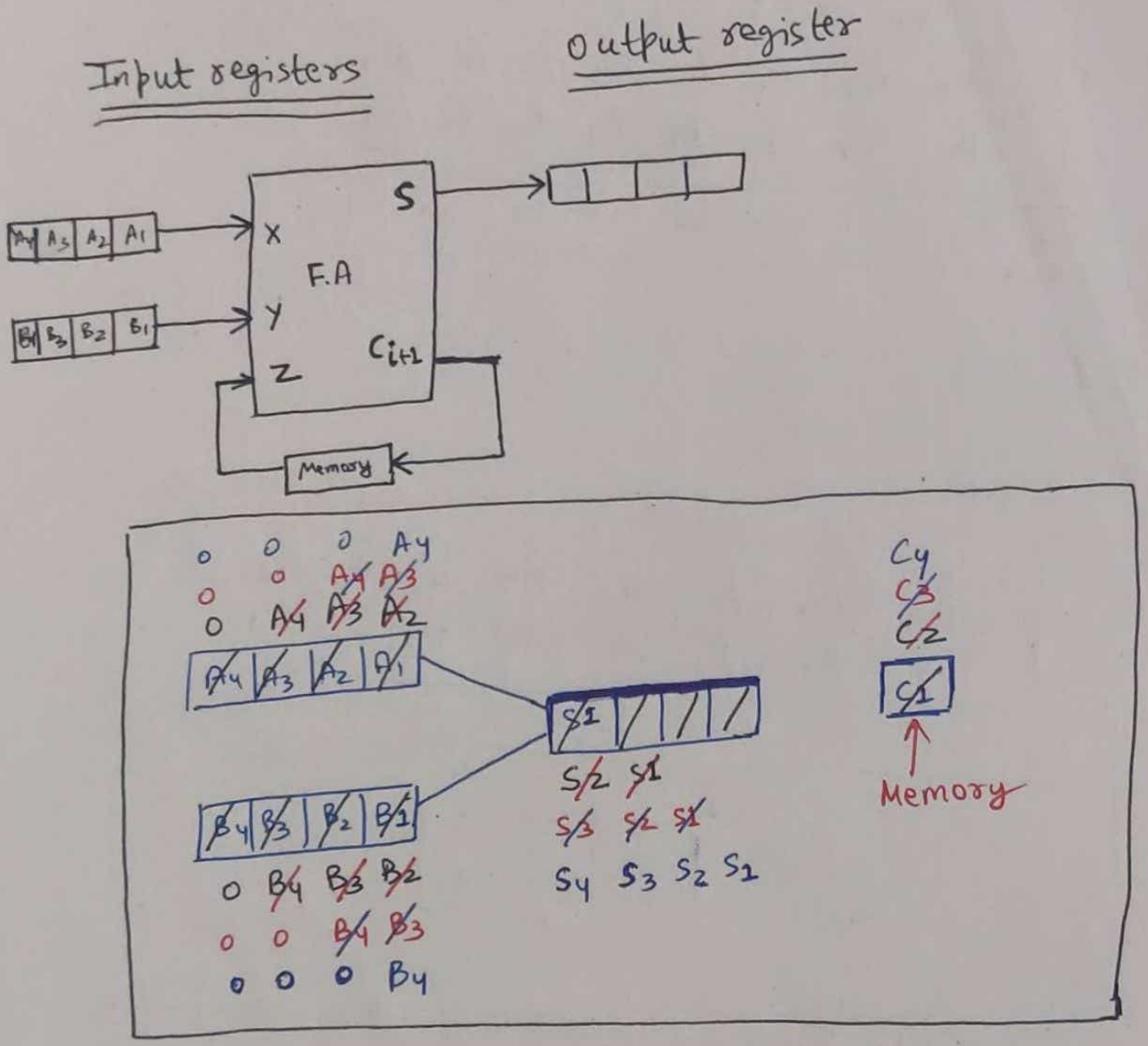

Out of 8 bits First calculate the 4 least significant bits & then Carry which is generated from First gives to second or left 4 most significant bits are calculated.

Actually we add the sum of 4 bits & then with the same device we add the sum of another 4 bits. so that the device is not very complex & be fast enough.

# Serial Adder:

(It comes under sequential circuit)

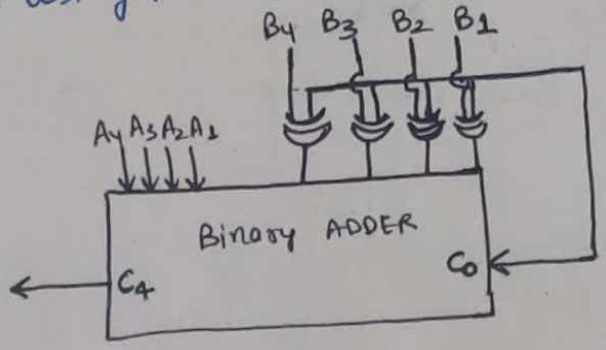
uses only one Full adder to add the 2 Binary numbers




Actually serial Adder is very very slow but it is the cheapest one, As here we need only one full adder.

## #Binary ADDER-SUBSTRACTOR:

Now, we need to add two Binary numbers & one number is directly provided as Input & other Input is provided using x-OR gates.




Now, $B \oplus 0 = B$
$B \oplus 1 = \bar{B}$
} If X=0, then the o/p of XOR gates will be same ($B_4B_3B_2B_1$) & it behaves as Binary Adder.

If X=1, then the o/p of XOR gates will be complimented ($\bar{B}_4\bar{B}_3\bar{B}_2\bar{B}_1$) = 1st compliment

If X=1, A + (1's complement of B) + 1
= A + 2's complement
= A - B & it behaves as Binary ~~Adder~~ Substractor

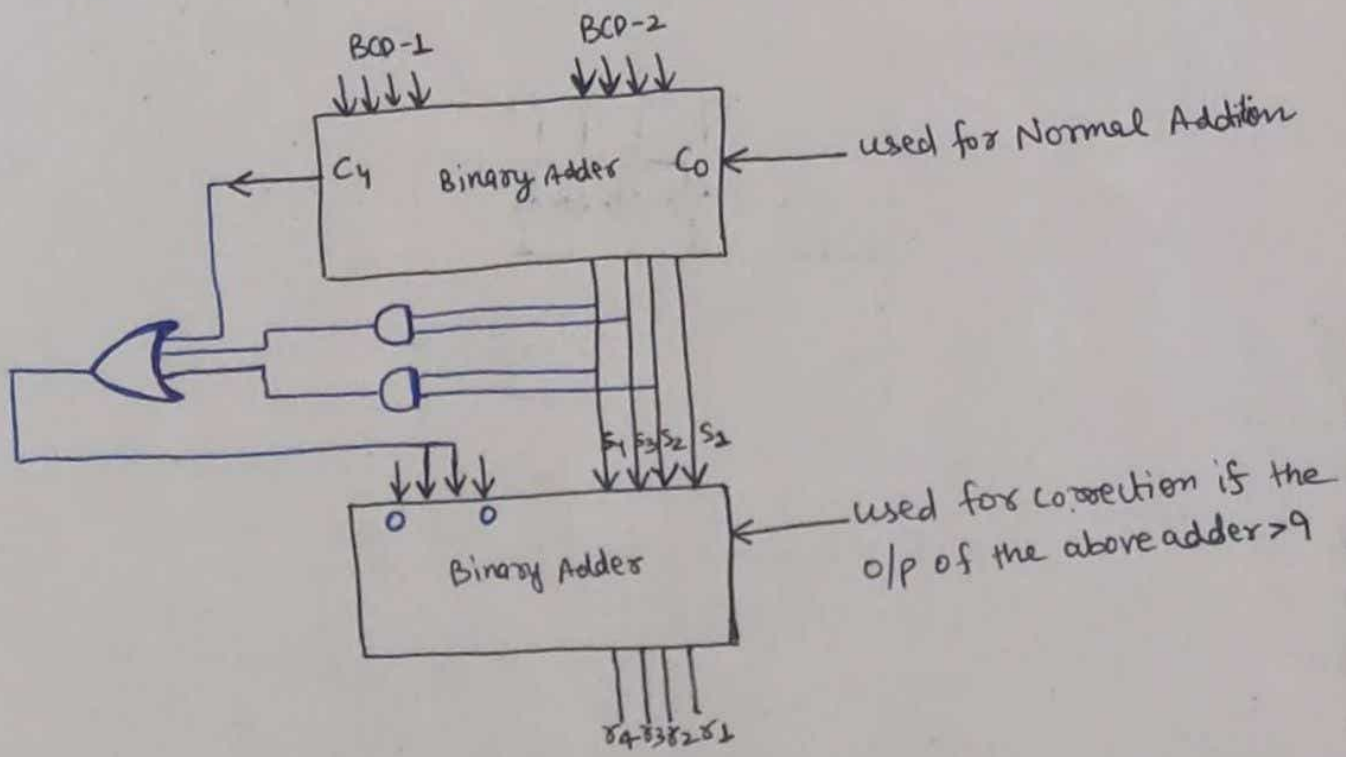
∴ The Adder (behaves as Adder when X=0) & (behaves as substractor when X=1)

The above Circuit not only Functions as Adder & Substractor :-

| A | B | X | Function |
|---|---|---|---|
| BCD | 0011 | 0 | BCD → Ex-3 converter |
| Ex-3 | 0011 | 1 | Ex-3 → BCD converter |
| 1001 | BCD | 1 | (9-B) ⇒ 9's complement of given number B in (BCD) |
| 1010 | BCD | 1 | (10-B) ⇒ 10's complement of given number B in (BCD) |

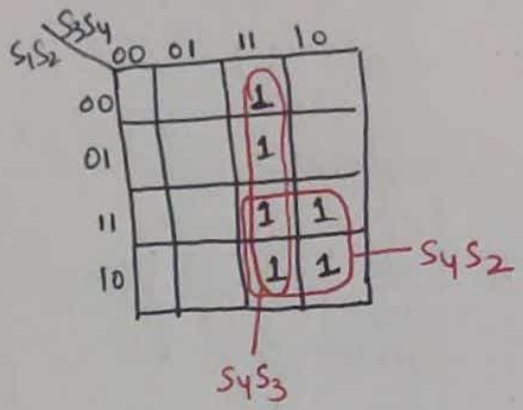
# BCD Adders:

The Output is Invalid if o/p is >9, & we do a small correction (we add 6 to the result) so, we need 2 Adders




Connection is Required when $S_4S_3S_2S_1 > 9$ or if $C_4 = 1$,
Now let us form a function where the sum of $S_4S_3S_2S_1 > 9$,
Now the combinations for which the sum is greater than 9 (>9) are :-

| S.NO | $S_4$ | $S_3$ | $S_2$ | $S_1$ | F |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 |
| 2 | 1 | 0 | 1 | 1 | 1 |
| 3 | 1 | 1 | 0 | 0 | 1 |
| 4 | 1 | 1 | 0 | 1 | 1 |
| 5 | 1 | 1 | 1 | 0 | 1 |
| 6 | 1 | 1 | 1 | 1 | 1 |

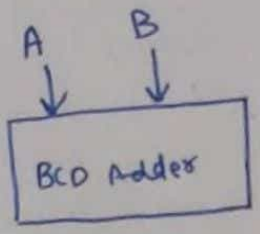



$F = S_4S_3 + S_4S_2$

Now, If the sum have to be greater than 10 then either the $S_4S_2$ should be 1 or $S_4S_3$ should be 1, irrespective of other combinations.

If $S_4S_2$ or $S_4S_3$ are 1 then sum is greater than 9 (sum>9) & when $C_4=1$, then also the sum is greater than 9

## # Invalid combinations of the BCD Adder:




A & B are each of 4 bits Hence, 16 combinations are possible for A & 16 combinations for B, out of which the sum (0-9) [10 numbers] are valid.

Method-1

Invalid combinations = Total combinations − Valid combinations

$= (16 * 16) - (10 * 10)$

$= 256 - 100$

$= 156$

Method-2

A (10-15) → 6 are invalid

B (10-15) → 6 are invalid

$\Rightarrow (6 * 16 + 6 * 16) - (6 * 6)$ → Intersection

$\Rightarrow 156$

Invalid combinations = 156

# 2 Bit Comparator:

$00 < 10$
$10 > 01$
$11 = 11$
} Size of input = 2 bits

Exclusive-NOR is going to act as a Bit comparator

[For 2 Bits]

Method-1

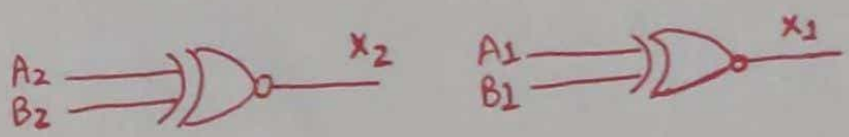
| A | | B | | A>B | A=B | A<B |
|---|---|---|---|---|---|---|
| $A_2$ | $A_1$ | $B_2$ | $B_1$ | | | |
| ↑ MSB | ↑ LSB | ↑ MSB | ↑ LSB | | | |

Method-2

XNOR

| A | B | A⊙B |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

The XNOR produces 1 iff both A & B are same, Now apply this concept to Comparator




Case1: A=B if $x_1=1$ & $x_2=1$ ie. $x_1 \cdot x_2 = 1$

Case2: A>B if $(A_2 > B_2)$ or $(A_2 = B_2 \ \& \ A_1 > B_1)$

$x_2=1$, $A_2=1$, $B_2=0$, $A_1=1$, $B_1=0$

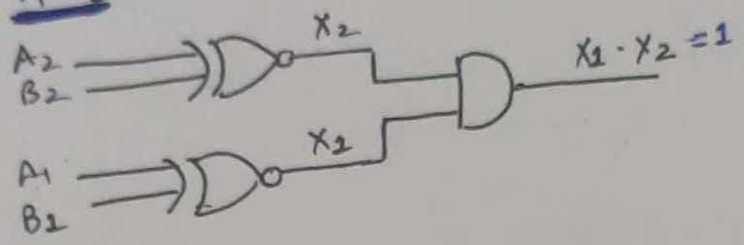
$$F = A_2\overline{B_2} + x_2\overline{B_1}A_2 = 1$$

Case3: A<B if $(A_2 < B_2)$ or $(A_2 = B_2 \ \& \ A_1 < B_1)$

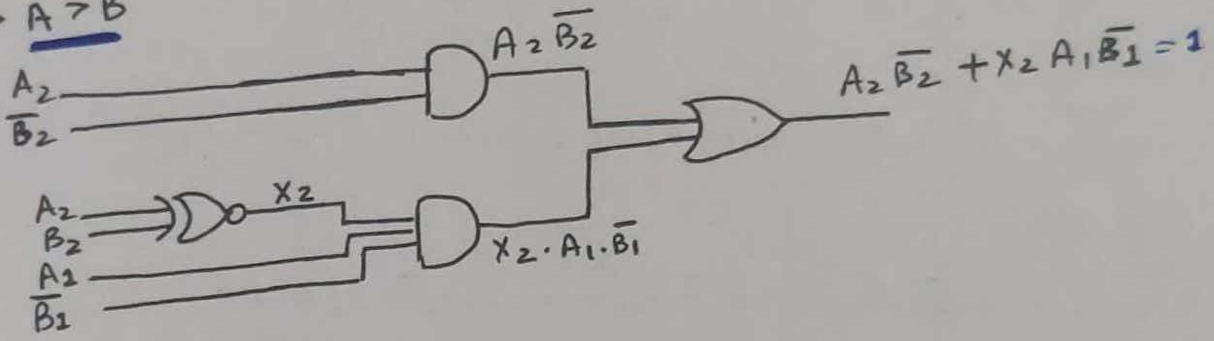
$x_2=1$, $A_2=0$, $B_2=1$, $A_1=0$, $B_1=1$

$$F = \overline{A_2}B_1 + x_2\overline{A_1}B_1 = 1$$

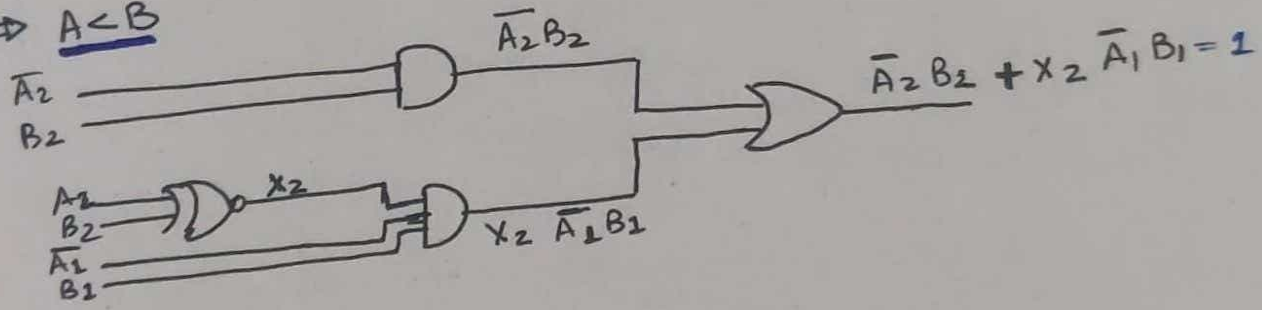
⇒ A=B




⇒ A>B




⇒ A<B




# 3 bit comparator:

$A = A_3 A_2 A_1$

$B = B_3 B_2 B_1$

Case 1 : $A = B$

$x_3 \cdot x_2 \cdot x_1 = 1$

Case 2 : $A > B$

$A_3 \overline{B_3} + x_3 A_2 \overline{B_2} + x_3 x_2 A_1 \overline{B_1} = 1$

Case 3 : $A < B$

$\overline{A_3} B_3 + x_3 \overline{A_2} B_2 + x_3 x_2 \overline{A_1} B_1 = 1$

## # 4 Bit comparator:

$A = A_4 A_3 A_2 A_1$

$B = B_4 B_3 B_2 B_1$

Case 1: $A = B$

$$x_4 \cdot x_3 \cdot x_2 \cdot x_1 = 1$$

Case 2: $A > B$

$$A_4\bar{B_4} + x_4 A_3\bar{B_3} + x_4 x_3 A_2\bar{B_2} + x_4 x_3 x_2 A_1\bar{B_1} = 1$$

