Digital Electronics

- Electronics.
  the study of motion of electron inside the semi-conductor is knowas Electronic.

note:
Semi-conductor is having Control Conductivity

- Boolean logical idea :- there are 3 classification
  1. producting the constant 0,1 [null, identity]
  2. Uniary operation [transfer, complementary]
     - transfer → Buffer
     - complementary → NOT
  3. Binary operation [AND, OR, NAND, NOR, XOR, XNOR, Inhibition, Implication].

note
For n i/p variable we will get 2^n combination & 2^{2^n} possible function.

- Truth table

"1" corresponding to minterm
"0" correspond to maxterm

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

$F_0 = 0$ Null

$F_1 = x.y$ AND
$x \wedge y$

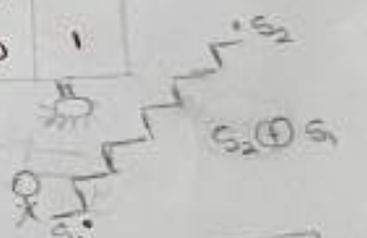
$F_2 = x.\bar{y}$
Inhibition
(x but not y)

$F_3 = x$ transfer (Buffer)

$F_4 = \bar{x}.y$
Inhibition
(y but not x)

$F_5 = y$ transfer Buffer

$F_6 = x \oplus y$ EXOR
$= \bar{x}y + x\bar{y}$
[For two way switching operation like staircase, Escalators, X-OR logic is used].



S_2	S_1	Bulb
0	0	0
0	1	1
1	0	1
1	1	0

$F_7 = x + y$ OR
$= x \cup y$

$F_8 = \overline{x+y}$ NOR
$x \downarrow y$

$F_9 = x \odot y$ EXNOR
$\bar{x}\bar{y} + xy$

[EXNOR is also knowas Coincidence logic gate, equivalence logic gate].

$F_{10} = \bar{y}$ compl^n of y (NOT)

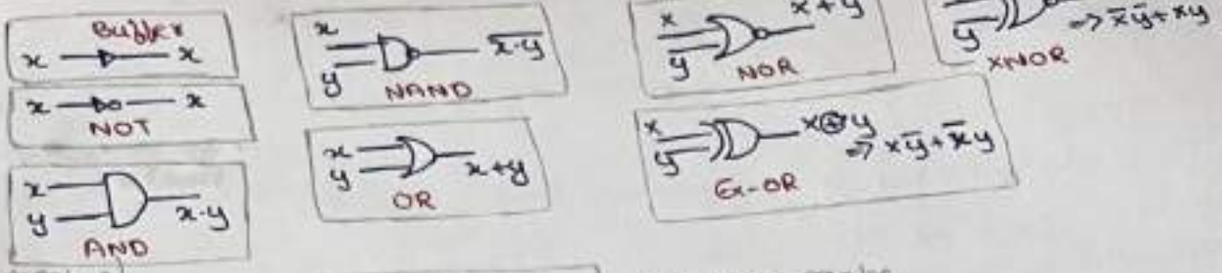
$F_{11} = x + \bar{y}$ Implication

$F_{12} = \bar{x}$ compl^n of x (NOT)

$F_{13} = \bar{x} + y$ Implication

$F_{14} = \overline{x.y}$ NAND
$x \uparrow y$

$F_{15} = 1$
Identity.



universal gate

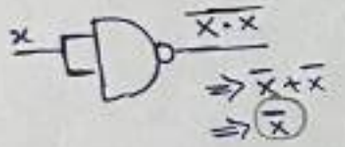
- NOT gate can be construct using NAND gate

	NAND ← 4 → NOR	
NOT	1	1
AND	2	3
OR	3	2
Ex-OR	4	5
Ex-NOR	5	4

- 4 NAND gate required for 1 NOR gate
- 4 NOR gate required for 1 NAND gate
- these is applicable only on individual not for expression

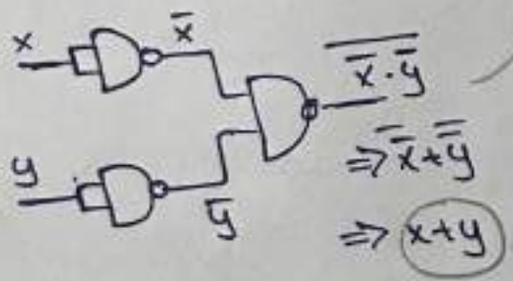
NAND

1. NOT

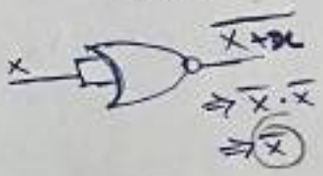


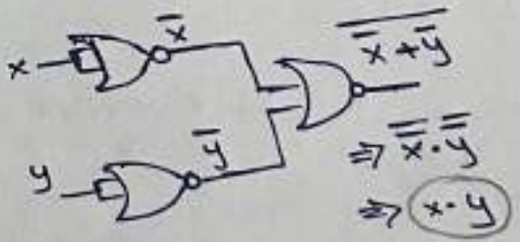
2. AND

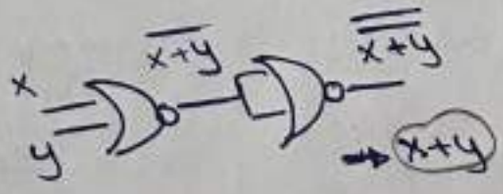
3. OR



NOR.



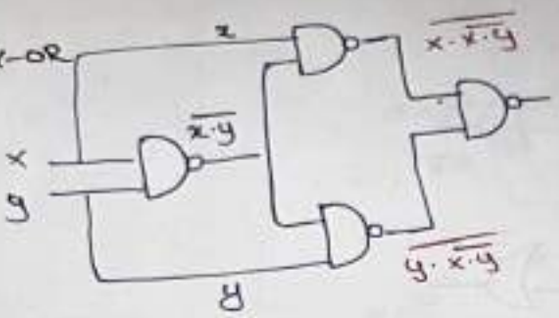


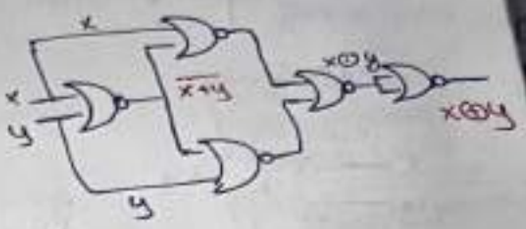


NAND

NOR

④ Ex-OR





$\rightarrow \overline{\overline{x \cdot \overline{x \cdot y}} \cdot \overline{y \cdot \overline{x \cdot y}}}$

$\rightarrow \overline{\overline{x \cdot \overline{x \cdot y}}} + \overline{\overline{y \cdot \overline{x \cdot y}}}$

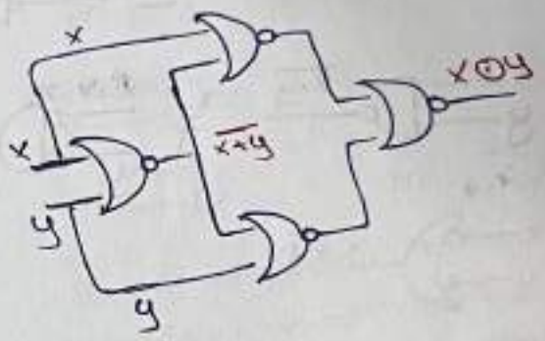
$\rightarrow x \cdot \overline{x \cdot y} + y \cdot \overline{x \cdot y}$

$\rightarrow x[\bar{x}+\bar{y}] + y[\bar{x}+\bar{y}]$

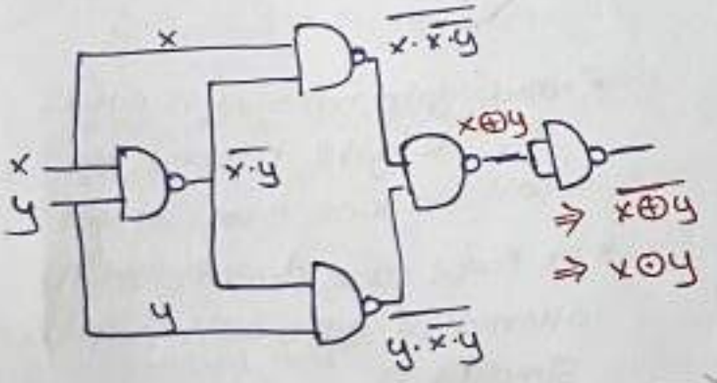
$\rightarrow x\bar{x} + x\bar{y} + y\bar{x} + y\bar{y}$

$x\bar{x}$ = 0, $x\bar{y} + y\bar{x} = x \oplus y$, $y\bar{y}$ = 0

$\Rightarrow x \oplus y$



⑤ EXNOR



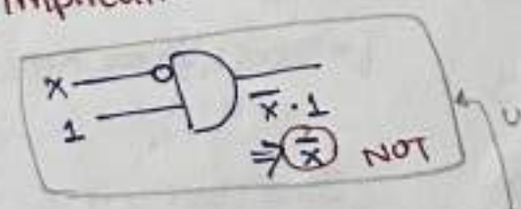
$\Rightarrow \overline{x \oplus y}$

$\Rightarrow x \odot y$

Note:- Inhibition, Implication are universal gate when provided 0 & 1

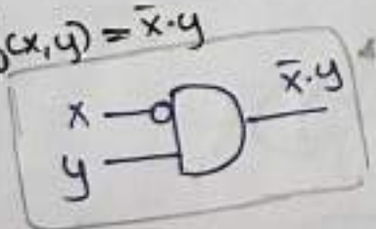
ex: $f(x,y) = \bar{x} \cdot y$

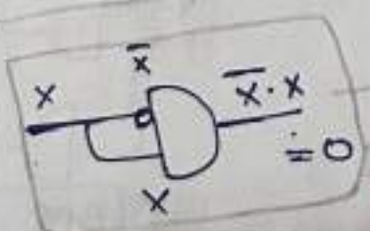
$f(x,y) = x \cdot \bar{y}$



using these both you can create NAND gate which is universal gate

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

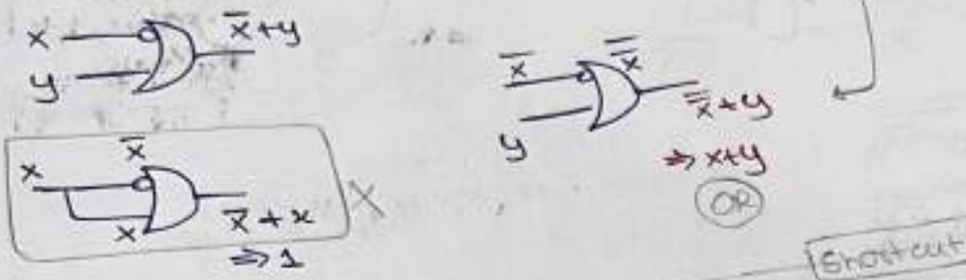




~~Not~~ is not possible

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

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

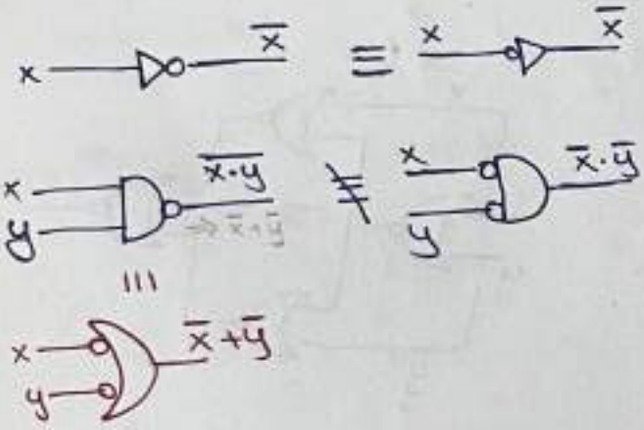


Shortcut

bubbled

NAND → OR
NOR → AND
AND → NOR
OR → NAND

- Alternative logic gate



NAND → OR
NOR → AND
AND → NOR
OR → NAND

- "NAND" gate bubbled i/p is equal to "OR" gate bubbled i/p will produce same result.
- In some case Conversion to alternative gate will help us to simplify it.

- Logic Minimization technique:-

SOP & POS form

	x	y	minterm	z (Given)	
0	0	0	$\bar{x} \cdot \bar{y}$	0	
1	0	1	$\bar{x} \cdot y$	1	$\bar{x} \cdot y$
2	1	0	$x \cdot \bar{y}$	0	
3	1	1	$x \cdot y$	1	$x \cdot y$

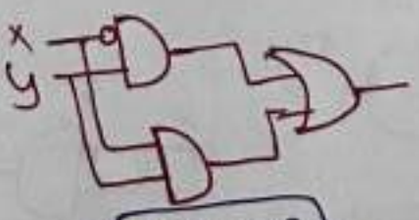
minterm

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

$f = \Sigma m(1,3) = \pi M(0,2)$

01 11

$\Sigma m \; f = \bar{x} \cdot y + x \cdot y$



AND-OR

(product term)

SOP form

also can be said as

AND-OR

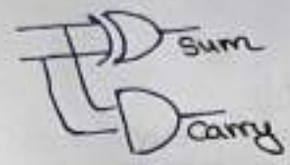
x	y	carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$\bar{x}\cdot y$, $x\cdot\bar{y}$

Halfadder

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

$= x \oplus y$ (modulo addition)



SOP (OR) Disjunctive normal form

POS (OR) conjunctive normal form

EX-NOR

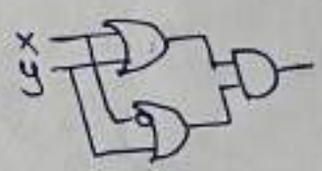
$x \odot y = x\bar{y} + xy$

x	y	
0	0	1 ($\bar{x}\cdot\bar{y}$)
0	1	0
1	0	0
1	1	1 ($x\cdot y$)

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

POS or **OR-AND** gate

x	y	f (maxterm)	
0	0	0	$x+y$
0	1	1	
1	0	0	$\bar{x}+y$
1	1	1	



OR

x	y	f = x+y	
0	0	0	$x+y$
0	1	1	
1	0	1	
1	1	1	

AND

x	y	f = x·y	
0	0	0	
0	1	0	
1	0	0	
1	1	1	$x\cdot y$

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

minterm & maxterm will give same o/p for a circuit.

(we can say that minterm is complement of maxterm & viceversa)

- when maxterm is to be calculated then in the functn we see for "0" and draw the expression for it.
- when minterm is to be calculated then in the functn we see for "1" and draw the expression for it.
- minterm which fn^ns can't be sub divided

ex: $\bar{x}\bar{y}$

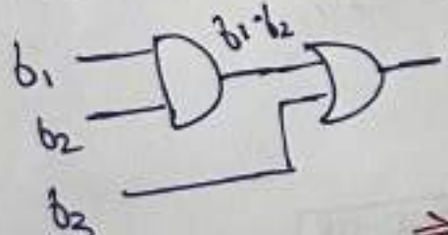
$\bar{x}$ $\bar{y}$ X (not same)

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

$b_1 = \Sigma m(0,1,2,5,7)$

$b_2 = \Sigma m(2,3,4,5,6,7)$

$b_3 = \Sigma m(0,1,2,4,6)$



$f = b_1 \cdot b_2 + b_3$

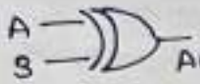
$(b_1 \cap b_2) \cup b_3$

$\Rightarrow \Sigma m(2,5,7) + \Sigma m(0,1,2,4,6)$

$\Rightarrow \Sigma m(0,1,2,4,5,6,7)$

$A = \Sigma m(0,1,4,6,7)$

$B = \Sigma m(2,3,4,5,6,7)$.



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

$\bar{A}B$ → B but not A

$A\bar{B}$ → A but not B.

$\Rightarrow \bar{A}B = \Sigma m(2,3,5)$

$A\bar{B} = \Sigma m(0,1)$

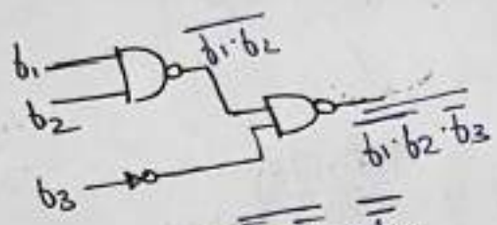
$\bar{A}B + A\bar{B} = A \oplus B$

$\Rightarrow \Sigma m(0,1,2,3,5)$

ex. Given $b_1 = \Sigma m(0,1,3,5)$

$b_2 = \Sigma m(6,5)$

$b_3 = \Sigma m(1,4,5)$



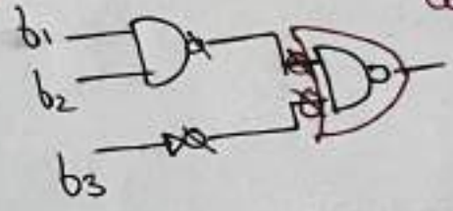
$\Rightarrow \overline{\overline{b_1 + b_2} + \bar{b}_3}$

$\Rightarrow b_1 \cdot b_2 + \bar{b}_3$

$\Rightarrow (5) + (1,4,5)$

$(1,4,5)$

Can use alternative gate to cancel compliment.



• Duality.

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

$B = \{0, +\}$

$B = \{0, 1\}$

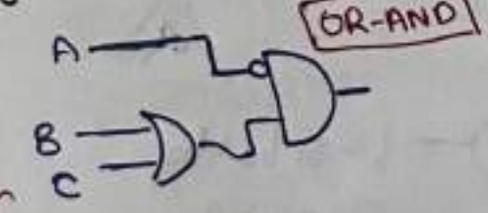
Interchange

Step 1 $\{0, +\}$

Step 2 $\{0, 1\}$

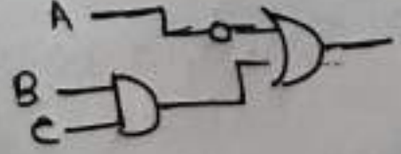
ex.

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



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

AND-OR



Note

- $(f^D)^D = f$
  It happen for every Boolean Expression.
- If expression is tautology then it's dual is also tautology

Order of Priority
1. ( )
2. NOT
3. AND
4. OR

- Self Duality $[f^D = f]$

ex: $f = AB + AC + BC$

$f^D = (A+B)(A+C)(B+C)$

$\Rightarrow [A+(B+C)](BC)$

$\Rightarrow A\cdot B + A\cdot C + BCB + BCC$

$\Rightarrow AB + AC + BC + BC$

$\Rightarrow \boxed{AB+AC+BC} = f$

So, $\boxed{f^D = f}$

if, f is Self Dual
It happen only for Some expression.

- Rules to check Self Duality.

① It should be neutral function. [we can say that equal no. of 0 & 1]

② It should not contain mutually Exclusive term.

$f(A,B,C)$ (Given)

	A	B	C	f	
0	0	0	0	0	M_0
1	0	0	1	1	M_1
2	0	1	0	1	M_2
3	0	1	1	0	M_3
4	1	0	0	1	M_4
5	1	0	1	0	M_5
6	1	1	0	1	M_6
7	1	1	1	0	M_7

neutralize
no. of terms
Minterms.
(m_1, m_2, m_4, m_6)
=
no. of maxterm
(M_0, M_3, M_5, M_7)

Condition.

ex: a) $f = \Sigma m(0,1,3)$ ✗ , ✗
b) $f = \Sigma m(0,1,5,6)$ I , ✗
c) $f = \Sigma m(2,3,4,5)$ I , ✗
d) $f = \Sigma m(0,1,2,3)$ I , II

$0 \to 000$ $(0,7)$
$7 \to 111$
$1 \to 001$ $(1,6)$
$6 \to 110$

if sum of 2 no. is coming out to be $(2^n - 1)$ then it's mutually exclusive
where "n" is no. of variable in funcn

$2 \to 010$ $(2,5)$
$5 \to 101$

So, all these pair are mutually exclusive.

If there Sum is coming out to be (7) if Variable are ABC (3)
$2^n - 1$

• no. of self Dual form

ex: $f(A,B,C)$
$n = 3$ variable

(0,7) (1,6) (2,5) (3,4)
↓ ↓ ↓ ↓
2 possibility × 2 × 2 × 2 = 16

$f = \Sigma m(0,1,2,3)$
$f = \Sigma m(7,1,2,3)$
$f = \Sigma m(0,1,2,4)$
⋮
16 combination.

So, 2^4
$\Downarrow$
2^{2^2}
$\Downarrow$
$2^{2^{3-1}}$
Self Dual.

So, general formula will be $2^{2^{n-1}}$

Shortcut

$x \oplus$	x	$= 0$
$x \oplus$	$\bar{x}$	$= 1$
$x \oplus$	1	$= \bar{x}$
$x \oplus$	0	$= x$

Your Shortcut

$x \odot x = 1$
$x \odot \bar{x} = 0$
$x \odot 1 = x$
$x \odot 0 = \bar{x}$

$\overline{x \oplus y}$
$\bar{x} \oplus y$
$x \oplus \bar{y}$
$= x \odot y$

$\bar{x} \oplus \bar{y} = x \oplus y$

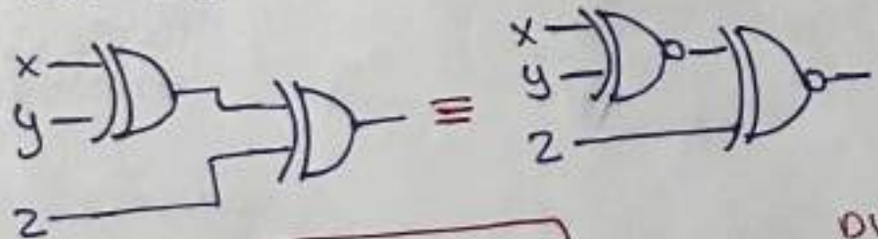
$\overline{x \oplus y} = x \odot y$

$(x \oplus y) \oplus z = (x \odot y) \odot z$

$\overline{[(w \oplus x) \oplus y] \oplus z} = [(w \odot x) \odot y] \odot z$

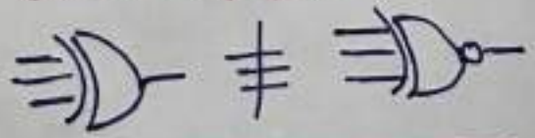
For 2 i/p gate
- For even no. of variable
  $\overline{EXOR} = EXNOR$
- For odd no of variable
  $EXOR = EXNOR$

Two I/p gate



$EXOR \equiv EXNOR$

Direct 3 i/p gate



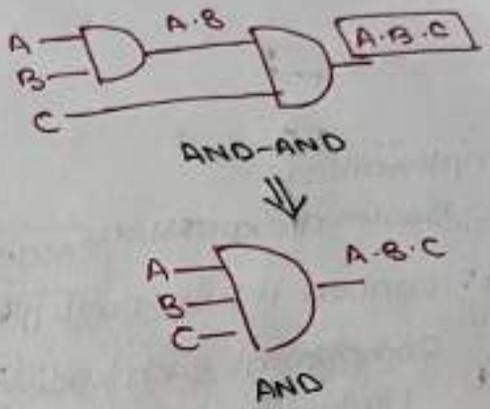
then $EXOR \neq EXNOR$

x	y	z	x⊕y	(x⊕y)⊕z	xyz	x⊙y	(x⊙y)⊙z
0	0	0	0	0	0 0 0	1	0
0	0	1	0	1	0 0 1	1	1
0	1	0	1	1	0 1 0	0	1
0	1	1	1	0	0 1 1	0	0
1	0	0	1	1	1 0 0	0	1
1	0	1	1	0	1 0 1	0	0
1	1	0	0	0	1 1 0	1	0
1	1	1	0	1	1 1 1	1	1

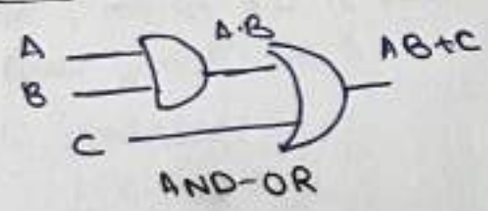
- **Degenerative forms.**

If a 2-level logic gate system o/p is expressed by a single logic gate then that 2 level logic gate system is known as Degenerat form for a single logic gate.

ex: AND-AND is the degenerate for the single AND gate



non-Degenerative form.



Shortcut

AND-OR ⟷ (dual form) OR-AND

NAND-NAND ⟷ NOR-NOR

NOR-OR ⟷ NAND-AND

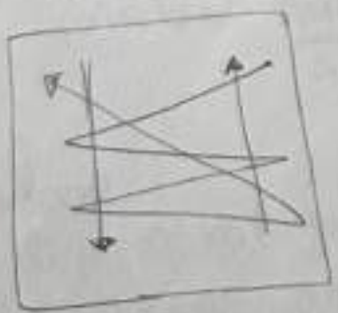
OR-NAND ⟷ AND-NOR

all these form are non-Degenerate form

all the Corresponding are Dual form

- combination present in the same row of above representation are Dual form

method to remember



- positive & negative logic

(+)ve logic	(-)ve logic
High → 1	High → 0
Low → 0	Low → 1

ex: $-7V$ $(-5V)$ - Highest
0 1

(+ve) logic

(+ve) AND logic

A	B	$f=A\cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

(-ve) AND = (+ve) OR

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

- -ve AND is equal to (+ve) OR & vise-versa.
- -ve NAND is equal to (+ve) NOR & vice-versa.

- Complimenting the Boolean expression.

Step 1: convert it into dual form.

Step 2: Compliement each individual variable.

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

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

Step 2: $\bar{f}=(\bar{\bar{A}}+\bar{B})\cdot\bar{C}$

$\Downarrow$

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

Note

- NAND & Nor are universal gate through which we can create all other gate. So, its functionally complete.

Generalize;
If we have 'k' variable & 'm' minterms present then $^{2^k}C_m$

- If we have 2 variable then total no. of functn can be $2^{2^2} \Rightarrow 2^4 = 2^{16}$
- out of 16 only 7 are mostly used which are also implimented in form of logic gate.

$f(x,y) = xy \rightarrow$ AND
$f(x,y) = \bar{x}y + x\bar{y} + xy \rightarrow$ OR
$f(x) = \bar{x} \rightarrow$ NOT
$f(x,y) = \bar{x}\bar{y} + \bar{x}y + x\bar{y} \rightarrow$ NAND
$f(x,y) = \bar{x}\,\bar{y} \rightarrow$ NOR
$f(x,y) = \bar{x}y + x\bar{y} \rightarrow$ XOR
$f(x,y) = xy + \bar{x}\bar{y} \rightarrow$ XNOR

- Type of gate

Basic or fundamental gate
- AND → ∞ i/p can be 'n'
- OR → " " " "
- NOT (Inverter or bubbled) → i/p can be only one

Derived gate
- NAND → universal gate
- NOR → universal gate
- XOR → Ringsum $x \oplus y$
- XNOR → $x \odot y$

ex: How many minterm will be present in 'n' variable AND functn. is 1

2 variable
$f(x,y) = x \cdot y$

3 variable
$f(x,y,z) = x \cdot y \cdot z$

ex: How many minterm will be present in 'n' variable OR functn
$2^n - 1$
only one absent when all are zero.

ex: no. of minterm in NOR is
OR → NOR
$2^n - 1 \rightarrow 1$

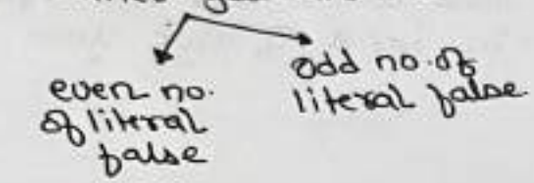
ex: no. of minterm in NAND will be
$\overline{AND} \rightarrow$ NAND
$\overline{1} \rightarrow 2^n - 1$

no. of minterm in XOR is
$2^{n-1} \rightarrow \frac{2^n}{2}$

- XOR
o/p of xor gate is '0' if even no. of input are 1 & o/p is 1 when odd no. of i/p are 1.

- XNOR
- O/P of XNOR gate is 1 if even no. of I/P are 0.

In 'n' variable either you can have



- O/P of XNOR gate is 0. if odd no. of I/P are 0.
- no. of minterm in XNOR is $\frac{2^n}{2}$ or 2^{n-1}

- If odd no. of variable then XOR ≡ XNOR
- If even no. of variable then XOR ≡ $\overline{\text{XNOR}}$

- Principle of duality

1	0
0	1
.	+
+	.

- NAND & NOR are dual of each other
- XOR & XNOR are dual of each other
- AND & OR are dual of each other.

- boolean function is combination of boolean term or minterms.

ex $F(A,B,C) = (B+C)(A+B)$ is in standard POS
No, as in 1st term A is missing & in 2nd term C is missing
- Hence above functn is in POS but not SPOS.

XNOR &
- XOR variable of 'n' variable is.

neutral functn

- Self Dual.

no. of neutral functn for 3 variable?

2^3 → 8 minterm

& half should be present ${}^8C_4 = \frac{8\times7\times6\times5}{4\times3\times2\times1}$

= 70 functn

for "n" variable functn then neutral functn

$^{2^n}C_{2^{n-1}}$

• Orthogonal functn
A functn whose
Dual & complement
both are same
ex: $f = A \oplus B$
$f^D = A \odot B$
$f^C = A \odot B$

for 'n' variable the orthogonal functn are

2^n minterm

$\frac{2^n}{2} = 2^{n-1}$ pair } and you need to choice half of it

$\frac{2^{n-1}}{2} = 2^{n-2}$

so from

$^{2^{n-1}}C_{2^{n-2}}$

K-Map.

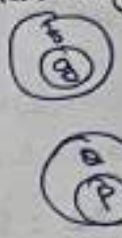
ex:

CD \ AB				
	1	1	1	1
	1	1	1	1
	1	1	1	1
	1	1	1	1

then these will be represented as (1)
& literal count will be (zero)

• In logic/B.A we say f covers g means $g \to f$ is a tautology.

A	B	f_1	f_2	$f_1 \to f_2$
0	0	0	0	1
0	1	1	1	1
1	0	1	1	1
1	1	0	1	1



$g \subseteq f$
$g \to f$
$P \to Q$

f_2 covers f_1 : $f_1 \to f_2$ is Tautology

• Implicant
given 2 functn f_1 & f_2 s.t. $f_1 \subseteq f_2$ then f_1 is called implicant of f_2
i.e $f_1 \to f_2$ is tautology
[Simply we can say that every subset of boolean functn is an implicant]

i.e if functn "f" has "n" minterms then how many implicant.

• (2^n) no. of subset.

if functn "f" has "x" minterms of "n" variable then how many functn can covers "f"?

$n \to (2^n)$ minterm

$2^n - x$ are absent

& for these we have 2 choice present or absent

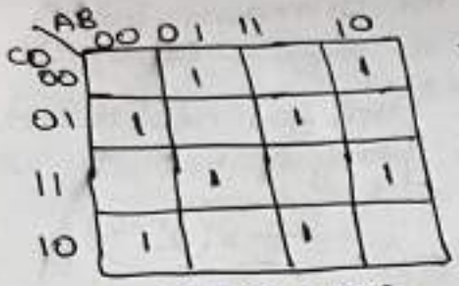
$2^{2^n - x}$

ex: $f(A,B,C) = AB + BC$
How many functn can cover f?

$f(A,B,C) = \Sigma m(6,7,3)$
3 are present

total minterm $2^3 = 8$ } so, $8 - 3 = 5$

So, $2^5 = 32$ functn



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

⇒ XOR (odd no. of literal true) if even then its XNOR

ex: How many funct^n does $f_1 \cdot f_2 \neq f_1 + f_2$ represent?

$f_1(a,b,c) = \sum m(0,2,4) + d(3,5,7)$

$f_2(a,b,c) = \sum m(2,3) + d(1,6,7)$

$f_1 \cdot f_2 = \sum m(2) + d(3,7)$

$f_1 + f_2 = \sum m(0,2,3,4) + d(1,5,6,7)$

a	bc	f_1	f_2	$f_1 \cdot f_2$	$f_1 + f_2$
0	00				
0	01				
0	10				
0	11				
⋮					

← same copy

small example taken

ex: $f_1(a,b) = \sum m(1) + d(0,2)$

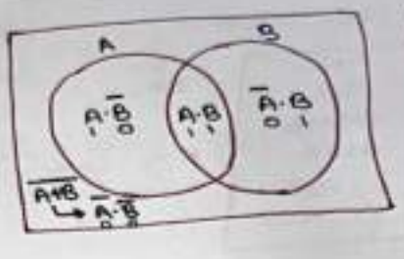
$f_2(a,b) = \sum m(2) + d(0,1,3)$

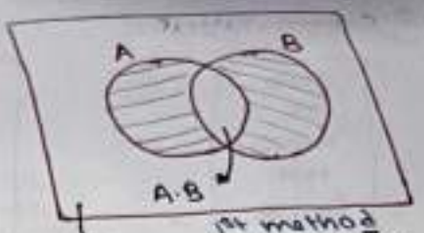
a	b	f_1	f_2	$f_1 \cdot f_2$	$f_1 + f_2$
0	0	x	x	x	x
0	1	1	x	x	1
1	0	x	1	x	1
1	1	0	x	0	x

• Venn Diagram

i.e $x+x\cdot y=x$

i.e $x+\bar{x}\cdot y=x+y$





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

1st method

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

$= A \oplus B$

2nd method →

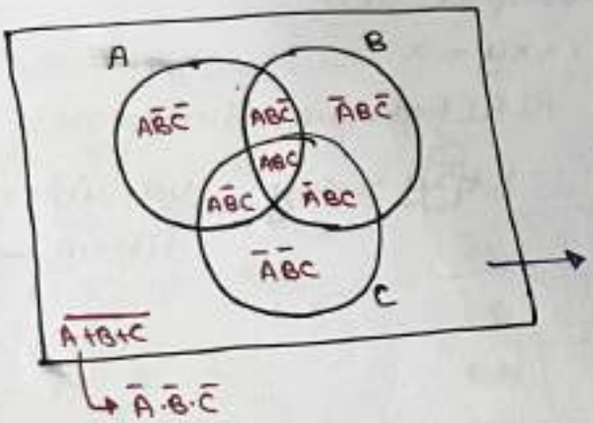
→ $\overline{A\cdot B+\bar{A}\bar{B}}$

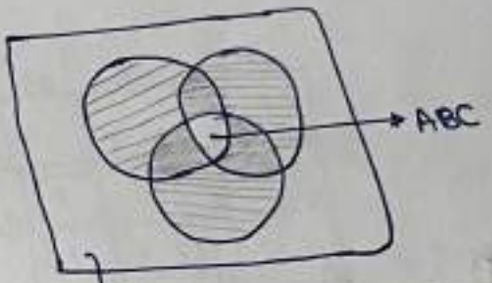
→ $\overline{A \odot B}$

→ $A \oplus B$

taking unshaded area's and complementing it.

ex:





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

• Boolean laws

NOT

$(A')' = A$

Involutionary theorem.

AND

$x\cdot x = x$

$x\cdot 0 = 0$

$x\cdot 1 = x$

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

OR

$x+x = x$

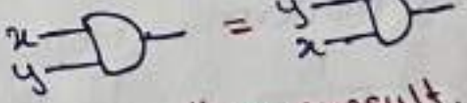
$x+1 = 1$

$x+0 = x$

$x+\bar{x} = 1$

• Commutative law

- $x\cdot y = y\cdot x$
- $x+y = y+x$



// Same result.

• Associative law.

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

↳ $(x+y)+z = x+(y+z)$

• Distributive law

↳ $x\cdot(y+z) = x\cdot y + x\cdot z$

↳ $x+(y\cdot z) = (x+y)\cdot(x+z)$

• Demorgan law.