Case 3: $A < B$

$$\bar{A_4} B_4 + x_4\bar{A_3} B_3 + x_4 x_3\bar{A_2} B_2 + x_4 x_3 x_2\bar{A_1} B_1 = 1$$

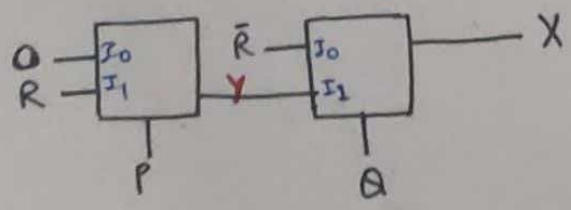
Hence, If we have 2 $n$-bits number, then the number of combinations in which

$A = B \longrightarrow 2^n$

$A > B \longrightarrow \frac{2^{2n} - 2^n}{2}$

$A < B \longrightarrow \frac{2^{2n} - 2^n}{2}$

Q Consider the two cascaded 2-to-1 multiplexer as shown in figure




$Y = \bar{P}.0 + PR$

$= PR$

$X = \bar{Q}.\bar{R} + Q.PR$

$= PQR + \bar{Q}\bar{R}$

The minimal SoP form of output X is -

(a) $\bar{P}\bar{Q} + PQR$

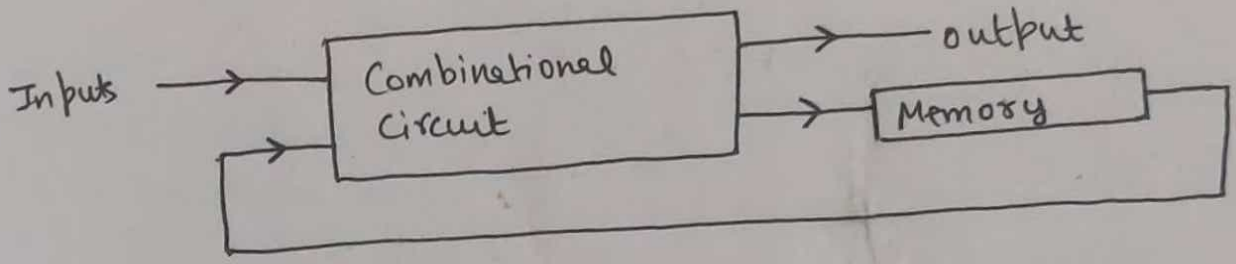
(b) $\bar{P}Q + QR$

(c) $PQ + \bar{P}QR$

(d) $PQR + \bar{Q}\bar{R}$

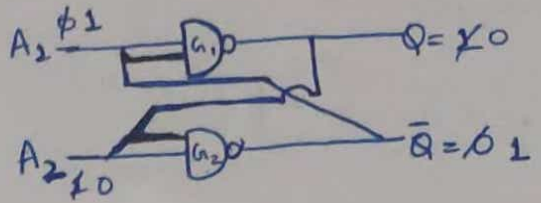
# Introduction to sequential circuits:

- The external output of a sequential circuit depends on external inputs & on present contents of memory element.
- The present contents of a memory elements are called present state & the new contents of a memory elements are obtained by taking external inputs & present state. This is called next state.



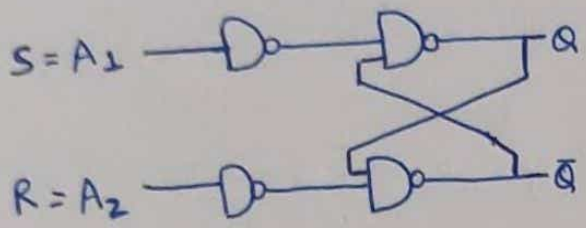

# Latch & Flip flop:

- A flipflop is going to store 1 bit of information
- A memory element is some medium in which one bit of information can be stored or retained until necessary, & there after its content can be replaced by new values.
- The basic binary or digital memory circuit is known as flipflop.
- It has 2 stable states which are known as '1' or '0' so it is also called as bistable multi-vibrator.




LATCH OR LOCKING

## # SR - Flipflop:




Let the present state be represented by $Q_n$ & the next State be represented by $Q_{n+1}$

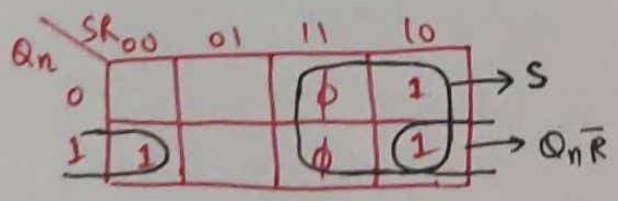
$$Q_{n+1} = F(S, R, Q_n)$$

↳ The output is not just a function of the present Inputs but also previous outputs.

### Characteristics Table:-

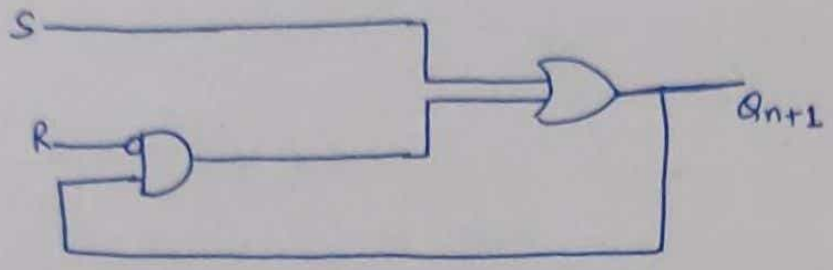
| S | R | $Q_n$ | $Q_{n+1}$ | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | } Latch |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | } Reset |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | } Set |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | $\phi$ | } Invalid (o/p is irrespective of the values of Q & $\bar{Q}$) |
| 1 | 1 | 1 | $\phi$ | |

we can use the SR-Flipflop in only 3 modes ⇒ Latch, set & Reset modes.




Hence, $Q_{n+1} = S + Q_n\bar{R}$ } Characteristics equation

## Realization of SR Flip flop:




⇒ Excitation table represents the present output & what is the output that you are expecting in the next state & what are the combinations that we should provide to get the output of next state that we have assumed.

| $Q_n$ | $Q_{n+1}$ | S | R |
|---|---|---|---|
| 0 | 0 | 0 | $\phi$ → R can be '0' or '1' so $\{\phi\}$ |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | $\phi$ | 0 |

} Excitation Table

Minimized characteristic table is called "function table".

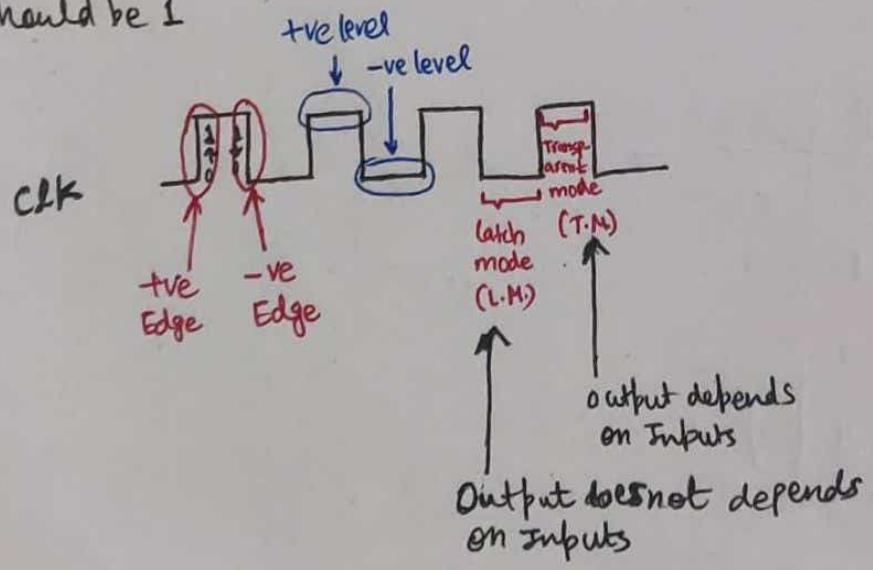
| S | R | $Q_{n+1}$ |
|---|---|---|
| 0 | 0 | $Q_n$ |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | $\phi$ |

# clocked Flipflops:

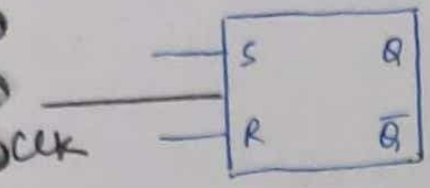
Now, what happens in general with S-R flipflops/flipflops is, because of some external disturbances the i/p symbol will suddenly fluctuate & then the output will change & so what we are supposed to do is we should make the Inputs work only during sometime & we should not let it change during the other time.

clock varies b/w 0 & 1

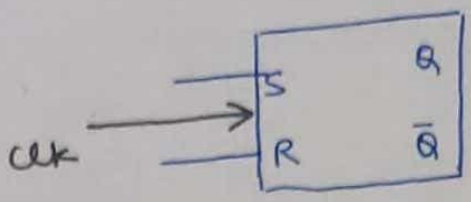
when clock is '0' then the o/p of NAND gates will be 1 & 1 irrespective of the values of S,R It is as good as having S,R as 0,0 ⇒ Latch mode is enabled. So, when we want to change the o/p depending on S,R then the clock value should be 1



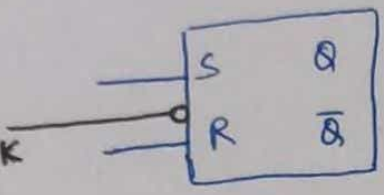

Edge triggered means the output of the flipflops depends on the Input only on some particular edge so, when we trigger that particular edge then the o/p of Flipflop depends on Input.



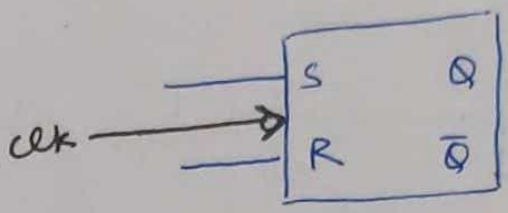

(+ve level)




(+ve Edge)




(-ve level)

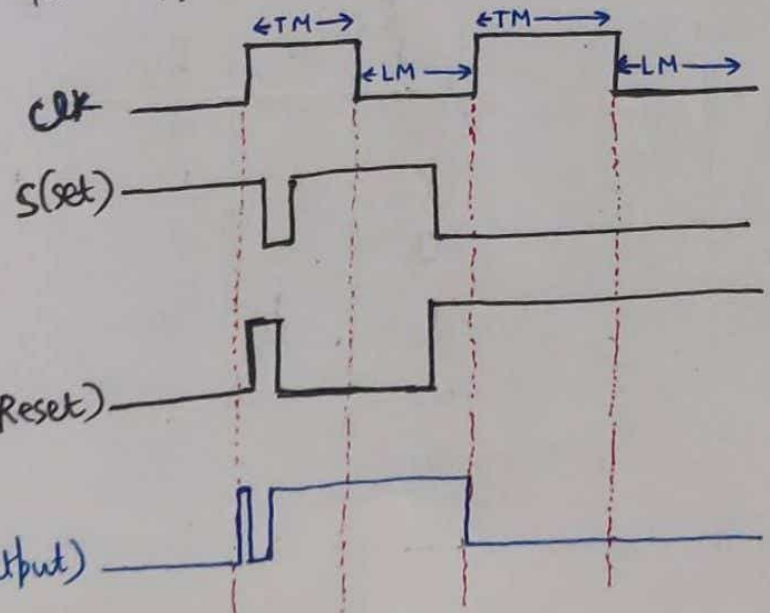



(-ve Edge)

## #Positive Level Triggered:

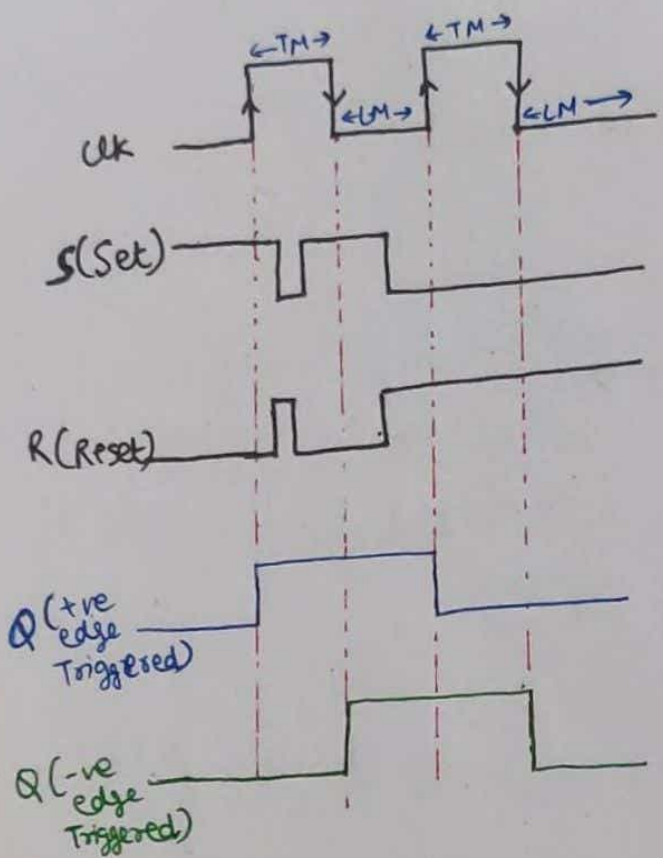
Now, let see how the positive level triggered flipflop (FF) react to the Inputs.

In SR flipflop 'S' means set & 'R' means Reset. The reason is whenever 'S' is 1 the o/p is always going to be 1 & whenever 'R' is 1 the output is always going to be 0 & when both S, R = 0, then the o/p will be in latch mode.




TM = Transparent Mode
LM = Latch Mode

# Edge Triggered (+ve as well as -ve):




TM = Transparent Mode.
LM = Latch Mode.

# JK Flip-flop:

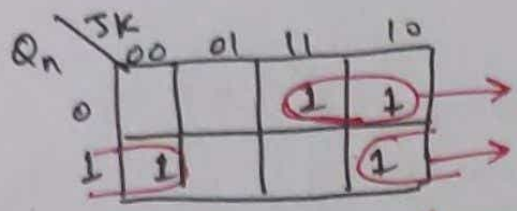
The JK flipflop is same as SR flipflop (J=S, K=R).
It is defined for J=K=1 also & the flipflop performs Complementation for (J=K=1)

Characteristic Table:-

| J | K | Qn | Qn+1 | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Same as SR flip flop. |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 1 | Complementation |
| 1 | 1 | 1 | 0 | |

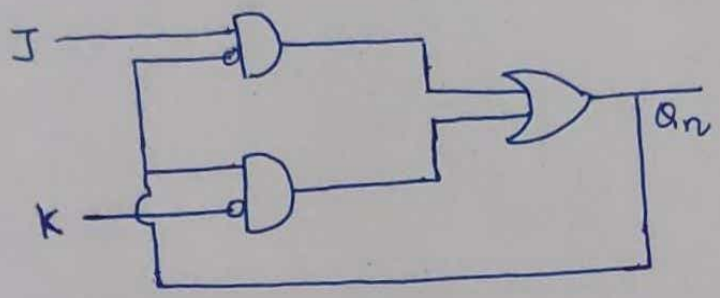
Function Table:-

| J | K | Qn+1 |
|---|---|---|
| 0 | 0 | Qn |
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 1 | $\overline{Qn}$ |




$$Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$$

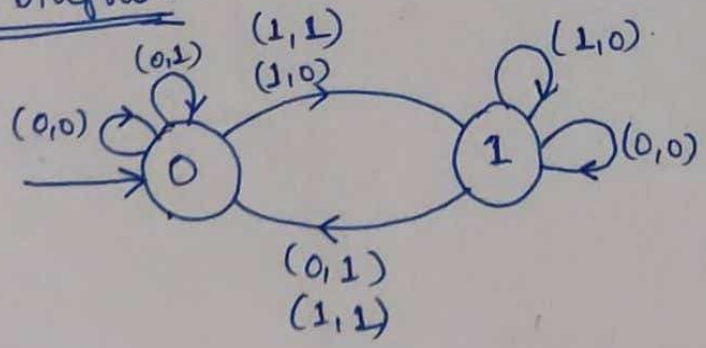
} characteristic equation




Excitation table:

| $Q_n$ | $Q_{n+1}$ | J | K |
|---|---|---|---|
| 0 | 0 | 0 | $\phi$ |
| 0 | 1 | 1 | $\phi$ |
| 1 | 0 | $\phi$ | 1 |
| 1 | 1 | $\phi$ | 0 |

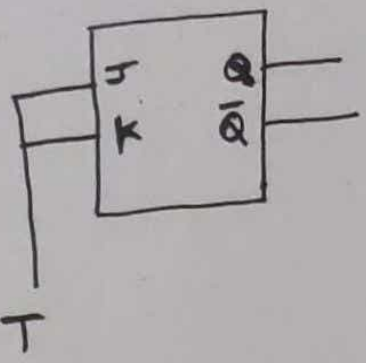
State Diagram:




} look at function Table & draw

# T-Flipflop:

↑
Toggle flipflop (This flipflop is either latching or complementing so it is called Toggling)

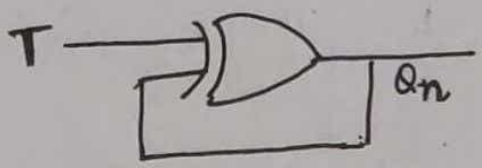



Function Table:-

| T | $Q_{n+1}$ |
|---|---|
| 0 | $Q_n$ |
| 1 | $\bar{Q}_n$ |

Characteristic Table:

| T | $Q_n$ | $Q_{n+1}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




$Q_{n+1} = \bar{T} Q_n + T \bar{Q}_n$

$= T \oplus Q_n$

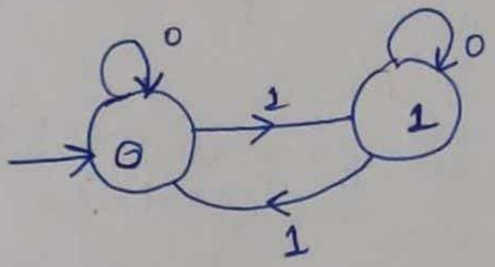
(⊕ : EXOR)

$\boxed{Q_{n+1} = T \oplus Q_n}$ } characteristic equation

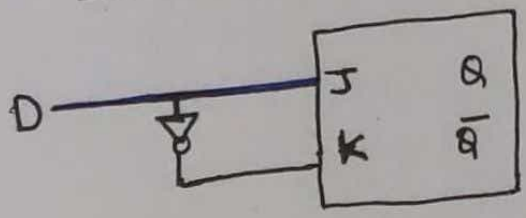
## Excitation Table:-

| $Q_n$ | $Q_{n+1}$ | T |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## State Diagram:-




## # D-Flipflop:




If D=0,
J=0 & K=1

If D=1,
J=1 & K=0

## Function Table:

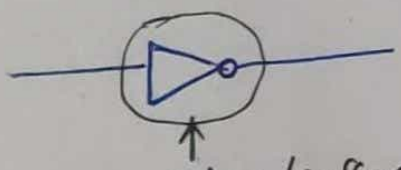
| D | $Q_{n+1}$ |
|---|---|
| 0 | 0 |
| 1 | 1 |

## Characteristic Table :-

| D | $Q_n$ | $Q_{n+1}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

$$Q_{n+1} = D\bar{Q_n} + DQ_n$$
$$= D(\bar{Q_n} + Q_n)$$
$$= D$$

$$\boxed{Q_{n+1} = D}$$ } characteristic equation.



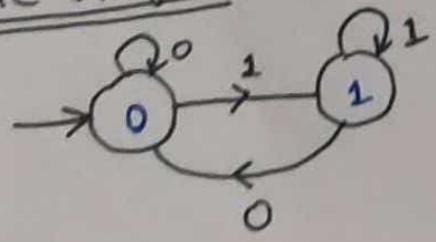
Delay / Buffer

[what are giving at the Input, the same Input will be displayed as output after some delay, so it is called delay flipflop].

## Excitation Table :-

| $Q_n$ | $Q_{n+1}$ | D |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## State Diagram :-



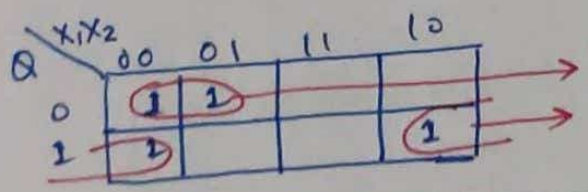
# If the characteristic equation of a flipflop is $Q_n = \bar{x}_1\bar{Q} + \bar{x}_2 Q$. Define the behaviour of this

| $x_1$ | $x_2$ | $Q_n$ | |
|---|---|---|---|
| 0 | 0 | 1 | → set |
| 0 | 1 | $\bar{Q}$ | → Toggle |
| 1 | 0 | Q | → latch |
| 1 | 1 | 0 | → Reset |

$Q_n = \bar{x}_1\bar{Q} + \bar{x}_2 Q$

## Characteristic Table:-

| $x_1$ | $x_2$ | Q | $Q_n$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

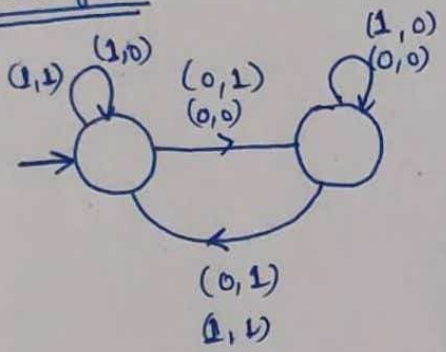



$Q_n = \bar{x}_1\bar{Q} + \bar{x}_2 Q$ } characteristic equation

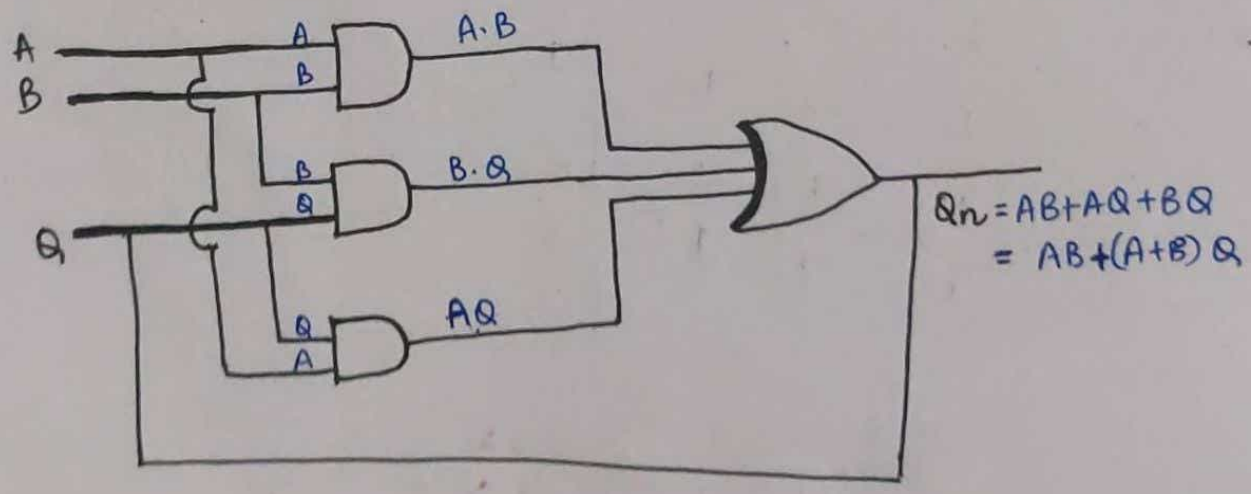
## Excitation Table:

| Q → Qn | $X_1$ | $X_2$ |
|---|---|---|
| 0 → 0 | 1 | $\phi$ |
| 0 → 1 | 0 | $\phi$ |
| 1 → 0 | $\phi$ | 1 |
| 1 → 1 | $\phi$ | 0 |

## State Diagram:




# Consider the following realization. construct the excitation Table



$$Q_n = AB + (A+B)Q$$

| A | B | $Q_n$ | |
|---|---|---|---|
| 0 | 0 | 0 | → Reset |
| 0 | 1 | Q | → latch |
| 1 | 0 | Q | → latch |
| 1 | 1 | 1 | → set |

Characteristic Table:

| A | B | Q | $Q_n$ |
|---|---|---|---|
| 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 |

Excitation Table:

| $Q \rightarrow Q_n$ | A | B |
|---|---|---|
| 0 → 0 | 0 | $\phi$ |
| 0 → 1 | 1 | 1 |
| 1 → 0 | 0 | 0 |
| 1 → 1 | 1 | $\phi$ |
| 0 → 0 | 1 | 0 |
| 1 → 1 | 0 | 1 |

0 → 0 {0,0 / 0,1 / 1,0}

1 → 1 {0,1 / 1,0 / 1,1}

# Introduction to Flip Flop Inter Conversion:

## ⇒ Conversion of given Flipflop to another Flipflop :-

(1) Get the characteristic table of target FlipFlop
(2) Replace the next state using Excitation of the given FlipFlop.
(3) Obtain the Expressions for the i/p of given Flipflop & realize them.

### eg1: Conversion of J-K flipflop to T-flipflop

① Target flipflop = T-Flipflop
⇒ Characteristic Table =

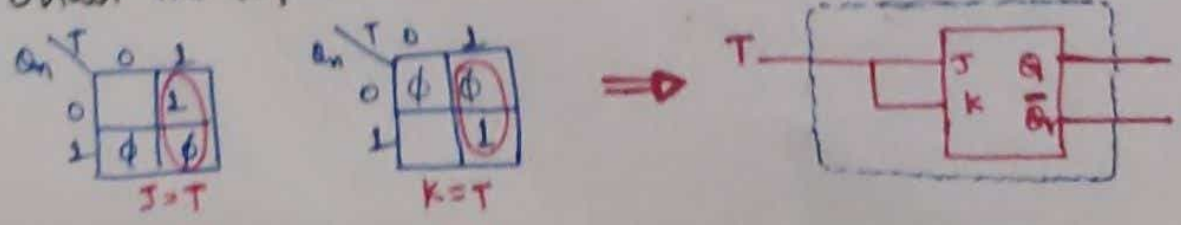
| T | $Q_n$ | $Q_{n+1}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

② Replace the next state with excitation

| T | $Q_n$ | $Q_{n+1}$ | J | K |
|---|---|---|---|---|
| 0 | 0 → | 0 | 0 | ϕ |
| 0 | 1 → | 1 | ϕ | 0 |
| 1 | 0 → | 1 | 1 | ϕ |
| 1 | 1 → | 0 | ϕ | 1 |

write down the values of J, K for which output changes from ($Q_n \rightarrow Q_{n+1}$) looking at characteristic Table of JK

③ obtain the expression for the i/p of given flipflop



## eg2: Conversion of T flipflop to JK flipflop

① Target flipflop = J K flipflop

⇒ characteristic Table =

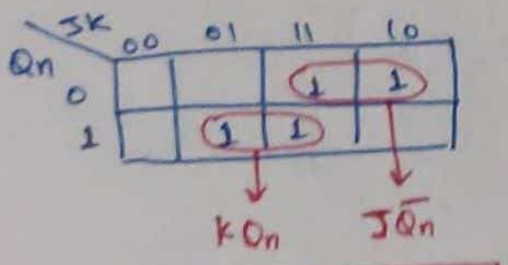
| J | K | $Q_n$ | $Q_{n+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

② Replace the next state with excitation

| J | K | $Q_n$ | $Q_{n+1}$ | T |
|---|---|---|---|---|
| 0 | 0 | 0 → | 0 | 0 |
| 0 | 0 | 1 → | 1 | 0 |
| 0 | 1 | 0 → | 0 | 0 |
| 0 | 1 | 1 → | 0 | 1 |
| 1 | 0 | 0 → | 1 | 1 |
| 1 | 0 | 1 → | 1 | 0 |
| 1 | 1 | 0 → | 1 | 1 |
| 1 | 1 | 1 → | 0 | 1 |

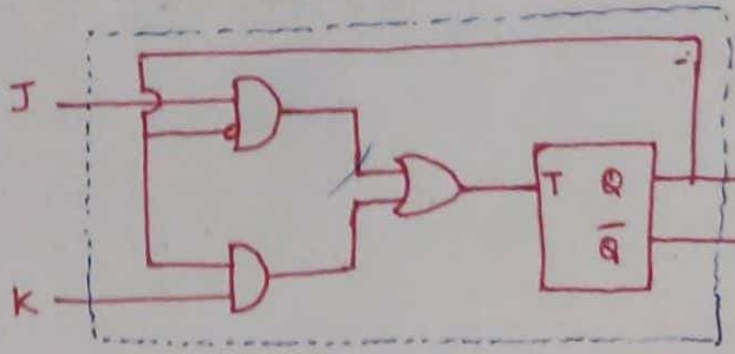
write down the values of T for which output changes from ($Q_n \rightarrow Q_{n+1}$) looking at characteristic Table of T

③ obtain the expression for the i/p of given flipflop




$T = J\bar{Q}_n + KQ_n$

⇒

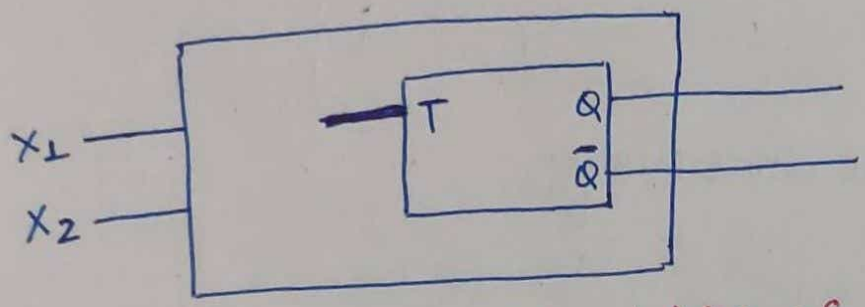


# Interconversion of flipflops

**eg 1**

A new Flipflop $X_1X_2$ has characteristic expression $Q_n = \bar{x_1}\bar{Q} + \bar{x_2}Q$. Realise it using T- flipflop.

Realise it using T-FlipFlop means the given flipflop is T & using T-flipflop. we want to recognise new flipflop.