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

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

Commulative

AND
OR
NAND
NOR
Ex-OR
Ex-NOR

Association

AND
OR
~~NAND~~
~~NOR~~
Ex-OR
Ex-NOR

note :-
↳ NAND, NOR commulative but not association.

- Consensus theorem.

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

Redundant

- Transposition theorem.

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

- Absorption law.

$x + xy = x$

RLR (Redundant literal role)

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

variable → 2 {x, y}

literal → 5

{x, $\bar{x}$, y, x, y}

- K-Map.

↳ it is the modification of the venn diagram

↳ it is the group of adjacent cell. each cell is represented by minterm.

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

MSB		LSB
A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

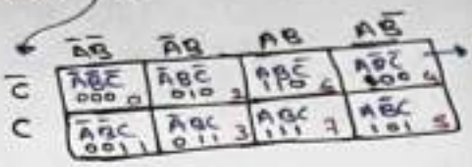
{(ABC)

MSB

	$\bar{B}\bar{C}$	$\bar{B}\bar{C}$	BC	$B\bar{C}$
$\bar{A}$	$\bar{A}\bar{B}\bar{C}$ 000 0	$\bar{A}\bar{B}C$ 001 1	$\bar{A}BC$ 011 3	$\bar{A}B\bar{C}$ 010 2
A	$A\bar{B}\bar{C}$ 100 4	$A\bar{B}C$ 101 5	ABC 111 7	$AB\bar{C}$ 110 6

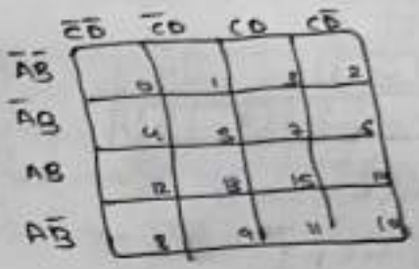
- you should be careful about the 2 format whether its LSB or MSB (By Default)
- Bc ans. get different with different format.

F(A B C)
MSB LSB



b/c C is LSB in MSB position.

ex: f(CAB)

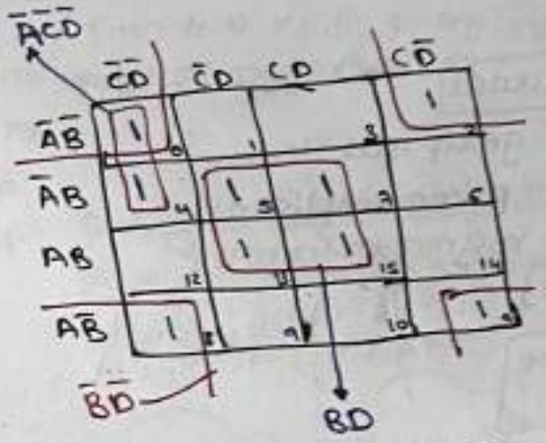


f(A B C)
MSB LSB

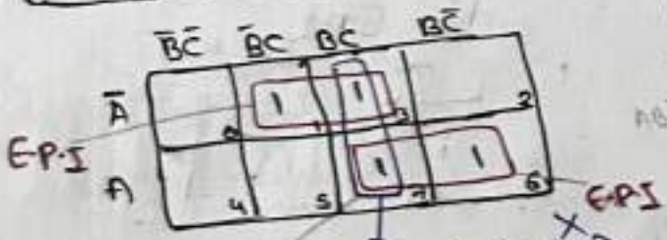
$f = \Sigma m(0,1,2,6)$ → 000, 001, 010, 110

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

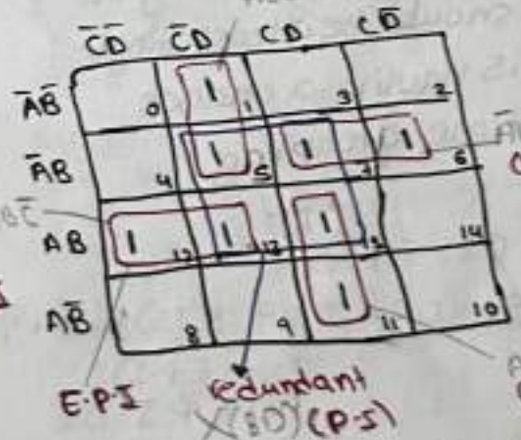
$f = \bar{A}\bar{B} + B\bar{C}$



Redundant grouping in k-map



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



note:-
always the grp. must be done from higher order to the lower order.

octet
quad
pair

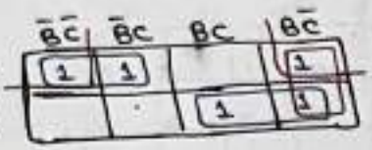
- it is that part of the circuit with or without it there is no effect of the circuit result.

we represent it by (1)

- **Universal gate**
- NAND & NOR are universal know as universal logic gate
- So by using these gate any boolean funct^n can be designed thats why NAND & NOR are know as functionally complete

- in the same manner inhibition, implication provide with 1,0 are also universal gate & treated as functionally complete

ex:



$f = \Sigma m(0,1,2,6,7)$

(a) $\bar{A}\bar{B} + \bar{A}\bar{C} + B\bar{C} + AB$
(b) $\bar{A}\bar{B} + \bar{A}\bar{C} + AB$
(c) $\bar{A}\bar{B} + B\bar{C} + AB$
(d) none

- **Implicant**
A single one or a group of 1's in K-map is called implicant

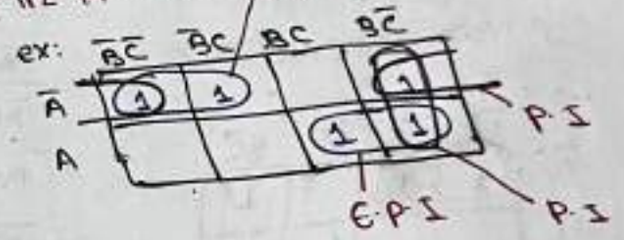
$\bar{A}\bar{B} \rightarrow$ E·P·I
$\bar{A}\bar{C}$, $B\bar{C}$ } one of there → P·I becomes → P·I Redundent. } selective P·I
$AB \rightarrow$ E·P·I

- any single 1 (or) group of 1's that can't be included in bigger rectangle in K-map

prime Implicant

- all possible grouping a K-map are know as prime Implicant [only from higher order to the lower order] should not be subset of other implicant

ex:



essential prime Implican

- it is that prime implicant only in which there should be atleast single one which is having a chance of only one time grouping.

- we need to find a single 1 in a group that can only form with one group then it's E·P·I

ex: the no. E·P·I. in "0"

$f(w,x,y,z) = \Sigma m(0,2,6,7,8,9,13,15)$ is.

(zero E·P·I) no E·P·I b/c there is no single "1" available which is only associated with one group.

So, its cyclic prime Implicant

- **Cyclic Prime Implicant:**
  If the no. of EPIs are zero it is said to be cyclic prime Implicant.

- **non-prime Implicant.**
  it is that one in the K-map which is not able to get any grouping

ex:

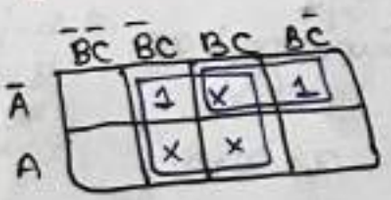
nonprime Implicant [non-P.I]

	ABC	f	(minterm
0	000	1	$\bar{A}\bar{B}\bar{C}$
1	001	0	
2	010	0	$\bar{A}BC$
3	011	1	
4	100	0	
5	101	0	$A\bar{B}C$
6	110	1	$AB\bar{C}$
7	111	1	ABC

- **Don't care condition in the K-map**

↳ it is an undefine o/p corresponding to unoccured i/p

$f = \Sigma m(1,2) + d(3,5,7)$



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

- If there is don't care then the pair we are making should be in the higher order

octet
quad
pair

- don't make pair of only don't care

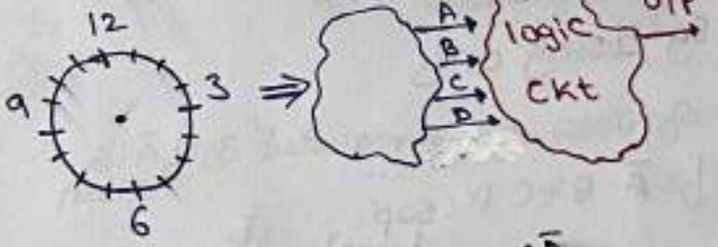
↳ neither EPI nor EPI Implicant
- **selective prime Implicant**
  those prime Implicant which are under the process of selection treated as selective prime Implicant.
- always occur in pair

- **minterms**
  ↳ it indicate one in the K-map (it must consist of all the variable)

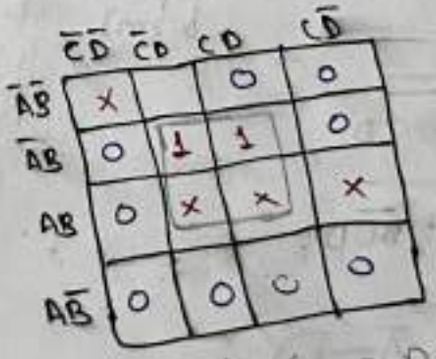
- **Implicants**
  ↳ it indicate one in the K-map (it may or may not consist all variable in the K-map)

ex: $f = \bar{A}\bar{B}\bar{C} + BC + AB$
Implicant.

ex: Design a logic circuit of 4 I/p which give an alarm at 5:00 clock & 7'o clock.



ABCD	
0.	X
1.	0
2.	0
3.	0
4.	0
5.	1
6.	0
7.	1
8.	0
9.	0
10.	0
11.	0
12.	0
13.	X
14.	X
15.	X



~~ABD~~ B·D.

• POS form in K-map

wrong method

$\overline{SOP} = POS$ ✗

ex. $f = \Sigma m(0,1,6,7)_{SOP} = \Pi M(2,3,4,5)$

	$\bar{B}\cdot\bar{C}$	$\bar{B}\cdot C$	BC	$B\cdot\bar{C}$
$\bar{A}$	1	1		
A			1	1

$f = \bar{A}\cdot\bar{B} + A\cdot B$ SOP

Step 1:

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

Step 2:

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

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

// Correct method

Step 1: Interchange $(\cdot, +)$

Step 2: Compln Individual variable.

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

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

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

$\bar{A}A \rightarrow 0$, $\bar{B}\bar{B} \rightarrow 0$

***

• Implementation of the Boolean expression

↳ By using completely by NAND Gate.

↳ By using only

Step 1: Bar Bar karo

Step 2: Inner Bar ko Expand karo.

Q. no. of NAND gate required 3; if $\bar{A}$ is given

ex: $f = \bar{A}\cdot B + C\cdot D$ (SOP form)

≠ 4; if $\bar{A}$ not given it required 1 extra.

$\Rightarrow \overline{\overline{\bar{A}\cdot B} + CD}$

$\Rightarrow \overline{\overline{\bar{A}\cdot B}\cdot\overline{CD}}$

no. of NAND gate 4 if $\bar{B}$ is given

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

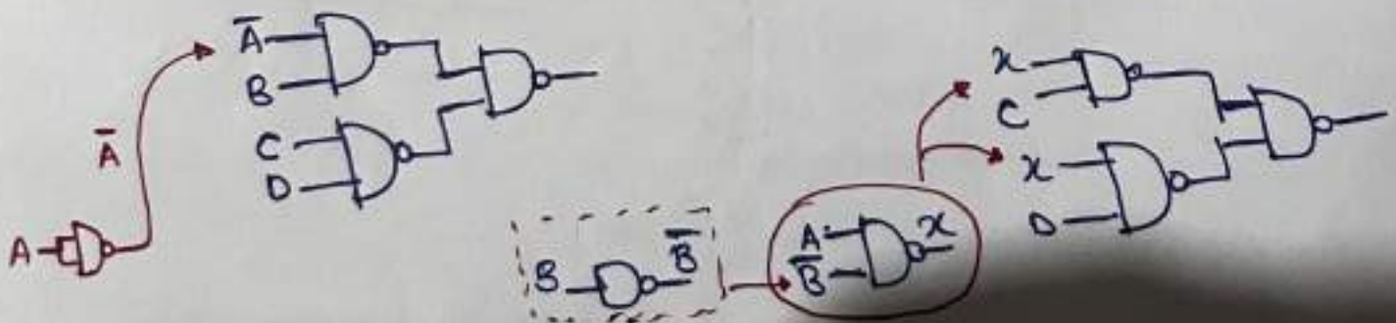
Suppose it as x ✗

$x\cdot(C+D)$

$\Rightarrow \overline{\overline{xC + xD}}$

$\Rightarrow \overline{\overline{xC}\cdot\overline{xD}}$

≠ 5 if $\bar{B}$ is not given

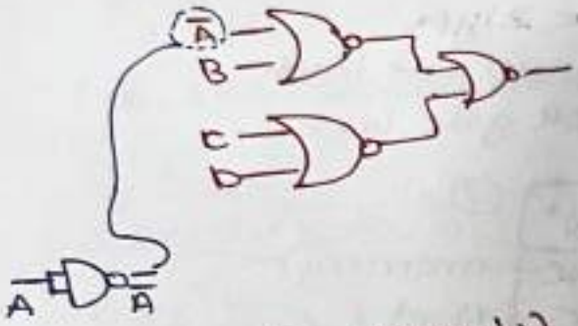


using NOR gate.

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

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

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



SO, 3 NOR is required if $\bar{A}$ is given.

else 4 NOR is required if $\bar{A}$ is not given.

using NOR gate

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

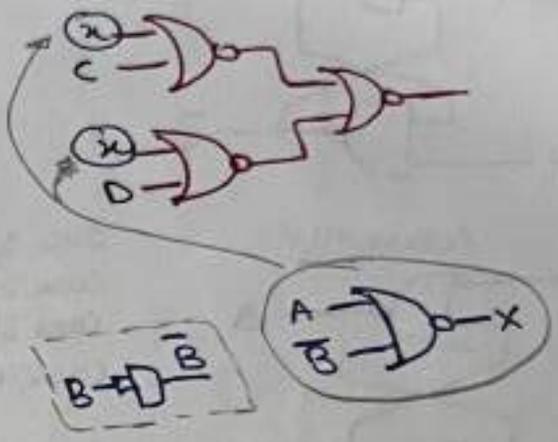
$\downarrow$

$x + C\cdot D$

$\Rightarrow \overline{\overline{(x+C)\cdot(x+D)}}$ POS.

$\Rightarrow \overline{\overline{(x+C)} + \overline{(x+D)}}$

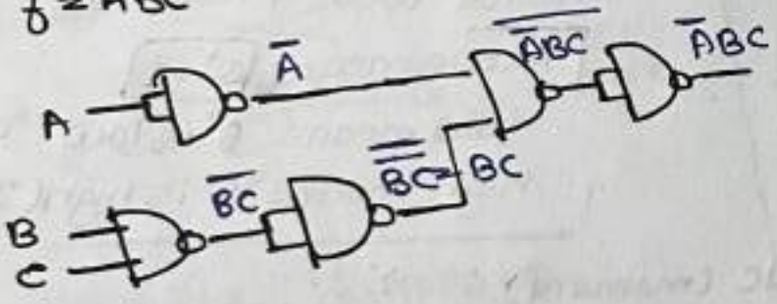
④ NOR gate required if $\bar{B}$ is given else ⑤



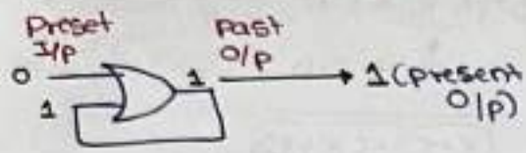
- NAND is friendly to SOP
- NOR is friendly to POS

(means's required min. no gate)

$f = \bar{A}BC$



• **Sequential circuit**

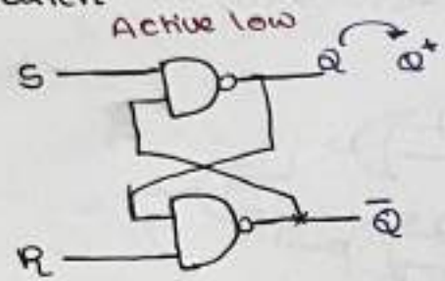


• the present o/p not only depend on present I/p, it also depend on past o/p

• Logic gate with feedback connection can be treated as sequential circuit ex: Latch's

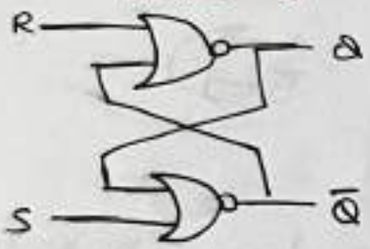
• Latch

Active low

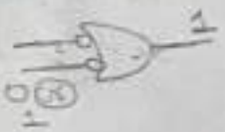


• Latch are 2 type
  - 1. NAND gate latch
  - 2. NOR gate latch.

Active High

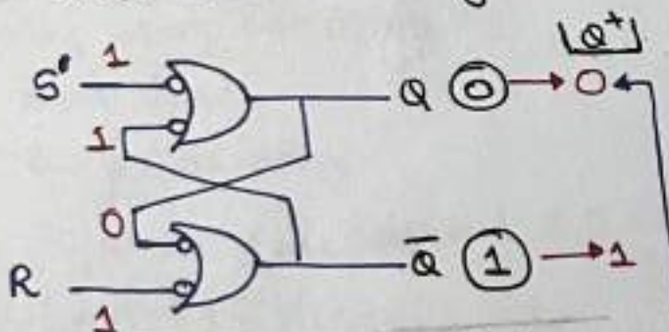


	S	R	Q^+	
Case 1:	1	1	NC	no change (memory)
Case 2:	1	0	0	Reset
Case 3:	0	1	1	Set
Case 4:	0	0		Don't Care (unused)

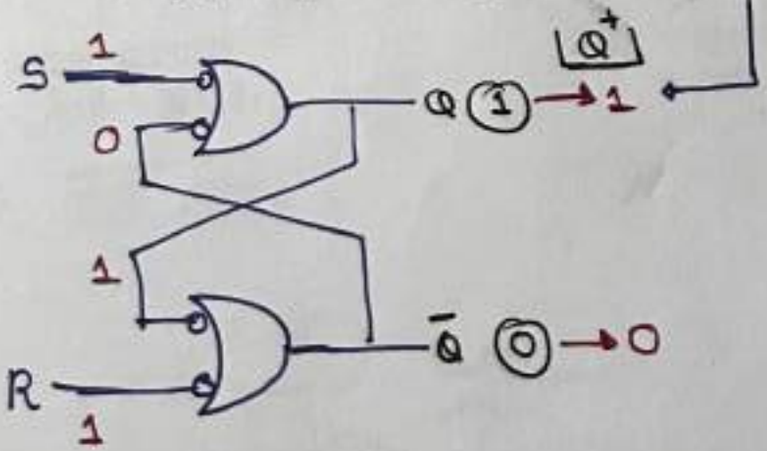


Case I

• S=1, R=1 Initially Q=0.



• S=1, R=1, Initially Q=1



• Active low ≠ Active high

Active means $Q^+ = 1$

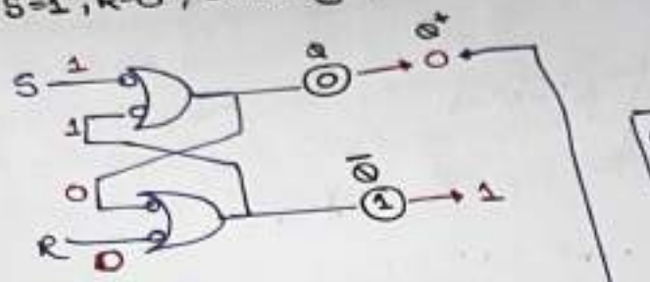
low means "S" is low ("0")

high means "S" is high ("1")

NC (memory)

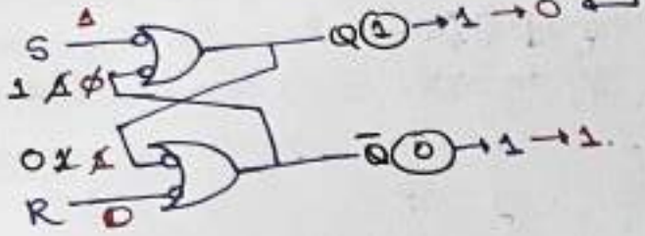
Case II

S=1, R=0, Initially Q=0.



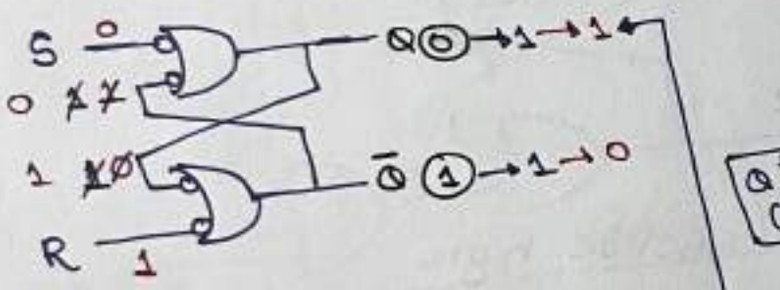
$Q^+ = 0$ (Reset)

S=1, R=0 Initially Q=1



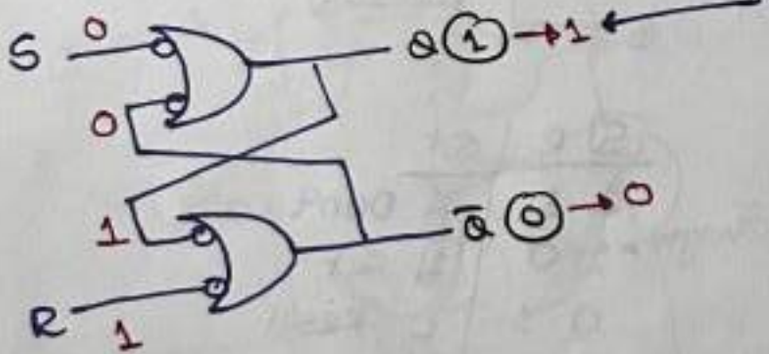
Case III

S=0, R=1 Initially Q=0.



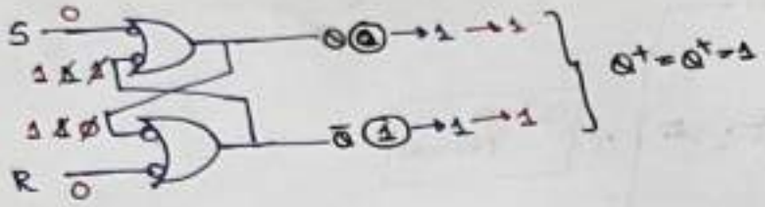
$Q^+ = 1$ (Set)

S=0, R=1, Initially Q=1.

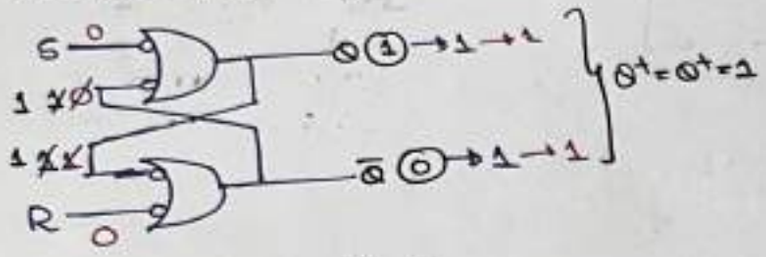


Case IV

- S=0, R=0, initially Q=0



- S=0, R=0, initially Q=1.

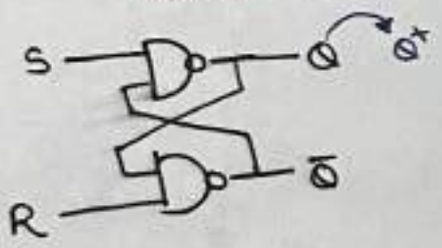


- Key point of operation.

- feedback connection should be operated until we get stable o/p

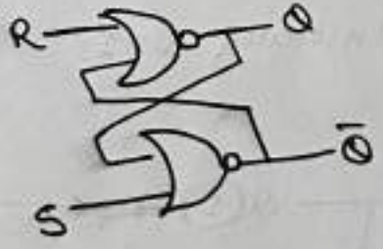
- apply the operation until you get new value. (unstable) once you the same value (stable value) then write in table.

Active low.



S	R	Q+
1	1	NC (memory)
1	0	0 Reset
0	1	1 set
0	0	Don't care (unused)

Active. high.



S	R	Q+
1	1	x Don't care
1	0	1 set
0	1	0 Reset
0	0	NC (memory)

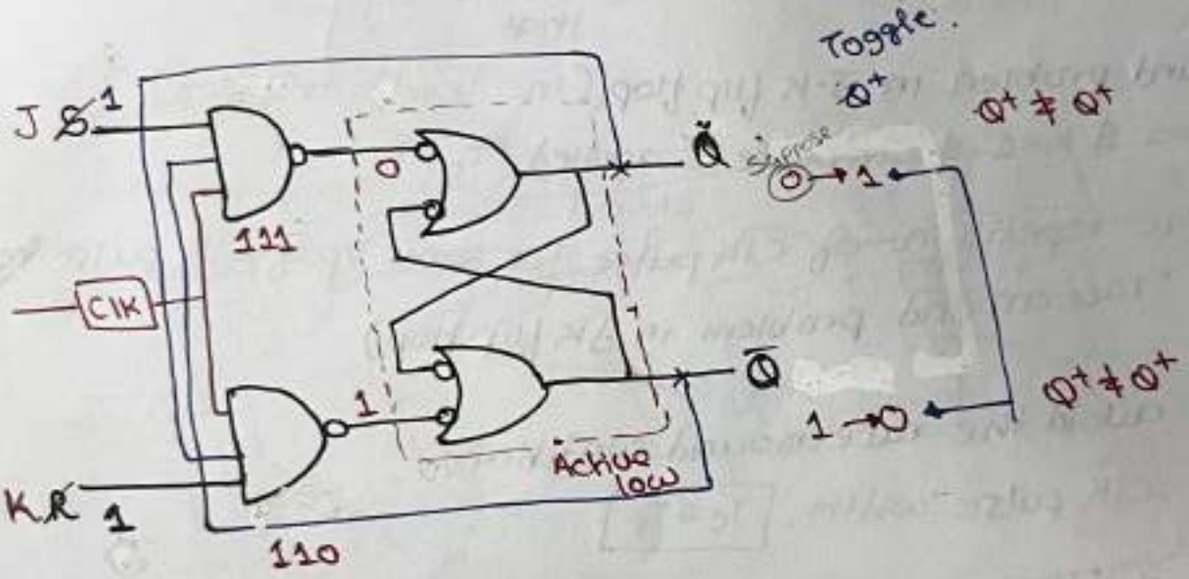
high

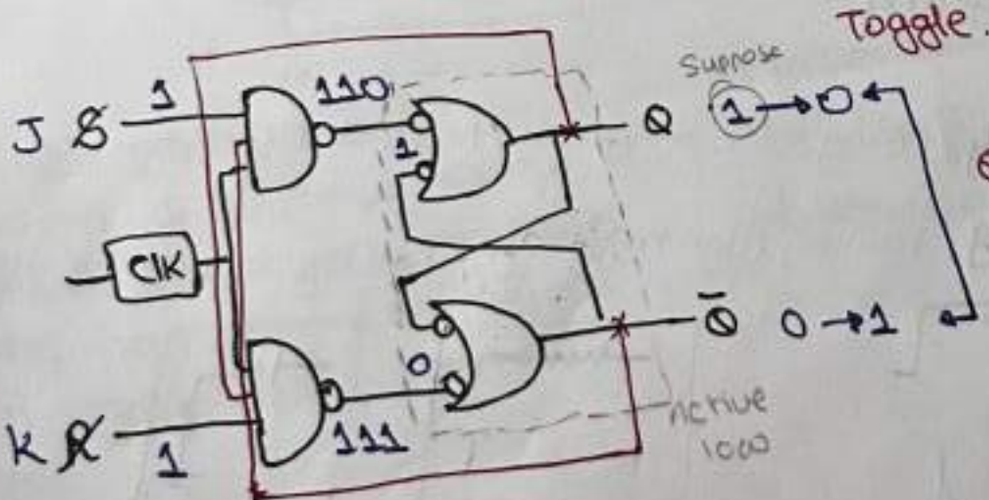
- The SR latch can be controlled with the help of the switch know as "gated SR latch"
- the controlling can also be done with the help of a clock generator. Knownas "clocked SR flip flop".

- J-K Flip-flop

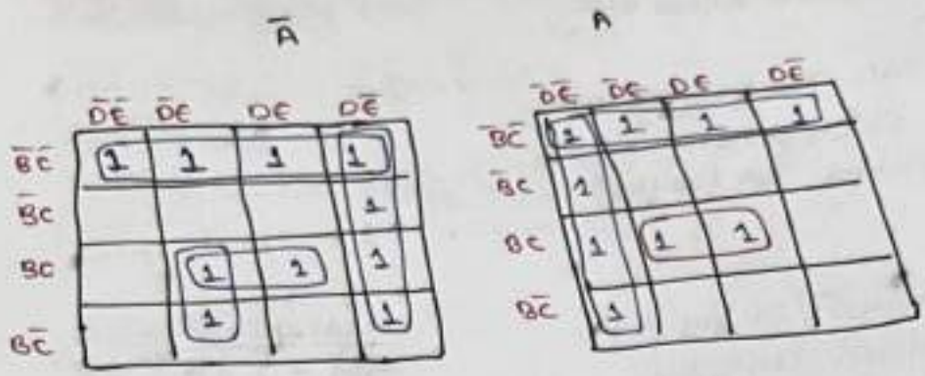
it is the modification of SR flip flop with external feedback connection. [when J=1 & K=1 & when Clk pulse is applied we get $Q^t = \bar{Q}$ then we get Toggle condition.

Clk	J	K	Q^t
⎍	1	1	$\bar{Q}$ (Toggle)
⎍	1	0	1 set
⎍	0	1	0 reset
⎍	0	0	N (memory)





• 5-variable K-map



(to check wheather the two pair or grp are matching or not in two different table then we draw expression & check wheather there is a difference of only one or not [not more than that]

ex: $\bar{A}D\bar{E}$, $AD\bar{E}$

• Race around problem in J-K flip flop. (in level trigger)

when J=1 & K=1 & clk pulse is applied $T_c > T_p$ (propagation time, clk pulse time)

we get the repetition of clk pulse for some Vp of clk pulse know as "race around problem in JK flip flop).

• Method to avoid the race around condition

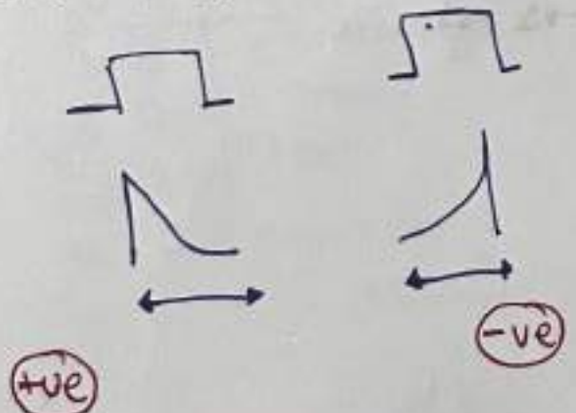
1. Reduce the clk pulse width. $T_c \leq T_p$

2. By using Edge triggering

↳ it is of 2 type.

I. (+ve) Edge triggering / Rising Edge triggering / Leading Edge triggering

II. (-ve) Edge triggering / Trailing Edge triggering / Falling edge triggering

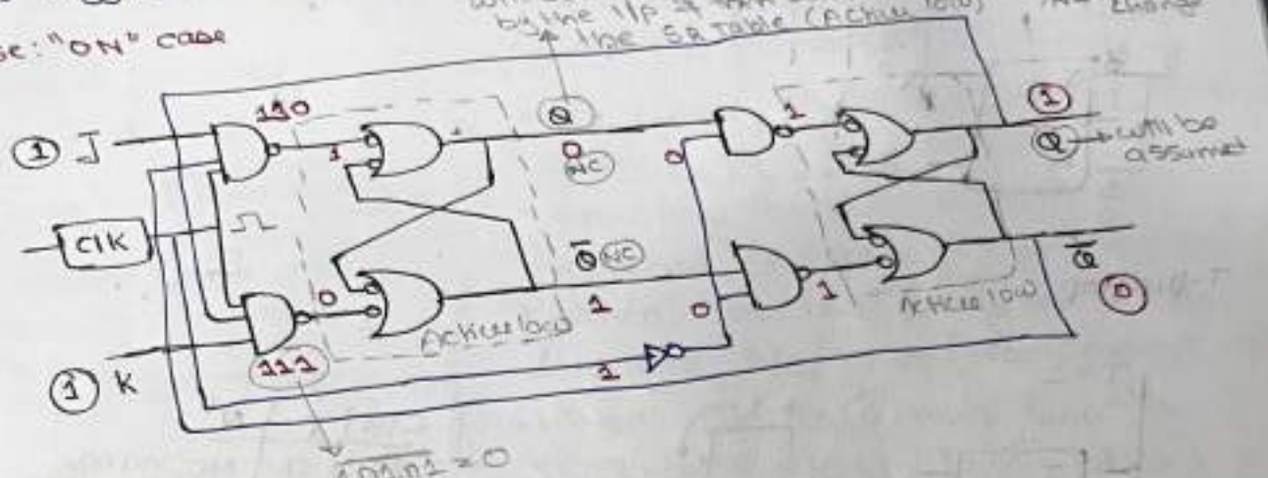


3. By using Master slave J-K flip-flop race around problem can be solved.

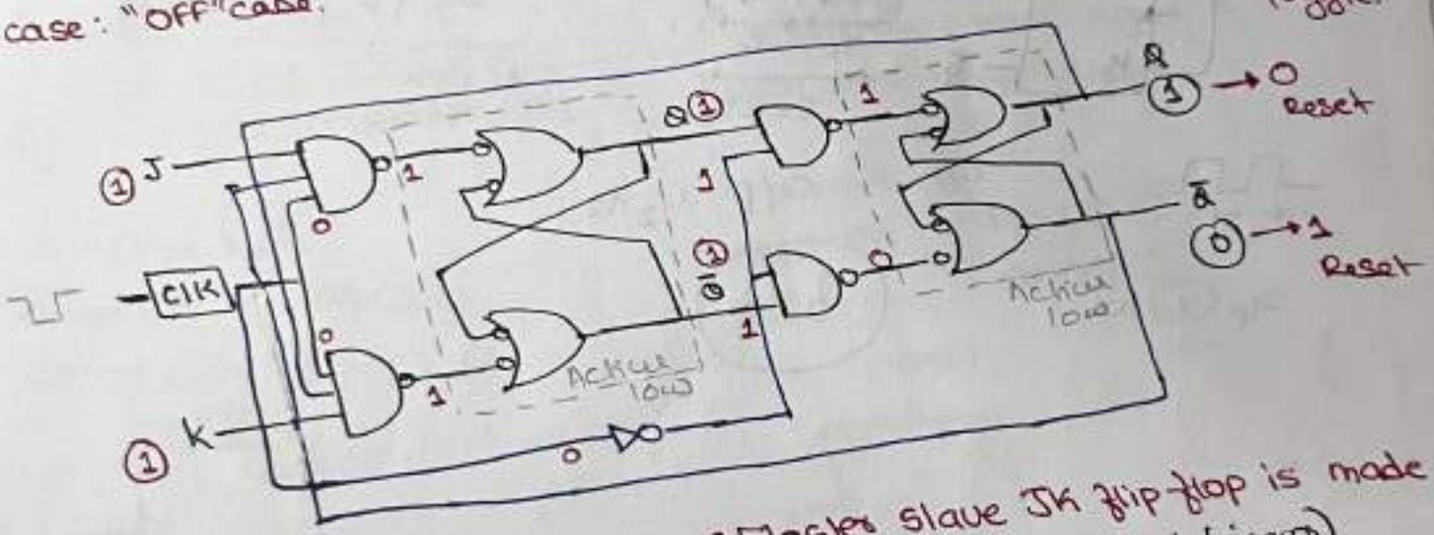
* key point

- when slave section is "ON" & toggle occurs. at that instance of time Master section is in the "off" state so there is no repetition of toggle & race around problem will be solved.

case: "ON" case



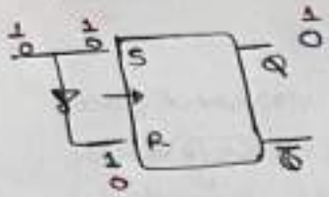
case: "OFF" case.