∴ characteristic tables of $x_1x_2$ should be written first & then excitation tables of T should be written

- Function Table:

| $x_1$ | $x_2$ | $Q_n$ |
|---|---|---|
| 0 | 0 | 1 ← set |
| 0 | 1 | $\bar{Q}$ ← Toggle |
| 1 | 0 | Q ← Latch |
| 1 | 1 | 0 ← Reset |

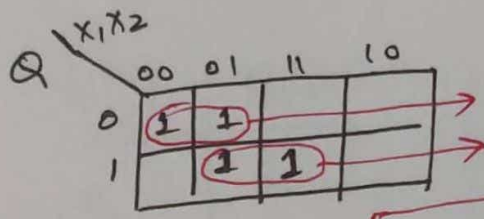
⇒

Characteristic table:

| $x_1$ | $x_2$ | Q | $Q_n$ | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | } set |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 1 | } Toggle |
| 0 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 0 | } Latch |
| 1 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 0 | } Reset |
| 1 | 1 | 1 | 0 | |

## Excitation Table:

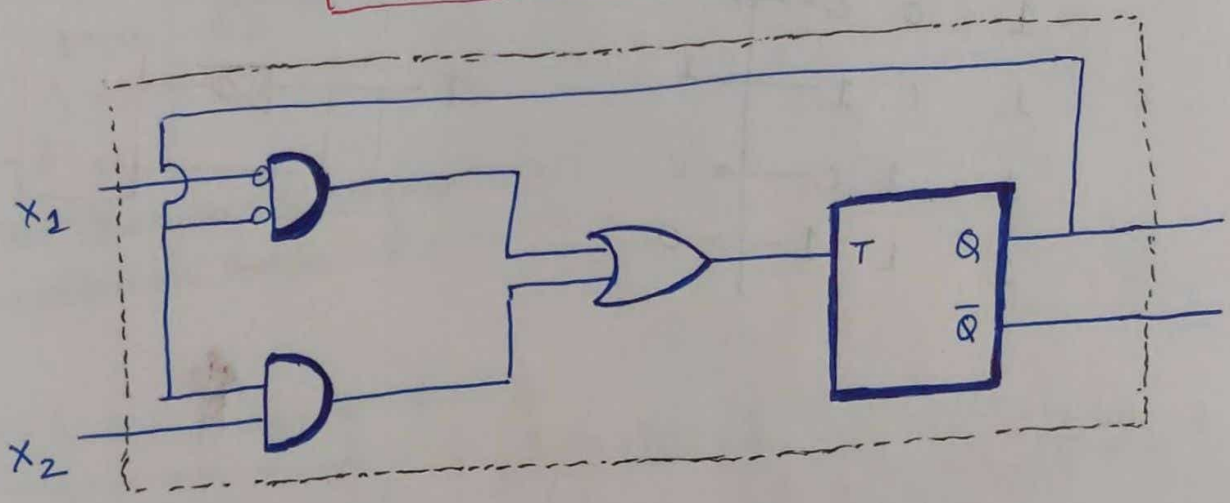
| $x_1$ | $x_2$ | $Q \to Q_n$ | T |
|---|---|---|---|
| 0 | 0 | 0 → 1 | 1 |
| 0 | 0 | 1 → 1 | 0 |
| 0 | 1 | 0 → 1 | 1 |
| 0 | 1 | 1 → 0 | 1 |
| 1 | 0 | 0 → 0 | 0 |
| 1 | 0 | 1 → 1 | 0 |
| 1 | 1 | 0 → 0 | 0 |
| 1 | 1 | 1 → 0 | 1 |




$\bar{x_1}\,\bar{Q}$

$x_2\,Q$

$T = \bar{x_1}\,\bar{Q} + x_2 Q$ } characteristics equation.




eg 2:

$x_1 x_2 \to T$

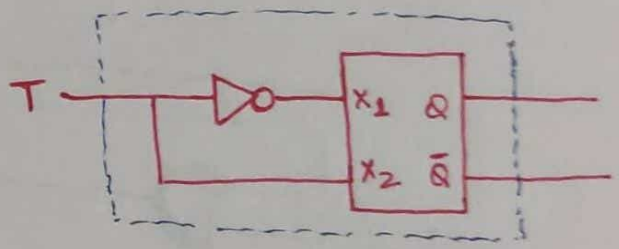
$Q_n = \bar{x_1}\,\bar{Q} + \bar{x_2}\,Q$ } Construct T flipflop using $x_1 x_2$

## Function Table :

| $X_1$ | $X_2$ | $Q_n$ ( $\bar{Q}_n$ ) |
|---|---|---|
| 0 | 0 | 1 ← set |
| 0 | 1 | $\bar{Q}$ ← Toggle |
| 1 | 0 | Q ← Latch |
| 1 | 1 | 0 ← Reset |

## Characteristic Table :

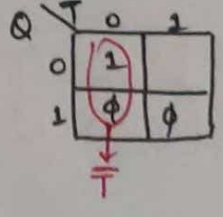
| $X_1$ | $X_2$ | Q | $Q_n$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |




## Excitation Table :

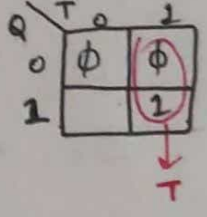
| T | Q | $Q_n$ | $X_1$ | $X_2$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | $\phi$ |
| 0 | 1 | 1 | $\phi$ | 0 |
| 1 | 0 | 1 | 0 | $\phi$ |
| 1 | 1 | 0 | $\phi$ | 1 |

for $X_1$



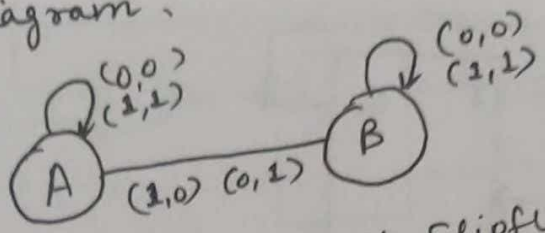

$X_1 = \bar{T}$

for $X_2$

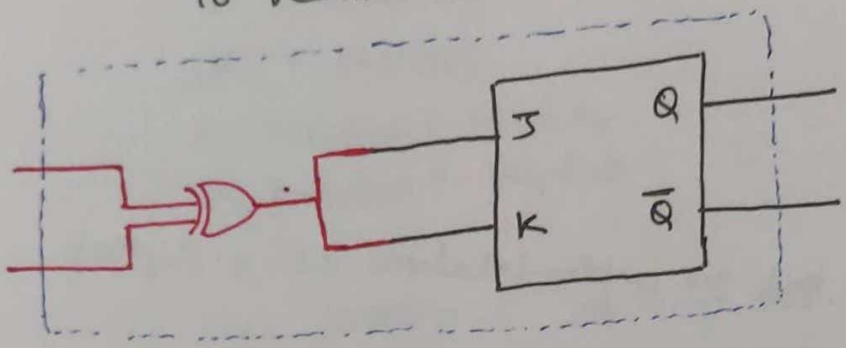



$X_2 = T$

Eg3 X, Y Flipflop is represented by the following state diagram.




To realise it using JK Flipflop. And J & K.

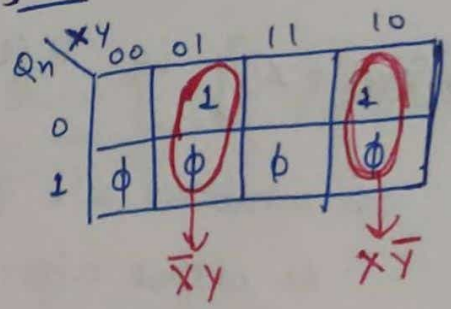



| X | Y | Qn+1 |
|---|---|---|
| 0 | 0 | Qn |
| 0 | 1 | $\overline{Qn}$ |
| 1 | 0 | $\overline{Qn}$ |
| 1 | 1 | Qn |

Now, the characteristic Equation of X, Y & excitation table of J, K

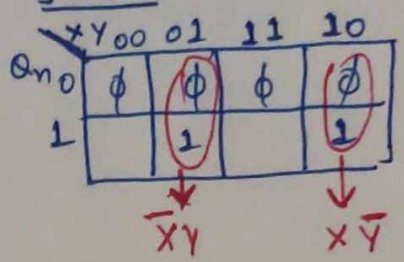
| X | Y | Qn | Qn+1 | J | K |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | $\phi$ |
| 0 | 0 | 1 | 1 | $\phi$ | 0 |
| 0 | 1 | 0 | 1 | 1 | $\phi$ |
| 0 | 1 | 1 | 0 | $\phi$ | 1 |
| 1 | 0 | 0 | 1 | 1 | $\phi$ |
| 1 | 0 | 1 | 0 | $\phi$ | 1 |
| 1 | 1 | 0 | 0 | 0 | $\phi$ |
| 1 | 1 | 1 | 1 | $\phi$ | 0 |

for J:




$J = \bar{X}Y + X\bar{Y}$
$= X \oplus Y$

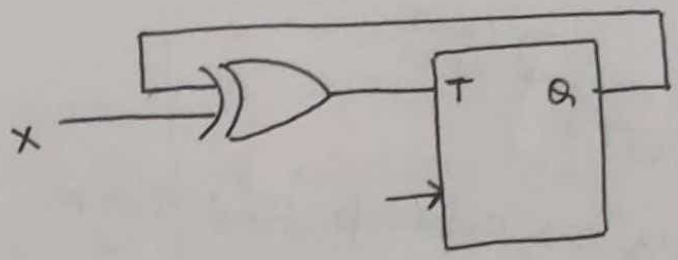
for K:




$K = \bar{X}Y + X\bar{Y}$
$= X \oplus Y$

eg4:

what is the behaviour of the following one-input Flipflop 'x'.




(a) D-flipflop
(b) T-flipflop
(c) Inverted D-flipflop
(d) Inverted T-flipflop

This problem is nothing but the implementation of X-flipflop using T-flipflop

Characteristic Table of X:

| X | $Q_n$ | $Q_{n+1}$ | $T = x \oplus Q_n$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

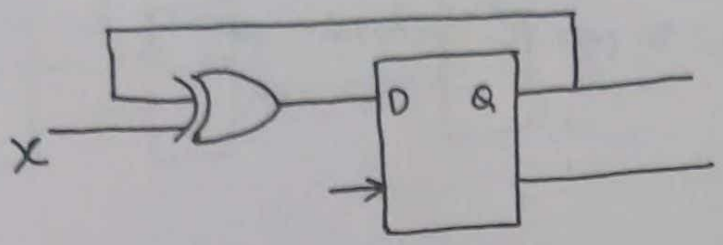
The output $Q_{n+1}$ is a combination of x, $Q_n$, T
So first Find $T = x \oplus Q_n$ ← then Find $Q_{n+1}$

∴ $Q_{n+1} \Rightarrow x$ } ie D flipflop

⇒ whatever we give as Input, is displayed as output after some delay so, it acts as the Delay flipflops.

eg5:

what is the behaviour of the following one Input flipflop 'x'




(a) D-flipflop
(b) T-flipflop
(c) Invested D-flipflop
(d) Inrested T-flipflop

This problem is nothing but the implementation of x-flipflop using D-flipflop

Characteristic Table of x:

| X | $Q_n$ | $Q_{n+1}$ | $D = X \oplus Q_n$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

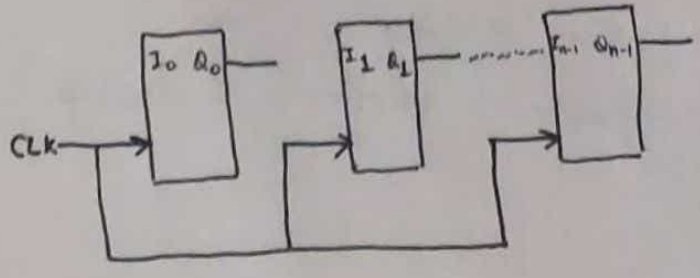
$\therefore$ when $x=0$, $Q_n = Q_{n+1}$
$x=1$, $Q_n = \overline{Q_{n+1}}$
} ie T flipflop

⇒ By comparing $Q_{n+1}$ with $Q_n$ & X we get the given/required flipflop is T flipflop.

## # Introduction to Counters:

- The Counters are used to provide accurate timing & control signals
- They are of 2 types -
  - Synchronous
  - Asynchronous
- In synchronous counters all the flipflops respond the same clock instances.
- In Asynchronous counters, the output of one flipflop drives the clock of another flipflop.
- Synchronous counters are faster than Asynchronous Counter.
- Due to simplicity of design, Asynchronous counters are used in Ic Fabrication.
- The simplified version of Synchronous Counter is called shift counters.
- The basic element in shift counter is D-flipflop.
- The Ring counter & Thonson counter are further simplified version of shift counter

### Synchronous Counter:



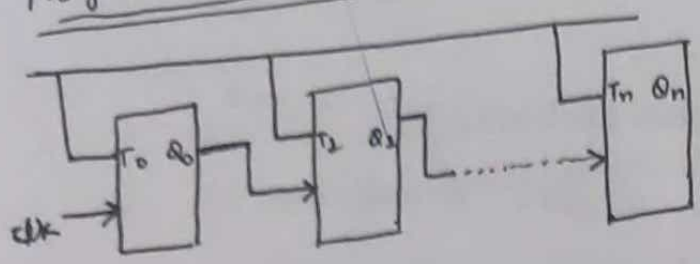

Propogation delay in synchronous counter is -

$T_{pdsyn} = T_{FF} + T_{Combinational}$

$T_{clk} \geq T_{pdsyn}$

$T_{clk}$ means the amount of time we should wait before sending the next Input signal

## Asynchronous Counter :




Propogation delay in Asynchronous counter is -

$T_{pdasyn} = N * T_{FF} + T_{Combinational}$

$T_{clk} \geq T_{pdasyn}$

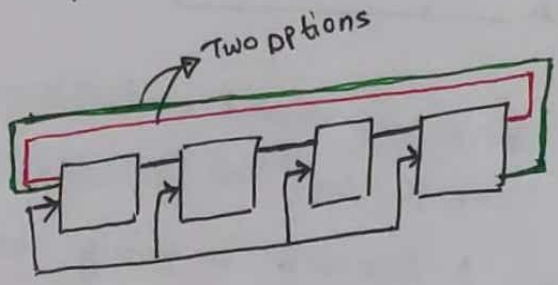
## # shift counters

### ⇒ synchronous counter :

$$I_i = f(Q_0, Q_1 \ldots\ldots Q_{N-1})$$

$$0 \leq i \leq N-1$$

Its Design is more complex, so simplifications are done as follows :-




① $I_i = Q_{i-1}$ $\boxed{1 \leq i \leq N-1}$

$I_1 = Q_0$

$I_2 = Q_1$

$I_3 = Q_2$

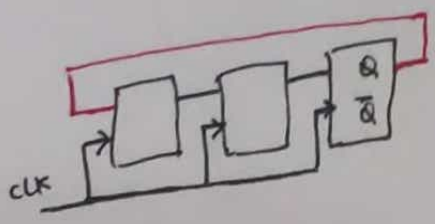
⋮

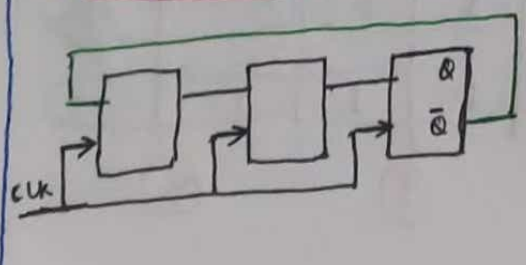
Here, Data is shifted from one flipflop to another flipflop so this is called shift counter (D-flipflops are used)

② $I_0 = f(Q_{N-1})$

$I_0 = (Q_{N-1})$ (Ring counter)

$I_0 = (\bar{Q}_{N-1})$ (Johnson Counter) → Twisted Ring counter



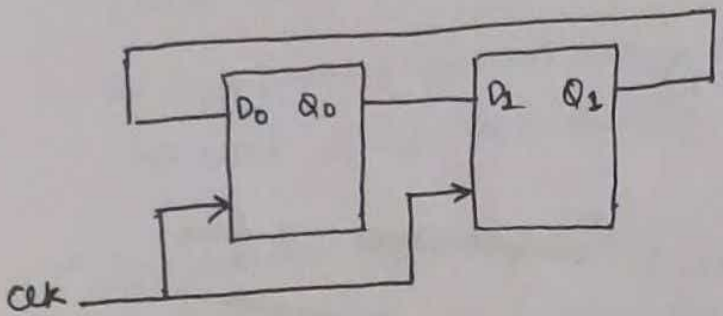



# Mod-2 Ring Counters:

Generally there are 2 types of Questions asked -

(1) They will give you the counter & ask you what it is counting.

(2) They will tell you what you should count & ask you to design the counter.




First we should understand what are the states in which the flipflops can be, if we observe there are 2 flipflops & 1 flipflop is going to give $Q_0$ state & another flipflop is going to give $Q_1$ state.

(Here, Assume $Q_0$ = LSB
$Q_1$ = MSB)

| | | Present state | | Next state | |
|---|---|---|---|---|---|
| $D_1$ | $D_0$ | $Q_1$ | $Q_0$ | $Q_{1N}$ | $Q_{0N}$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| (2) | | (1) | | (3) | |

Next state depends on 2 things

(1) Input

(2) In case any combinational circuit is present what is the i/p to the combinational circuit.

Here, $Q_{0N}$ depends on $D_0$ & $D_0$ Depends on $Q_1$
(By looking at Diagram)

$\therefore$ $\boxed{Q_{0N} = Q_1}$

i.e $(Q_{0N} = D_0 = Q_1)$

$\Rightarrow (Q_{0N} = Q_1)$

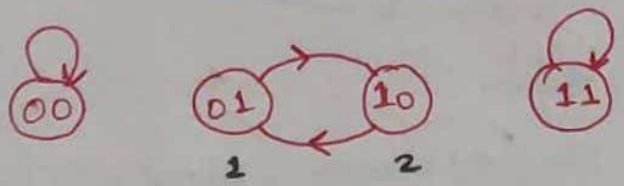
Now, $Q_{1N}$ is nothing but Input given at $D_1$

$\boxed{Q_{1N} = D_1}$

Now, observe the marked circles (present state, Next state) & find how they behave

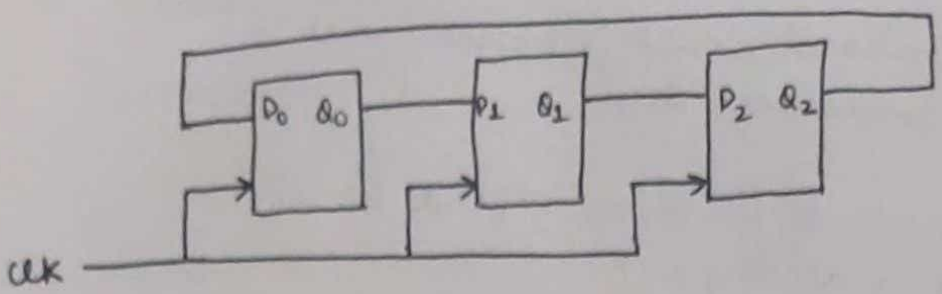
(a) when Input $\begin{cases} Q_0Q_1 = 00 \\ Q_0Q_1 = 11 \end{cases}$ then they remain in same state

(b) when Input $\begin{cases} Q_0Q_1 = 01 \\ Q_0Q_1 = 10 \end{cases}$ then they toggle or complimented.