- Race condition when Q & $\bar{Q}$ are same then we can't predicate the next o/p & o/p depend on gate delay, not on I/p

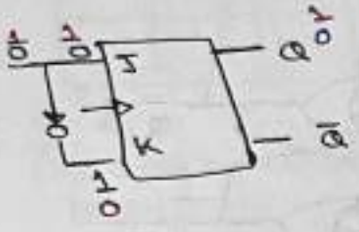
- Master slave JK flip-flop is made by (2) SR flip-flop (level trigger) but work as (1) JK (-ve) edge trigger

D-flip lop (Active high)

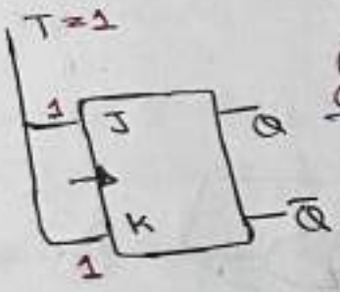


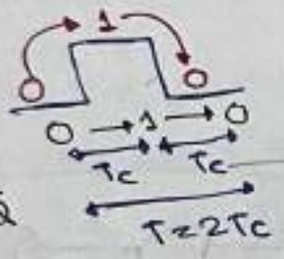
CK	D	Q^+
↑	0	0
↑	1	1

D- flipflop (Active high)



T-flipflop





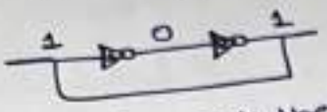
CLK	T	Q^+
↑	0	No change
↑	1	$\bar{Q}$ Toggle

I/p (2) Hz.

O/p 1 Hz.

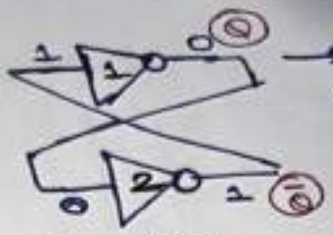
$f = \frac{1}{T}$

- 1 Bit Memory Cell



Latch: we can call these as latch as it can lock 1bit

redraw ⟺



→ Bi-state multivibrator.

i.e Latch (Cross Coupled NOT gate)

Clock

- Time & frequency

$T = \frac{1}{F}$ & $F = \frac{1}{T}$

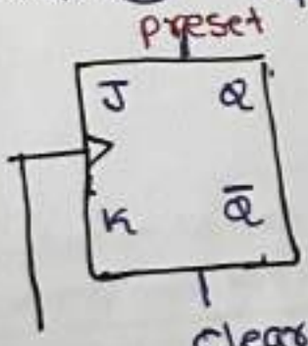
Time	Frequency
Sec	HRZ
MS(10^{-3})	KHRZ
µS(10^{-6})	MHRZ
nS(10^{-9})	GHRZ
PS(10^{-12})	THRZ

- Flip flop are know as Synchronˢ circuit bc change in o/p occur in synchronization with clk

i.e Active high

Preset (1) o/p(Q) (1)

Clear (1) o/p($\bar{Q}$) (0)



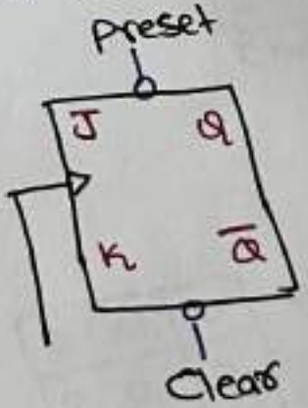
- In gated SR latch we can replace Enable pin by clk
- Resulting circuit is called as clocked SR latch (or) SR Flipflop
- when clk is (1) only then o/p changes A/c to I/p [transparent mode]
- when clk is (0) then $Q_{n+1} < Q_n$ regardless of I/p. [latch mode]

- Flip flop are level trigger & latch are edge trigger

i.e Active low

preset (0) o/p (Q) (1)

clear (0) op ($\bar{Q}$) (0)



• Characteristic table & equation

① JK flip-flop

	Q	J	K	Q^+	
0	0	0	0	0	→ NC
1	0	0	1	0	→ Reset
2	0	1	0	1	→ Set
3	0	1	1	1	→ $\bar{Q}$ toggle
4	1	0	0	1	→ NC
5	1	0	1	0	→ Reset
6	1	1	0	1	→ Set
7	1	1	1	0	→ $\bar{Q}$ Toggle.

Q \	$\bar{J}\bar{K}$	$\bar{J}K$	JK	$J\bar{K}$	
$\bar{Q}$			1	1	$J\bar{Q}$
Q	1			1	$\bar{K}Q$

$Q^+ = J\bar{Q} + \bar{K}Q$

Characters" equation.

② SR flip flop

	Q	S	R	Q^+	
0	0	0	0	0	→ NC
1	0	0	1	0	→ Reset
2	0	1	0	1	→ Set
3	0	1	1	x	→ Don't care
4	1	0	0	1	→ NC
5	1	0	1	0	→ Reset
6	1	1	0	1	→ Set
7	1	1	1	x	→ X don't care.

Q \	$\bar{S}\bar{R}$	$\bar{S}R$	SR	$S\bar{R}$
$\bar{Q}$			x	1
Q	1		x	1

$Q^+ = S + \bar{R}Q$.

③ D-flip-flop

Q	D	Q^+
0	0	0
0	1	1
1	0	0
1	1	1

Q \	$\bar{D}$	D
$\bar{Q}$		1
Q		1

$Q^+ = D$

are characteristic eqn of D-flip flop.

④ T flipflop

Q	T	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

Q \	$\bar{T}$	T
$\bar{Q}$	0	1
Q	1	0

$Q^+ = T\bar{Q} + \bar{T}Q$

$Q^+ = T \oplus Q$

Characteristic eqn of T flipflop

• Excitation tables or Transition tables

Q	Q^+	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

		S	R
Q	Q^+	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

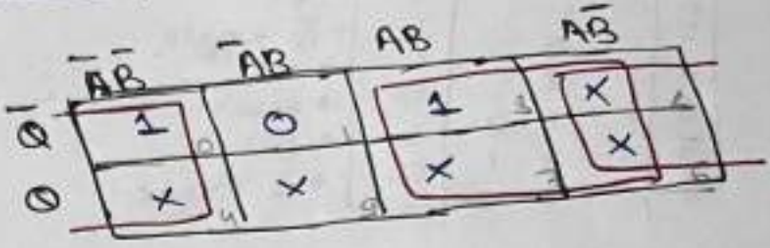
**.

- Designing of flip-flops
  - J-K to AB

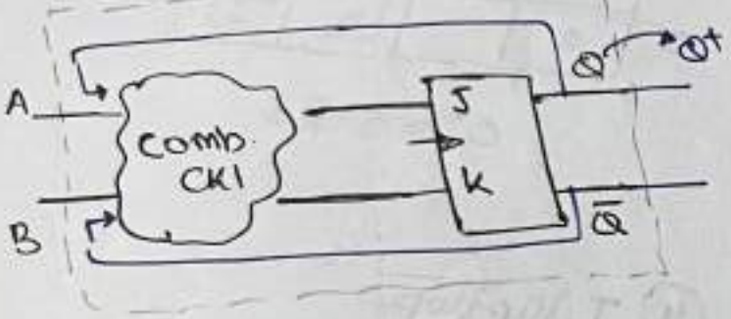
given

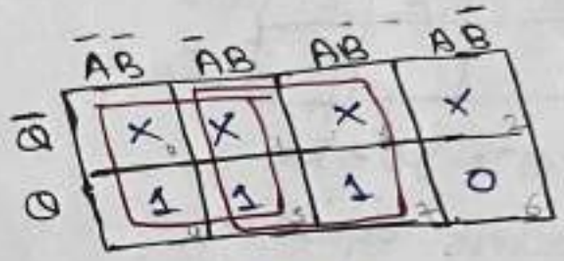
Q	A	B	Q^+	J	K
0	0	0	1	1	X
0	0	1	0	0	X
0	1	0	X	X	X
0	1	1	1	1	X
1	0	0	0	X	1
1	0	1	0	X	1
1	1	0	1	X	0
1	1	1	0	X	1

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

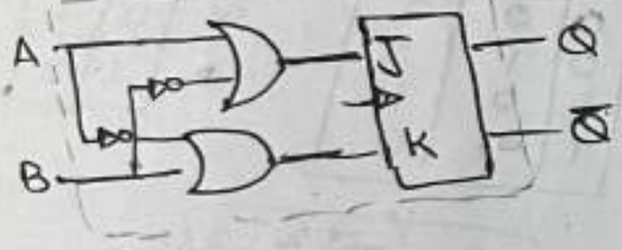


$J = A + B'$

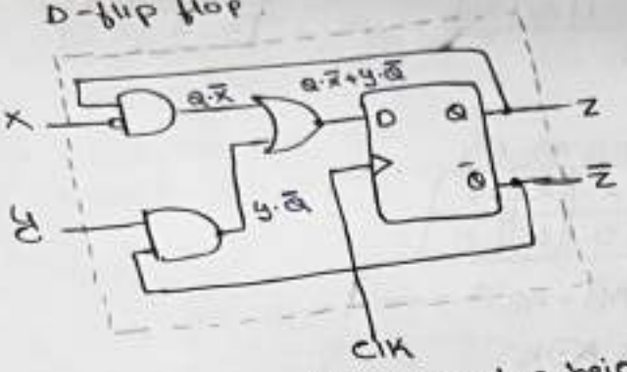




$K = \bar{A} + B$



6 A sequential circuit using D-flip flop



So, $D = z\bar{x} + \bar{z}y$

By option checking put $x=K$ & $y=J$ & $z=Q$

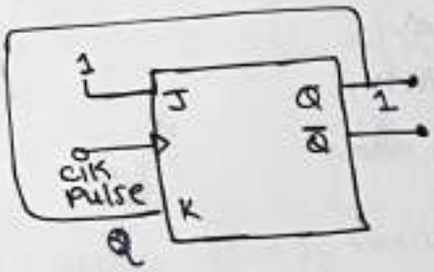
$D = z\bar{x} + \bar{z}y$

$D = z\bar{K} + \bar{z}J$

$D = Q^+ = J\bar{z} + \bar{z}K$

This is the charact^n eq^n of J-K F-F

④ initially $Q=1$ with clk pulse being given



So, $J=1$ & $K=0$ (from digram).

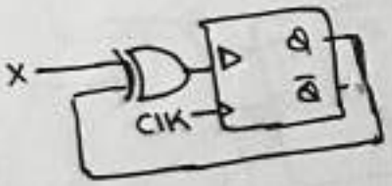
we know charact^n eq^n of JK FF.

$$Q^+ = J\bar{Q} + \bar{K}Q$$
$$= 1\cdot\bar{Q} + \bar{Q}Q$$

$\therefore Q^+ = \bar{Q}$ [Hence subsequent states will be complement of previous value]

Subsequent state: 0,1,0,1,0,1.....

⑤



we know the character equation of D FF

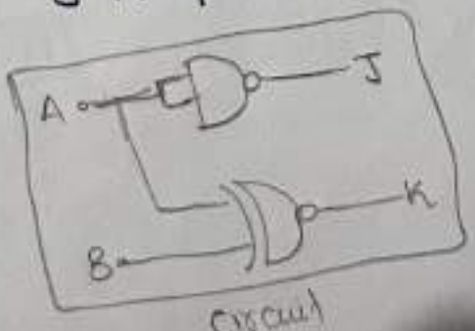
$\Rightarrow Q^+ = D$

$\Rightarrow Q^+ = D = x \oplus Q$ i.e T- flip flop.

⑦

A	B	Q_{n+1}
0	0	$\bar{Q}_n$
0	1	Q_n
1	0	Q_n
1	1	0

$J = \bar{A}$ & $K = A \odot B$.



Circuit

Q	A	B	Q^+	J	K
0	0	0	1	1	x
0	0	1	1	1	x
0	1	0	0	0	x
0	1	1	0	0	x
1	0	0	0	x	1
1	0	1	1	x	0
1	1	0	1	x	0
1	1	1	0	x	1

For J

	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{Q}$	1	1	0	0
Q	x	x	x	x

$J = \bar{A}$

For K

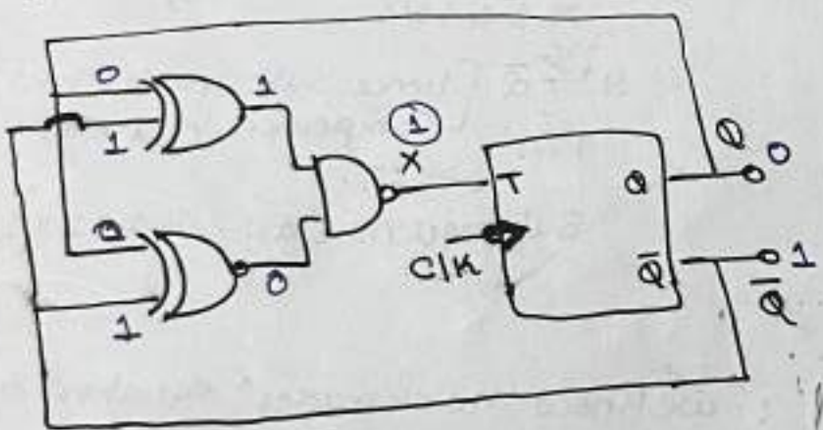
	$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{Q}$	x	x	x	x
Q	1	0	1	0

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

$K = A \odot B$

Q	Q^+	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

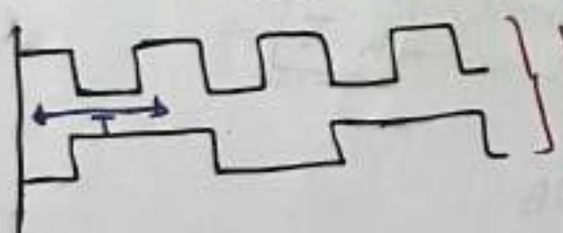
(10) The CLK frequency applied to the digital circuit shown in figure below is 1KHz. If the initial state of the o/p "Q" of the flip-flop is '0', then the frequency of the o/p waveform Q in kHz.



So, $x = 1$ & $x = (Q \oplus \bar{Q}) \cdot (\bar{Q} \oplus Q)$

If I/p is always "1" at T

then it's (-ve) edge triggered.



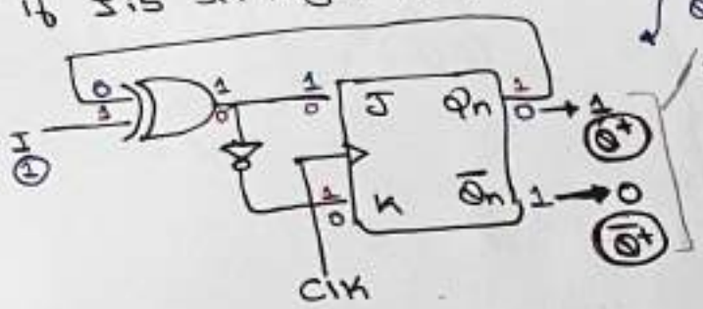
Hence obtained from waveform time period Q is doubled hothat of CLK. So, $f_Q = \frac{f_C}{2} = \frac{1}{2} = 0.5\text{KHz}$

(12) Latch constructed with NOR & NAND gate tend to remain in the latched condition due to which configuration feature is cross coupling.

b/c Latch is a type of bi-stable multivibrator having 2 stable states.

- Both the i/p of a Latch are directly connected to the other's o/p. Such structure is know as cross coupling.

(13) If J is set high then Q^+



Suppose $Q_n = 0$; then.

then $J=1$ & $K=0$.

then, it's a set condition.

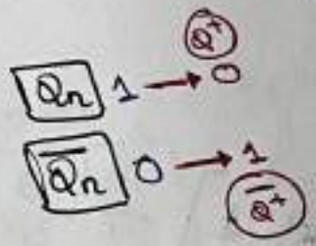
it means that

$Q = \overline{Q^+}$

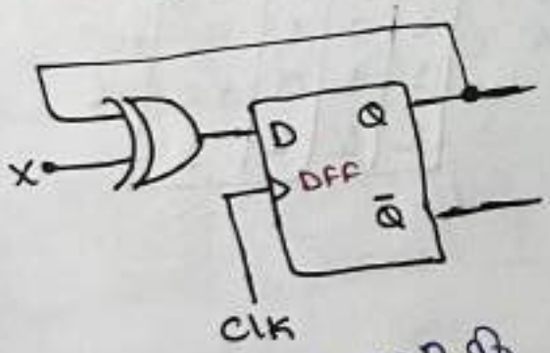
Suppose $Q_n = 1$; then

$J=0$ & $K=1$.

Still the condition $Q = \overline{Q^+}$ is satisfy.



(42)



- the charactn eqn of circuit will be.

$Q^+ = D$

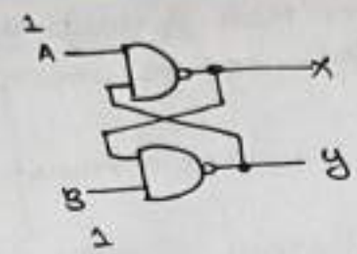
& $D = X \oplus Q$

so, $Q^+ = X \oplus Q$

(T1) initially

A = 1 & B = 1

↓ replaced by 1010101....



then o/p x & y?

(T3) A/c to question

both x=y then

[1,1 then set
  else Reset &
  0,1 then compln
  else No change]

Q	x	y	Q^+	
0	0	0	0	
0	0	1	$\bar{Q}$	→ 1
0	1	0	NC	→ 0
0	1	1	1	
1	0	0	0	
1	0	1	$\bar{Q}$	→ 0
1	1	0	NC	→ 1
1	1	1	1	

	$\bar{x}\bar{y}$	$\bar{x}y$	xy	$x\bar{y}$
$\bar{Q}$	0	1	1	0
Q	0	0	1	1

So, $Q^+ = Qx + \bar{Q}y$

• Flip-flop conversation

① given x y flip flop

↳ funct^n table

x	y	Q^+
0	0	$\bar{Q}$
0	1	$\bar{Q}$
1	0	Q
1	1	0

→ Q_{n+1}

→ Q_n

• Truth table

x	y	Q	Q^+
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

• charact^n equation

$Q^+ = f(x,y,Q)$

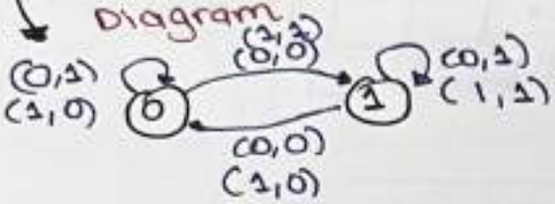
$= \Sigma m(0,3,6,7)$

Q \ xy	00	01	11	10
0	1		1	
1		1	1	

$Q^+ = \bar{x}\bar{y}\bar{Q} + xy + yQ$

From

• State transition Diagram



• Excit^n table

Q	Q^+	x	y
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

⑪ given: funct^n table

LM Flipflop

L	M	Q^+
0	0	0
0	1	0
1	0	$\bar{Q}$
1	1	$\bar{Q}$

• Truth table

L	M	Q	Q^+
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

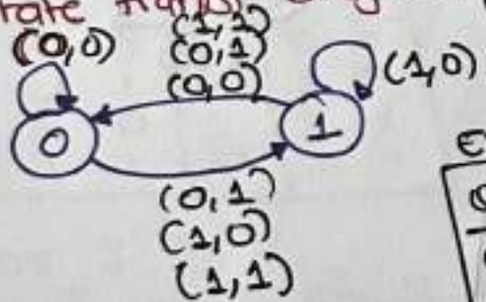
• charact^n equat^n

$Q^+ = \Sigma m(2,3,5,6)$

Q \ LM	00	01	11	10
0		1	1	1
1				1

$Q^+ = M\bar{Q} + L\bar{M}$

• State trannst^n diagram



• Excit^n table

ET_1

Q	Q^+	L	M
0	0	0	0
0	1	x	1
1	0	0	x
1	1	1	0

(56)

ET_2

Q	Q^+	L	M
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

(58)

ET_3

Q	Q^+	L	M
0	0	0	x
0	1	1	x
1	0	x	0
1	1	x	1

ET_4

Q	Q^+	L	M
0	0	x	0
0	1	x	1
1	0	0	x
1	1	1	x

- Flip flop conversion

(i) Convert SR FF to JK FF & visa versa

given FF ⇒ Desired FF

- Exitn table

Q	Q+	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

- Truth table of JK

J	K	Q	Q+	S	R
0	0	0	0	0	x
0	0	1	1	x	0
0	1	0	0	0	x
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	x	0
1	1	0	1	1	0
1	1	1	0	0	1

R

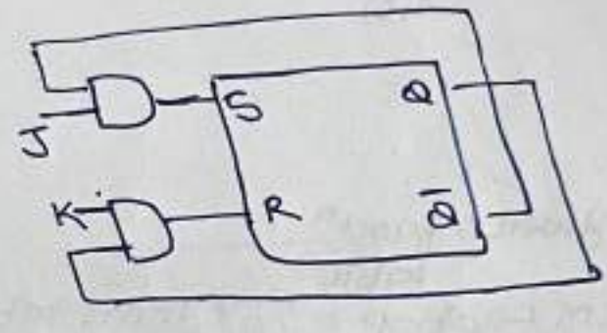
Q \ JK	00	01	11	10
0	x	x		
1		1	1	

R = KQ

S as in term of JKQ

Q \ JK	00	01	11	10
0			1	1
1	x			x

S = JQ̄



(ii) Construct D FlipFlop from SR FlipFlop

- Truth table of Desired FF

D	Q	Q+	S	R
0	0	0	0	x
0	1	0	0	1
1	0	1	1	0
1	1	1	x	0

S

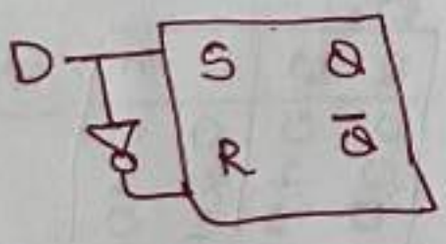
Q \ D	0	1
0		1
1		x

S = D

R

Q \ D	0	1
0	x	
1	1	

R = D̄

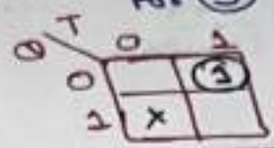


(iii) Construct T FF from SR FF

- Truth table

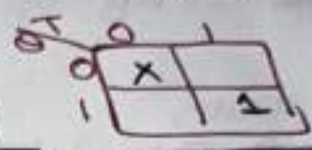
T	Q	Q^+	S	R
0	0	0	0	x
0	1	1	x	0
1	0	1	1	0
1	1	0	0	1

For (S)

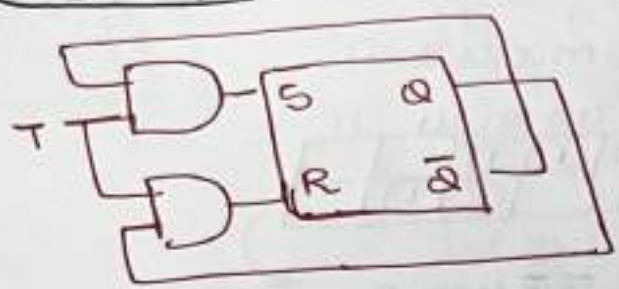


$S = T\bar{Q}$

For (R)



$R = TQ$



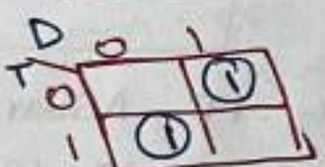
(iv) Construct TFF from D FF

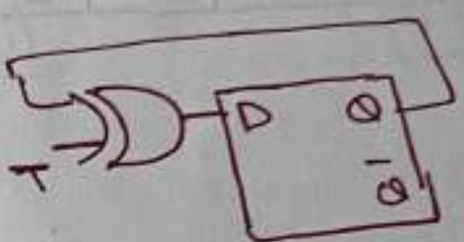
- Truth table

T	Q	Q^+	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

$D = \bar{T}Q + T\bar{Q}$

$D = T \oplus Q$

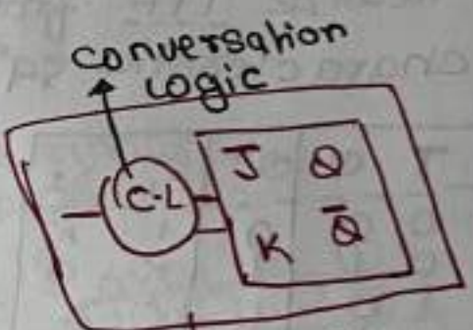


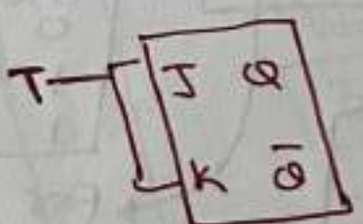


(v) Construct TFF from JK FF.

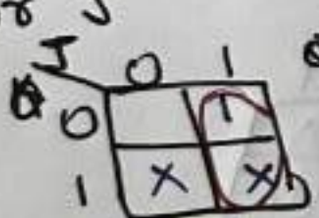
Desired — given

T	Q	Q^+	J	K
0	0	0	0	x
0	1	1	x	0
1	0	1	1	x
1	1	0	x	1

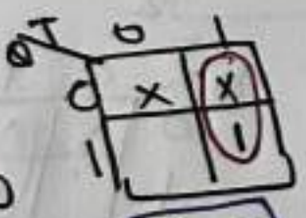




For J



$J = T$



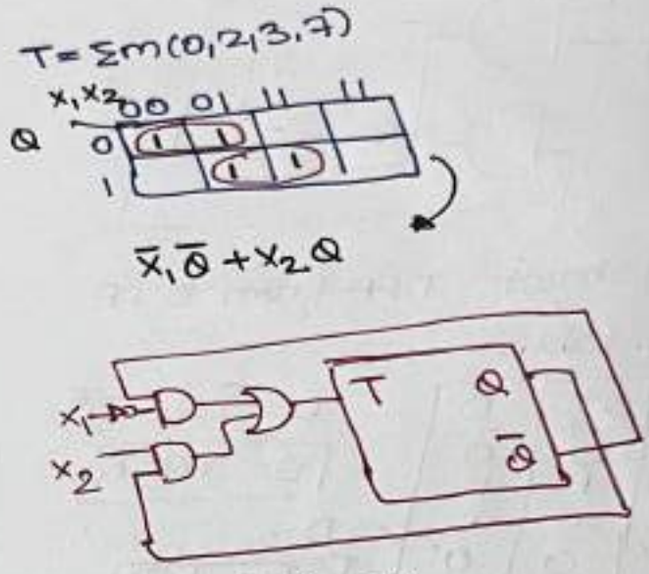
$K = T$

Q. Characteristic eqn of a FF is $Q^+ = \bar{x}_1\bar{Q} + \bar{x}_2 \cdot Q$ (desired)
Realize it by by T FF (given)

K-map

(1st) draw Truth table of Desired one

x_1	x_2	Q	Q^+	T
0	0	0	1	1
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	0	1

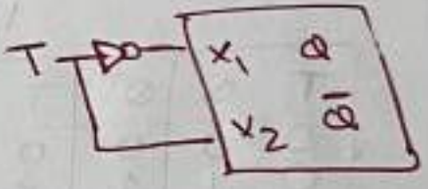


(2) Realize T FF from x_1, x_2 FF. Assume (desired) characteristic eqn of x_1x_2 FF is $Q^+ = \bar{x}_1\bar{Q} + \bar{x}_2Q$ (given)

T	Q	Q^+	x_1	x_2
0	0	0	1	x
0	1	1	x	0
1	0	1	0	x
1	1	0	x	1

• Excitn table

Q	Q^+	x_1	x_2
0	0	1	x
0	1	0	x
1	0	x	1
1	1	x	0



• Truth table

x_1	x_2	Q	Q^+
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

for x_1

Q \ T	0	1
0	1	
1	x	x

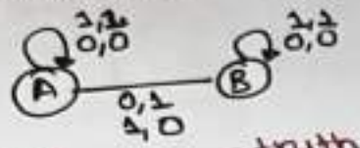
$x_1 = \bar{T}$

for x_2

Q \ T	0	1
0	x	x
1		1

$x_2 = T$

Q. State transition diagram of xy FF is.. Realize it using JK FF
(Desired) (given)



• Funct^n table

x	y	Q^+
0	0	Q
0	1	$\bar{Q}$
1	0	$\bar{Q}$
1	1	Q

⇒

• truth table

x	y	Q	Q^+	J	K
0	0	0	0	0	x
0	0	1	1	x	0
0	1	0	1	1	x
0	1	1	0	x	1
1	0	0	1	1	x
1	0	1	1	x	0
1	1	0	0	0	x
1	1	1	0	x	1

For J

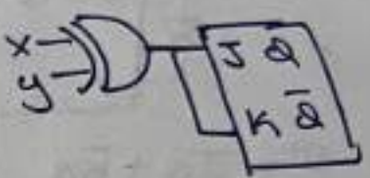
Q \ xy	00	01	11	10
0		1		1
1	x	x	x	x

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

For K

Q \ xy	00	01	11	10
0	x	x	x	x
1		1		1

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



Q. Construct AB FF from TFF
(Desired) (given)

A	B	Q^+
0	0	0
0	1	$\bar{Q}$
1	0	Q
1	1	$\bar{Q}$

• truth table of Desired flipflop

A	B	Q	Q^+	T
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
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

→ T in term of ABQ

Q \ AB	00	01	11	10
0		1	1	1
1	1	1		

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

Ques

this Flipflop is (D)

$Q^+ = T \oplus Q$

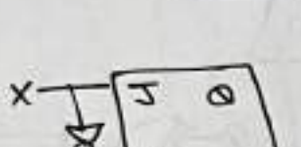
$= 2 \oplus Q \oplus Q$

$x \oplus 0$

$= x$

$Q^+ = x$

ex:



$Q^+ = J\bar{Q} + \bar{K}Q$

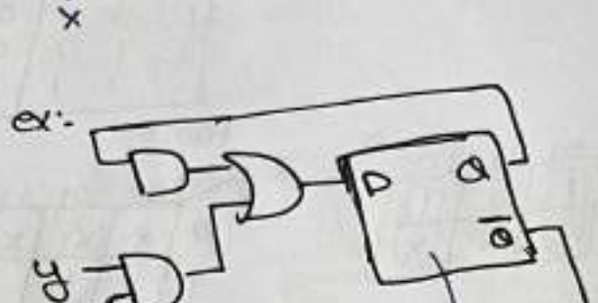
$x\bar{Q} + \bar{\bar{x}}Q$

$x\bar{Q} + xQ$

$x(Q + \bar{Q})$

$Q^+ = x$ D flip flop

ex:



$Q^+ = D$

$Q^+ = \bar{x}Q + y\bar{Q}$

$Q^+ = J\bar{Q} + \bar{K}Q$

instead of D flipflop we replace it with T (FF)

$Q^+ = T \oplus Q$

$= (xQ + y\bar{Q}) \oplus Q$

$= (xQ + y\bar{Q}) \cdot \bar{Q} + \overline{(xQ + y\bar{Q})} \cdot Q$

$y\bar{Q} + \overline{xQ} \cdot \overline{y\bar{Q}} \cdot Q$

$y\bar{Q} + (\bar{x} + \bar{Q}) \cdot (\bar{y} + Q) \cdot Q$

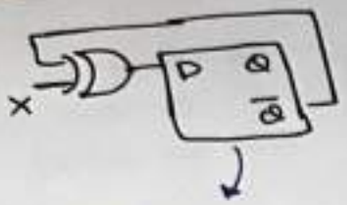
$y\bar{Q} + (\bar{x} + \bar{Q})(\bar{y}Q + Q)$

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

$Q^+ = y\bar{Q} + \bar{x}Q$

$Q^+ = J\bar{Q} + \bar{K}Q$

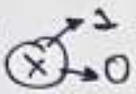
ex:

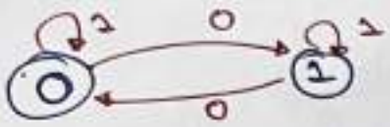


$Q^+ = D$

$= x \oplus \bar{Q}$

$Q^+ = x \odot Q$





$Q^+ = x \odot Q$

$0 \odot 0 = 1$

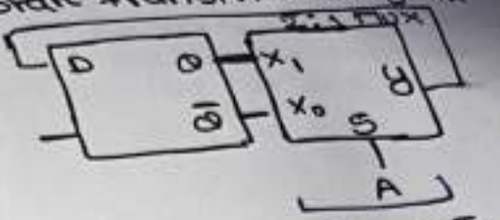
$1 \odot 0 = 0$

$Q^+ = x \odot Q$

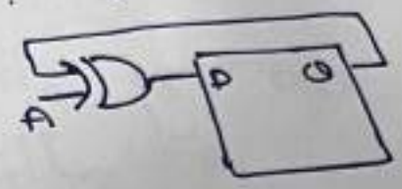
$0 \odot 1 = 0$

$1 \odot 1 = 1$

ex: State transition diagram

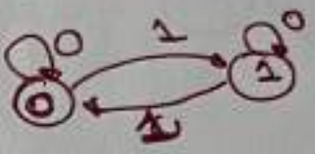


$\bar{A}Q + A\bar{Q}$

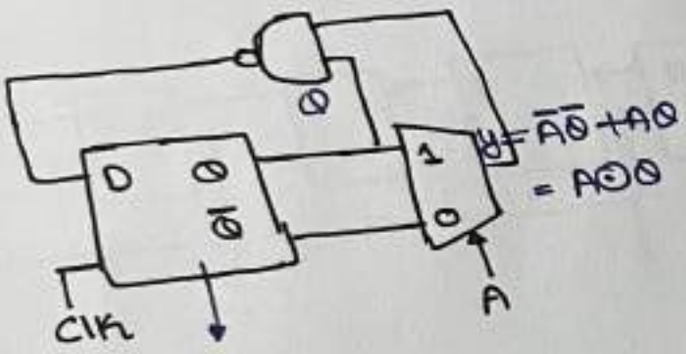


$Q^+ = D$

$Q^+ = A \oplus Q$



ex:



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

$= A \odot Q$

$Q^+ = D$

$= \overline{Q \cdot A \odot Q}$

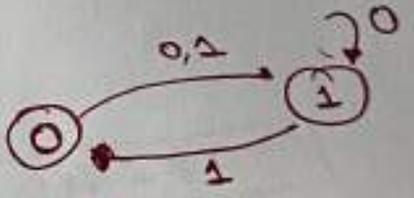
$Q^+ = \bar{Q} + A \oplus Q$

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

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

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

$Q^+ = \overline{Q \cdot A}$



▸ $Q^+ = \overline{Q \cdot A}$

Q

▸ $Q^+ = \overline{Q \cdot A}$

$1 \cdot 1$

$\Rightarrow 0$

& $\overline{Q \cdot A}$

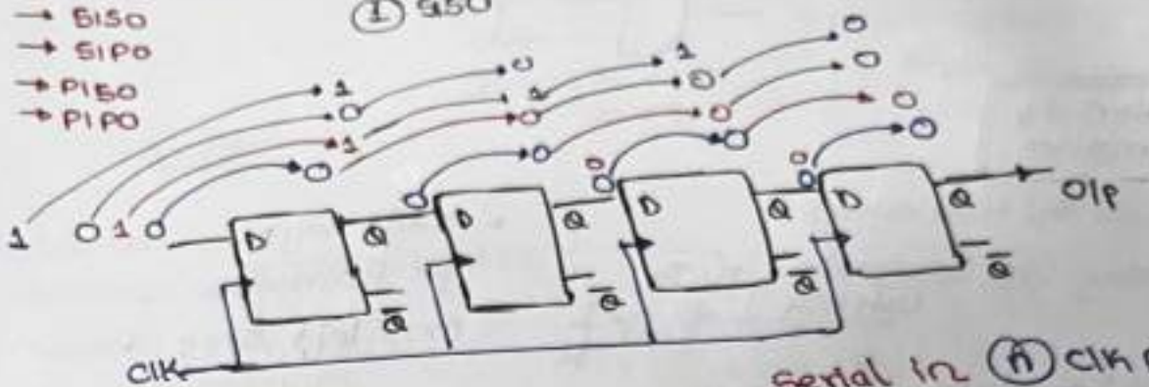
$1 \cdot 0$

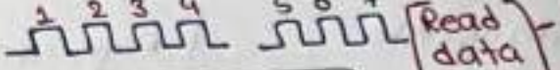
$\Rightarrow 1$

- shift register
- flip flop connected in the cascaded manner } easy to design
- only D-flip flop are used
- 4 type of shift register
  - → SISO
  - → SIPO
  - → PISO
  - → PIPO

- Counter is also connected in cascaded manner but are not easy to design

① SISO

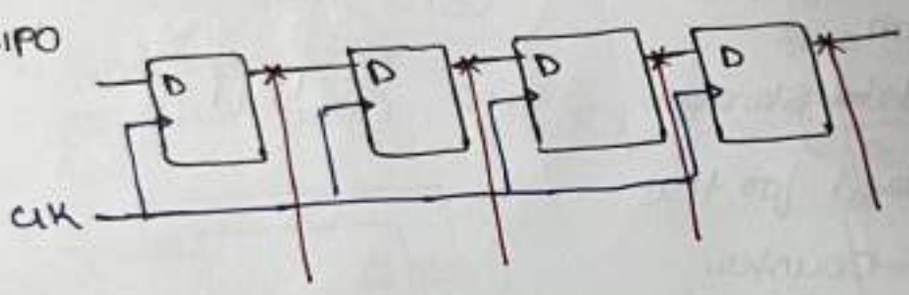




Serial in (n) clk pulse
Serial out (n-1) clk pulse

SISO → n $\updownarrow$ (n-1)

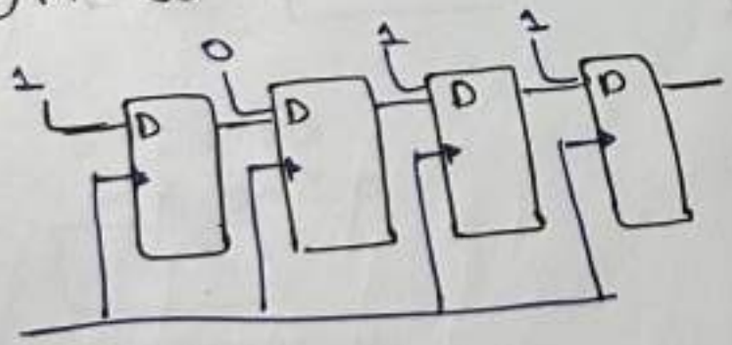
② SIPO



SIPO → n $\updownarrow$ 0

n for I/p
$\updownarrow$ 0 for O/p
b/c its parallel

③ PI SO

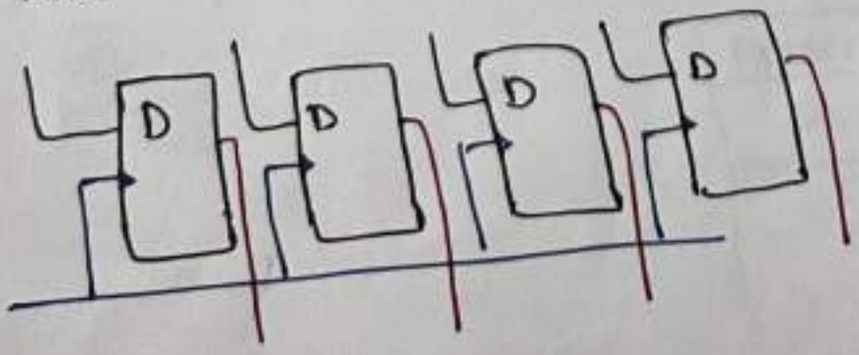


PISO ⇒ 1 $\updownarrow$ (n-1)

1 for I/p as it's parallel from we need to put clock on it
$\updownarrow$ (n-1) for O/p

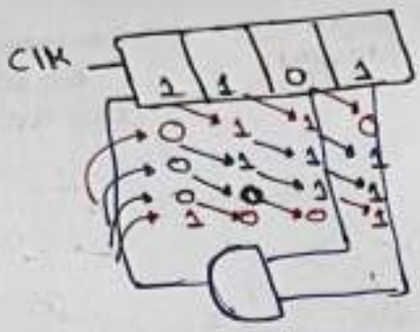
④ PIPO

PIPO → 1 $\updownarrow$ 0



•After 3 clk pulse?

CLK	1	1	0	1
1	0	1	1	0
2	0	0	1	1
3	1	0	0	1



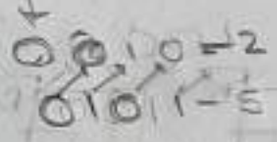
Application of Shift register

- It provide the time delays
  Tmax delay ⓐ longest latency } n·To ⇒ n·1/fc
- It is used for data conversion
  ex: SIPO convert Serial form of data into parallel form &
  PISO convert parallel form of data into Serial form
- used for temporary data storage.
- Shift register are used for the designing of Ring counter.

note:

- Right Shift Register have an error of (0.5) for odd. no.

- Shift register are used for Arithmetic operation
  ex: left shift Register is Multiplication of 2 circuit.
  & Righ shift Register is Division by 2 circuit



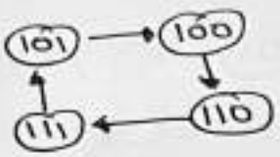
but 5/2 ⇒ 2.5
so (0.5) is error

• State of counter o/p of all flip flops taken together is called state

• Counters are used to count clock pulse by jumping from one state to another

• mod of counter → no. of states in counting sequence of counter

ex.



3FF
& mod 4 counter.

ex: to create 2^m counter then (m) flip flop is required

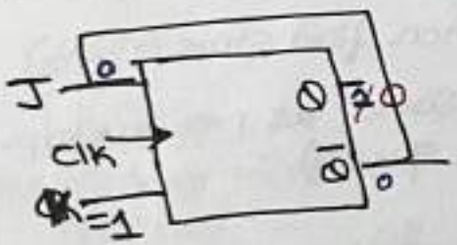
ex. mod 16 counter require 4FF.

Counter

Synchronous
- Same clk
- Easy to made
- Random sequence can be made

ASynchronous
- Different clk
- difficult to Design

ex if present state is (1) then find next state. (After 1 clk pulse) so, o/p is (1)



$Q^+ = JQ̄ + K̄Q$

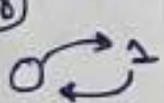
$= Q̄ \cdot Q̄ + \bar{1} \cdot Q$

$Q^+ \Rightarrow Q̄$

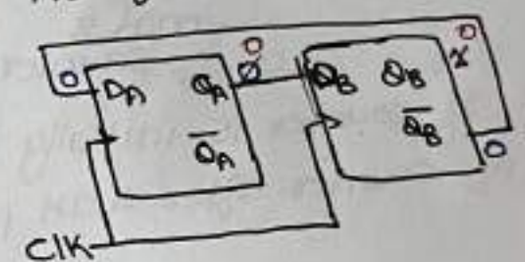
counting sequence

$0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \ldots$

(01)



ex: if present state $Q_A Q_B$ is 01 then find nxt state



Q_A Q_B

0 1

0 0 → nxt state

$D_A \rightarrow \bar{Q}_B$

$D_B \rightarrow Q_A$

$Q^+ = D$

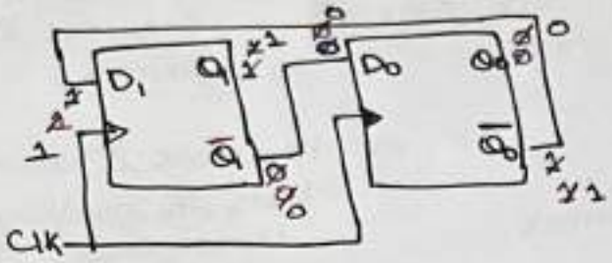
$\Rightarrow$

$Q_B^+ = Q_A$

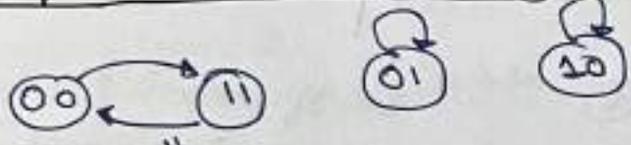
$Q_A^+ = \bar{Q}_B$

Present State		next State	
Q_A	Q_B	$Q_A^+ (\bar{Q}_B)$	$Q_B^+ (Q_A)$
0	1	0	0
0	0	1	0
1	0	1	1
1	1	0	1

ex: If present state Q_0Q_1 is 01 then find state of ckt after 2 clk pulse



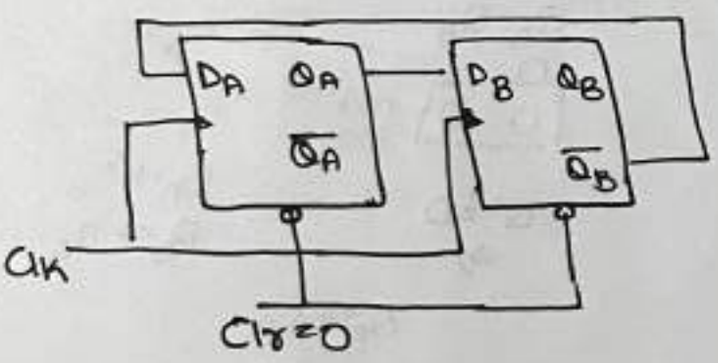
Present		next	
Q_0	Q_1	$Q_0^+(\bar{Q}_1)$	$Q_1^+(\bar{Q}_0)$
0	0	1	1
0	1	0	1
1	0	1	0
1	1	0	0



mod 2 counter.

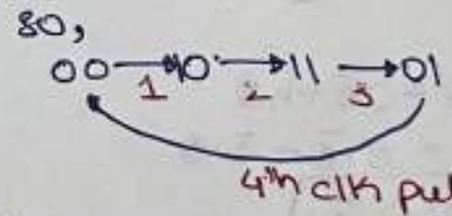
ex: If counter is initially cleared then find state (Q_AQ_B) of counter after 4 clk pulse.

(Same as previous to previous question)

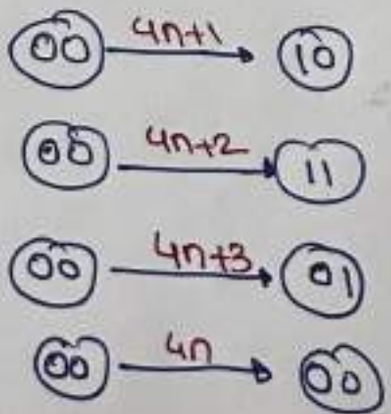


$Q_A = D_A = \overline{Q_B}$

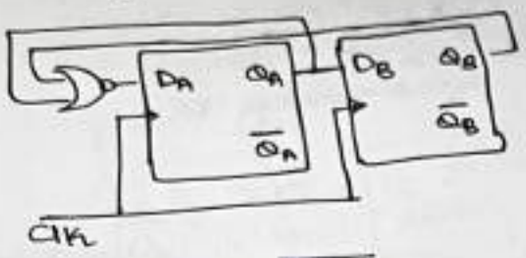
So,



when you start with 00
1, 5, 9... clk pulse $(4n+1 / n \geq 0)$



ex: If counter is initially cleared then find mod no. of counter



Q_A	Q_B	Q_A^+ $(\bar{Q}_A)$	Q_B^+ (Q_A)
0	0	1	0
1	0	0	1
0	1	0	0

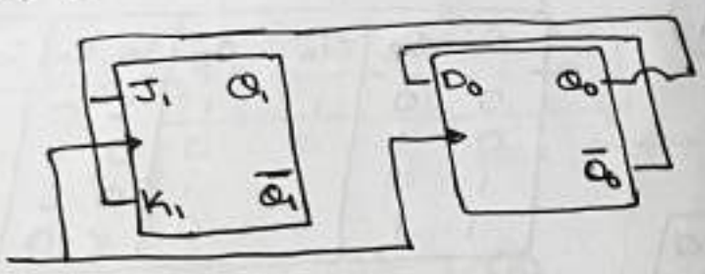
00 → 10 → 01

So, mod 3 counter

$Q_A = D_A = \overline{\bar{Q}_A + Q_B}$

$Q_B = D_B = Q_A$

ex: If present state Q_0Q_1 is 01 then find mod no.



Q_0	Q_1	Q_0^+ $(\bar{Q}_0)$	Q_1^+ $(Q_0 \oplus Q_1)$
0	1	1	1
1	1	0	0
0	0	1	0
1	0	0	1

01 → 11 → 00 → 10

So, mod 4 counter

$Q_0 = D_0 = \bar{Q}_0$

$Q_1 = J_1\bar{Q}_1 + \bar{K}_1Q_1$

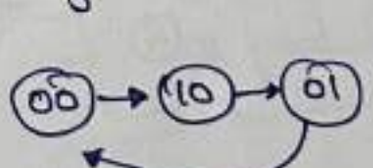
$Q_0\bar{Q}_1 + \bar{Q}_0Q_1$

$\boxed{Q_0 \oplus Q_1}$

Same questn but initially state ex is not given

Q_A	Q_B	Q_A^+ $(Q_A \oplus Q_B)$	Q_B^+ (Q_A)
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

⇒ State transition diagram



used state

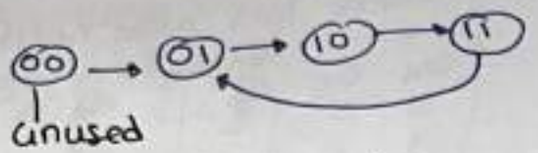
[mod 3 counter]

unused state

lock out problem

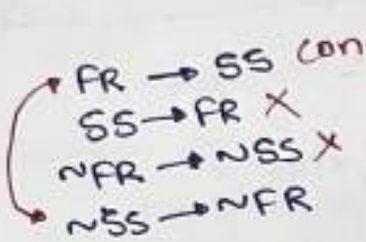
- Whenever a counter have lockout problem then it's not a self starting counter
- Free running: means every state should be used or present i.e in these case it should be mod 4 counter but it's not

i.e



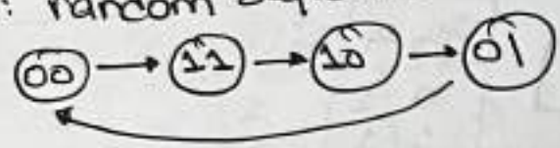
mod 3 counter

So it's self starting (SS)
but not free running (FR)

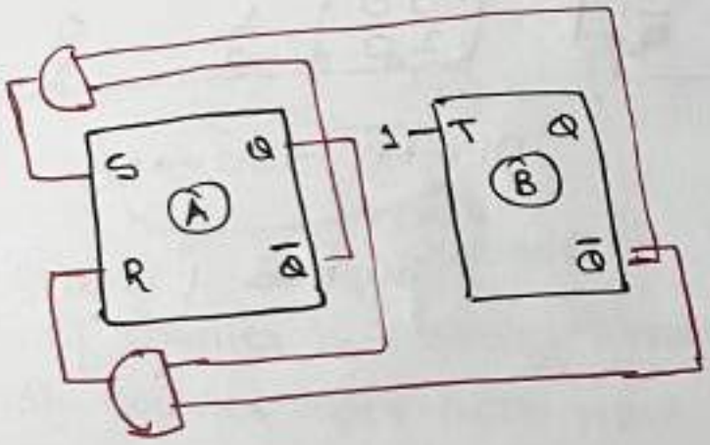


- up counter where we count in up direction
- down counter where we count in reverse direction.

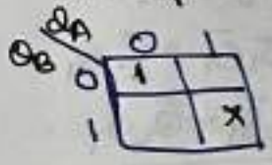
ex: random sequence



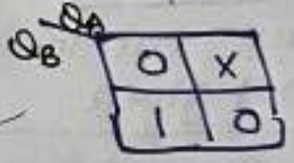
Q_A	Q_B	Q_A^+	Q_B^+	S_A	R_A	T_B
0	0	1	1	1	0	1
0	1	0	0	0	x	1
1	0	0	1	0	1	1
1	1	1	0	x	0	1



K-map for S

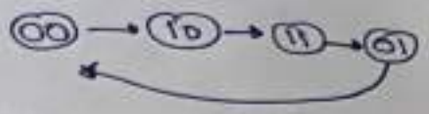


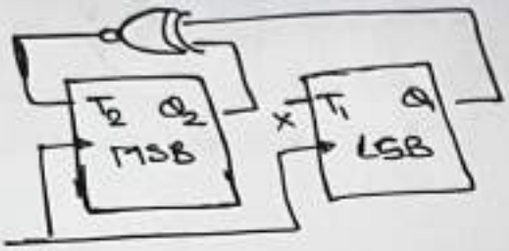
For R



gate 2004

ex: Partial implementation of a 2-bit counter using T-flip-flop following the sequence 0-2-3-1-0,



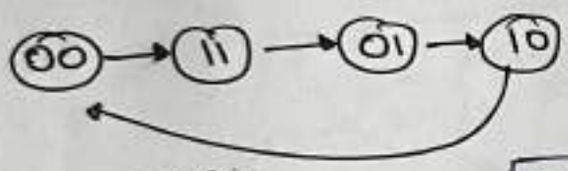


$SO, x = Q_1 \oplus Q_2$

Q_2	Q_1	Q_2^+	Q_1^+	T_2	T_1
0	0	1	0	1	0
0	1	0	0	0	1
1	0	1	1	0	1
1	1	0	1	1	0

XNOR XOR

Imp ex:



MSB → TFF

LSB → x y FF

to fill these we need excitn table of xy

Q_A	Q_B	Q_A^+	Q_B^+	T_A	x_B	y_B
0	0	1	1	1	1	X
0	1	1	0	1	X	0
1	0	0	0	1	0	X
1	1	0	1	1	X	1

• Functn table

given ff x y

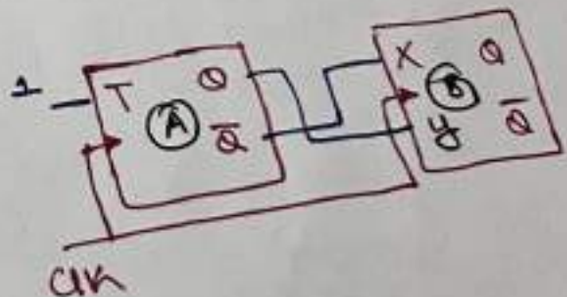
x	y	Q_n
0	0	0
0	1	Q_n
1	0	$\overline{Q_n}$
1	1	1

• truth table

x	y	Q	Q^+
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

• Excitn table

Q	Q^+	x	y
0	0	0	X
0	1	1	X
1	0	X	0
1	1	X	1

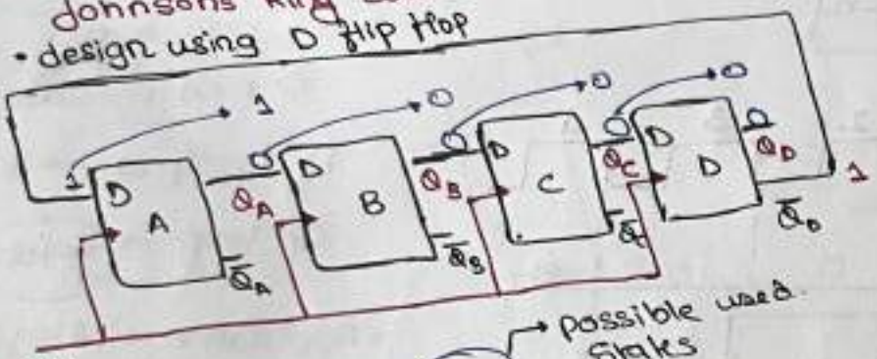


Counters

- It count no. of clk pulse applied to it.
- it consists of flip flop connected in the cascaded manner.
- Ring counter are special category of synchronous counter only.

- counters are of 2 type
  - Synchronous (parallel)
  - Asynchronous (Ripple)

- Ring Counter
- There are of 2 type.
  1. Twisted Ring counter or johnsons Ring counter.
     - also know as
       1. mobies
       2. walking
       3. switch tail
       4. Creeping
     - counter
  - design using D flip flop



clk	Q_A	Q_B	Q_C	Q_D	
0	1	0	0	0	:8
1	1	1	0	0	:12
2	1	1	1	0	:14
3	1	1	1	1	:15
4	0	1	1	1	:7
5	0	0	1	1	:3
6	0	0	0	1	:1
7	0	0	0	0	:0
8	1	0	0	0	X

(this sequence can start from any future)

$n \rightarrow 2 \cdot n$ → possible used states; → clk pulse.

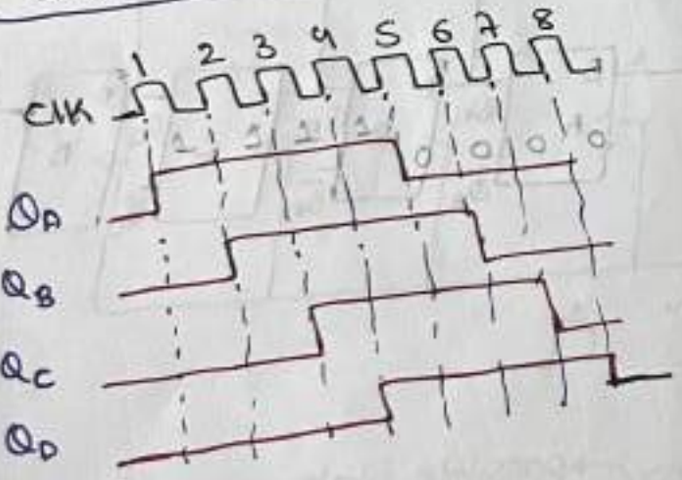
unused state → next state.

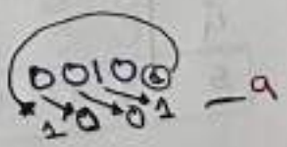
- 2 → 9
- 4 → 10
- 5 → 2
- 6 → 11
- 9 → 4
- 10 → 13
- 11 → 5
- 13 → 6

mod 8

used state: 8 - 12 - 14 - 15 - 7 - 3 - 1 - 0

unused state $2^n - 2n$

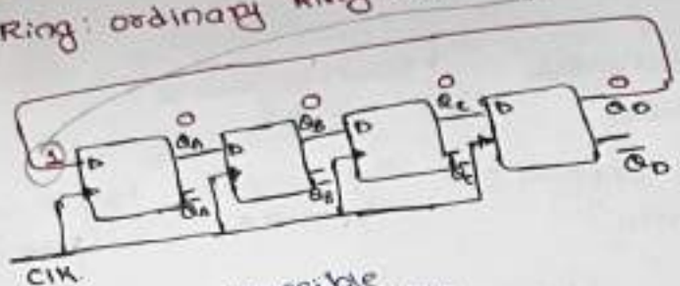




not a self corrected counter.

[lock out condition]

(ii) Ring: ordinary Ring counter: it's not a self starting so we apply ① in starting



CLK	Q_A	Q_B	Q_C	Q_D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	1	0	0	0

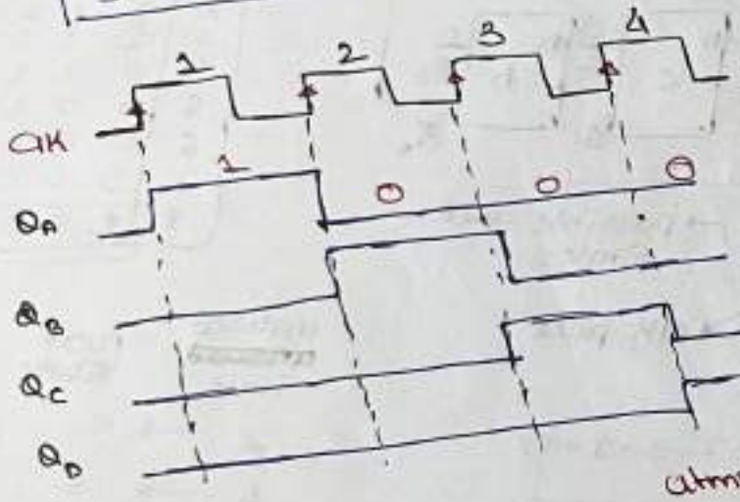
mod 4 ; X

mod 4

n → (n) → possible used state.
• clk pulse.

unused state } $2^n - n$

note
n bit ring counter
⇒ mod n counter
if i/p freq → x HRz
o/p freq → $\frac{x}{n}$ HRz

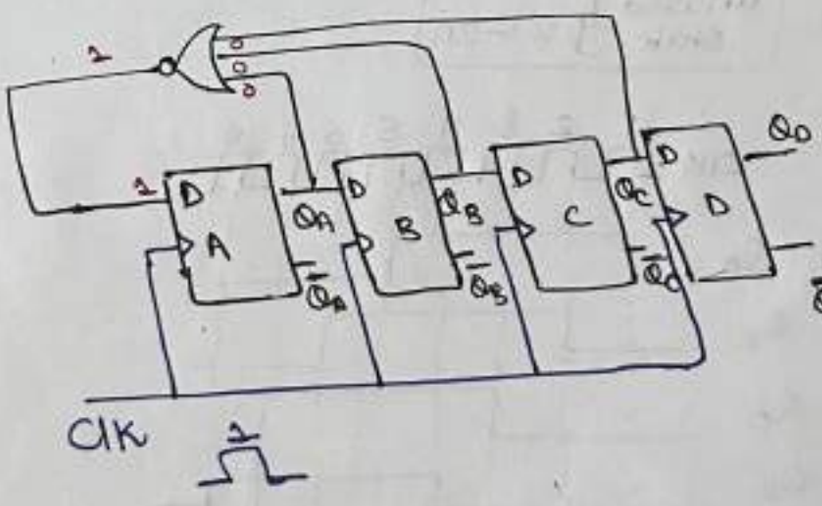


- Application of Ring counter
- timing & control signal which can enable/ Disable device in pu
- n-bit Ring counter ⇒ atmost (n-1) clk pulse it will come in counting loop

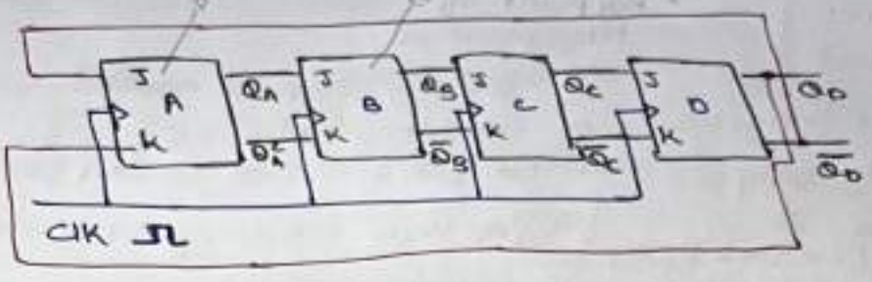
• Self Started Ring counter.

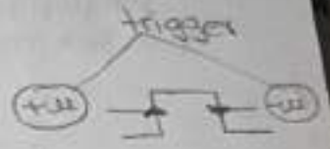
CLK	Q_A	Q_B	Q_C	Q_D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	1	0	0	0

mod 4 ; X

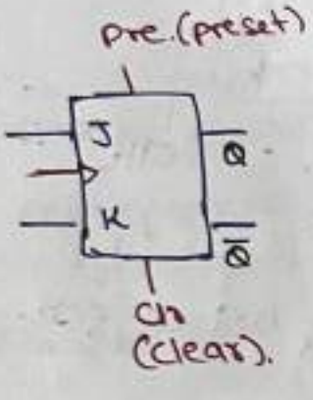


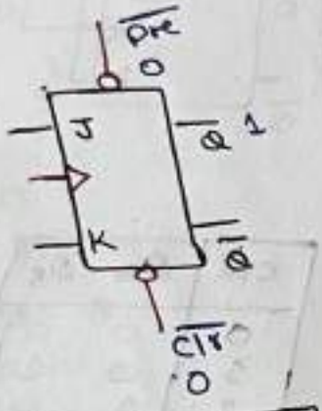
n → (n) → possible state
• clk pulse



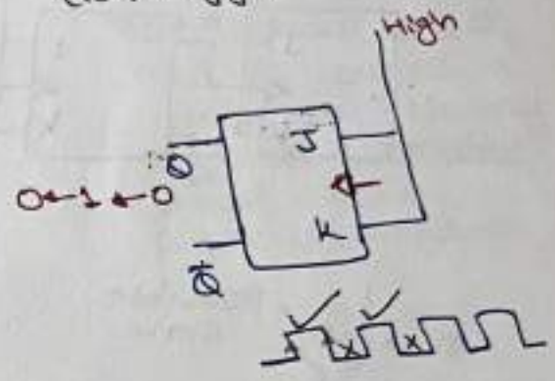


- Asynchronous I/p (05)
  Direct I/p (06)
  Overridding





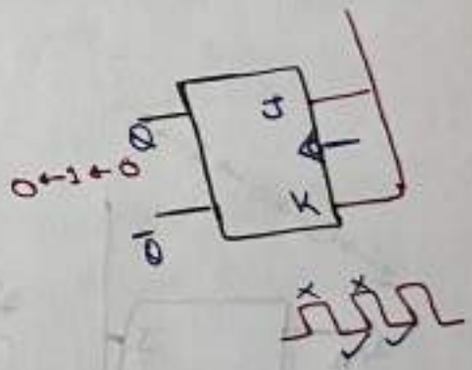
T-flipflop
(-ve) trigger



Pre	Clr	Q^+
0	0	ff
0	1	0
1	0	1
1	1	x don't care.

$\overline{Pre}$	$\overline{Clr}$	Q^+
0	0	x don't care
0	1	1
1	0	0
1	1	ff.

T-flipflop



Active low (0)

Active high (1)

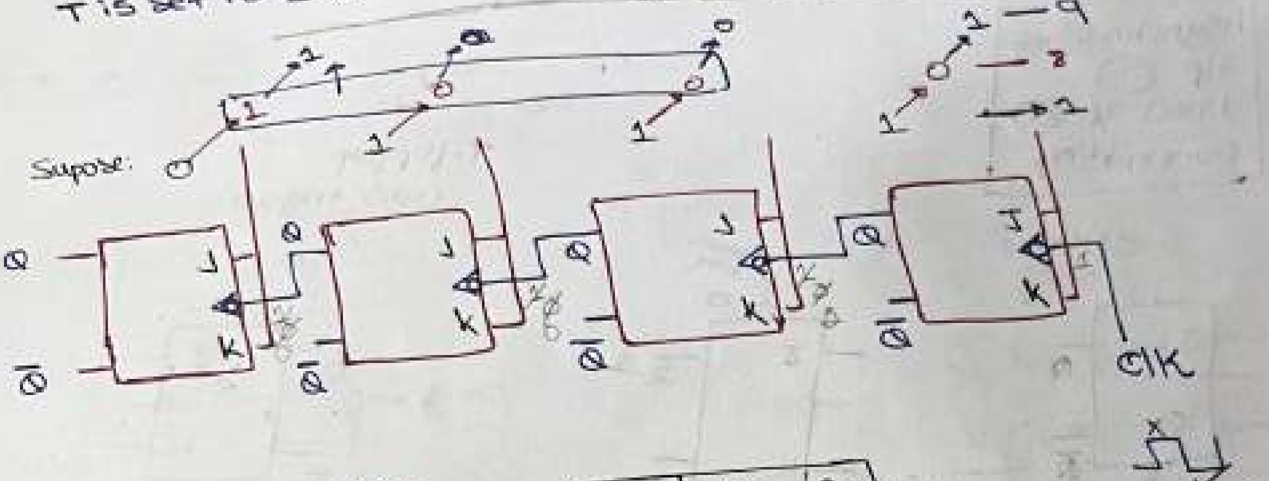
Asychno^n Counter [Ripple Counter]

- flip-flop are not trigger simultaneously. only T- flip flop is use & T is set to 1 [i.e T=1]

* key point of the operation.

- the affect of clk pulse applied should be consider upto its possible extend but if it stop at any particular flip flop then remaining state should be considered as it is.

Supose:

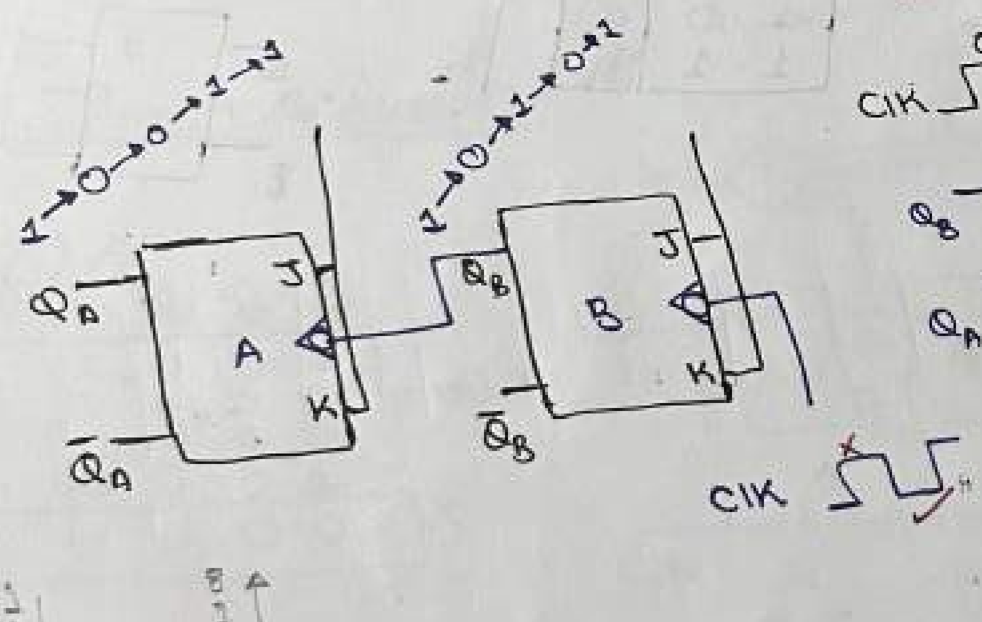


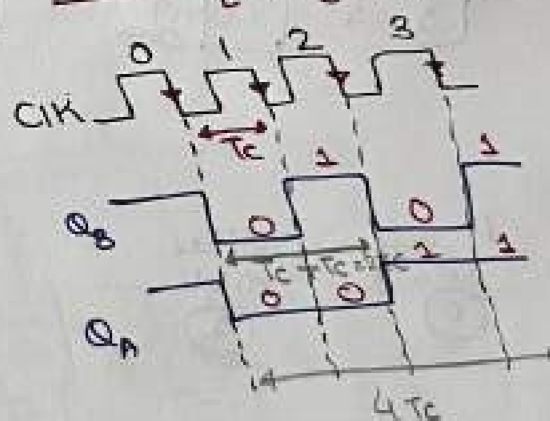
$n \rightarrow 2^n \rightarrow$ possible state
$\rightarrow$ clk pulse.

clk	Q_A	Q_B
0	0	0
1	0	1
2	1	0
3	1	1
4	0	0

mod 4

[Timing Diagram]

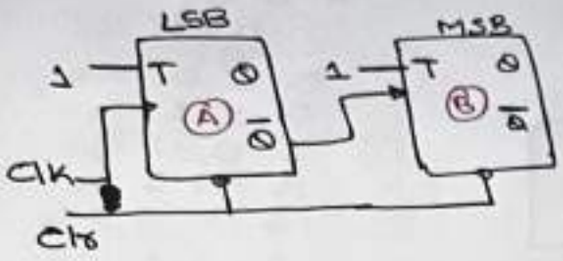


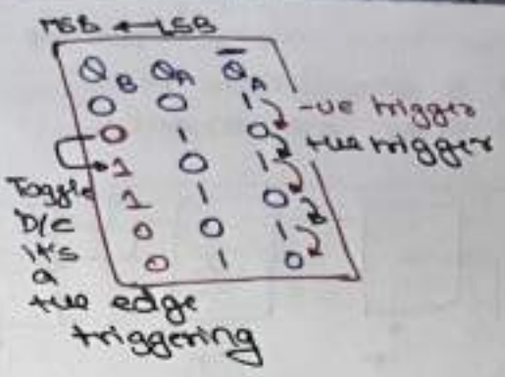


down counter up

up counter

ex: +ve trigger up counter





mod < 2^n

↳ counter can be constructed by (n bit)

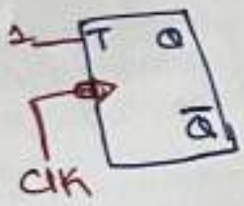
- Mod (10) counter (10 state)
(Decade counter)

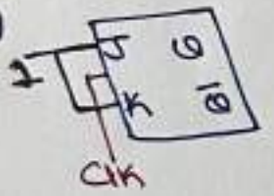
ex: Asynchronous counter can be designed by T FF only?

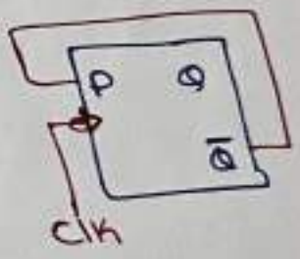
- Base of Async^n. Counter is that for every clk pulse it should Toggle.

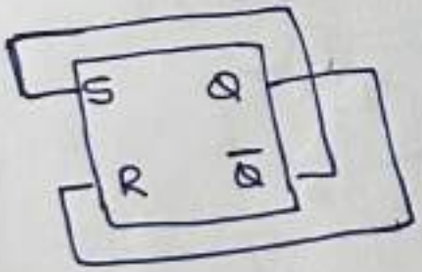
BCD counter
0000 → 1001

- its a special case of mod 10 counter in which we have only 10 state & all these sequence are in particular sequence

① 

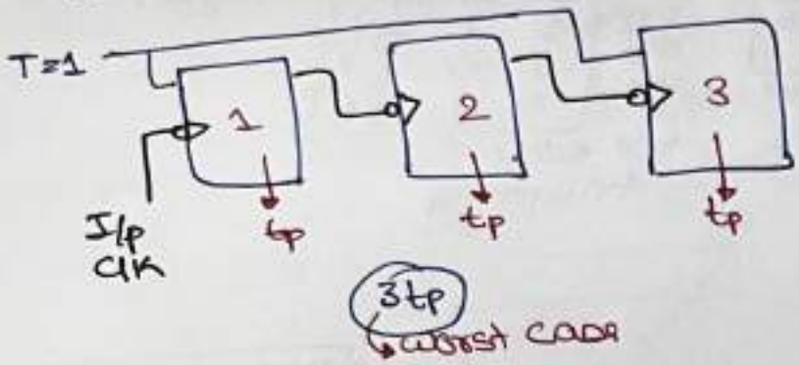
② 

③ 

④ 

- So, Async^n counter can be designed by all 4 flipflop

ex: If we have mod 8 ripple up counter & delay of each FF is tp then state will change



3tp
worst case

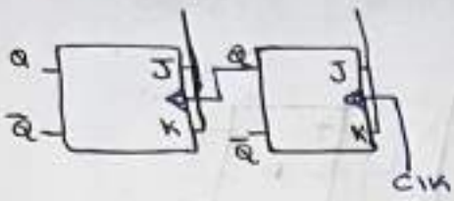
Q_3	Q_2	Q_1	
0	0	0	
0	0	1	tp
0	1	0	2tp
0	1	1	tp
1	0	0	3tp
1	0	1	tp
1	1	0	2tp
1	1	1	tp
0	0	0	3tp

time needed to change state by ripple counter ⇒ **no. of bit toggled * tp**

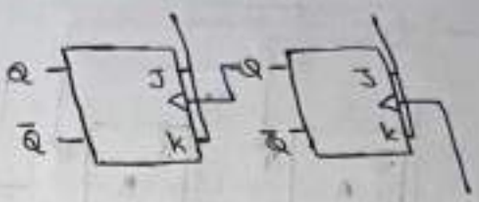
note
- In Synchronous counter the max. delay will be (tp)
- Synchronous counter are faster than the Asynchronous counter.

Short cut.

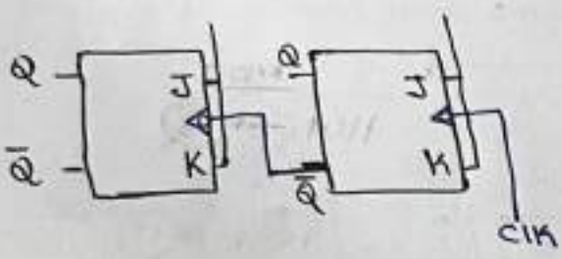
1. (-ve) trigger up counter



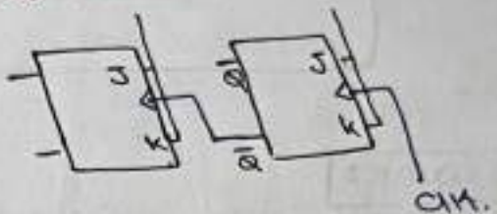
2. (+ve) trigger down counter



(-ve) trigger down counter



(+ve) trigger up counter

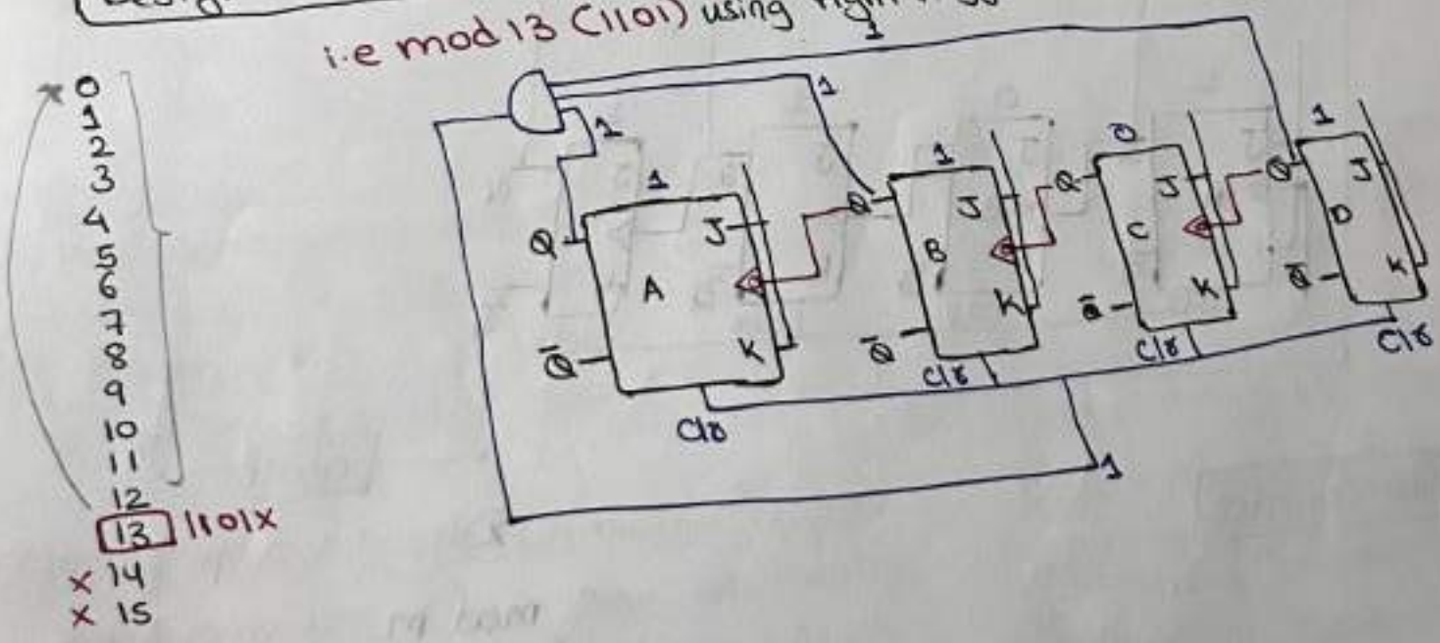


Note

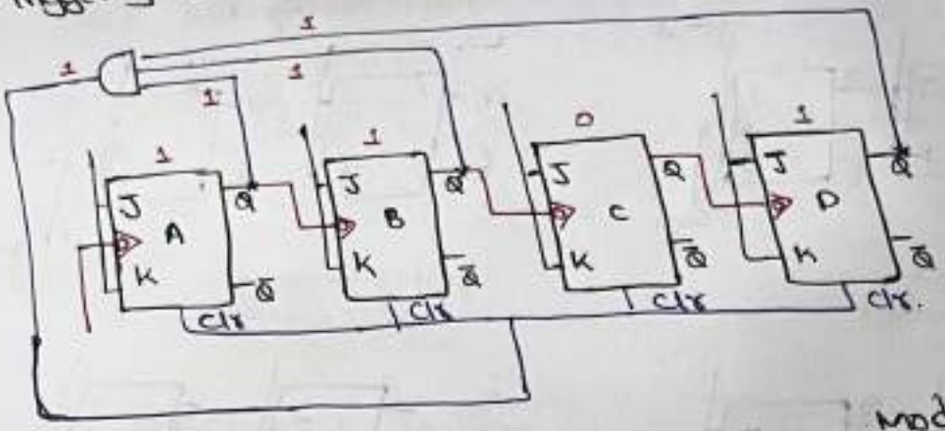
- all the circuit should be either (+ve) or (-ve) & it should be either (up) or (down) counter.

Asynchronous mod counter Design

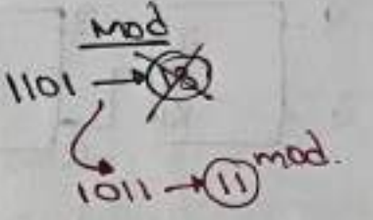
i.e mod 13 (1101) using right triggers.



i.e mod 13 using left triggering] we where heying to get mod 13 but got mod 11



up counter



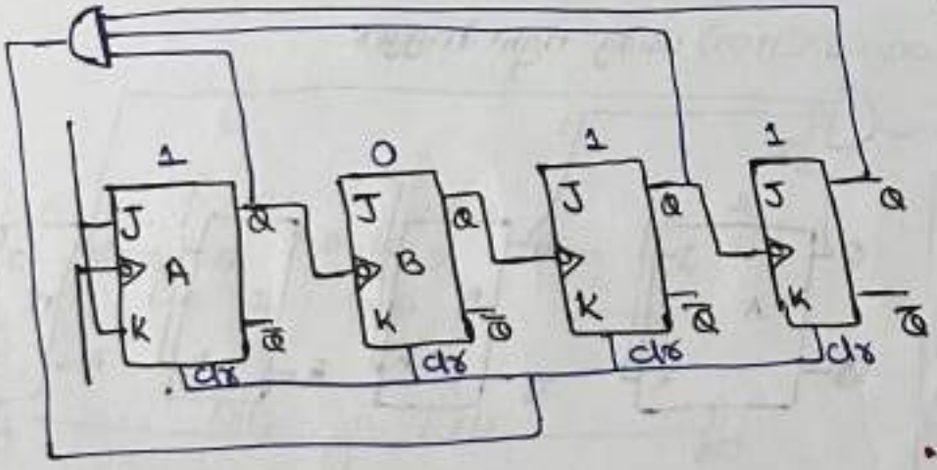
i.e if you want mod 13 in left triggering.

- swipe the binary code of mod 13 to.

1101 → 1011

to get it in left triggering.

	A B C D	A B C D
0	0 0 0 0	0 0 0 0
1	0 0 0 1	1 0 0 0
2	0 0 1 0	0 1 0 0
3	0 0 1 1	1 1 0 0
4	0 1 0 0	0 0 1 0
⋮	⋮	⋮
11	1 0 1 1	1 1 0 1 mod 11
12	1 1 0 0	0 0 1 1
13	1 1 0 1	1 0 1 1 mod 13



up-counter

note

- if (x) Hz is a i/p frequency of mod m counter then o/p frequency will be (x/m) Hz

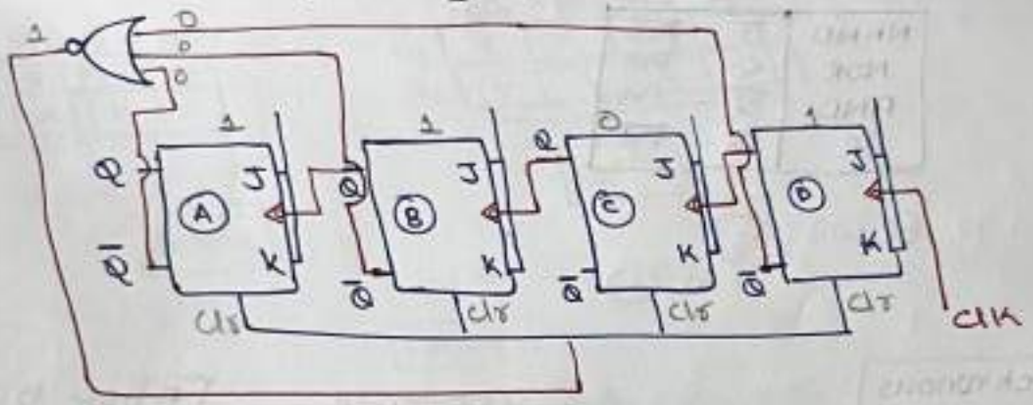
Procedure to find mod value for Asynchronous Up Counter

Step 1:- Keep 1 on those flip flop from which logic gate is having connections. (other flipflops should be kept at 0).

Step 2:- Check whether it's right triggered or left triggered.

Step 3:- Right triggered → Right Code
Left triggered → reverse code
} is the mod value.

- what if we use some different gate in place of AND gate which was providing i/p to the "clr" pin



:. Remember this table for up counters.

NAND	Q $\overline{clr}$
NOR	$\bar{Q}$ clr
AND	Q clr
OR	$\bar{Q}$ $\overline{clr}$

now my EE was (T2) ↳ ans (C)

Procedure to find mod value for Asynchronous Down counter.

↳ not many question have been asked.

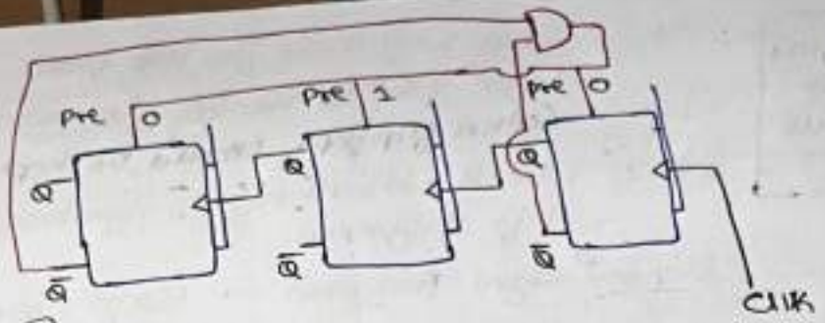
(EE) was (T2) ans (C)

Step 1: Keep zero's on those flipflops from which logic gate are having connection (other flipflop should be kept at 1).

Step 2: Check whether its right triggered (or) left triggered

Step 3:
Right triggered → Right code
Left triggered → Reverse code

Should be substracted from the max. state to get the mod value



010 → 2

7 - 2 = 5

mod 5 counter.

up connected
down connected
-ve triggered

Remember this table for "~~step~~" down Counter

NAND	$\bar{Q}$	~~Pre~~
NOR	Q	Pre
AND	$\bar{Q}$	Pre
OR	0	~~Pre~~

now by EC we (T5) → ans (d)

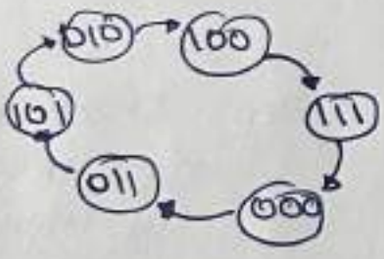
Synchronous Design

- any order of Sequence & any flip flop can be used.

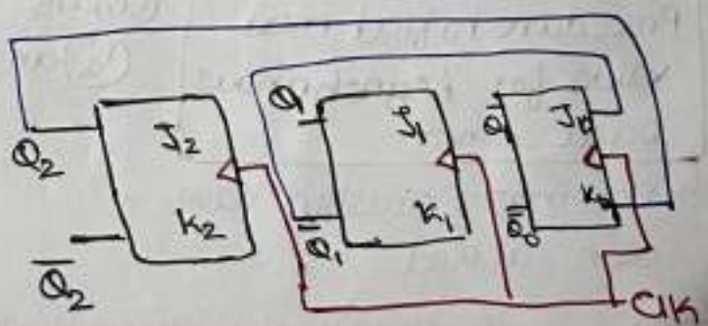
n → 2^n → possible States, clk Pulse.

SNESKSD
- State diagram
- next table
- excitn table
- K-map
- boolean expression
- design.

i.e 4 7 0 3 5 2



any sequence (random)



	Present State			next State								
	Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	J_0	K_0	J_1	K_1	J_2	K_2
4	1	0	0	1	1	1	1	x	1	x	x	0
7	1	1	1	0	0	0	x	1	x	1	x	1
0	0	0	0	0	1	1	1	x	1	x	0	x
3	0	1	1	1	0	1	x	0	x	1	1	x
5	1	0	1	0	1	0	x	1	1	x	x	1
2	0	1	0	1	0	0	0	x	x	1	1	x

For J_0

	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	Q_1Q_0	$Q_1\bar{Q}_0$
$\bar{Q}_2$	1	x	x	0
Q_2	1	x	x	x

place value a/c to

$J_0 = \bar{Q}_1$

For K_0

	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	Q_1Q_0	$Q_1\bar{Q}_0$
$\bar{Q}_2$	x	x	0	x
Q_2	x	1	1	x

$K_0 = Q_2$

For J_1

	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	Q_1Q_0	$Q_1\bar{Q}_0$
$\bar{Q}_2$	1	x	x	x
Q_2	1	1	x	x

$J = 1$

For K_1

	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	Q_1Q_0	$Q_1\bar{Q}_0$
$\bar{Q}_2$	x	x	1	1
Q_2	x	1	x	x

$K = 1$

For J_2

	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	Q_1Q_0	$Q_1\bar{Q}_0$
$\bar{Q}_2$	0	x	1	1
Q_2	x	x	x	x

$J_2 = Q_1$

For K_2

	$\bar{Q}_1\bar{Q}_0$	$\bar{Q}_1Q_0$	Q_1Q_0	$Q_1\bar{Q}_0$
$\bar{Q}_2$	x	x	x	x
Q_2	0	1	1	x

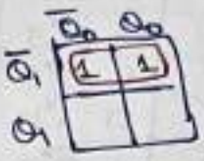
$K_2 = Q_0$

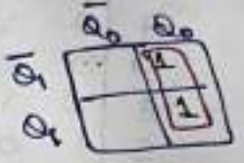
Co-8 (chapter-4)

(23)

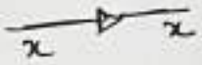
	Present State Q_1 Q_0		next state q_1 q_0		D_0	D_1
0	0	0	0	1	1	0
1	0	1	1	1	1	1
3	1	1	1	0	0	1
2	1	0	0	0	0	0

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

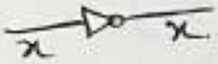


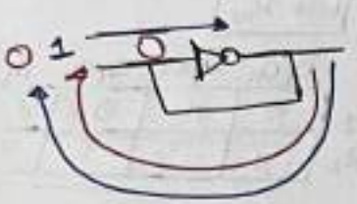


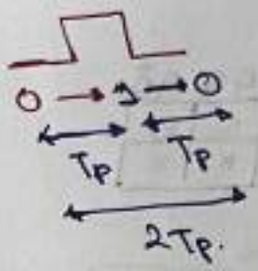
Buffer



Not gate



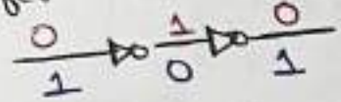


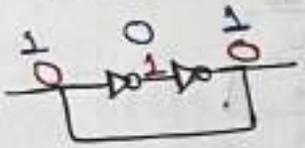


Astable Multivibrator or ring oscillator or wave form generator.

$f = \frac{1}{T} = \frac{1}{2t_p}$

Bffer





bistable multivibrators.

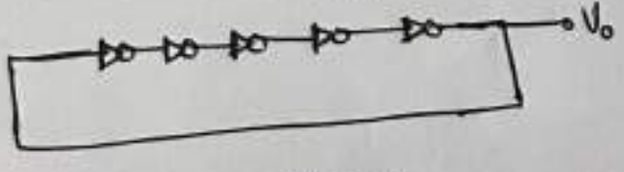
for "n" gates [n must be odd.]

$T = n(2t_p)$

$f = \frac{1}{T} = \frac{1}{n(2t_p)}$

EG (chapter-4).

(1) $P_d = 100$ pico sec (Propagation delay)



↳ it's ring oscillator.

$T = n(2t_p)$. & $n = 5$.

$P_d = 2t_p$.

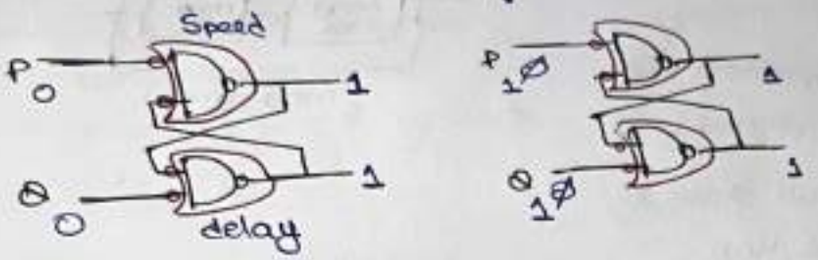
$\Rightarrow 2 \times 100 \times 10^{-12}$

So, $f = \frac{1}{T} = \frac{1}{5 \times 2 \times 100 \times 10^{-12}}$

So, it is (1GHz)

CC ③ given Initially P=Q=0 (given) Delay are not same

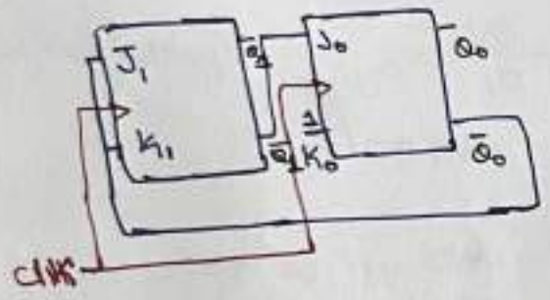
not P=Q=1



[It can't happen that delay are different so, we will distribute it in 2 half].

1st delay at P then 2nd delay at Q.

- How to find mod value of Synchronous counter.



clk	$\bar{Q}_0$ J_1	$\bar{Q}_0$ K_1	$\bar{Q}_1$ J_0	1 K_0	Q_1	Q_0	
					0	0	[assumed]
1	1	1	1	1	1	1	mod 3
2	0	0	0	1	1	0	
3	1	1	0	1	0	0	X
					0	1	(assume)
	0	0	1	1	0	0	mod 3
					1	1	
					1	0	
					0	0	X

Qs] Initially $Q_1Q_0 = 00$.
After 35 clk pulse.

33 clk pulse → 00
34 -"- → 11
35 -"- → 10

mod N → it will only contain value's which will repeat

If it's mod n then it will repeat it self after every nth table

n×1
n×2
n×3

we assumed 0,1 which is not in the sequence if we take 2 per in circuit then it is not repeating it self

Q_1Q_0 → 00, 11, 10, 00

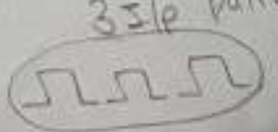
or Q_0Q_1 → 00, 11, 01, 00

or Q_1Q_0 → 11, 10, 00, 11

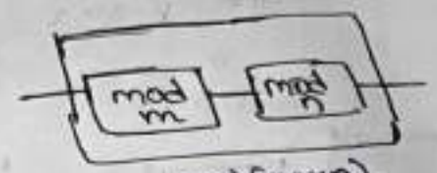
I/P (2) Hz

O/P (1/m) Hz

So here 3 I/p pulse required to produce 1 o/p



Cascaded counter



mod(m×n).

Try E.C Ques → ans

•Step to find mod value for synchronous counter.

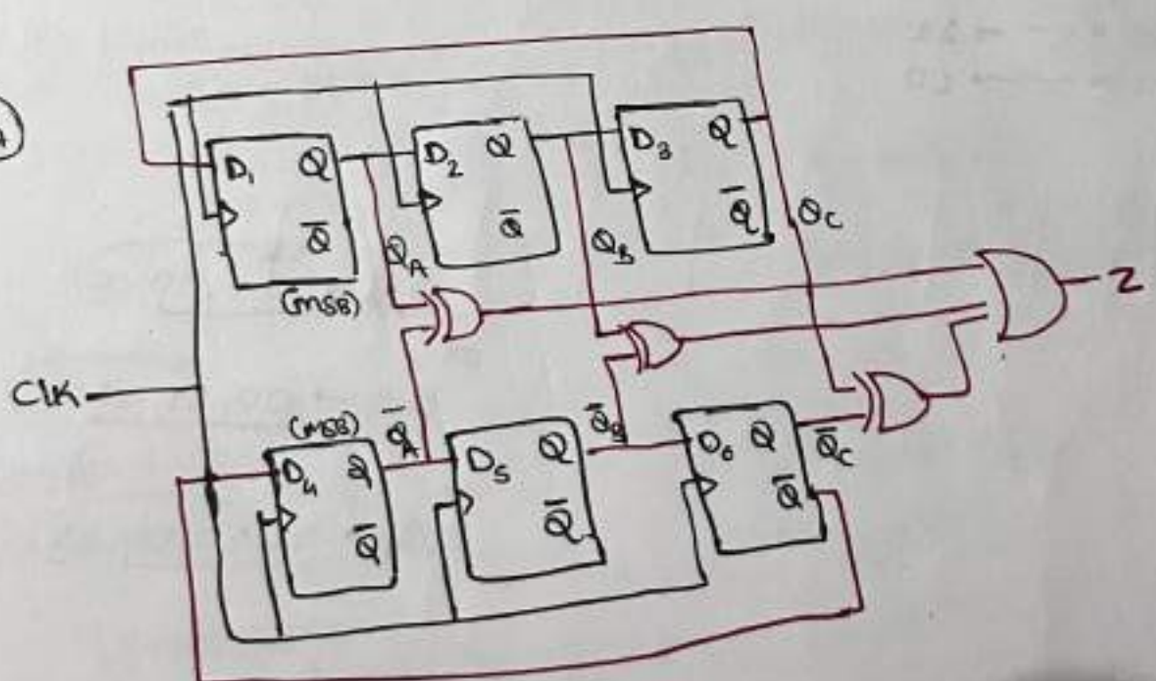
Step 1: write the I/p & o/p in the form of table.

Step 2: write the corresponding I/p connection on there head.

Step 3: keep some default state & make a horizontal line

(T3)

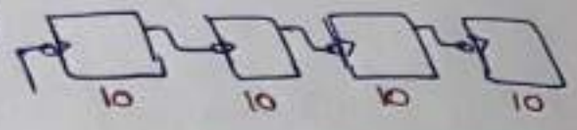
(T4)



ex: If we have mod-6
(a) ripple up counter
(b) Synchronous counter

& delay of each FF is 10nsec
then what must be the
time period & freq of i/p clock

(a)



max delay will be 40ns

So, Time period $\geq$ 40ns

& frequency $= \frac{1}{40 \times 10^{-9}}$

$\frac{1000}{40} \times 10^6 = 25$ MHz

(b) Synchronous.

So, Total delay to change
State = 10 nsec
(min)

min. Time
period = 10nsec

min. frequency $= \frac{1}{10 \times 10^{-9}} = \frac{10^9}{10}$

$\Rightarrow$ 100 MHz

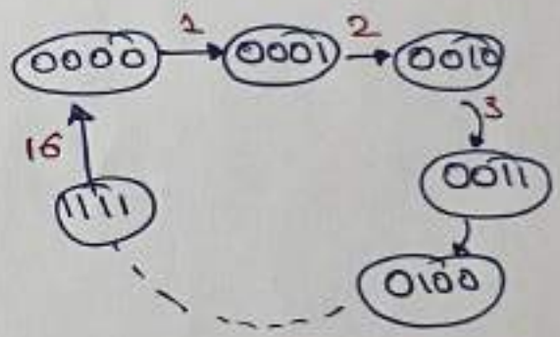
ex. what can be i/p clk
frequency to 4bit ripple
up counter if delay of
each FF is 2nsec.

Total
delay
to change $\Rightarrow 4 \times 2 = 8$ nsec

freq. $= \frac{1}{8} \times 10^9$

$\Rightarrow$ 125 MHz
max.

ex: If a ripple up counter
has initial state 0000
after applying 5 clk
pulse final state is 0011



So, $3+k*16$

$k = 0, 1, 2 \cdots$

Q_C Q_A Q_B $\overline{Q_C}$ $\overline{Q_A}$ $\overline{Q_B}$	
D_1 D_2 D_3 D_4 D_5 D_6	Q_A Q_B Q_C $\overline{Q_A}$ $\overline{Q_B}$ $\overline{Q_C}$
	1 0 0 1 0 0
0 1 0 1 1 0	0 1 0 1 1 0
0 0 1 1 1 1	0 0 1 1 1 1
1 0 0 0 1 1	1 0 0 0 1 1
0 1 0 0 0 1	0 1 0 0 0 1
0 0 1 0 0 0	0 0 1 0 0 0
1 0 0 1 0 0	1 0 0 1 0 0

mod ⑥

P.y.Q

311 no of pulses are need to change the content of a 8-bit upcounter from 10101100 to 00100111 (right most bit is the LSB)?

So, we can do is

$\underbrace{10101100}_{172}$ to $\underbrace{00100111}_{39}$.

* it range to 256 (0 to 255).

So, 172
39
133

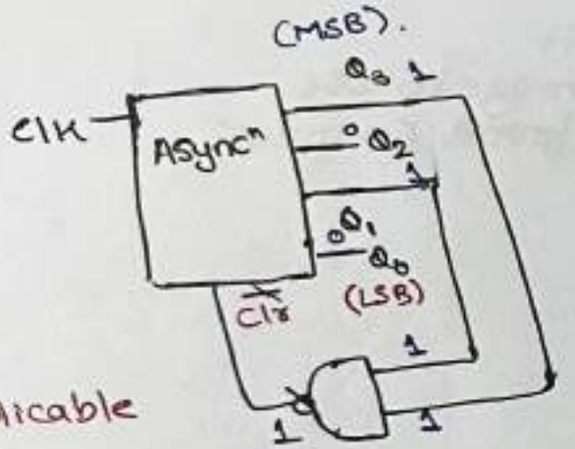
So, 256
133
123

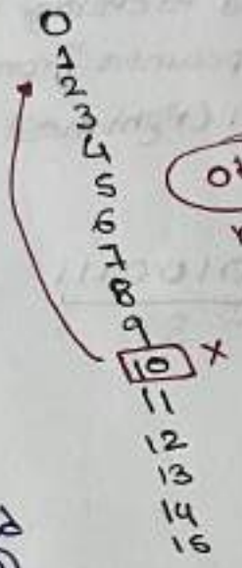
- Difference B/w Asynchronous & Synchronous clear.
- In case of Asynchronous clear, the clear funcn will be operated without depending on clock
- In the case of synchronous clear funcn, it will be operated along with applied clock.

(I) In the case of Asynchronous counter " jo code hai wahi mod ha"
In the case of Synchronous counter " jo code mila that +1" is mod value.

(II) if gate delay is 1ns more than CLK in both Asynchronous & synchronous the mod value increased by 1.

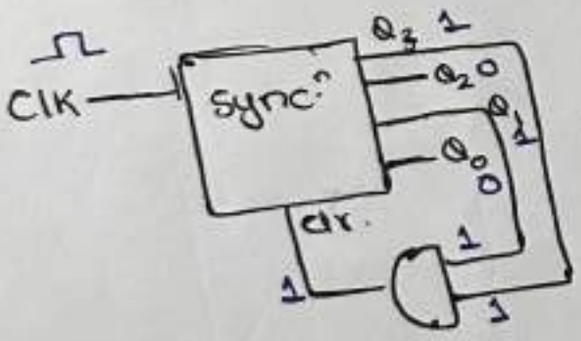
(1)



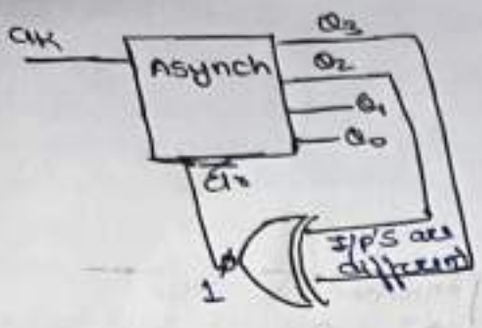


applicable for
NAND
NOR
AND
OR

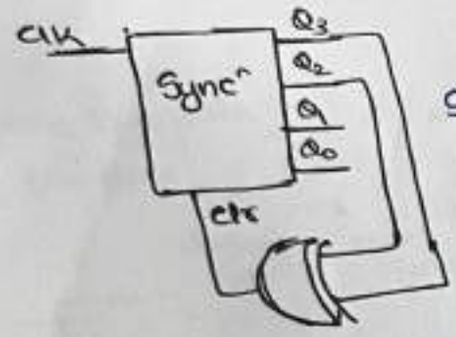
So, 1010 → mod (10)



1010 → mod (10)
+1
mod (11)



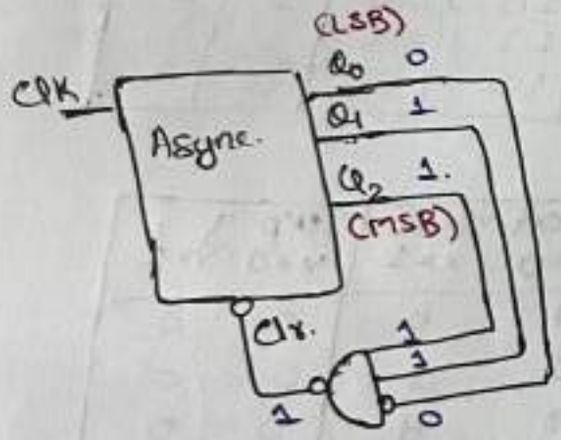
	Q_3	Q_2	Q_1	Q_0	
0	0	0	0	0	mod 4
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	X



So, ↓

Same reason as above but plus 1

Sq mod (4)
+ 1
mod (5)



In Asynchronous,

110 → mod (6)

and + 1 for delay

mod (7)

In Synchronous.