$\Rightarrow$ mod 2 counter (only 2 states are in counting)

# Mod-3 Ring Counters :



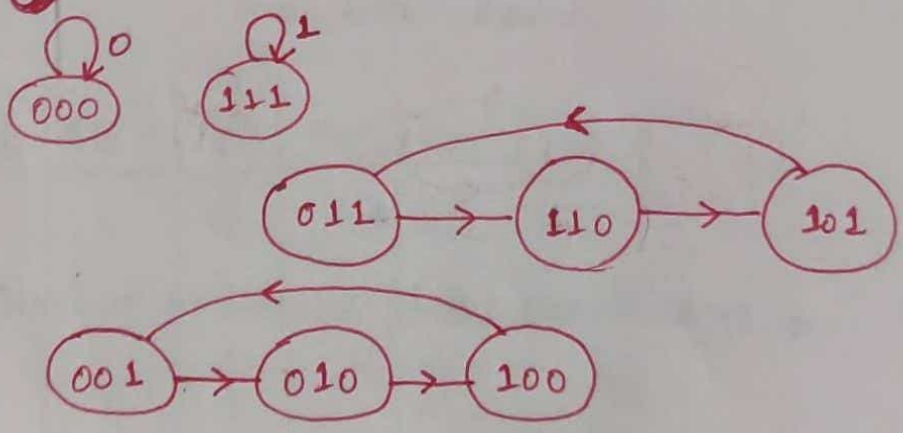

$Q_0$ ← LSB
$Q_1$
$Q_2$ ← MSB

$D_0 = Q_2$
$D_1 = Q_0$
$D_2 = Q_1$

$\Rightarrow$

$Q_{0N} = D_0$
$Q_{1N} = D_1$
$Q_{2N} = D_2$

| Present state | | | Next state | | | | | |
|---|---|---|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Q_{2N}$ | $Q_{1N}$ | $Q_{0N}$ | $D_0$ | $D_1$ | $D_2$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | ① | | ③ | | | ② | |

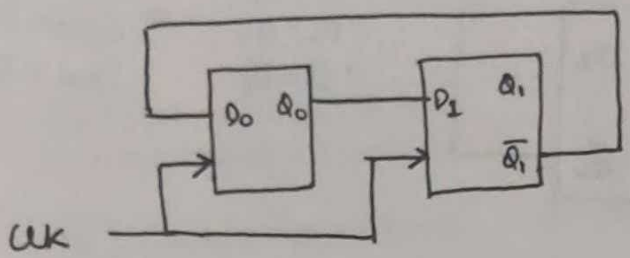



⇒ If a Ring Counter has n-"D flipflops" then it can perform 'mod n' calculations.

⇒ In Above Ring Counter it has 3 flipflops so it can perform mod3 operation.

# Mod 4 Johnson Counter:

(Twisted Ring Counter or Johnson counter)



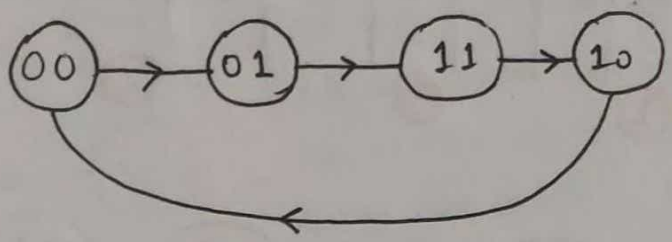

$D_0 = \bar{Q_1}$
$D_1 = Q_0$
$\Rightarrow$ $Q_{1N} = D_1$
$Q_{0N} = D_0$

There are 2 states $Q_0$ & $Q_1$

Present state — Next state

| $Q_1$ | $Q_0$ | $Q_{1N}$ | $Q_{0N}$ | $D_0$ | $D_1$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |

① ③ ②

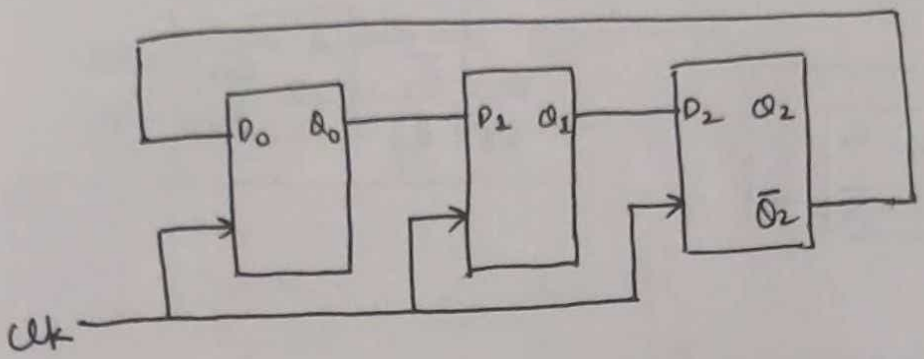



mod 4 Counter

$\Rightarrow$ If a twisted Ring counter (Johnson counter) has n-"D flipflops" then it can perform as 'mod 2n' counter

# Mod 6 Johnson Counters:

(Twisted Ring counter or Johnson Counter)

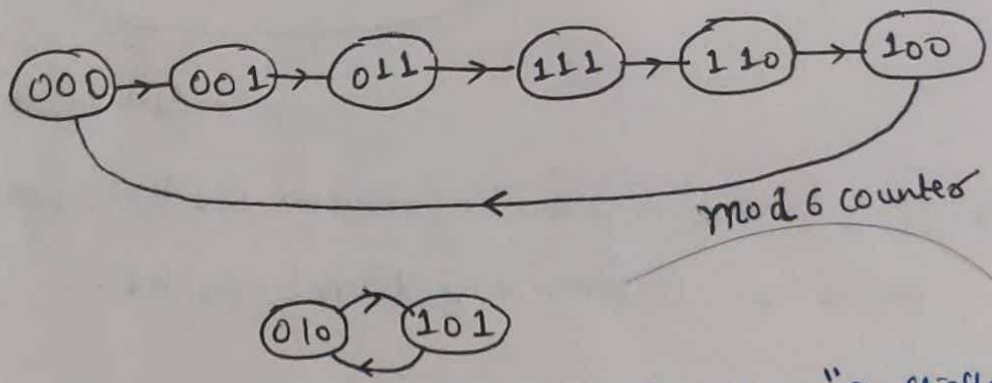



$D_2 = Q_1$
$D_1 = Q_0$
$D_0 = \bar{Q}_2$
$\Rightarrow$
$Q_{2N} = D_2$
$Q_{1N} = D_1$
$Q_{0N} = D_0$

Present state | Next state

| $Q_2$ | $Q_1$ | $Q_0$ | $Q_{2N}$ | $Q_{1N}$ | $Q_{0N}$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |

(1) (3) (2)




$\Rightarrow$ If a Twisted Ring counter (Johnson Counter) has $n$-"D-flipflops" then it can perform as 'mod $2n$' calculations/Counter

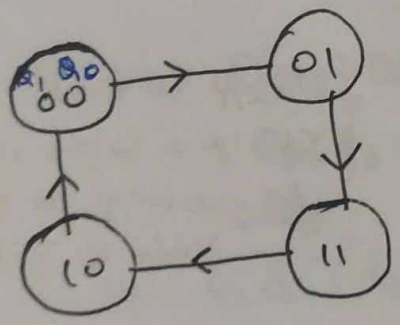
# Mod-4 GRAY counter using T-flipflop

Now, let us say how to design a counter with specifications

## Design a counter

1. Get the state table from state Diagram
2. Identify flipflops to be used & replace the concerned next state using excitation table of associated flipflop
3. Get the expressions for Inputs & realise them.

Design the synchronous counter for the following using T-flipflops




⇒ Here two flipflops are enough, at any instant of time the circuit will be is 00 or 01 or 10 or 11 state so each state has 2 bits So, 2 flipflops are enough.

Now, we have to connect these 2 flipflops in such a way that when a clock signal is applied it should make any of the above transitions (00→01, 01→11, 11→10, 10→00)

Now, we need to find the Input combinations that lead to the above transition (This is also called Excitation table)

Now, first draw state table.

For 2 flipflops Two inputs

| Present State | | Next State | | Excitation (Inputs) | |
|---|---|---|---|---|---|
| $Q_1$ | $Q_0$ | $Q_{1N}$ | $Q_{0N}$ | $T_1$ | $T_0$ |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |

written by seeing at the Diagram

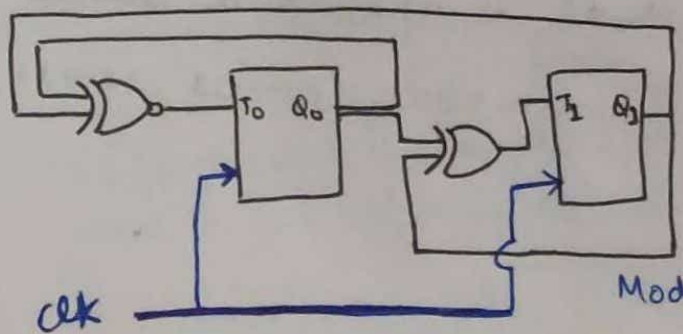
Now, to find $T_1$ check $Q_1$ & $Q_{1N}$

| $Q_1$ | $Q_{1N}$ | $T_1$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

{ If o/p is compliment, then T = 1,
If o/p is same, then T = 0,
(By looking at the function Table of T-flipflop)

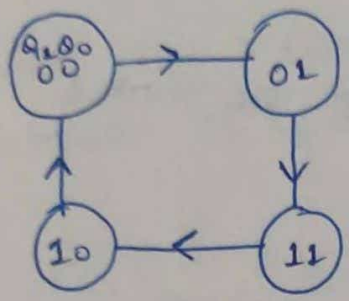
$$T_1 = \bar{Q}_1 Q_0 + Q_1 \bar{Q}_0$$
$$= \underline{Q_1 \oplus Q_0}$$

$$T_0 = \bar{Q}_1 \bar{Q}_0 + Q_1 Q_0$$
$$= \underline{Q_1 \odot Q_0}$$




Mod 4 Gray-counter

# Mod-4 Gray counter using D-flipflops:

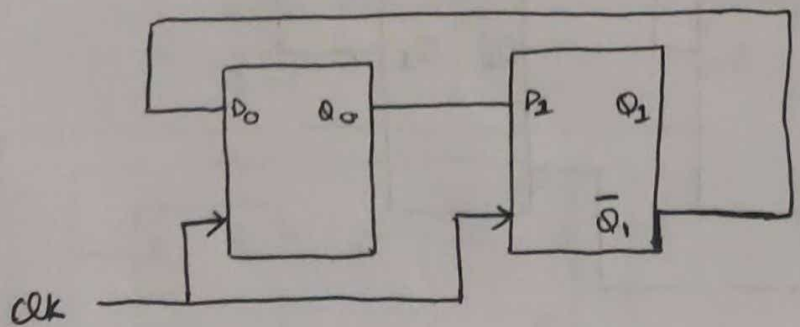



⟹ 2 flipflops are needed

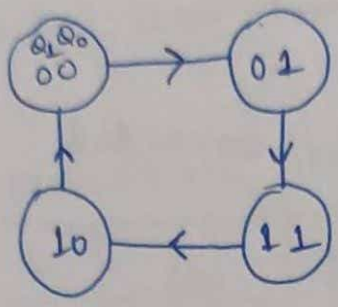
| present state | | Next state | | Excitation (Inputs) | |
|---|---|---|---|---|---|
| $Q_1$ | $Q_0$ | $Q_{1N}$ | $Q_{0N}$ | $D_1$ | $D_0$ |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |

$$D_1 = \bar{Q}_1 Q_0 + Q_0 Q_1$$
$$= Q_0(\bar{Q}_1 + Q_1)$$
$$= Q_0$$

$$D_0 = \bar{Q}_0 \bar{Q}_1 + \bar{Q}_1 Q_0$$
$$= \bar{Q}_1 (\bar{Q}_0 + Q_0)$$
$$= \bar{Q}_1$$



# Mod-4 Gray Counter using 1 D & 1 T Flipflop

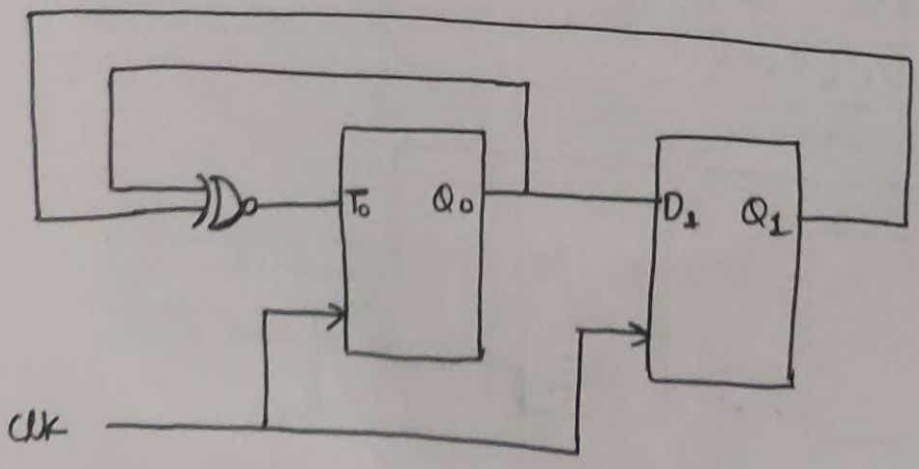



⇒ 2 flipflops are needed

| present state | | Next state | | Excitation (Inputs) | |
|---|---|---|---|---|---|
| $Q_1$ | $Q_0$ | $Q_{1N}$ | $Q_{0N}$ | $D_1$ | $T_0$ |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 |

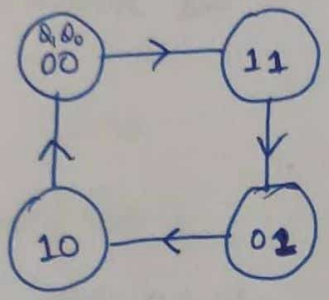
$$D_1 = \bar{Q}_1 Q_0 + Q_1 Q_0$$
$$= Q_0(\bar{Q}_1 + Q_1)$$
$$= Q_0$$

$$T_0 = Q_0 Q_1 + \bar{Q}_0 \bar{Q}_1$$
$$= Q_1 \odot Q_0$$



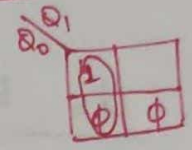
# Counter using Two different flipflops:

Consider the following state diagram which is to be designed using T-flipflops for MSB & XY for LSB.
The behaviour of XY is given below:-



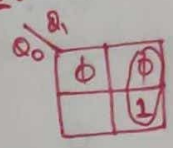

| X | Y | $Q_n$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | Q |
| 1 | 0 | $\bar{Q}$ |
| 1 | 1 | 1 |

For X:




$X = \bar{Q_1}$

For Y:




$Y = Q_1$

| $Q_1$ | $Q_0$ | $Q_{1N}$ | $Q_{0N}$ | $T_1$ | X | Y |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 | ϕ |
| 1 | 1 | 0 | 1 | 1 | ϕ | 1 |
| 0 | 1 | 1 | 0 | 1 | ϕ | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | ϕ |

$\Rightarrow X = \bar{Q_1}$, $Y = Q_1$ [using k-map]

Given Function Table → characteristic Table → Excitation Table

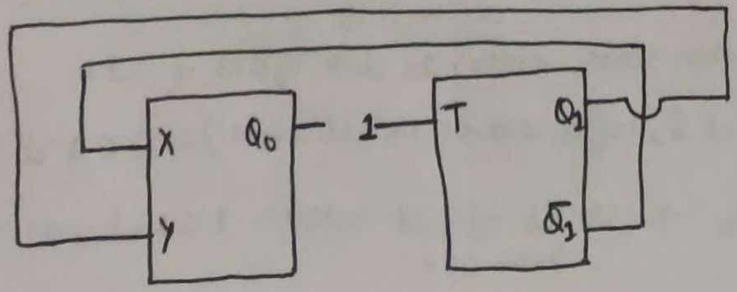
| X | Y | $Q_n$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | Q |
| 1 | 0 | $\bar{Q}$ |
| 1 | 1 | 1 |

→

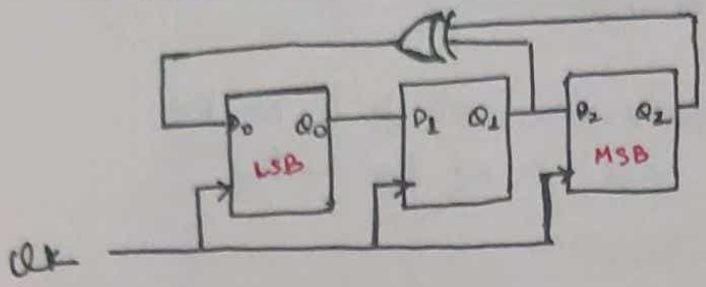
| X | Y | Q | $Q_n$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

→

| Q | $Q_n$ | X | y |
|---|---|---|---|
| 0 | 0 | 0 | ϕ |
| 0 | 1 | 1 | ϕ |
| 1 | 0 | ϕ | 0 |
| 1 | 1 | ϕ | 1 |



# Model on Analysis Counting states & sequence Generation:




① If the initial state is S0 & S0 then after 5 clocks, what is going to happen?