110 → mod (6)

+1 --- for Synchronous.

+1 --- for delay

mod (8)

- State Reduction Technique

→ State Diagram:
It is a graphical representation which consist of present state, I/p, next state O/p

- write procedure.

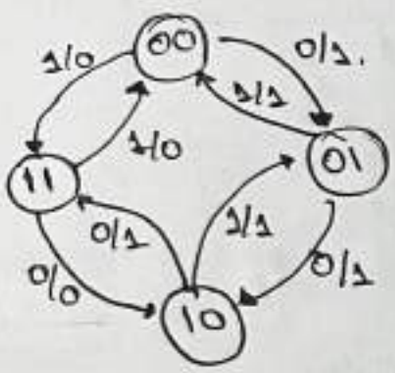
Step 1: Draw the state table for given State diagram.

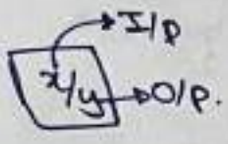
Step 2: Identify the equivalent state

Step 3: If two state are equivalent then eliminate one of the state

Step 4: Repeat the above process untill we get all different state.

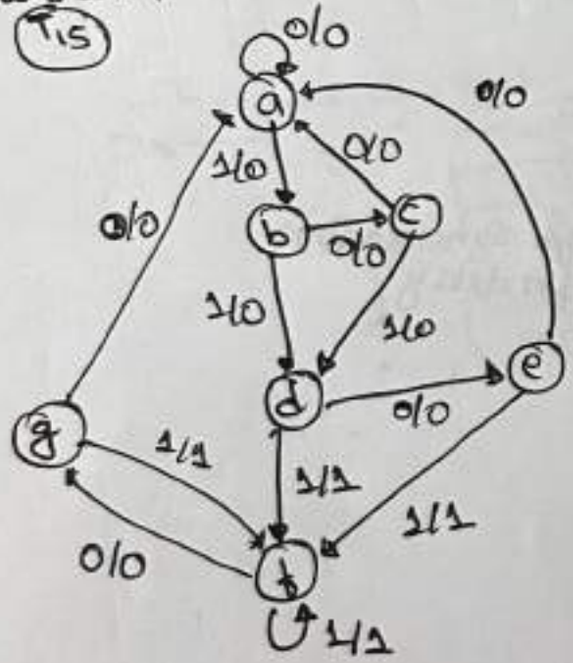
equivalent state
↳ 2 state are said to be equivalent if nxt state & O/p is equal



So, 

Present State		next State x=0	next State x=1	O/p y x=0	O/p y x=1
0	0	01	11	1	0
0	1	10	00	1	1
1	0	11	01	1	1
1	1	10	00	0	0

Q.8 (Chapter 4)



Present State	next State x=0	next State x=1	O/p y x=0	O/p y x=1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

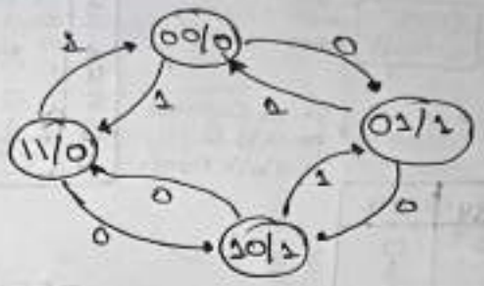
same

will also become equal

• State Reduction are of 2 type.

↳ mealy method
• in these method the present o/p not only depends on present state but also on present I/p.
[Above ex. are mealy only]

↳ moore method.
↳ In these method the present o/p depends only on present state.



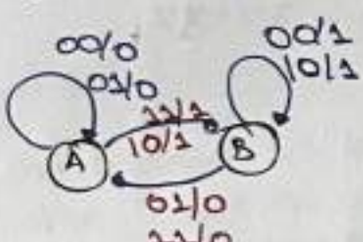
• State Diagram of flipflop

assume
0 → A
1 → B

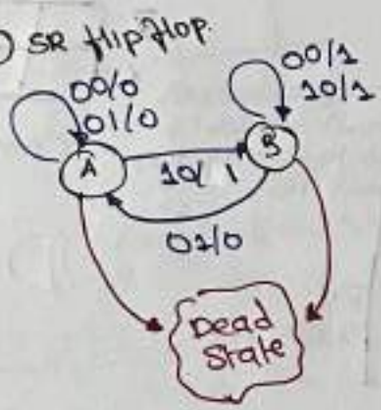
present state | next state

1	J	Q	Q^+
1	K	$\bar{Q}$	

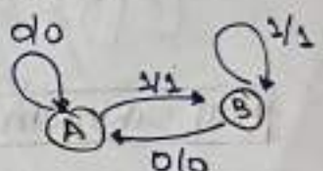
① JK flip flop

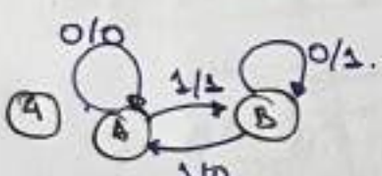


② SR flip flop



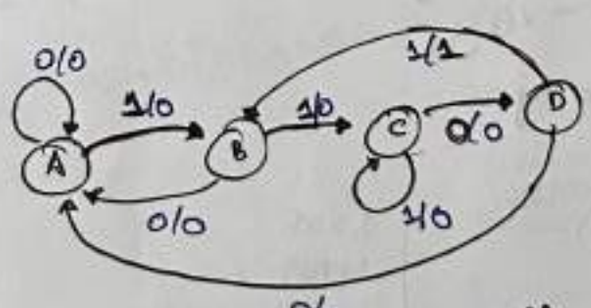
③ D-flip flop





T-Flip Flop

[or to detect sequence]



• Sequence Detector (or) state machine

• overlapping allowed

I/p → 0 1 0 1 1 0 1 0 1 1 0 1 1 0 1

O/p → 0 0 0 0 0 0 1 0 0 0 1 0 0 1

to Detect the sequence
• move until you get o/p as (1) until then move forward.
and after reaching end you have 2 way, you can go any way until you find o/p as "1"

Combinational Circuit → with no memory

I/p's → Comb^{n} Circuit → O/p's

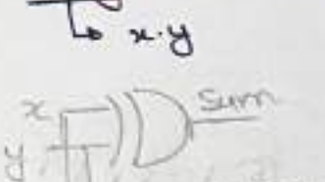
Half adder • It is capable to add 2, 1bit num.

x	y	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$\bar{x}\cdot y$, $x\cdot\bar{y}$

$Sum = \bar{x}\cdot y + x\cdot\bar{y}$
$= x \oplus y$

Carry
↳ $x \cdot y$



Full adder capable of adding 3bit at a time

	x	y	z	Carry	Sum
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1

Majority Ckt. [means getting atleast 2 one's]

x \ yz	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{x}$		1		1
x	1		1	

$Sum = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$

$Sum = [x \oplus y] \oplus z$

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

x \ yz	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{x}$			1	
x		1	1	1

$Carry = xy + xz + yz$

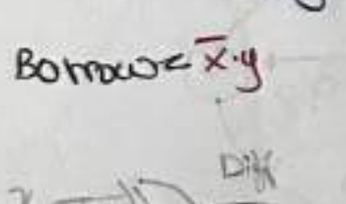
Half Subtractor → it's a combinational circuit which is used to subtract 2, 1bit no.

X	y	Borrow	Diff
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

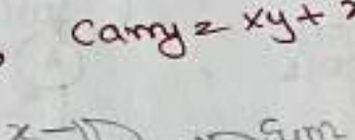
$\bar{x}y$, $x\cdot\bar{y}$

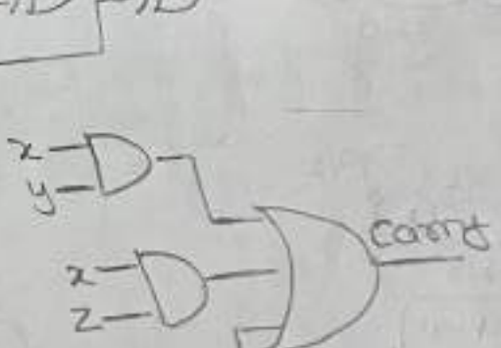
$Diff = \bar{x}\cdot y + x\cdot\bar{y}$
$= x \oplus y$

$Borrow = \bar{x}\cdot y$



1 XOR
1 NOT
1 AND
gate





Full Subtractor

	x	y	z	Borrow	Diff
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

used to subtract 2, 1bit no. & previous borrow.

Borrow: $\bar{x}\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz + xyz$

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

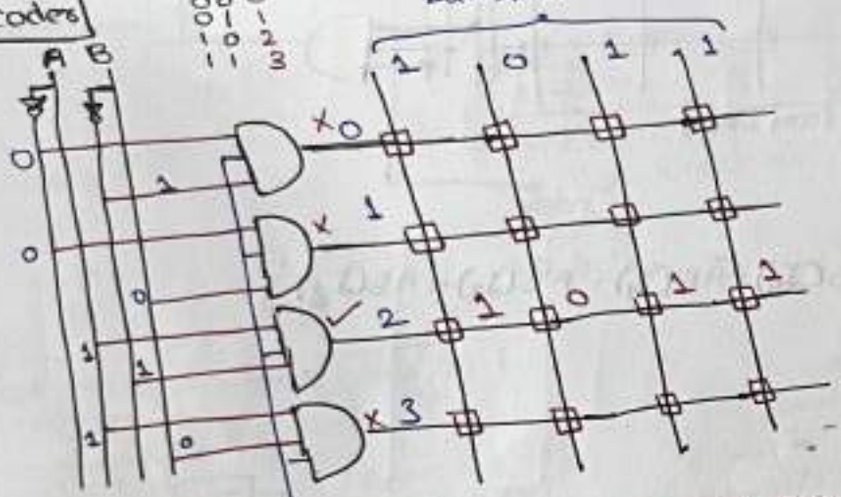
Diff: $x \oplus y \oplus z$

Design
- 1 (3 i/p) XOR
- [2 (2 i/p) XOR]
- 1 NOT
- 3 AND
- 1 (3 i/p) OR
- [2 (2 i/p) OR]

4:1 Multiplexer

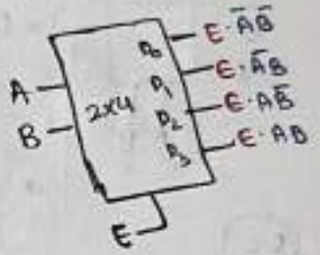
Decoder

AB: 00 → 0, 01 → 1, 10 → 2, 11 → 3



n → 2^n

Select line



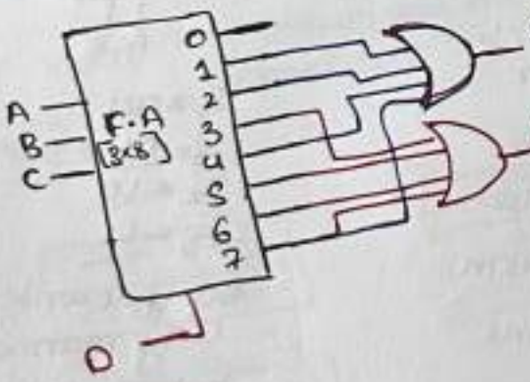
Sum = $\Sigma m(1,2,4,7)$.

Carry = $\Sigma m(3,5,6,7)$.

Enable
[the Additional we are adding to get the control so that we to store or not]

→ if we attach these enable to Decoder then its become **de Multiplexer**

if we remove Enable then it becomes Decoder.



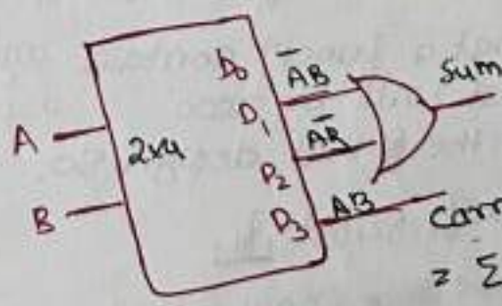
AND-OR

H.A

	A	B	carry	Sum
0	0	0	0	0
1	0	1	0	1
2	1	0	0	1
3	1	1	1	0

Sum = $\Sigma m(1,2,4,7)$

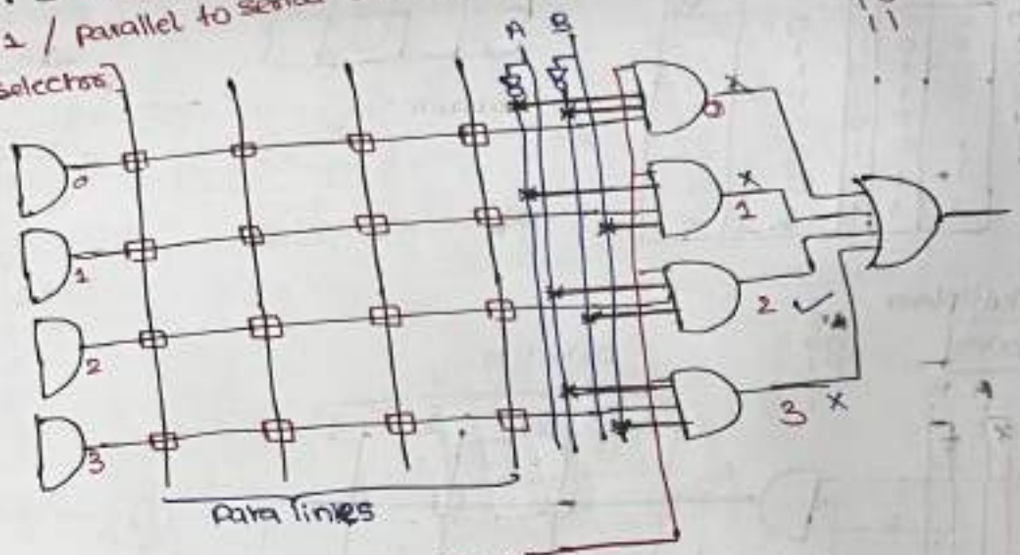
Carry = $\Sigma m(3,5,6,7)$

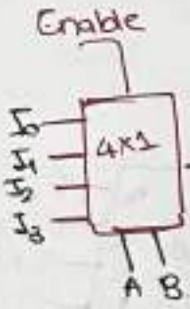


• Multiplexer [MUX]

[many to 1 / parallel to serial converter

/data selector]





$f = E\{\bar{A}\bar{B}(I_0) + \bar{A}B(I_1) + A\bar{B}(I_2) + AB(I_3)\}$

E.E

1) the following switching functions are to be implemented using a Decoder.

$f_1 = \Sigma m(1,2,4,8,10,14)$

$f_2 = \Sigma m(2,5,9,11)$

$f_3 = \Sigma m(2,4,5,6,7)$.

The min. configuration of the decoder should be

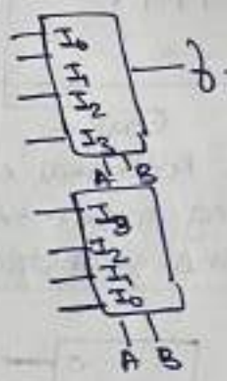
among the f_1, f_2, f_3 we will choice the max. no. of minterm & maxterm that a functn contain. and the rest will be acco in the B that design so.

$\max(f_1, f_2, f_3) = f_1$

In f_1 we should find max. no. so. $f_1 = \max(1,2,4,8,10,14)$

so, max is 14

that comeunder $2^3 < 14 < (2^4) \Rightarrow$ so, 4 to 16 decoder will be choosn.



$I_0 = 00$

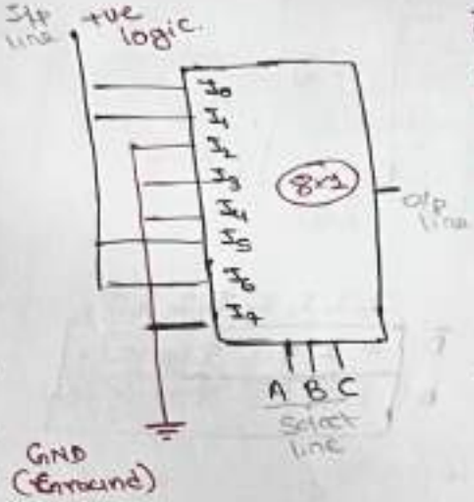
$I_1 = 01$

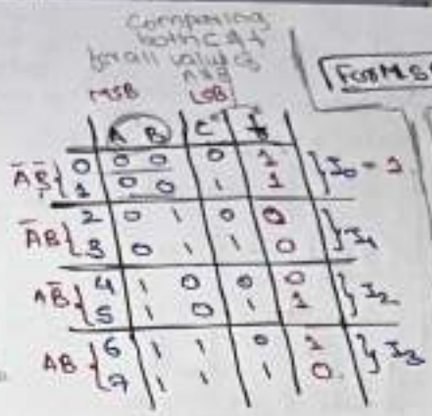
$I_2 = 10$

$I_3 = 11$

so you write in any manner but I_0 will always be (00) for I_0

$f(ABC) = \Sigma m(0,1,5,6)$



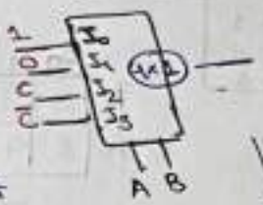


For MSB

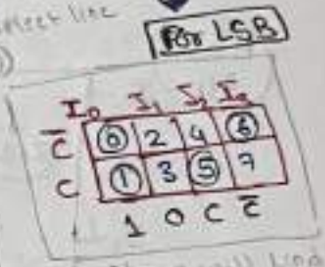
∴ if we are using C (LSB) as I/p. for these we will use the Short cut

↓

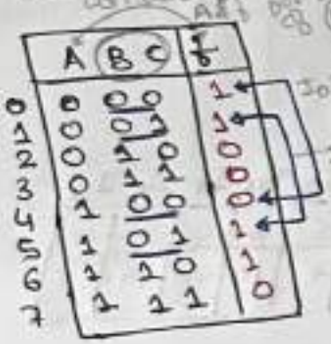
blla blla



For LSB



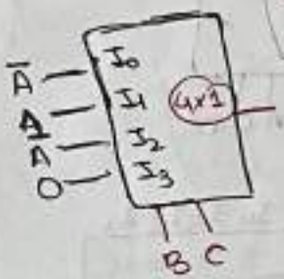
For MSB using MSB as I/p



then we are going to use the Short cut.

→ lalala

--→ lalala.



1st we will find the format based on the LSB & MSB b/c we took MSB as I/p we will use

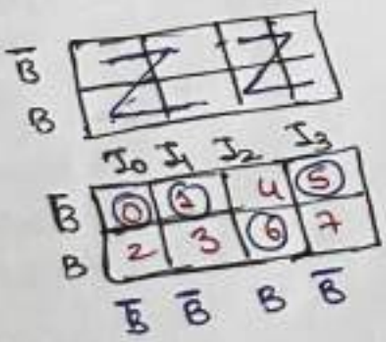
→ lalala

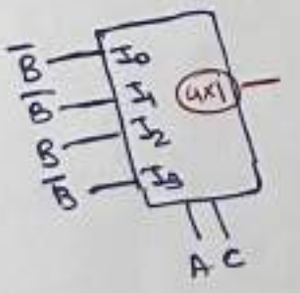
→ lalala

format & then we will mark the no. that are given & o/c no that we will Draw k-map

if we choose "B" as I/p then

you can use the Short cut Z1 Z2 normal numbering



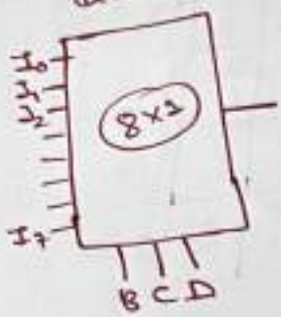


MSB			LSB
A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

A as I/P (MSB)

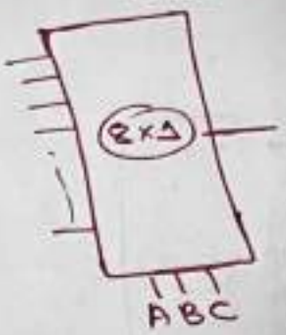
lalala →

lalala →



D as I/P (LSB)

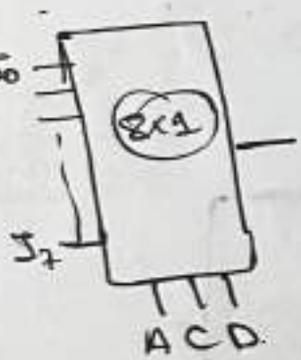
lalla lalla.



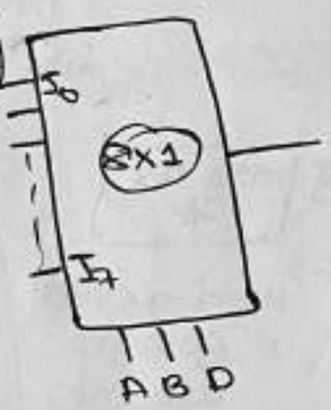
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{D}$	0	2	4	6	8	10	12	14
D	1	3	5	7	9	11	13	15

B as I/P



C as I/P



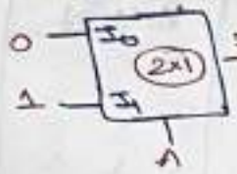
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{B}$	0	1	2	3	8	9	10	11
B	4	5	6	7	12	13	14	15

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{C}$	0	1	4	5	8	9	12	13
C	2	3	6	7	10	11	14	15

•Logic gate by Mux.

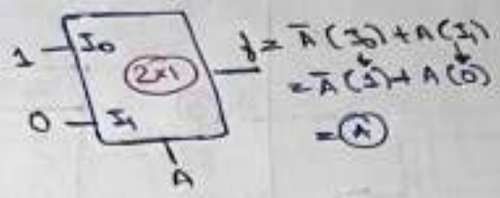
A → A

(1) Buffer



A → $\bar{A}$

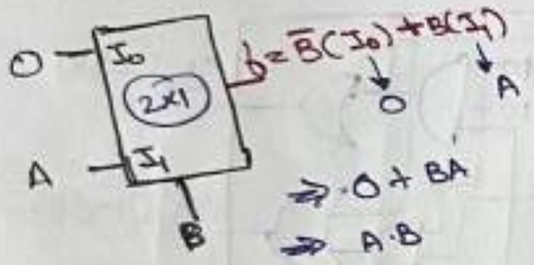
(2) NOT



(3) AND gate

	A	B	$f = A \cdot B$
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

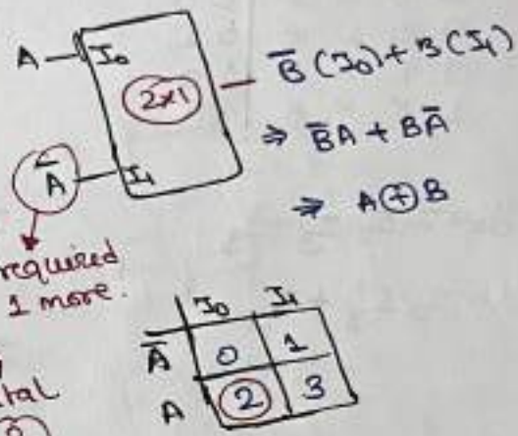
$f = \Sigma m(3)$.



(4) EXOR

	A	B	$f = A \oplus B$
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

$f = \Sigma m(1,2)$.



	I_0	I_1
$\bar{A}$	0	1
A	2	3

	I_0	I_1
$\bar{A}$	0	1
A	2	3
	0	A

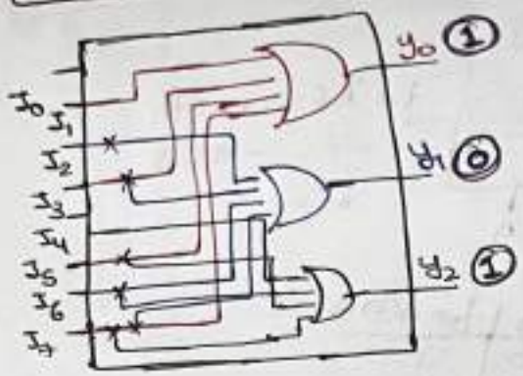
(2×1) mux how many required to build a gate.

NOT – 1
AND – 1
OR – 1
EXOR – (2)
EXNOR – 2
NAND – 2
NOR – 2

Half adder

Sum → EXOR 2
Carry → AND 1
3

Encoders — for single line I/p



$y_0 = I_1 + I_3 + I_5 + I_7$

$y_1 = I_2 + I_3 + I_6 + I_7$

$y_2 = I_4 + I_5 + I_6 + I_7$

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

for multiple line o/p

I_3	I_2	I_1	I_0	y_1	y_0
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

$y_0 = I_1 I_2' I_3' + I_3$

$y_1 = I_2 I_3' + I_3$

we will draw eqn where every there is 1 in y_1 & y_0

also don't take don't care

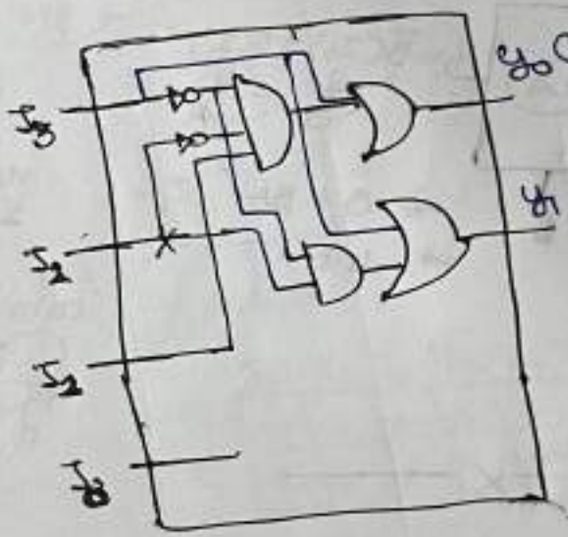
i.e $y_0 = I_1 I_2' I_3' + I_3$

represent (1)

represent (0)

as it contain bar

Priority Encoder



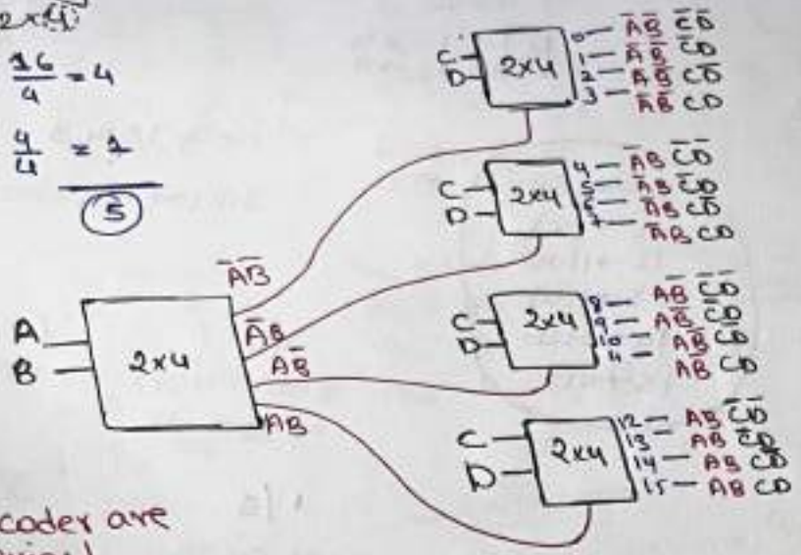
• Decoder / (Demux) if enable.

by using 2×4

$\frac{16}{4} = 4$

$\frac{4}{4} = 1$

(5)



• 5 Decoder are required

• 4 enable decode are present.

• Circuit to construct 3×8 using 2×4

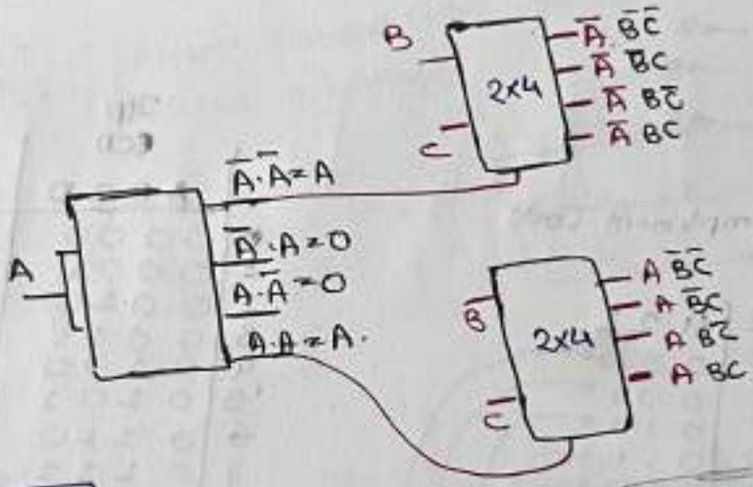
$\frac{8}{4} = 2$

$\frac{2}{4} \rightarrow (0.5)$

1

$\Rightarrow \boxed{3}$

For decoder we find by looking 4×16 by using 2×4

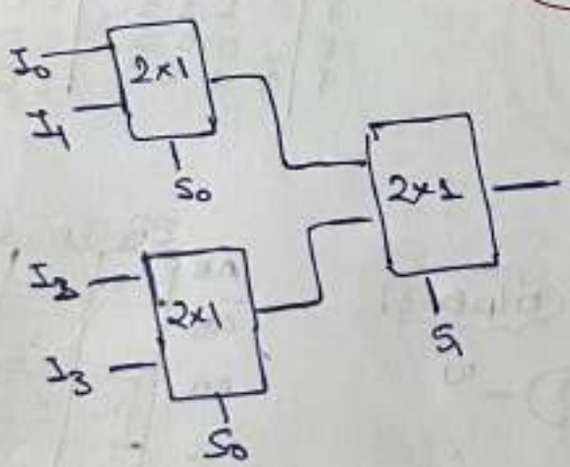


• MUX (4)×1 by using (2)×1

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

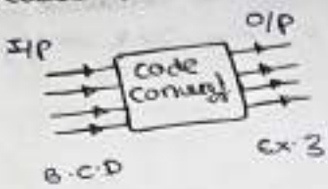
$\frac{2}{2} = 1$

$\boxed{3}$



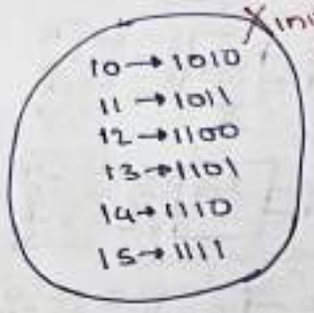
For encoder we find by looking (4)×1 by using (2)×1

• codes & code convertors



BCD
8 4 2 1 ----→ [max^m range is 8+4+2+1 = 15]
0 1 1 1 -→ 7
1 0 0 1 -→ 9

i.e 7/9/5/4/3
0111 1001 0101 0100 0011
B.C.D.



0 → 0000 B.C.D.
+11
0011 Ex-3

1 → 0001 B.C.D
+11
0100 Ex-3.

1/0
0001 0000
B.C.D

1/6
0001 0110

1/5
0001 0101
B.C.D.

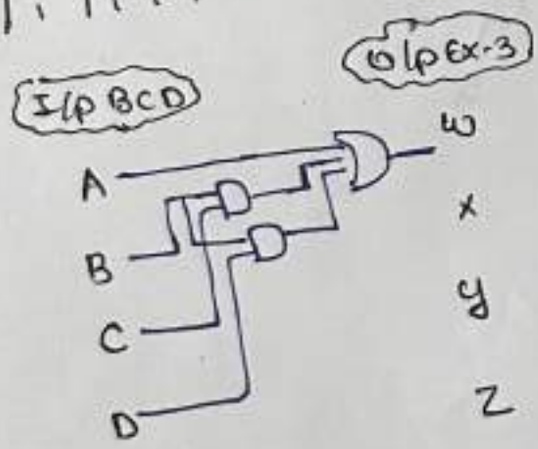
• 2421
0010 → 2
0001 → 5
1011 → 5.

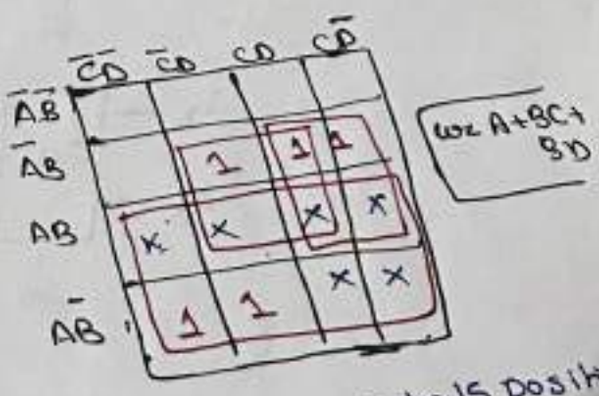
Self Complement Code

	2	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	1	0 1 1
6	1	1 0 0
7	1	1 0 1
8	1	1 1 0
9	1	1 1 1

	I/P BCD A B C D	O/P Ex-3 W X Y Z
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	

are allowed in excess -3 but not in BCD



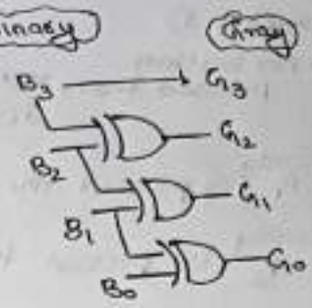


'x' will be but in 10 to 15 position as it doesn't belong to BCD.
↓
Don't Care

Binary code ⟷ gray code

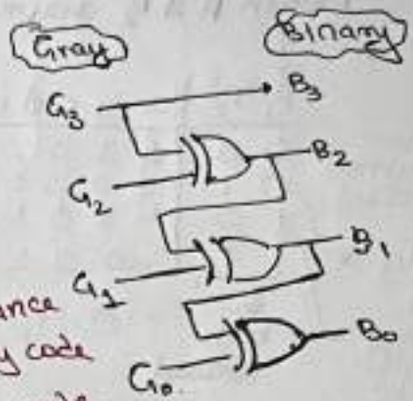
binary (MSB) → Gray (MSB)

$G_3 = B_3$
$G_2 = B_3 \oplus B_2$
$G_1 = B_2 \oplus B_1$
$G_0 = B_1 \oplus B_0$



Gray → binary (MSB)

$B_3 = G_3$
$B_2 = B_3 \oplus G_2$
$B_1 = B_2 \oplus G_1$
$B_0 = B_1 \oplus G_0$

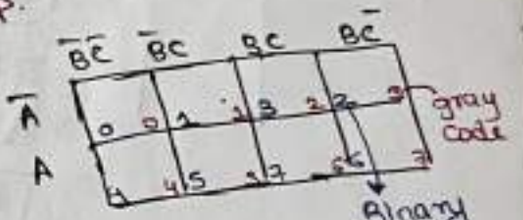


Note

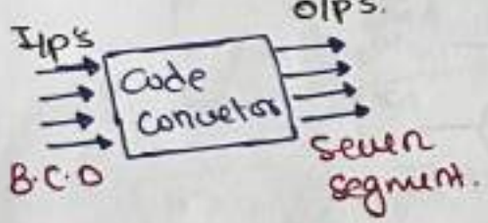
- Hamming distance is "1" in gray code
- unit distance code
- used in k-map.

	binary	gray	
0	0 0	0 0	1
1	0 1	0 1	1
2	1 0	1 1	2
3	1 1	1 0	1

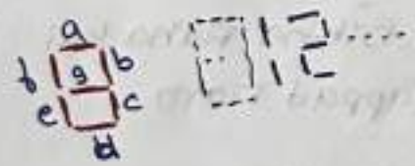
are the difference b/w 2 codes.



- Seven segment display



	I/P BCD A B C D	o/p seven segment a b c d e f g	
0	0 0 0 0	1 1 1 1 1 1 0	'0'
1	0 0 0 1	0 1 1 0 0 0 0	
2	0 0 1 0	1 1 0 1 1 0 1	
3	0 0 1 1		
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		



u.B.C chapter 3)

Fig

- no button pressed then "0" is displaced
- only P_1 is pressed then "2"
- only P_2 is pressed then "5"
- both A & P_2 are pressed "E"

P_1	P_2	a	b	c	d	e	b	g
0	0	1	1	1	1	1	1	0
0	1	1	0	1	1	0	1	1
1	0	1	1	0	1	1	0	1
1	1	1	0	0	1	1	1	1

no button pressed

only P_2

only P_1

both pressed

For g

	$\bar{P}_2$	P_2
$\bar{P}_1$	1	1
P_1	1	1

$g = P_1 + P_2$

For d

	$\bar{P}_2$	P_2
$\bar{P}_1$	1	1
P_1	1	1

For c

	$\bar{P}_2$	$\bar{P}_2$
P_1	1	1
$\bar{P}_1$		

$C = \bar{P}_1$

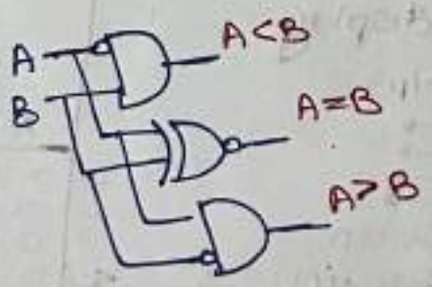
For e

	$\bar{P}_2$	P_2
$\bar{P}_1$	1	
P_2	1	1

$e = \bar{P}_2 + P_1$

- one bit comparator

A	B	$\bar{A}B$ A<B	A⊙B A=B	$A\bar{B}$ A>B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0



Note

- arrow that is pointing then that variable will get bar
  ie A<B ⇒ $\bar{A} \cdot B$

- NAAK Jisko Dikha Raha hai Usko EK Tappad Maro
- Jiske Upper tapped Hai usko Naak Dikhaoo.

$A \rightarrow A_3 A_2 A_1 A_0$

$B \rightarrow B_3 B_2 B_1 B_0$

if $A_3 > B_3$

1 x x x
0 x x x

A > B

if $A_3 < B_3$

0 x x x
1 x x x

A < B

if $A_3 = B_3$

if $A_2 > A_3$

A_3 1 x x
B_3 0 x x

A > B

if $A_3 = B_3$

if $A_2 < B_2$

A_3 0 x x
B_3 1 x x

A < B

So. A = B

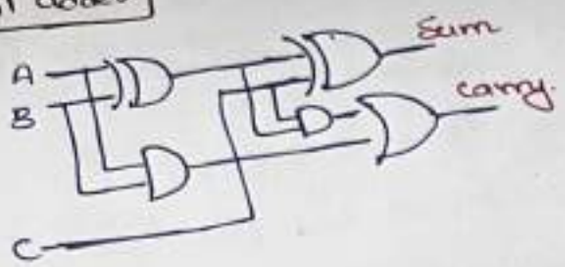
↳ $(A_3 \odot B_3)(A_2 \odot B_2)$
$(A_1 \odot B_1)(A_0 \odot B_0)$

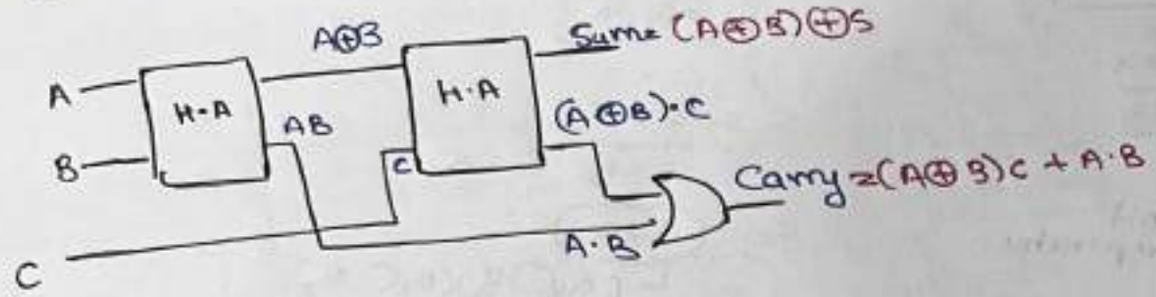
- 4 bit comparator.

$A_3 B_3$	$A_2 B_2$	$A_1 B_1$	$A_0 B_0$	A>B	A=B	A<B
$A_3 > B_3$	x	x	x	1	0	0
$A_3 < B_3$	x	x	x	0	0	1
$A_3 = B_3$	$A_2 > B_2$	x	x	1	0	0
$A_3 = B_3$	$A_2 < B_2$	x	x	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 > B_1$	x	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 < B_1$	x	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 > B_0$	1	0	0
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 < B_0$	0	0	1
$A_3 = B_3$	$A_2 = B_2$	$A_1 = B_1$	$A_0 = B_0$	0	1	0

$A > B \Rightarrow A_3 \bar{B}_3 + (A_3 \odot B_3) \cdot (A_2 \cdot \bar{B}_2)$
$+ (A_3 \odot B_3) \cdot (A_2 \odot B_2)(A_1 \cdot \bar{B}_1)$
$+ (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \bar{B}_0)$

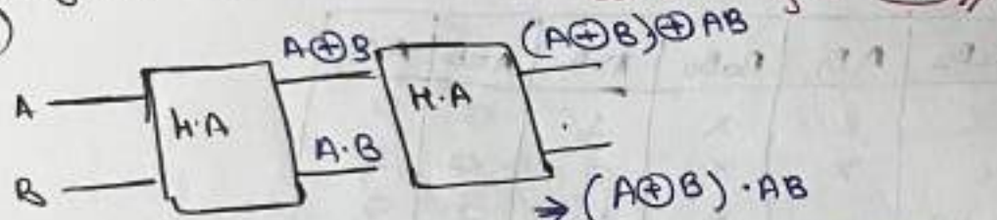
Full adder.





w.B (pg. no. 65).

(T6)



$\Rightarrow (A \oplus B) \cdot AB$

$\Rightarrow (\bar{A}B + A\bar{B}) \cdot AB$.

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

$= 0 + 0$

$= 0$ // carry

$A \rightarrow A_3A_2A_1A_0$

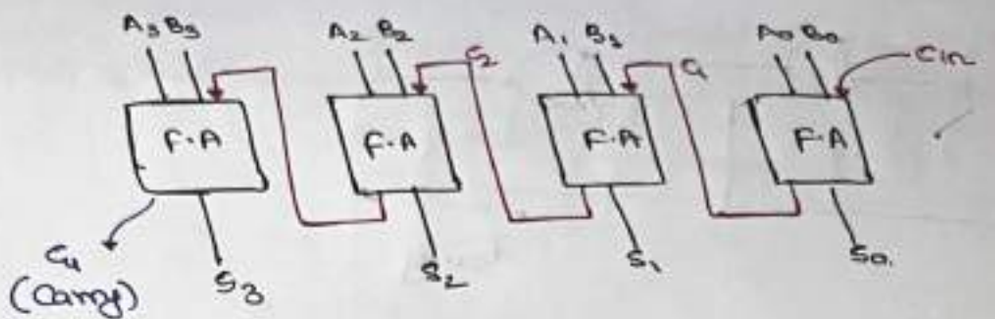
$B \rightarrow B_3B_2B_1B_0$

$[A_3A_2A_1A_0]$
1 1 1 1
1 1 1 1
$[B_3B_2B_1B_0]$
$C_3\ C_2\ C_1$
1 1 1 C_0 ← C_{in}
C_4 1 1 1 0

Ripple carry adder

(or)

Parallel binary adder.



"N" F.A (Full adder) [if carry in present (Cn)]

(N-1) (F.A) + 1 H.A [if carry in not present] ($C_0 = 0$)

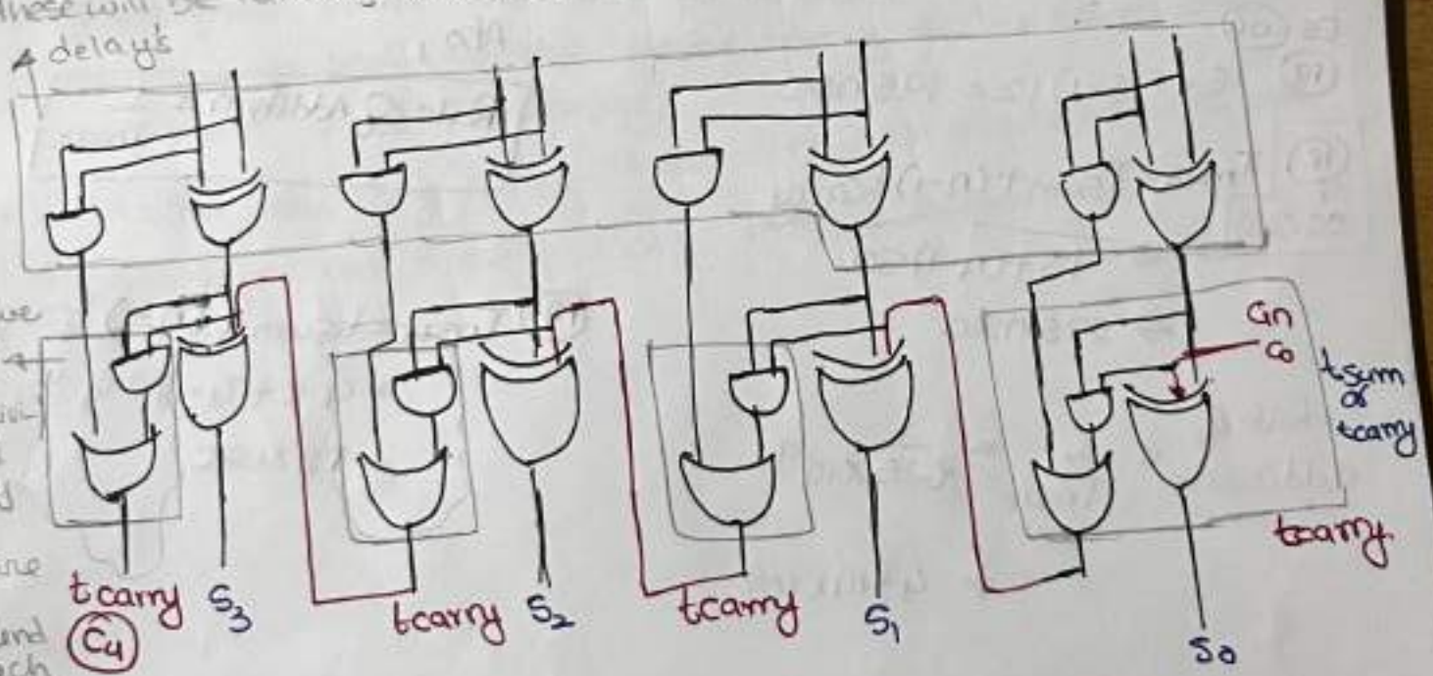
So, (N-1) [2 H.A + 1 "OR" gate] + 1 H.A.

(2N-1) H.A + (N-1) OR gate

Ex: N=4
→ (2×4-1)
⇒ 8-1 = 7 H.A

(N-1) = 4-1
= 3 "OR" gate

these will be running all in parallel so we don't need to add individual delays



Hereone need to add individual delay b/c The are inter dependent on each other

$T_{max} = t_{sum} + 3\,t_{carry}$

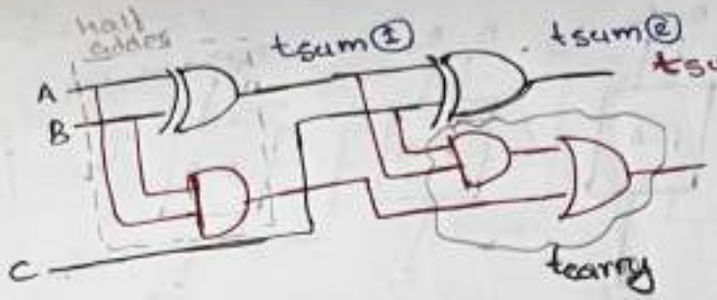
t_{sum} ② + t_{sum} ④ + 3 t_{carry}.

t_{carry}

in generally,

$T_{max} = t_{sum} + (N-1)\,t_{carry}$

• Time delay in 1 full adder.



$t_{sum} = t_{sum①} + t_{sum②}$

Here is b/c both will 2 level so we can take these

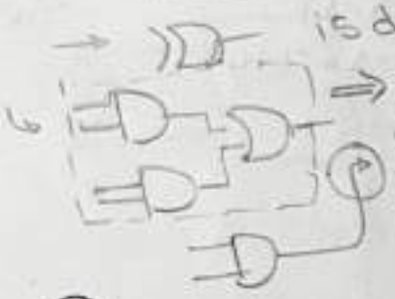
if $t_{sum②} = t_{carry}$

$T_{max}[delay] = t_{sum}$ or worst case delay

$t_{sum①} + t_{sum②}$ or t_{carry}.

$T_{max}[delay] = t_{sum①} + max[t_{sum②}, t_{carry}]$

$t_{sum②} \neq t_{carry}$



is divided in 2 level

⇒ So by the time XOR will complete it's operation parallel these "AND" gate will also complete it's delay. So will write the max delay among it if it's parallel connected

Also,

Rate of Addition $= \frac{1}{T_{max}}$

CC (WB)

(19) $15 + (16-1)12 = 195$ nsec.

(18) or CS (23) $T_{max} = T_{sum} + (n-1)t_{carry}$

$\Rightarrow 75 + (4-1)50$

$\Rightarrow 225$ nsec

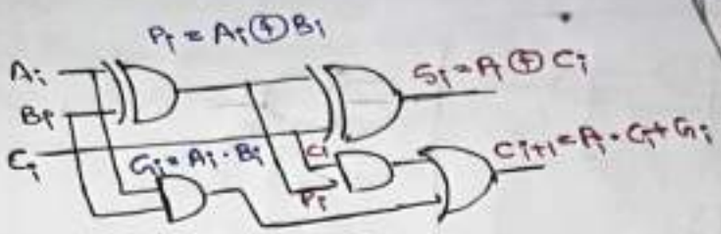
Rate of additions $= \frac{1}{T_{max}} = \frac{1}{225 \times 10^{-9}}$

$\Rightarrow 4.44 \times 10^{6}$.

(17) $T_{max} = t_{sum} + (n-1)t_{carry}$

$= 4.8 + (4-1)2.4$

$= 12$ μsec



—[Note]—

- For "n" bit parallel binary adder carry propagation take place through "2·n" logic gate

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

$C_{i+1} = C_{0+1} = C_1 = P_0 C_0 + G_0$

$= C_{1+1} = C_2 = P_1 C_1 + G_1 = P_1 \cdot [P_0 C_0 + G_0] + G_1$

$= P_1 P_0 C_0 + P_1 G_0 + G_1$ (C_1) ✗ A doesn't contain C so it's not dependent.

$= C_{2+1} = C_3 = P_2 C_2 + G_2 = P_2 [P_1 P_0 C_0 + P_1 G_0 + G_1] + G_2$

$= P_2 P_1 P_0 C_0 + P_2 P_1 G_0 + P_2 G_1 + G_2$

applied carry

$= C_{3+1} = C_4 = P_3 C_3 + G_3 = P_3 [P_2 P_1 P_0 C_0 + P_2 P_1 G_0 + P_2 G_1 + G_2] + G_3$

$= P_3 P_2 P_1 P_0 C_0 + P_3 P_2 P_1 G_0 + P_3 P_2 G_1 + P_3 G_2 + G_3$

—[Note]—

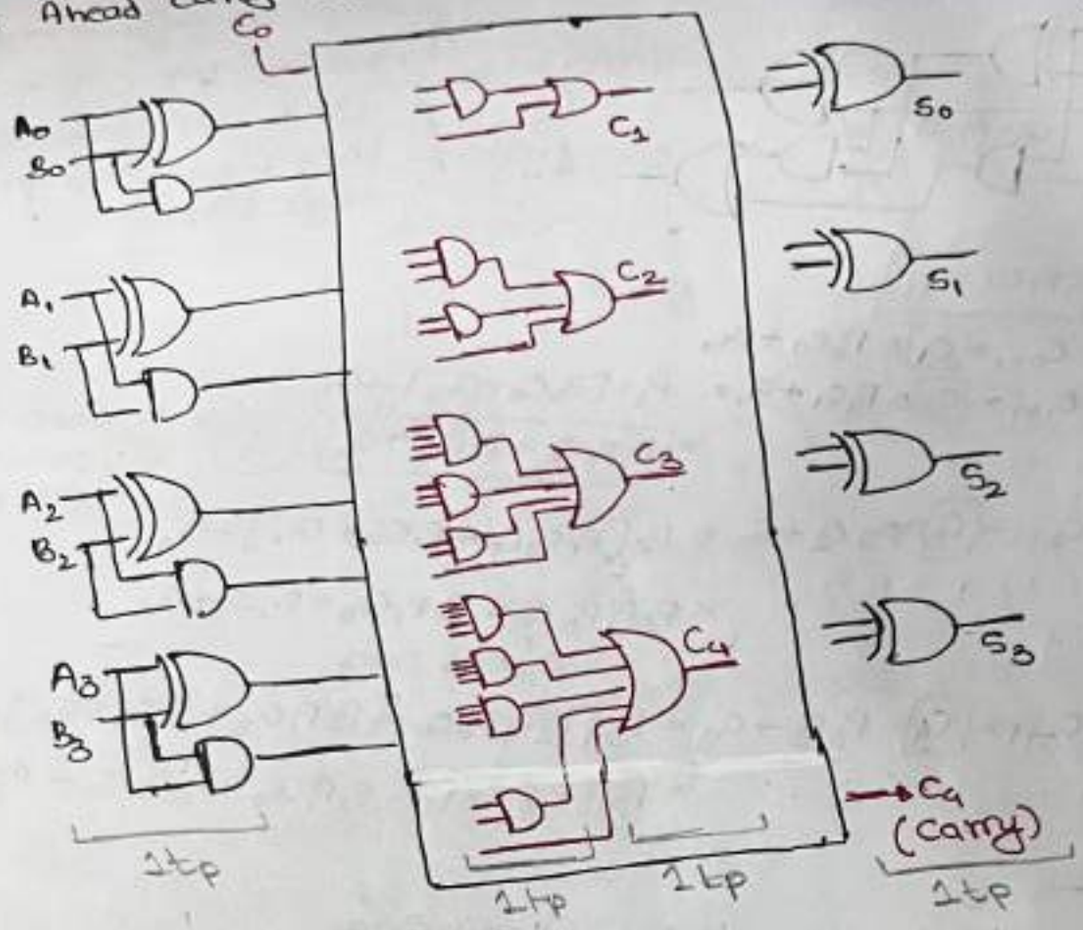
- Carry propagation delay is the disadvantage in the Ripple carry adder

To resolve these issue we use Look ahead carry adder.

↳ here the problem is the interdependency blw the Adder due to which there is a more delay

↳ In Lookahead Carry adder there is no such interdependency bcoz. the Adder, we are directly adding in that Adder the carry

- Look Ahead carry adder.

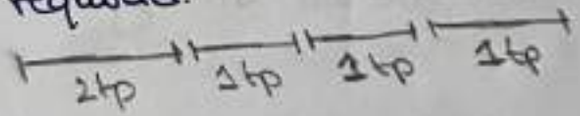


⇒ (4tp) in total time taken.
for all no. of bit i.e 4bit, 8bit - - - nbit.

- it is only (4tp) when

AND
OR
Ex-OR
Compliment IS available

- if XOR is not given then we need to construct it using "AND" & "OR".
So, "2" additional "tp" will be required.



- no. of stage are constant
Independent of 'n'

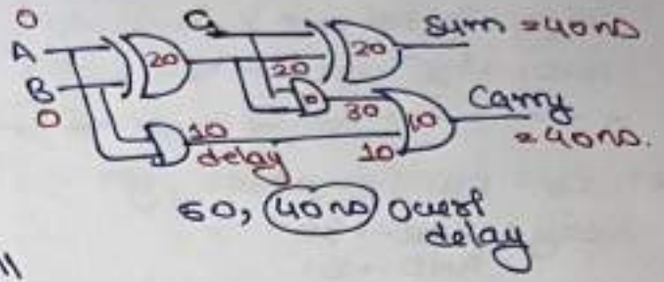
- Look ahead carry adder

ex: consider an implementation of FA using 2 HA

Delay of XOR → 20 nsec

AND/OR → 10ns

Total delay of F.A

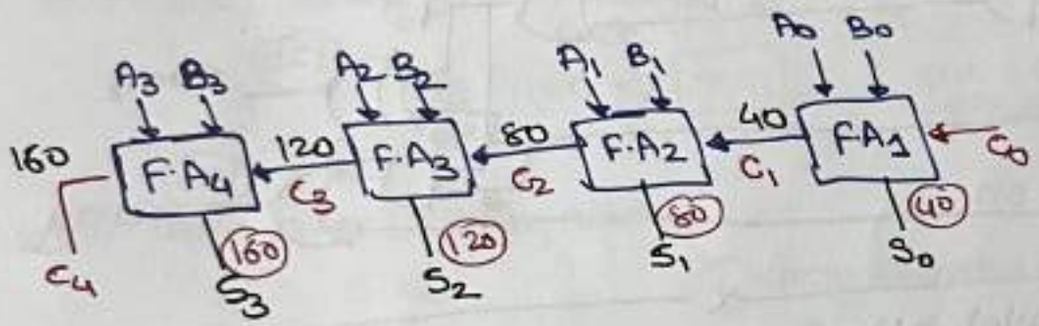


1 H.A delay ⇒ Sum = 20 delay } overall delay is 20ns.

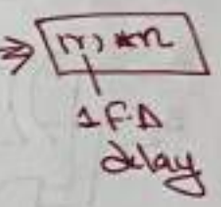
Carry = 10 delay }

ex: 4bit parallel adder (Ripple carry adder)

1FD → 40ns.



generalize :- 'n' bit parallel adders.

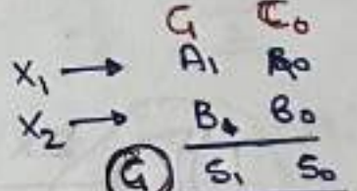


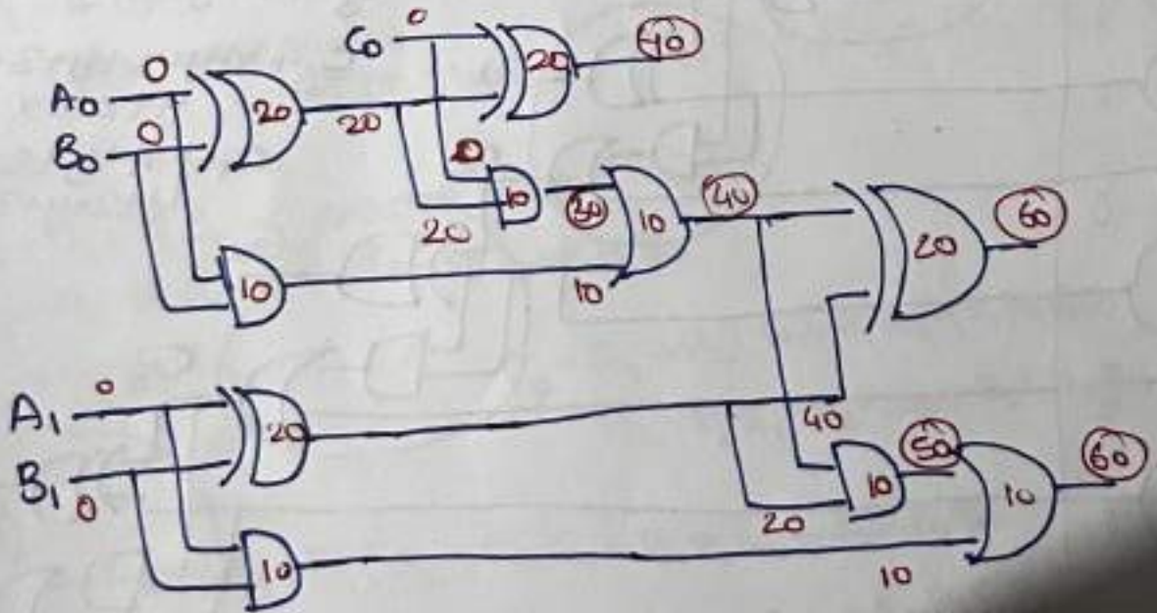
ex: 2bit parallel adder implemented by H·A

1XOR = 20ns

1AND/OR = 10ns

60 delay





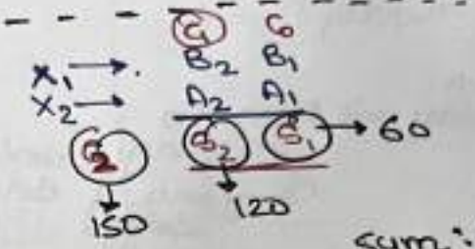
- Total delay for 'n' bit parallel adder if it's implemented using HA & delay of XOR → 'x', AND → 'y', OR → 'z'

Sum → 2x + (n−1)*(y+2) (for 1st bit sum; for (n−1) sum bit)

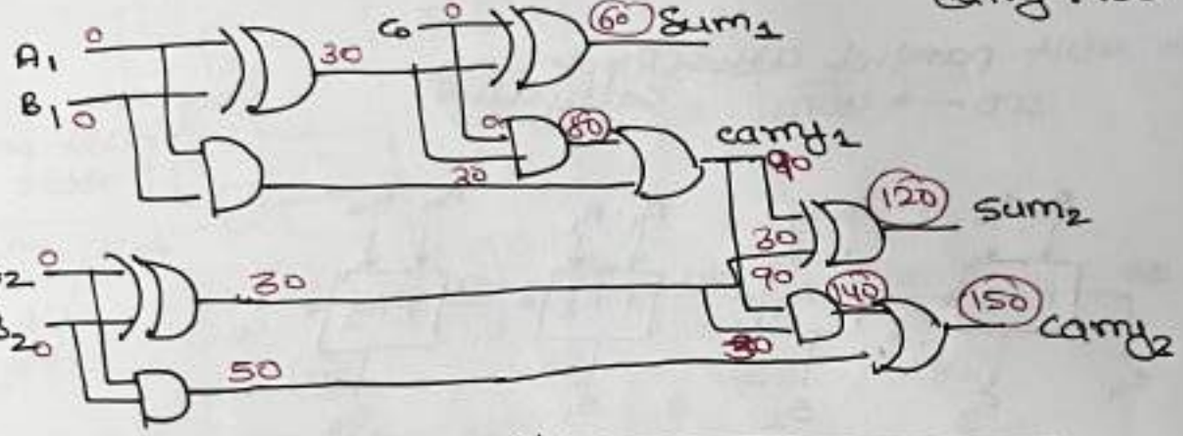
Carry → (x+y+z) + (n−1)(y+2) (for 1st bit carry; (n−1) carry bit)

ex: 2bit parallel adder

delay: XOR → 30
AND → 50
OR → 10

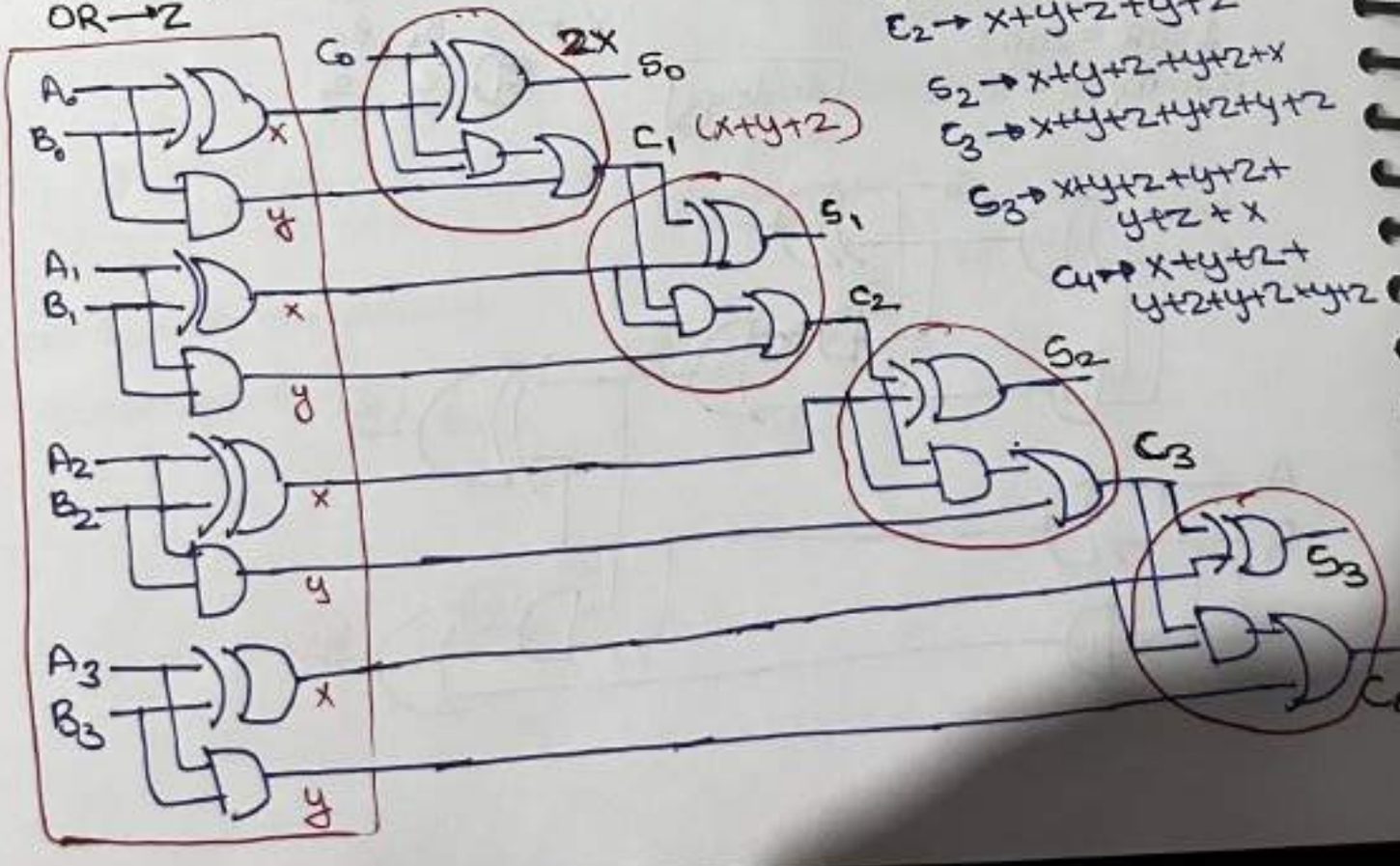


Sum: 120
Carry: 150



i.e 'n' bit parallel adder.

XOR → x
AND → y
OR → z



$S_0 \rightarrow 2x$

$C_1 \rightarrow x+y+z$

$S_1 \rightarrow x+y+z+x$

$C_2 \rightarrow x+y+z+y+z$

$S_2 \rightarrow x+y+z+y+z+x$

$C_3 \rightarrow x+y+z+y+z+y+z$

$S_3 \rightarrow x+y+z+y+z+y+z+x$

$C_4 \rightarrow x+y+z+y+z+y+z+y+z$

• Sum: $2x+(n-1)(y+z)$ → $x + x + (n-1)(y+3)$

• Carry: $(x+y+z)+(n-1)(y+z)$ → $x + (y+z) + (n-1)(y+z)$

only difference. so these will tell either sum ya carry is impacting

① Sum; $x > y+z$

② carry; $x < y+z$

• Disadvantage:

• 'n' bit parallel adder

• As 'n' increase then delay increase linearly

↳ if implementation by H·A ⇒ $O(n)$

↳ if implementation by F·A then Total delay ⇒ n * (delay of each F.A)

$n * c \Rightarrow O(n)$

▸ In case of H·A we are already calculating $A_0B_0, A_1B_1, A_2B_2 \ldots$ at some time i.e '0' time

▸ But in case of F·A we can't do that b/c until we get the 'C_i' we can't calculate futher.

4bit look ahead carry adder.

C_4 C_3 C_2 C_1 C_0

$n_1 \rightarrow$ A_3 A_2 A_1 A_0

$n_2 \rightarrow$ B_3 B_2 B_1 B_0

S_3 S_2 S_1 S_0

↳ carry generated at i^{th} stage

$G_i = A_i \cdot B_i$

$P_i = A_i \oplus B_i$

↳ carry propagated from i^{th} stage

① carry come's when.

① 0 / 1 / 0 ① 1 / 1 / 1 ① 1 / 0 / 0 ① 1 / 0 / 0

$C_1 \Rightarrow A_0 \cdot B_0 + (A_0 \oplus B_0) \cdot C_0$

$C_2 \Rightarrow A_1B_1 + (A_1 \oplus B_1) \cdot C_1$

$C_3 \Rightarrow A_2 \cdot B_2 + (A_2 \oplus B_2) \cdot C_2$

$C_4 \Rightarrow A_3 \cdot B_3 + (A_3 \oplus B_3) \cdot C_3$

$C_1 = G_0 + P_0 \cdot C_0$

$C_2 = G_1 + P_1 C_1$ $\Rightarrow G_1 + P_1(G_0 + P_0 \cdot C_0)$

$C_3 = G_2 + P_2 C_2$ $G_1 + P_1G_0 + P_1P_0C_0$

$C_4 = G_3 + P_3 \cdot C_3$

$C_3 = G_2 + P_2(G_1 + P_1G_0 + P_1P_0C_0)$

$G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

$C_4 = G_3 + P_3(G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0)$

$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$

- For 'n' bit look ahead carry adder.

Stage	gate	I/p	number
1	XOR	2	n
	AND	2	n
2	AND	$C_1 \to 2$ $C_2 \to 2,3$ $C_3 \to 2,3,4$ $C_n \to 2,3,4,\cdots(n+1)$	$C_1 \to 1$ $C_2 \to 2$ $C_3 \to 3$ $C_n \to n$ $\frac{n(n+1)}{2}$
3	OR	$C_1 \to 2$ $C_2 \to 3$ $C_3 \to 4$ $C_n \to n+1$	$C_1 \to 1$ $C_2 \to 1$ $C_3 \to 1$ $C_n \to \frac{1}{n}$
4	XOR	$S_0 \to 2$ $S_1 \to 2$ $S_2 \to 2$ $S_n \to 2$	$S_0 \to 1$ $S_1 \to 1$ $S_2 \to 1$ $S_n \to \frac{1}{n}$

↳ Stage 1 & 4 need only 2 I/p gate

↳ Stage 2 & 3 need upto (n+1) I/p gate

Variable

ex: Time complexity of 'n' bit look ahead carry adder is O(1) (const.)