② After 100 clocks what will happen?

This all depends on what the Initial state is.

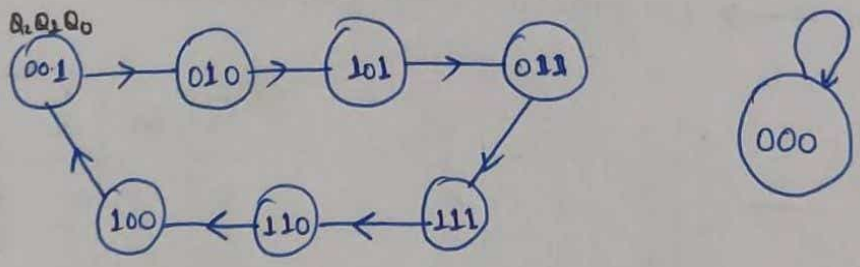
State Diagram Analysis:-

| $Q_2$ | $Q_1$ | $Q_0$ | $Q_{2N}$ | $Q_{1N}$ | $Q_{0N}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 |

$Q_{0N} = D_0 = Q_1 \oplus Q_2$

$Q_{1N} = D_1 = Q_0$

$Q_{2N} = D_2 = Q_1$




① How many states are these in mod Counter = 7

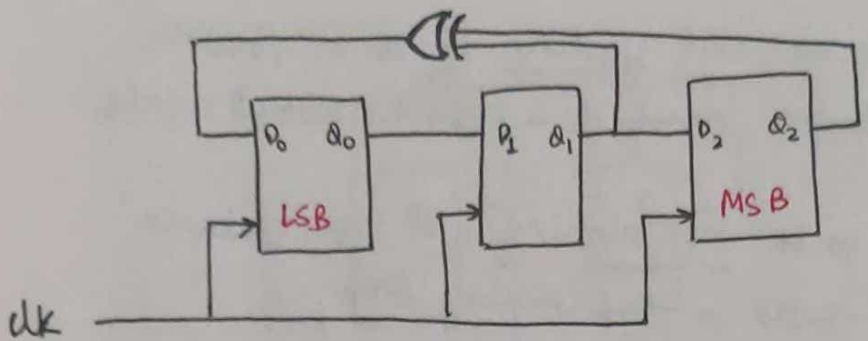
② Given Initial state = 011, then what happens after 4 clock cycles = state after 4 clock cycles = 001

③ Initial state = 001, then state after 10 clock cycles = 011

④ If tapping the output at $Q_2$ ⇒ ($Q_2$ values in all states) = 0010111

⑤ o/p is Tapped at $Q_0$ ⇒ (1st bit of all states from LSB) = 1011100

# Deriving the clock frequency:




$T_{FF} = 15ns$

$T_{comb} = 5ns$

which of the following clock frequency ensures proper counting ?

(a) 40 MHz
(b) 60 MHz
(c) 90 MHz
(d) 300 MHz

$T_{clk} \geq T_{FF} + T_{comb}$

$T_{clk} \geq 15ns + 5ns$

$T_{clk} \geq 20ns$

As, Time & frequency are inversely proportional.

$$f_{clk} \leq \frac{1}{20ns}$$

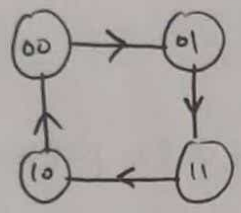
$$\leq \frac{1}{20 \times 10^{-9}}$$

$$f_{clk} \leq 50MHz$$

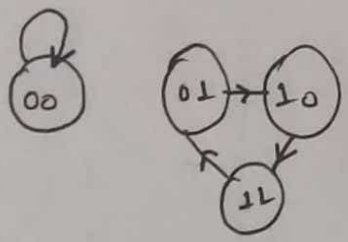
# Self starting & free running counter:

- A counter is said to be self starting if it is possible to enter counting loops irrespective of the initial state
- A counter is said to be free running if it contains all the possible states in the counting loop.

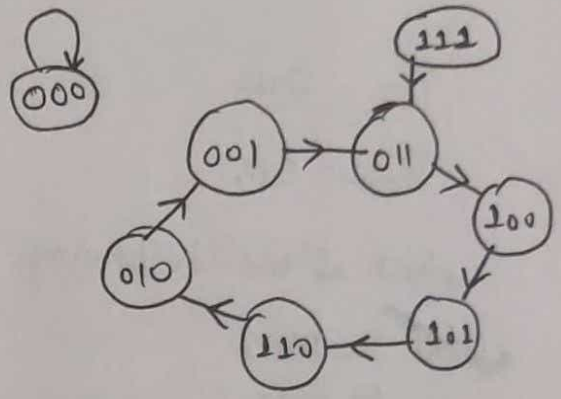
(a)



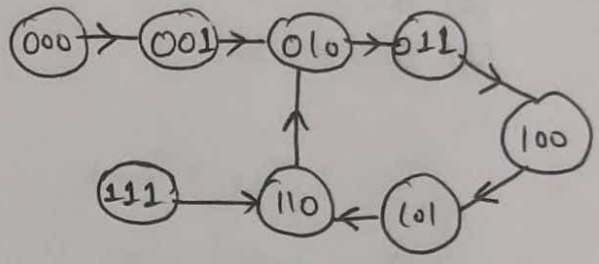

(b)



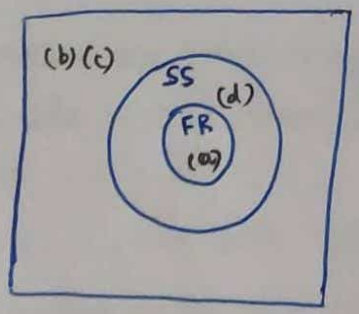

(c)



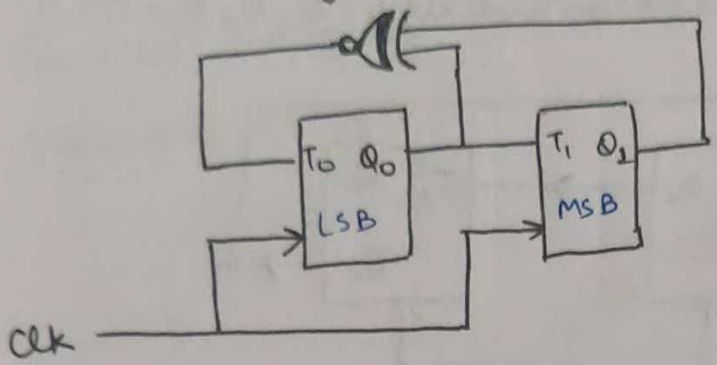

(d)




Every Free running Counter is self starting Counter but Every self starting counter is not necessarily be the free running Counter



Identify if the following Counter is self starting or Free running.




$Q_{1N} = Q_1$ , $T_1 = 0$

$= \overline{Q_1}$ , $T_1 = 1$

$Q_{0N} = Q_0$ , $T_0 = 0$

$= \overline{Q_0}$ , $T_0 = 1$

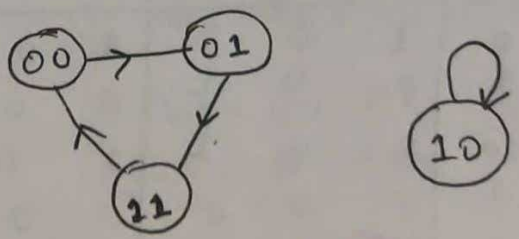
$\Rightarrow T_1 = Q_0$

$T_0 = Q_0 \odot Q_1$

| $Q_1$ | $Q_0$ | $Q_{1N}$ | $Q_{0N}$ | $T_1$ | $T_0$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| (1) | | (3) | | (2) | |

$T_1 = Q_0$

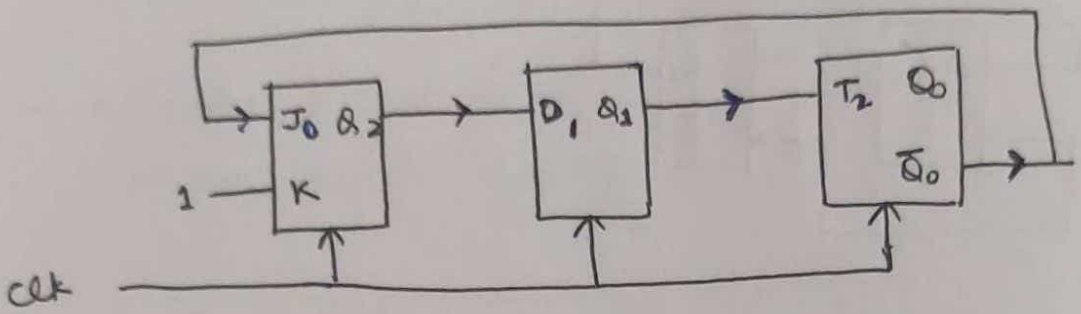
$T_0 = Q_0 \odot Q_1$




$\Rightarrow$ Not self starting Nor Free Running.

# Counter using 3 Different flipflops:

Consider the following counters, Initially $Q_2Q_1Q_0 = 000$




(1) what will be the states After 4 clocks?

(a) 000
(b) 010
~~(c)~~ 011
(d) 101

$D_1 = Q_2$
$T_2 = Q_1$
$J_0 = \overline{Q_0}$

$Q_{0N} = Q_0, T=0, Q_1=0 \;/\; \bar{Q}_0, T=1, Q_1=1$
$Q_{1N} = D_1 = Q_2$
$Q_{2N} = ?? = JK$

(2) Modulus of the Counter?

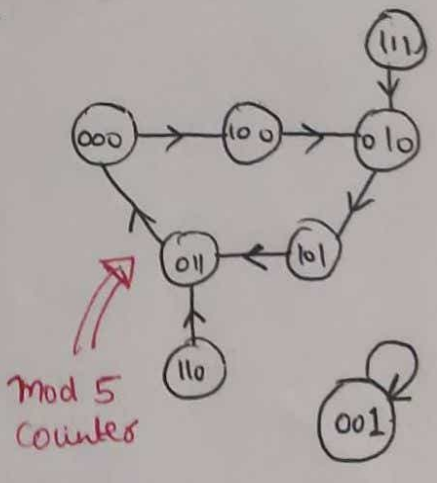
(a) 4
~~(b)~~ 5
(c) 6
(d) 7

| $\bar{Q}_0$ → J | K | $Q_{2N}$ |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | $\bar{Q}_N$ |

⇒

| $Q_0$ | K | $Q_{2N}$ |
|---|---|---|
| 1 | 1 | 0 |
| 0 | 1 | $\bar{Q}_N$ |

| $Q_2$ | $Q_1$ | $Q_0$ | $Q_{2N}$ | $Q_{1N}$ | $Q_{0N}$ | $J_0$ | $D_1$ | $T_2$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |



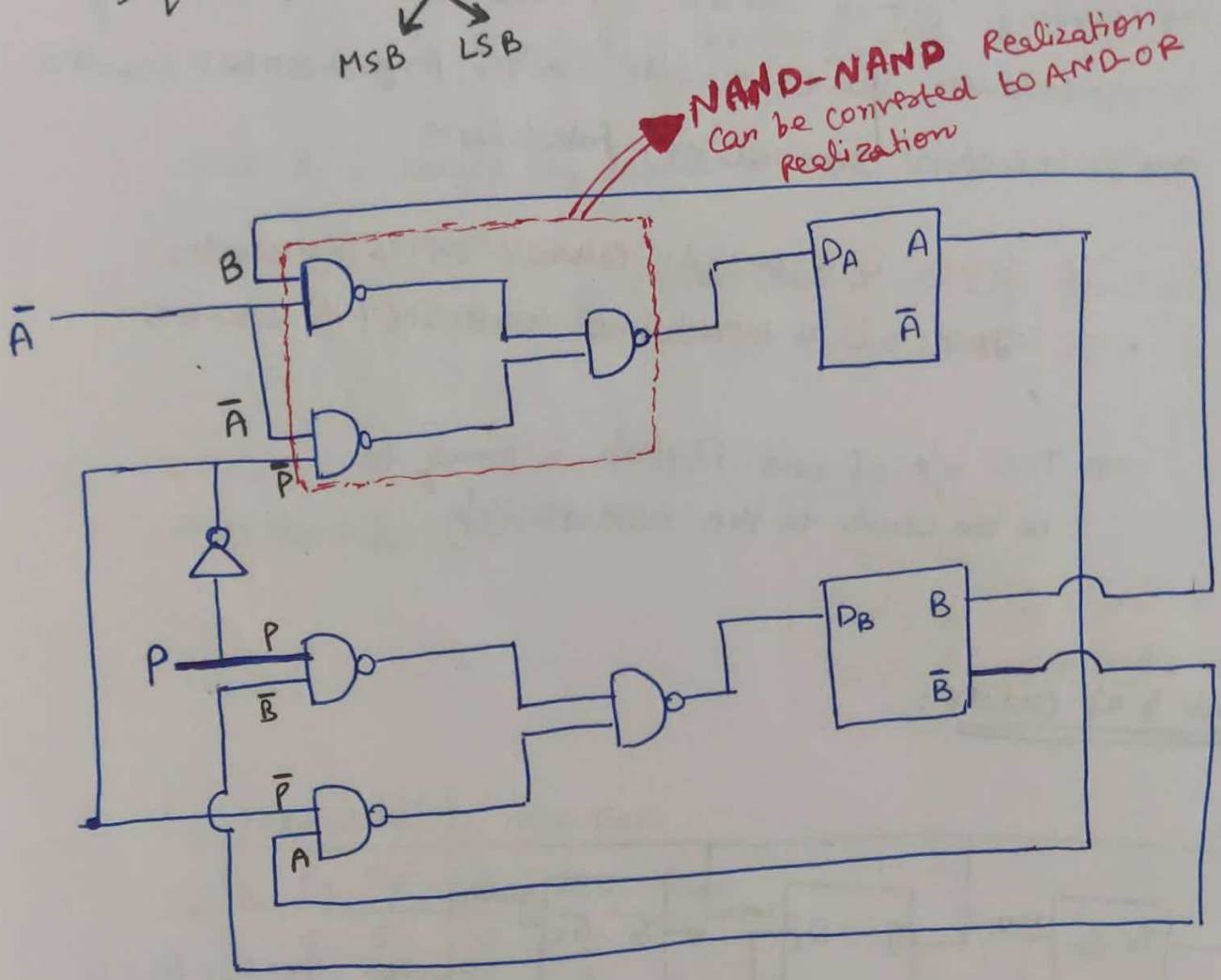

Mod 5 Counter

⇒ Not self starting
Nor free running

# Example on combinational circuits & flipflops:

Consider the following counter, If P=0, the Counting sequence of AB is

MSB LSB



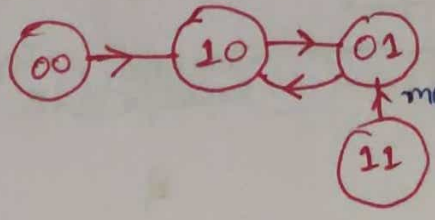

| A | B | An | Bn |
|---|---|---|---|
| 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 |

$A_n = D_A$
$B_n = D_B$
} If we observe these, they can be realised to AND-OR Realisation.

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

Hence, $A_n = \bar{A}$
$B_n = A$

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




⇒ self starting but not free running

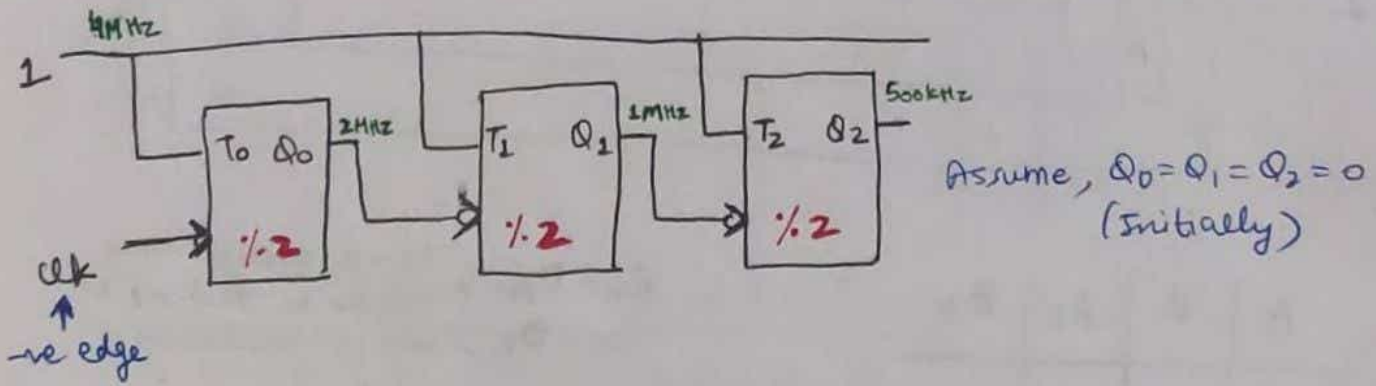
# Asynchronous Counters:

⇒ The asynchronous counters are also called Ripple Counter. The basic flipflop is T & basic counting is binary up counting. The up counters are used for implementing incrementation. Due to simpler design, Asynchronous counters are preferred in IC counter fabrications.

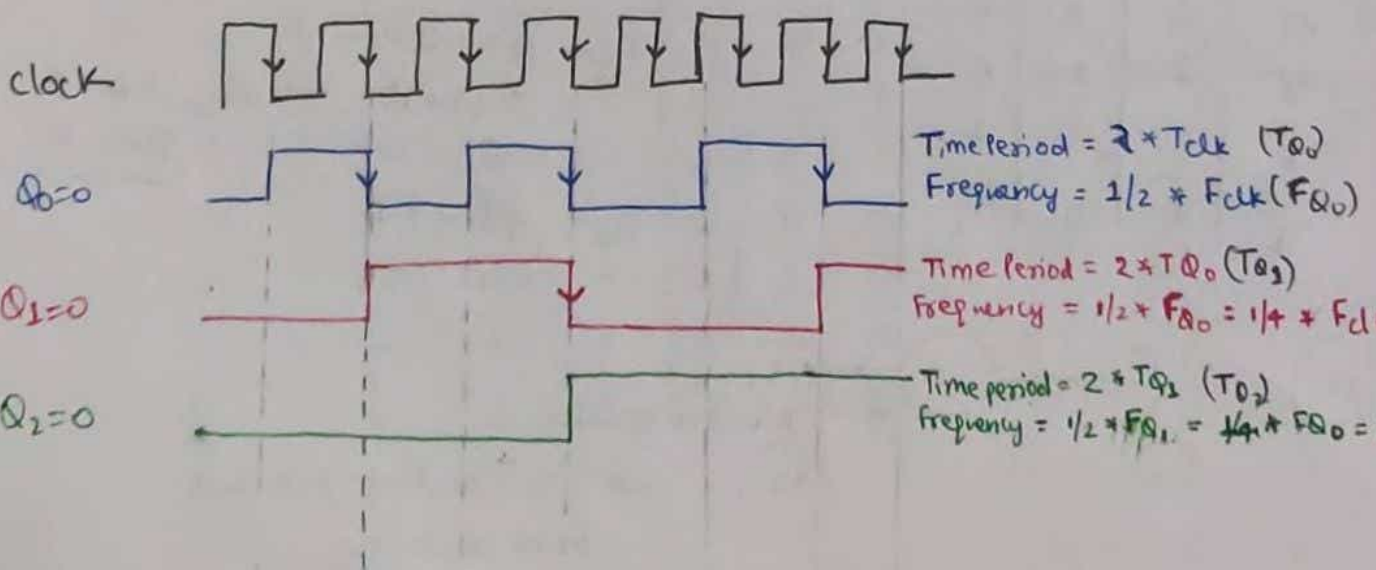
- IC 7490 is a Decade counter ie (%10) counter.
- IC 7492 is a hexadecimal counter ie (%16) counter.

⇒ The o/p of one flipflop is going to be the clock to the next flipflop.

# Mod-8 up counter:




Assume, $Q_0 = Q_1 = Q_2 = 0$ (Initially)




Time period = $2 * T_{clk}$ ($T_{Q_0}$)
Frequency = $1/2 * F_{clk}$ ($F_{Q_0}$)

Time period = $2 * T_{Q_0}$ ($T_{Q_1}$)
Frequency = $1/2 * F_{Q_0} = 1/4 * F_{clk}$ ($F_{Q_1}$)

Time period = $2 * T_{Q_1}$ ($T_{Q_2}$)
Frequency = $1/2 * F_{Q_1} = 1/4 * F_{Q_0} = 1/8 * F_{clk}$ ($F_{Q_2}$)

⇒ (3) ⅟2 Counters from ⅟8 counters

Now, Consider '$Q_0$'. It is having input 1 & it is -ve edge triggered & it is T flipflop so, T will become 1 & it will Toggle

$\therefore Q_{0N} = \overline{Q_0}$ for every clock.

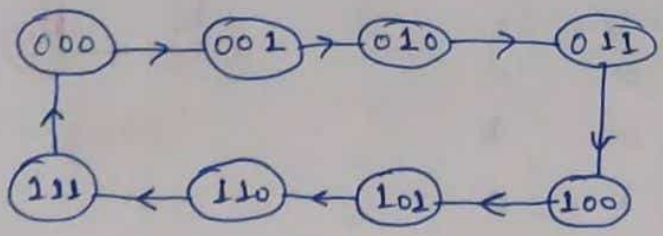
Now, $Q_1$ is changed / $Q_1$ depends on how $Q_{0N}$ is providing the clock signal

$\therefore Q_{1N} = \overline{Q_1}$ (when there is -ve edge from $Q_0 (1 \to 0)$)
⇒ $Q_0$ will toggle

Now, $Q_{2N} = \overline{Q_2}$ $(Q_1 : 1 \to 0)$

$Q_{0N} = \overline{Q_0}$
$Q_{1N} = \overline{Q_1}$ $(Q_0 : 1 \to 0)$
$Q_{2N} = \overline{Q_2}$ $(Q_1 : 1 \to 0)$

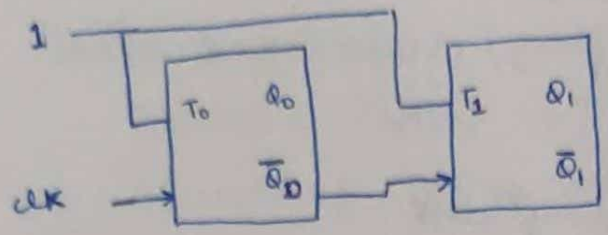
| Present state | | | Next state | | |
|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Q_{2N}$ | $Q_{1N}$ | $Q_{0N}$ |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |




⇒ Both self starting as well as free Running

⇒ Mod 8 up counter

# Mod-4 up counter:




what type of counting it is performing & what is the type of the counter?

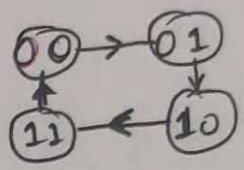
$Q_{0N} = \bar{Q}_0$ (for every clock)

$Q_{1N} = \bar{Q}_1$ $(\bar{Q}_0 : 0 \to 1)$

or

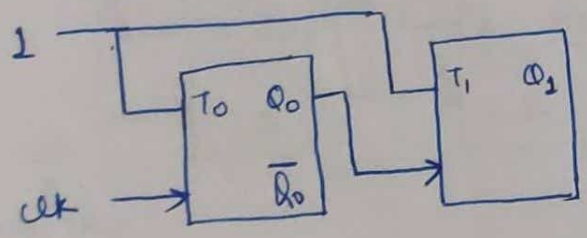
$(Q_0 : 1 \to 0)$

| Present state | | Next state | |
|---|---|---|---|
| $Q_1$ | $Q_0$ | $Q_{1N}$ | $Q_{0N}$ |
| 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |




⇒ Mod-4 up counter

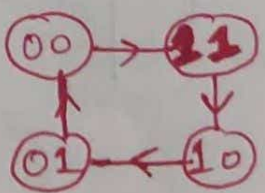
# Mod-4 Down counter:




$Q_{0N} = \bar{Q}_0$ (for every clock)

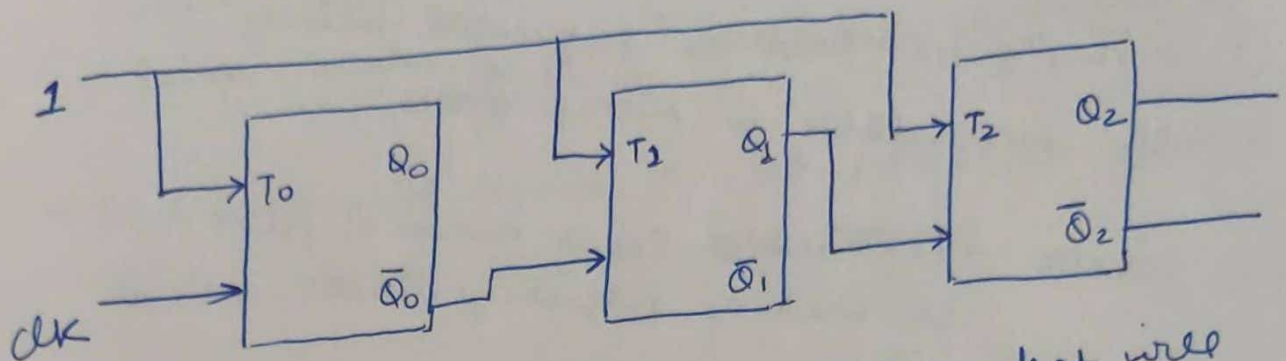
$Q_{1N} = \bar{Q}_1$ $(Q_0 : 0 \to 1)$

| Present state | | Next state | |
|---|---|---|---|
| $Q_1$ | $Q_0$ | $Q_{1N}$ | $Q_{0N}$ |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |




⇒ Mod-4 downcounter

# Mod-8 Random counter:




If the Initial state $Q_2Q_1Q_0 = 101$, what will be the state After 4 clock clycles.

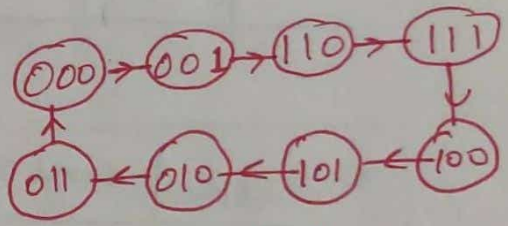
(a) 001
(b) 101
(c) 111
(d) 010

$Q_{0N} = \bar{Q}_0$

$Q_{1N} = \bar{Q}_1$ ($\bar{Q}_0$ ➡ 0→1)
or
($Q_0$ ➡ 1→0)

$Q_{2N} = \bar{Q}_2$ ($Q_1$ ⇒ 0→1)

In diagram It is positive edge so(0→1)




⇒ self starting as well as free running

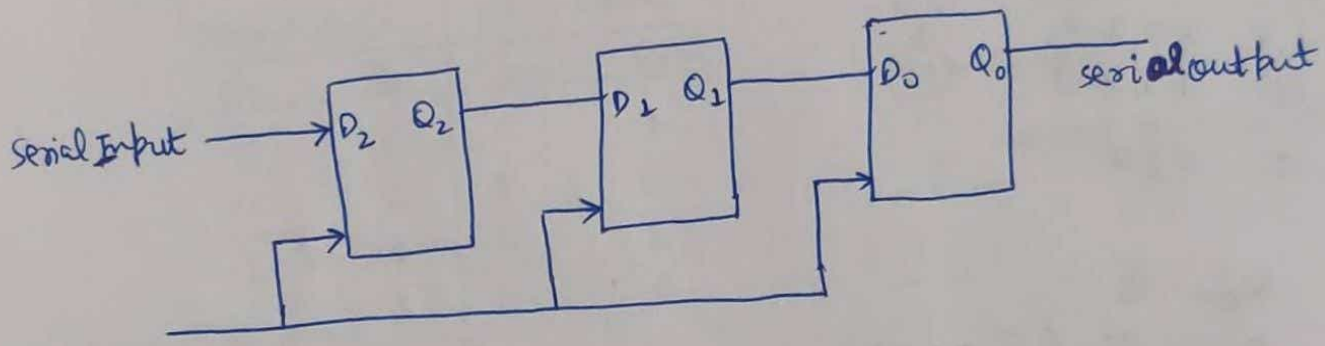
| present state | | | Next state | | |
|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Q_{2N}$ | $Q_{1N}$ | $Q_{0N}$ |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

# Applications of flipflops

(1) Shift Register → used as "sequential memory"
eg: Accumulator in Microprocessors.

(2) Counters → (1) used to count number of pulses.
(2) used as frequency divider.

# 3-bit shift Registers (Requires 3 D-flipflops)

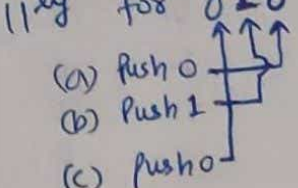



| clk | Serial Input | $Q_2$ $Q_1$ $Q_0$ |
|---|---|---|
| | | 0 0 0 |
| 0 | – | |
| 1 | 1 | 1 0 0 |
| 2 | 0 | 0 1 0 |
| 3 | 1 | 1 0 1 |

3 clock pulses Entire for Pushing Input on the shift Register.

Say I want to push 101 on the shift Register

(a) Push 1
(b) Push 0
(c) Push 1



Ily for 010

(a) Push 0
(b) Push 1
(c) Push 0

- Now, we have n bit shift Registers, which is a serial Input device then we require 'n' clock pulses inorder to place the Input on the flipflop

- In case of shift Registers 'n' clock pulses are required to place n-bits onto the shift Register.

| clk | $Q_2$ | $Q_1$ | $Q_0$ | output |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | → 1 |
| 1 | 0 | 1 | 0 | → 0 |
| 2 | 0 | 0 | 1 | → 1 |

To retreive the output we need 2 clock pulses

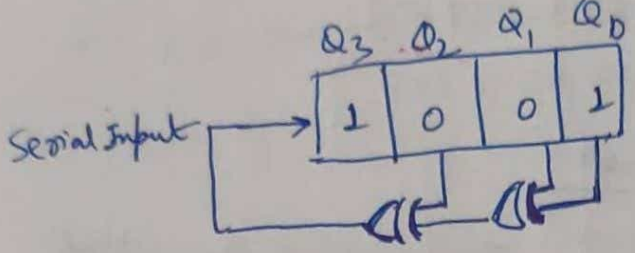
Hence, Totally 3clk + 2 clk pulses are needed for 3 bit serial Input & serial output

In general for n-bit serial Input & serial output we need (2n-1) Clock pulses

n → for serial I/P
n-1 → for serial O/P

# Example 1 on shift Right Register:

In the following right shift register, determine the number of clocks required to bring it to the Initial state of "1001"



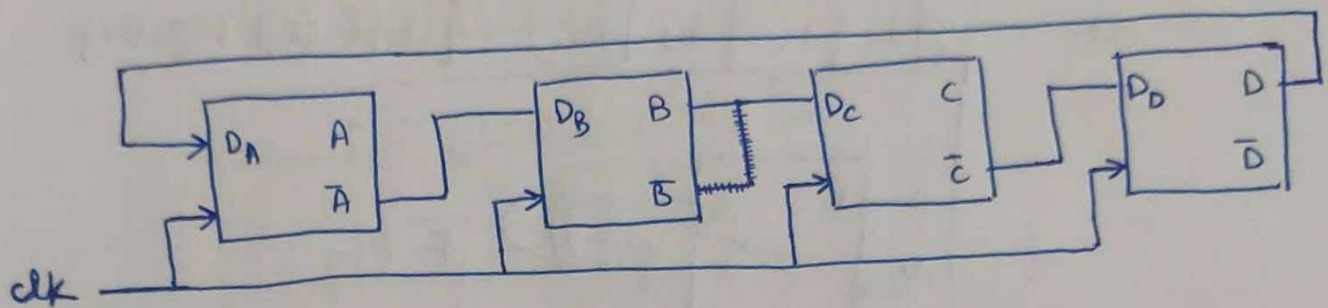

Serial I/P $\Rightarrow Q_2 \oplus Q_1 \oplus Q_0$

| Clk | Serial I/P | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|---|
| | — | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 |
| 2 | 0 | 0 | 1 | 1 | 1 |
| 3 | 1 | 1 | 0 | 1 | 1 |
| 4 | 0 | 0 | 1 | 0 | 1 |
| 5 | 0 | 0 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 | 0 | 1 |
| 7 | | | | | |

Hence, 7 clocks are required to bring it to the Initial state of 1001

# Example 2 on shift right Register:

Initial value of ABCD = 0 0 0 0, then what are the sequence of numbers, if the following circuit is counting.



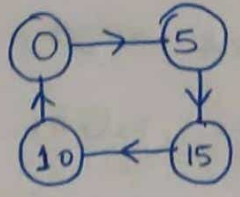

$D_A = D$

$D_B = \bar{A}$

$D_C = B$

$D_D = \bar{C}$

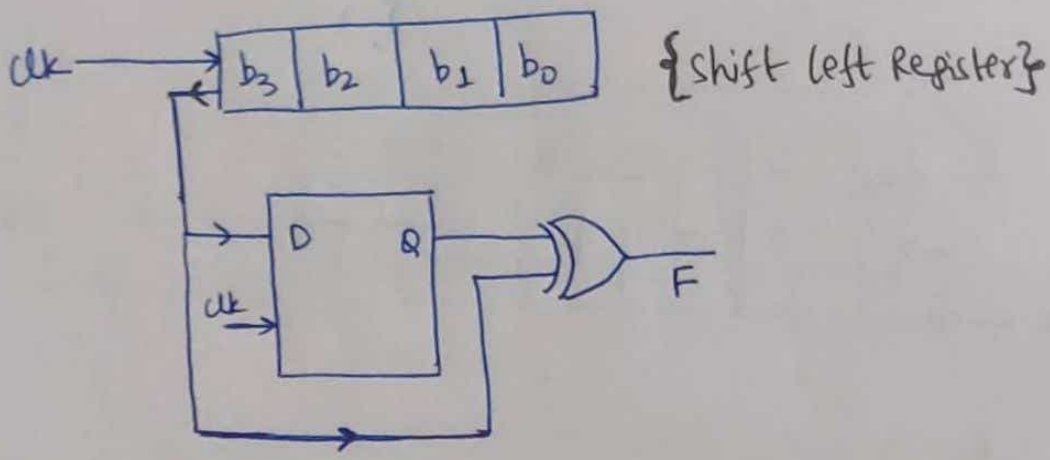
| Clk | A | B | C | D | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | → (0) |
| 1 | 0 | 1 | 0 | 1 | → (5) |
| 2 | 1 | 1 | 1 | 1 | → (15) |
| 3 | 1 | 0 | 1 | 0 | → (10) |
| 4 | 0 | 0 | 0 | 0 | → (0) |
| 5 | 0 | 1 | 0 | 1 | → (5) |




ie After 0, 5, 15, 10 zero is repeated Again & so on.

# Binary To Gray Converter:

Determine the function of the following circuit.

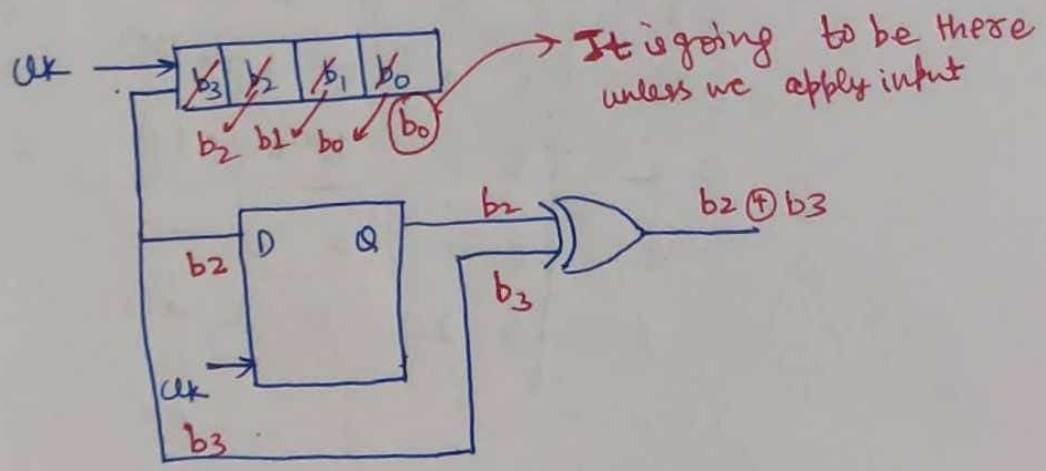



Initially Assume the output of D-flipflop is '0'

Now, $b_3$ is given to the output of D-flipflop ∴

The output of XOR gate will be $0 \oplus b_3 = b_3$

After Applying 1 clock



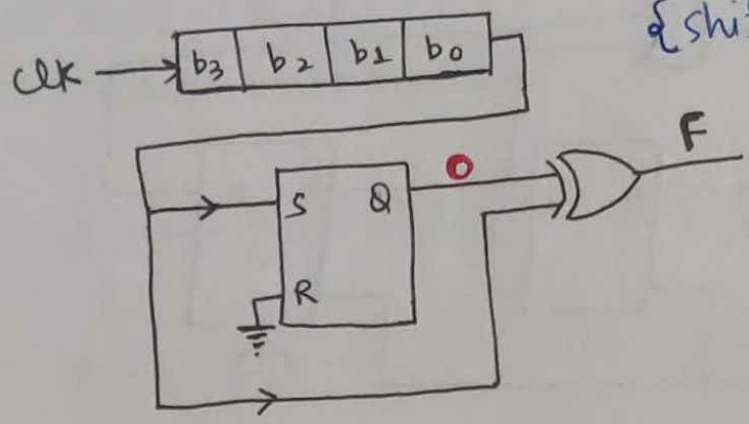

Continuing the procedure we get,

$b_3$, $b_3 \oplus b_2$, $b_2 \oplus b_1$, $b_1 \oplus b_0$

⟹ Gray code Convertion

# Finding 2's complement:

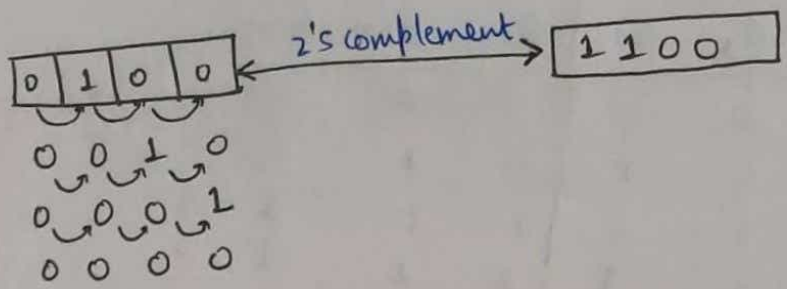
what is the function of the following circuit?




{Shift Right Register}

S R
0 0 → Latch
1 0 → Set

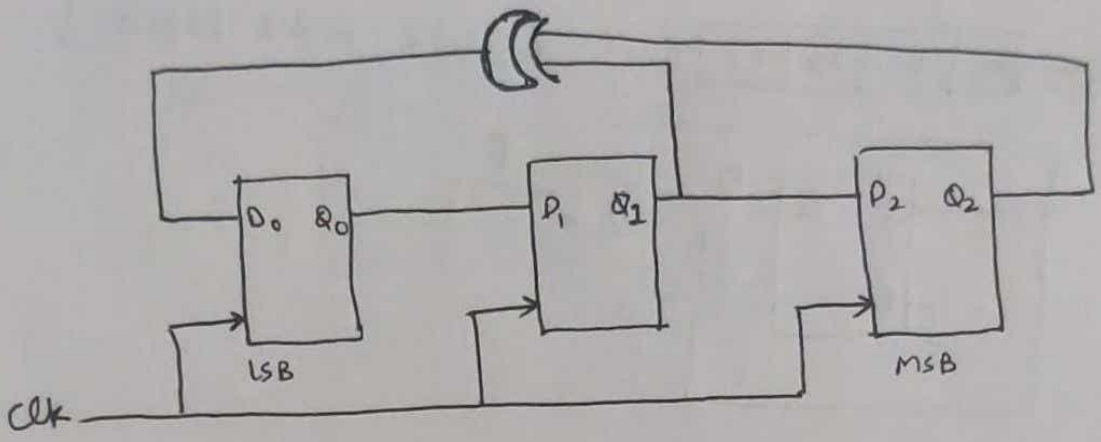
Here, R is always be in 0 state hence whenever S do not become 1, then it is in latch mode & produces the same output 0. And as soon as S become 1 then it is in 1 0 state ie set state.
(S R)




IIly for 0001 we get 1111

⇒ 2's complement conversion

Q Consider the circuit given below with initial state $Q_0 = 1$, $Q_1 = Q_2 = 0$. The state of the circuit is given by the value $4Q_2 + 2Q_1 + Q_0$




which of the following is a correct state sequence of the circuit?

(a) 1, 3, 4, 6, 7, 5, 2

(b) 1, 2, 5, 3, 7, 6, 4

(c) 1, 2, 7, 3, 5, 6, 4

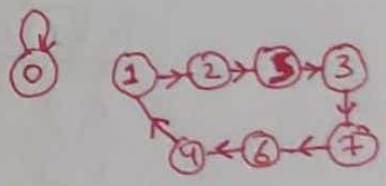
(d) 1, 6, 5, 7, 2, 3, 4

$Q_{2N} = Q_1$

$Q_{1N} = Q_0$

$Q_{0N} = Q_1 \oplus Q_2$

| Present state | | | Next state | | | |
|---|---|---|---|---|---|---|
| $Q_2$ | $Q_1$ | $Q_0$ | $Q_{2N}$ | $Q_{1N}$ | $Q_{0N}$ | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 2 |
| 2 | 0 | 1 | 1 | 0 | 1 | 5 |
| 3 | 0 | 1 | 1 | 1 | 1 | 7 |
| 4 | 1 | 0 | 0 | 0 | 1 | 1 |
| 5 | 1 | 0 | 0 | 1 | 1 | 3 |
| 6 | 1 | 1 | 1 | 0 | 0 | 4 |
| 7 | 1 | 1 | 1 | 1 | 0 | 6 |



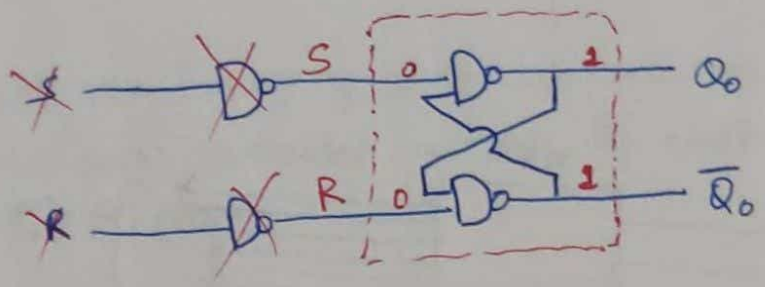
Q SR latch Made by Cross Coupling Two NAND gates if S = R = 0, then it will result in -

(a) $Q_0 = 0$, $\bar{Q}_0 = 1$

(b) $Q_0 = 1$, $\bar{Q}_0 = 0$

(c) $Q_0 = 1$, $\bar{Q}_0 = 1$

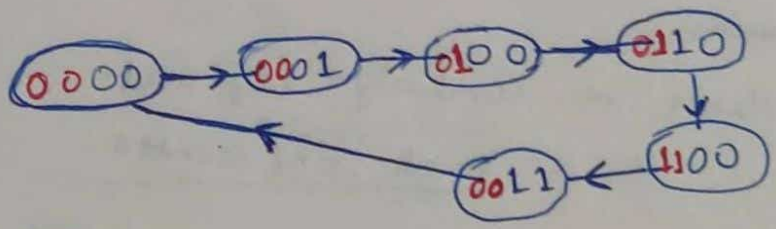
(d) Indeterminate state.

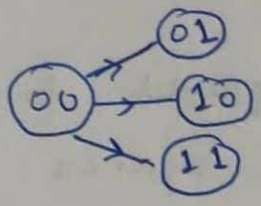



For S = R = 0, we get $Q_0 = 1$ & $\bar{Q}_0 \Rightarrow 1$

Q We want to design a synchronous counter that counts the sequence 0-1-0-2-0-3 & then repeats. The minimum number of J-K flipflops required to implement this counter is -

(a) 1
(b) 2
(c) 4
(d) 5



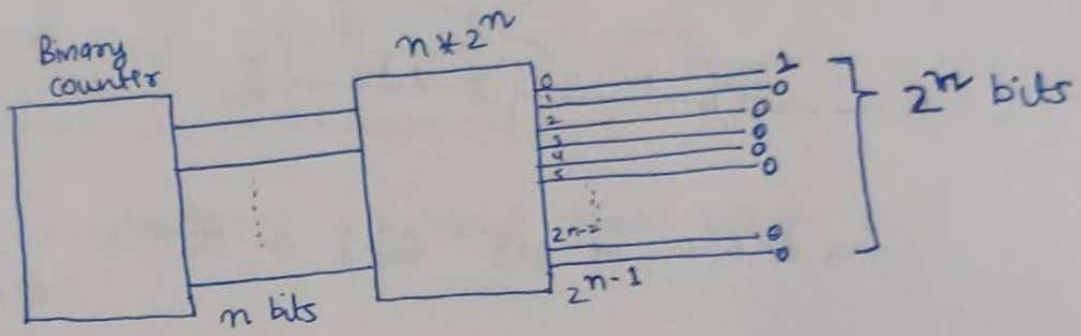




$\lceil \log_2 3 \rceil \Rightarrow 2$

Hence 4 J-K flipflops are required to implement this Counter

Q Let $k=2^n$, a circuit is built by giving the output of an $n$-bit binary counter as input to an $n$ to $2^n$ bit decoder. This circuit is equivalent to a

(a) k-bit binary up counter.
(b) k-bit binary down counter.
(c) k-bit ring counter
(d) k-bit Johnson counter.




As it outputs only one 1. Hence According to the Binary counter whether it is Binary up or Binary down.

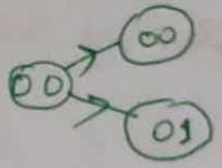
Hence, we combinely say that

It is k-bit ring counter.

Q The minimum number of JK flipflops required to construct a synchronous counter with the count sequence

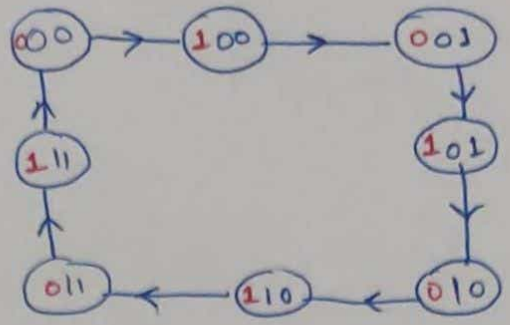
(0,0,1,1,2,2,3,3,0,0....) is —

(a) 0
(b) 1
(c) 2
(d) 3



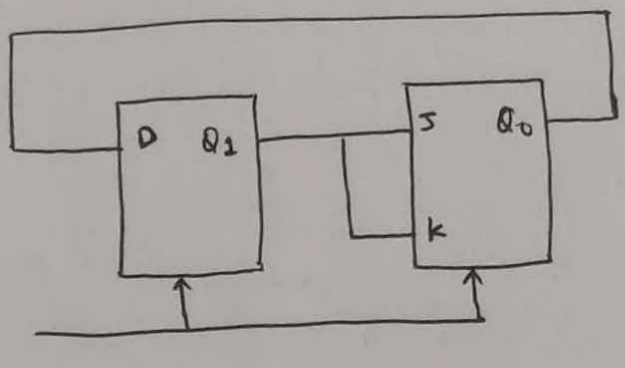

$\lceil \log_2 2 \rceil \Rightarrow 1$

Hence, 3 JK flipflops are required to implement this counter




⇒ 8 states
Hence, 3 JK flipflops are required

Q which of the following is the bit sequence (including Initial state) generated at the Q output of the JK flipflops.



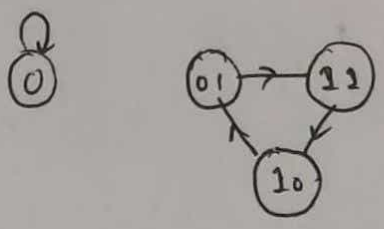

(a) 0 110110...
(b) 0 100100...
(c) 0 111 0111 0...
(d) 0 11001100...

$Q_{1N} = Q_0$

$Q_{0N} = J\bar{Q}_0 + \bar{K}Q_0$

$= Q_1\bar{Q}_0 + \bar{Q}_1 Q_0$

$= Q_1 \oplus Q_0$




| $Q_1$ | $Q_0$ | $Q_{1N}$ | $Q_{0N}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

0 1 10 110...

10 → 01 → 11 → 10 → 01 → 11