↳ carry is available in 3rd stage
↳ we need 4th stage for sum

Assuming (n+1) I/p AND & OR gates are available & each gate had const. delay.

ex: Amount of time needed to generate sum & carry by looking ahead carry adder if delay of XOR, AND, OR gate is 50, 10, 20. regardless of no. of I/p to these gates.

C_n [3 Stage]

$S_1 \rightarrow$ [XOR = 50, AND = 10] = 50

$S_2 \rightarrow$ [AND] = 10 $\Rightarrow$ 50+10 = 60

$S_3 \rightarrow$ [OR] = 20 $\Rightarrow$ 60+20 = 80

$C_n = 80$

S_{n-1} = 4th stage

80+50 = (180) //

S_{n-1} (Sum need 4 stage)

S_1 = 2 I/p XOR $\Rightarrow$ 5*2 = (10)
(Stage 1)
2 I/p AND = 2*2 = (4)
} (10) //

Stage 2 $\Rightarrow$ (AND)

$S_8 \Rightarrow$ 4 I/p AND = 2*4 = (8)

Stage 8 $\Rightarrow$ (OR)

C_3 = 4 I/p OR $\Rightarrow$ 4*1 = 4

Stage 4 $\Rightarrow$ (XOR)

2 I/p $\Rightarrow$ 5*2 = 10

**ex: amount of time needed to generate sum & carry by (4) bit look ahead carry adder if delay of XOR, AND, OR gate is 5n, 2n, n where n is no. of I/p to these gate

Carry need 3 stage

(C4) most time taken

▸ Stage 1 (S_1)

2 I/p XOR = 5x2 = (10) larger delay

2 I/p AND = 2x2 = (4)

▸ Stage 2 (S_2)

AND gate

2 I/p, 3 I/p, 4 I/p, (5 I/p) largest

So, 5 I/p AND = 2x5 = (10)

Total delay = $S_1 + S_2$ = 10+10 = (20)

▸ Stage 3 (S_3) [OR gate] 4 I/p = 2, 3, 4, (5)

5 I/p OR $\Rightarrow$ 5x1 = 5 = 5

Total delay = 20+5 = 25
$S_1 + S_2 + S_3$

Total delay = $S_1 + S_2 + S_3 + S_4$
= 10+8+4+10 = (32)

Sol. C_n ⇒ 3 stage

$S_1 = 10$

$S_2 = (n+1) * 2 = 2n+2$

$S_3 = (n+1) * 1 = n+1$

$C_n = 3n+13$

S_{n-1} (Sum) ⇒ 4 stage

$S_1 = 10$

$S_2 = n * 2 = 2n$

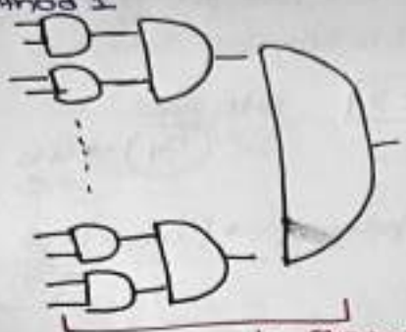
$S_3 = n * 1 = n$

$S_4 = 10$

$S_{n-1} = 3n+20$

• generalize

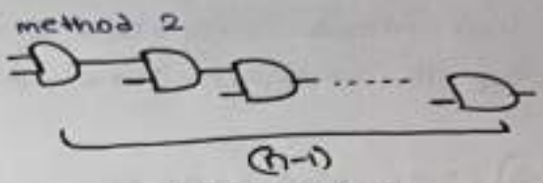
method 1



ex: 32 I/p AND gate given 2 I/p AND gate

k stage

method 2



(n-1)

Total delay

$(n-1)*x$ → delay

method 1:

$S_1 \rightarrow \frac{32}{2} = 16$

$S_2 \rightarrow \frac{16}{2} = 8$

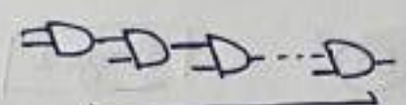
$S_3 \rightarrow \frac{8}{2} = 4$

$S_4 \rightarrow \frac{4}{2} = 2$

$S_5 \rightarrow \frac{2}{2} = 1$ (Stage 5)

31

method 2



31

$5 \times 2 = 10$

$30, 31 \times 2 \rightarrow (62)$

method 1.

$S_1 \rightarrow \frac{n}{2}$

$S_2 = \frac{n}{2^2}$

$S_3 = \frac{n}{2^3}$

⋮

K stage

$S_k = \left(\frac{n}{2^k}\right) = 1$

$\frac{n}{2^k} = 1 \Rightarrow n = 2^k$

$k = \log_2 n$

& Total delay is $x * \log_2 n$

▸ if it's not an power of 2 then $\text{stage} = \lceil \log_2 n \rceil$

□ x I/p AND gate realize using y I/p AND gate

$\text{Delay} \rightarrow \lceil \log_y x \rceil$

note

carry = ab+bc+ac

same as

carry = ab + (a⊕b)·c

ex: Amount of time needed to generate sum & carry by 4bit look ahead carryadder if delay of XOR, AND, OR gate of 2 I/P is 50, 10, 20. Time complexity : $\Theta(\log n)$

carry:

Stage 1: 2 I/P XOR → 50; 2 I/P AND → 10 } (50)

Stage 2: AND gate (C4) → 4+1 = 5

Substage → $\lceil \log_2 5 \rceil * 10 = 30$

Stage 3: OR gate (C4) → 4+1 = 5

Substage = $\lceil \log_2 5 \rceil * 20 = 60$

So, Total delay is $S_1+S_2+S_3 \Rightarrow 50+30+60 \Rightarrow (140)$

Sum:

Stage 1 ⇒ (50)

Stage 2 ⇒ AND gate

(S_3) = 3+1 = 4

Substage = $\lceil \log_2 4 \rceil * 10 = 2*10 = 20$

Stage 3 ⇒ OR gate

(S_3) → 3+1 = 4

Substage = $\lceil \log_2 4 \rceil * 20$

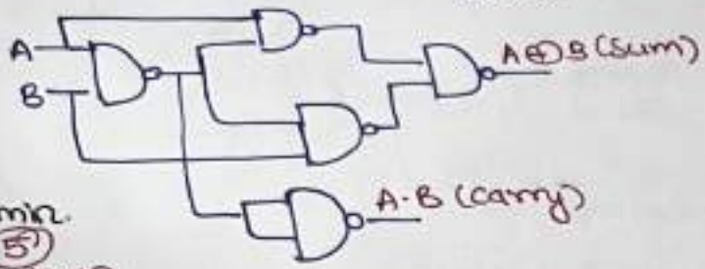
$= 2*20 = (40)$

Stage 4: XOR gate (50)

Total delay = $S_1+S_2+S_3+S_4 \Rightarrow 50+20+40+50$

$\Rightarrow (160)$ //

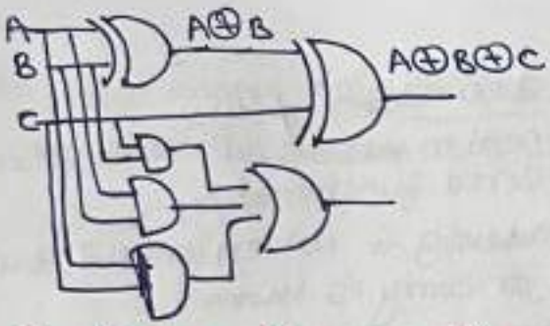
- to design Half adder we need 1 XOR & 1 AND gate
- no. of NAND gate to build H.A (Half adder)



min. (5) NAND gate

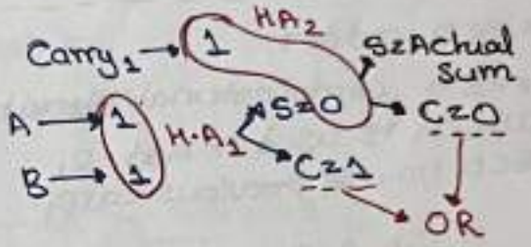
- min. no of NOR gate to build H.A
(5) NOR gate

- Full adder



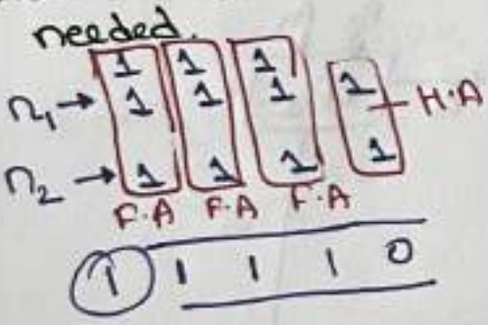
To realize FA we need 3 AND, 1 OR, 2 XOR gate

ex: How many HA are needed to realize F.A?



So, 2 H.A & 1 OR gate is required

ex: To add 2, 4bit no..
How many H.A & F.A are needed.



So, 3 F.A & 1 H.A
In worst case required

we can say that 4 F.A only

or 4 H.A only

generalize;
to add 2, n bit no.

1 H.A & (n-1) F.A only

only (2n-1) H.A

only n F.A

ex'. no. of H.A & F.A are needed to design 4bit parallel adder.

2, 4bit no. added & previous carry

4 F.A
0 & 8 H.A
} bc we are also adding previous carry

Generalize
'n' bit parallel adder
↳ n F.A
↳ 2n H.A
• if delay of each F.A = 'd'
Total delay = (n*d)

• BCD adder.
↳ it's a combinatorial circuit which is used to add 2, BCD no. & previous carry.

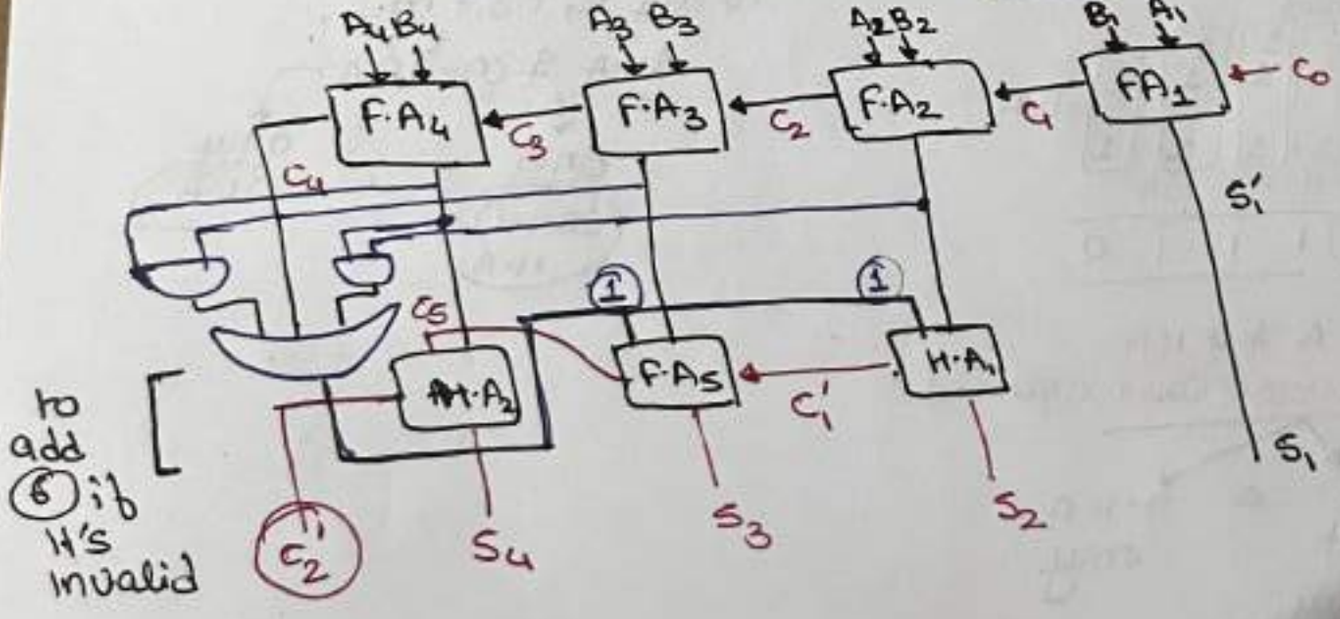
5 F.A & 2 H.A

ex'. C_3 C_2 C_1 C_0
A_4 A_3 A_2 A_1
B_4 B_3 B_2 B_1

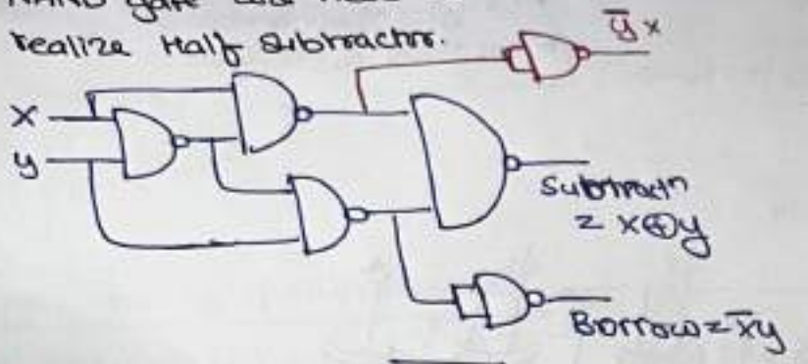
ex'. BCD adder ≠ 4bit parallel adder
need to add extra functionality
adding 6 when invalid BCD or carry is there

i.e 0111 → (7)10
0100 → (4)10

1011
+ 0110
carry 1 0001
1
(11)10



ex: min no. How many NAND gate are need to realize Half subtractor.

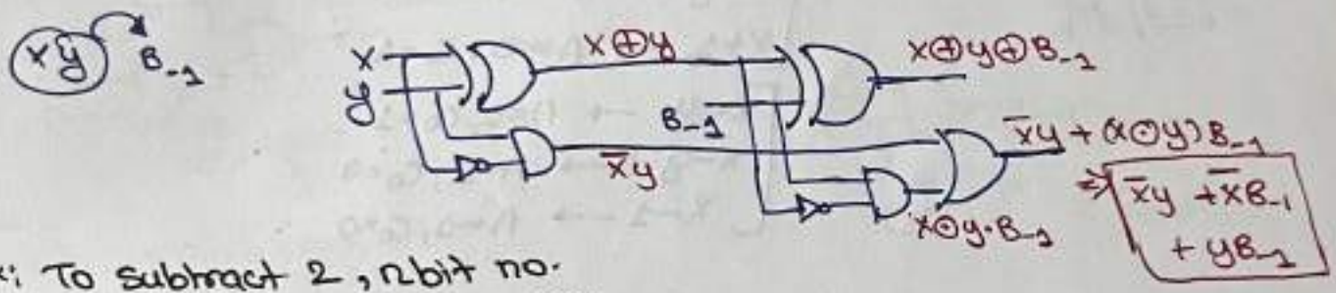


$\overline{\overline{x \cdot y} \cdot y} \Rightarrow \bar{y} + xy$

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

Half adder, subtractor then (5) Nand, NOR gate only

ex: How many Half Subtractor are needed to realize Full subtractor & what else do we need?



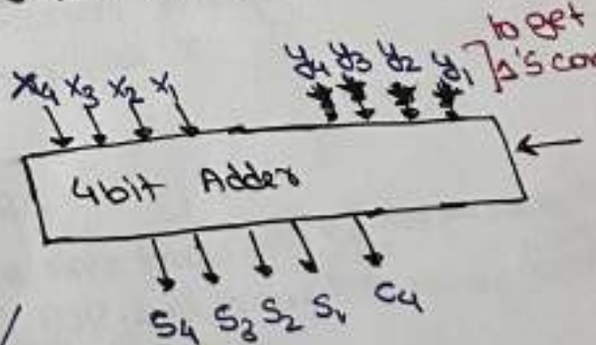
ex: To subtract 2, n bit no. How many H.S & F.S are needed?

(n-1) F.S & 1 H.S

if only F.S available then (n) F.S

& only H.S then (2n-1) H.S

• 4bit adder & subtractor



we can obtain

- x+y when $C_0 = 0$
- (x+y)+1 when $C_0 = 1$
- x+1 when $C_0 = 1$, $y = 0$

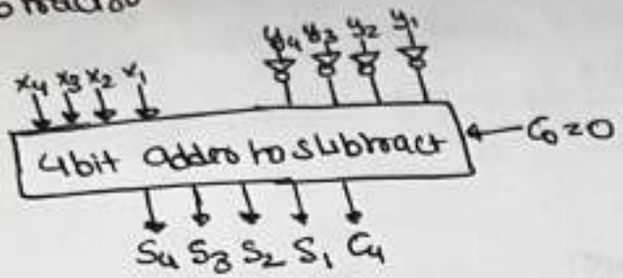
C_3 C_2 C_1 C_0

$x \rightarrow x_4 x_3 x_2 x_1$

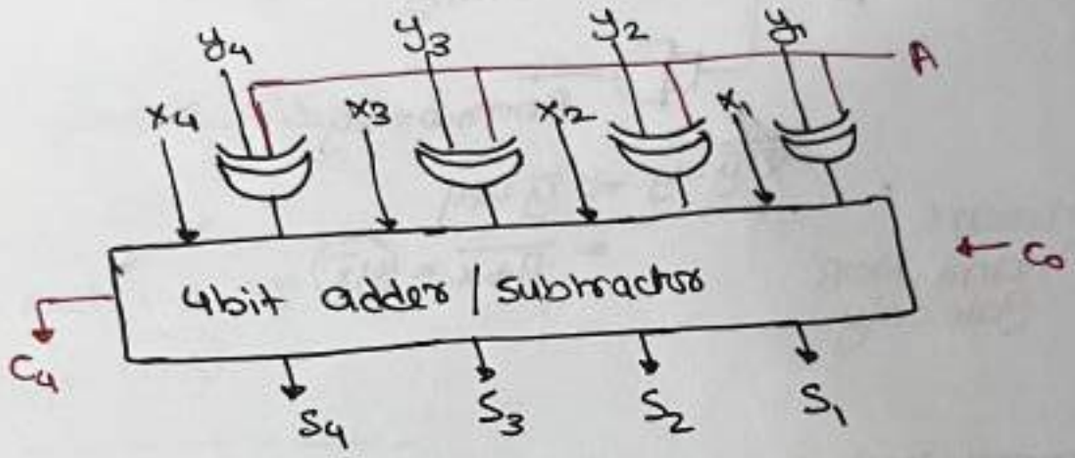
$+y \rightarrow y_4 y_3 y_2 y_1$

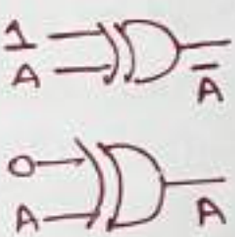
(C4) S_4 S_3 S_2 S_1

- Subtractor



- $(x-y)$ then $C_0=1$
- $(x-y)-1$ then $C_0=0$
- $x-1$ then $C_0=0, y=0$





$x+y \rightarrow A=0, C_0=0$

$x+y+1 \rightarrow A=0, C_0=1$

$x+1 \rightarrow A=0, C_0=1$

$x-y \rightarrow A=1, C_0=1$

$x-y-1 \rightarrow A=1, C_0=0$

$x-1 \rightarrow A=1, C_0=0$

H·A / H·S (Half Adder / Half Subtractor)

F·A / F·S

NAND	NOR
5	5
9	9

golden Rule

① SOP —(can easily be made by NAND)→ NAND —(to convert)→ POS

SOP ←(1st conversion)— POS

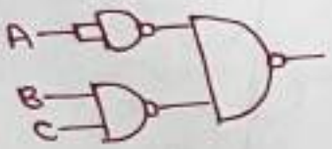
② POS → NOR ← SOP

POS —(1st conversion)→ SOP

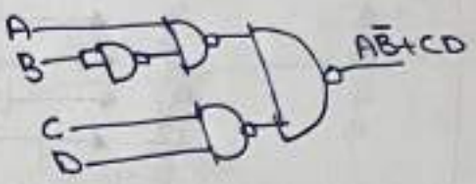
• min. no. of NAND/NOR gate • NAND

ex: $(A+B)(A+C) \Rightarrow A+B\cdot C$

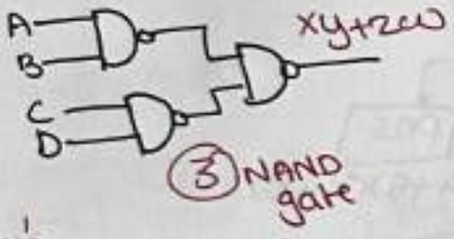
min. no. of NAND ③



ex: $A\bar{B}+CD$ (SOP) ④ NAND gate



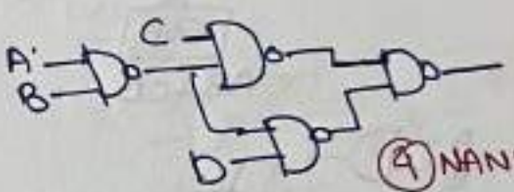
generalize



③ NAND gate

ex: $(\bar{A}+\bar{B})(C+D) \rightarrow \overline{AB}\cdot(C+D)$ (→ POS)

$\overline{AB}\cdot C + \overline{AB}\cdot D$



④ NAND

▸ Law's can't be applied on the expression

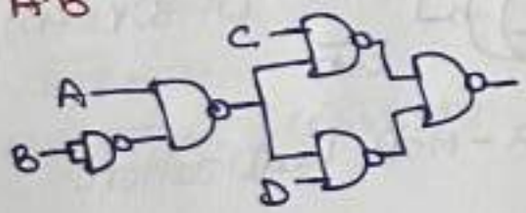
→ ▸ generalize Expression having Bar should be taken as x.

ex: $(\bar{A}+B)(C+D)$, $x = \bar{A}+B$

$x(C+D) = xC+xD$

$\bar{A}+B$ → $A\cdot\bar{B}$

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



ex: $\bar{A}B+\bar{B}C$

Some time there will be some exception that SOP will have more no. of NAND gate than POS NAND gate

↓ try to draw POS using K-map

$(\bar{A}+B)(B+C) \Rightarrow$ ④ NAND gate

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

ex. min no of 2 I/P NAND gate needed to realize

$x_1x_2 + x_3x_4 + x_5x_6 + \cdots + x_{2n-1}x_{2n}$

1 to 2n-1

$2+4+6+8+\cdots+2n \Rightarrow 2\underbrace{(1+2+3\cdots n)}_{n}$

$\Rightarrow \sum_{k=1}^{n} x_{2k-1}\, x_{2k}$

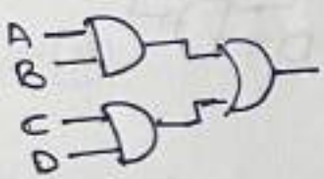
NAND

n	no. of NAND	
2	3	3(2-1) 3*1
3	6	3(3-1) 3*2
4	9	3(4-1) 3*3
5	12	3(5-1) 3*4
⋮	⋮	⋮
n		3(n-1)

funct^n

SOP

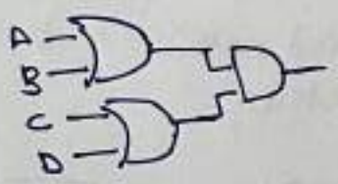
i.e AB+CD



AND-OR Realizat^n

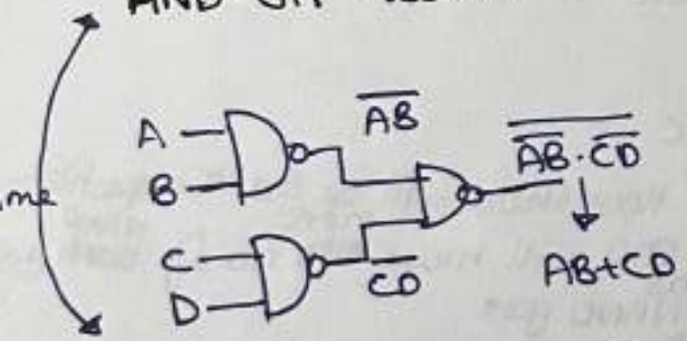
POS

(A+B)(C+D)

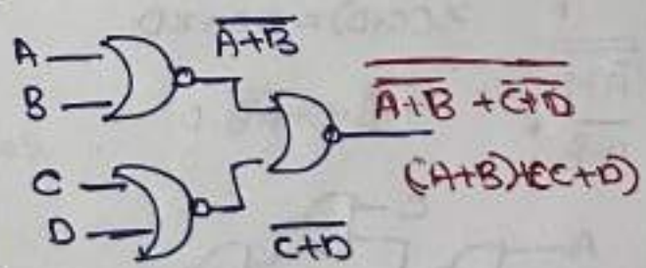


OR-AND Realizat^n

Same

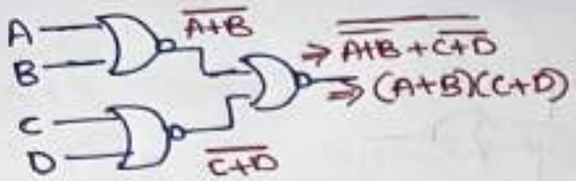


NAND-NAND Realizat^n



NOR-NOR Realization

$\underbrace{(A+B)(C+D)}_{POS}$ ③ NOR gate

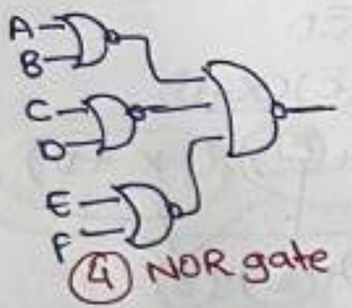


ex: $(A+B)(C+D)(E+F)$ min no. of NOR gate

(a) any no. of I/ps

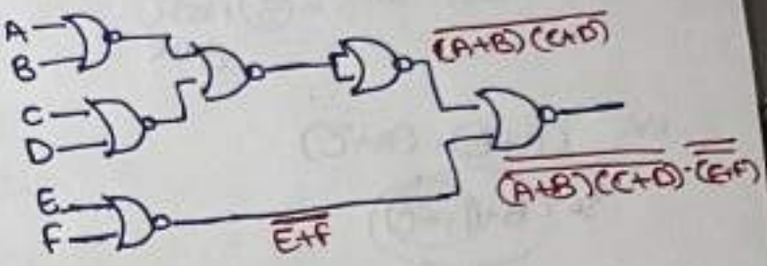
(b) 2 I/ps

(a)



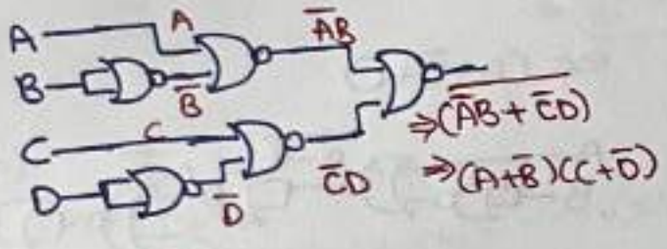
④ NOR gate

(b)



⑥ NOR gate

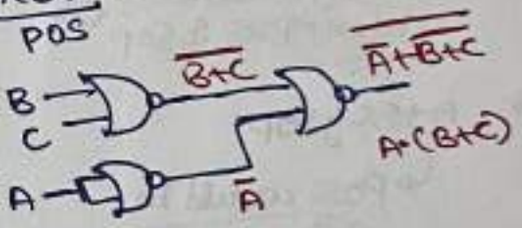
ex: $(A+\bar{B})\cdot(C+\bar{D})$



⑤ NOR gate

ex: $AB+AC$ } SOP

$\underbrace{A(B+C)}_{POS}$

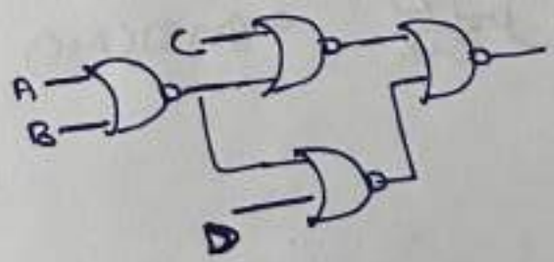


ex: $\bar{A}\bar{B}+CD$

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

$\overline{A+B}+CD$

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

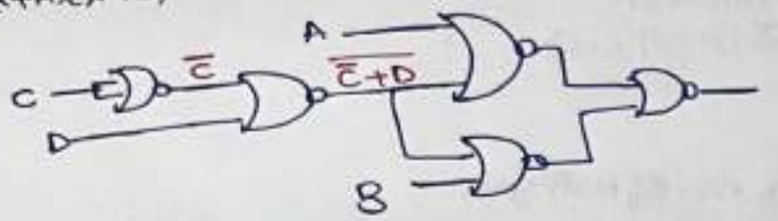


④ NOR gate

ex. $AB+\overline{C\overline{D}}$

$\Rightarrow (X+A)(X+B)$

$\left[\overline{\overline{C\overline{D}}}\;\;\overline{C+D}\right]$



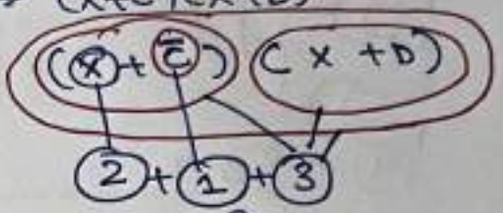
So, 2+3 = (5) NOR gate

ex. $\overline{A}B+\overline{C}D$

X → (2) NOR

$X+\overline{C}D$

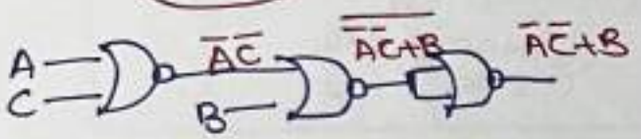
$\Rightarrow (X+\overline{C})(X+D)$



$\Rightarrow$ (6) NOR gate

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

$B+\overline{A}\cdot\overline{C}$



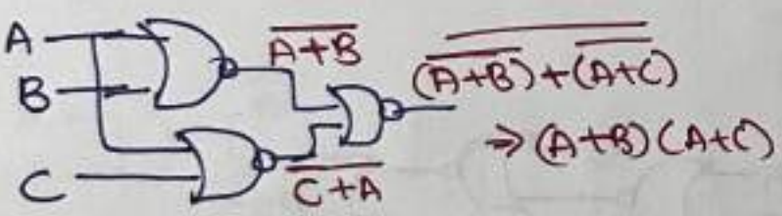
Note → when repetition of variable is there then try to find min b/w POS & SOP

ex. $A+BC$ } SOP

↳ POS could be

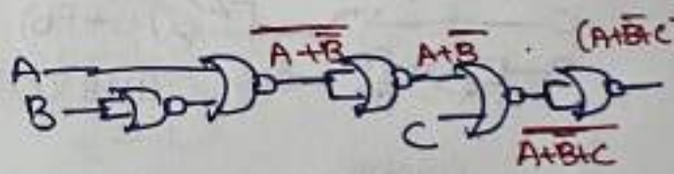
$(\overline{A}+B)(A+C)$

3 NOR gate



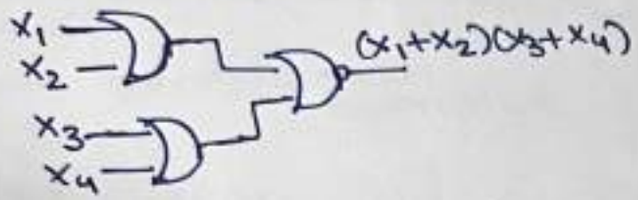
$\Rightarrow (A+B)(A+C)$

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

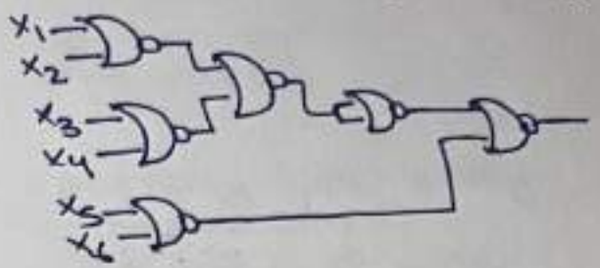


ex: min. no. of NOR gate needed to realize
$(x_1+x_2)(x_3+x_4)(x_5+x_6)\cdots\cdots(x_{2n-1}+x_{2n})$

n=2
$(x_1+x_2)(x_3+x_4)$



n=3
$(x+x_2)(x_3+x_4)(x_5+x_6)$



(n) 2 3 4 ... (n)

NOR	
3	3*1
6	3*2
9	3*3

(n) → $3(n-1)$ //

ex: $\overline{AB}+x_1\cdot x_2\cdot x_3\cdot\cdots x_n$

X → 2 NOR gate

$\Rightarrow 3(n-1)+2$
$\Rightarrow 3n-3+2$
$\Rightarrow (3n-1)$ //

ex: $\overline{AB}+x_1\cdot x_2\cdot x_3\cdot\cdots x_n$

X → 3 NOR gate

so, $3(n-1)+3$
$\Rightarrow (3n)$ //

ex. $x+x_1\cdot x_2x_3\cdots x_n$
$(x+x_1)\cdot(x+x_2)(x+x_3)\cdot\cdots(x+x_n)$
↳ require $3(n-1)$ NOR gate

Boolean Algebra

- Set of boolean funct^n ⑥ is called functionally complete or universal if any boolean function can be constructed from s.
  - → {AND, OR, NOT} is universal/ functionally complete
  - → {AND, NOT}
  - → {OR, NOT}
  - ↳ i.e (a+b, a·b, $\bar{a}$) OR AND NOT
- Boolean funct^n / Boolean operation

Digital Circuits

Digital circuit are used to realize Boolean or operators.

- A ckt / gate, or set of gate is called functionally Complete if any boolean funct^n can be realized using only instance of that ckt / gate or set of gate
- to check given set of function is functionally complete

Step 1: If only one funct^n is given then try to realize it by AND, OR, NOT gate & then treat it like black box

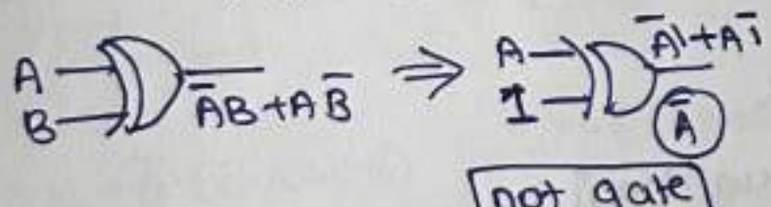
Step 2: Try to make {AND, NOT} or {OR, NOT} from given ckt.

- $\bar{A}+B$ & 0 (NOR) } P.F.C & $\bar{A}+B$ & $\bar{A}\cdot B$ here are P.F.C
- $\bar{A}\cdot B$ & 1 (NAND)

ex: $\bar{A}B+A\bar{B}$ is universal assuming V_{cc} & GND is available

↳ so, it's not an P.F.C

A ⊕ B. ↳ {AND, NOT} / {OR, NOT}



not gate

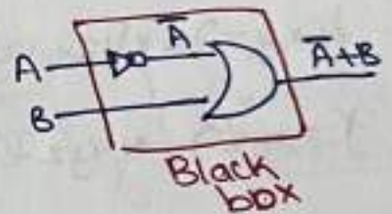
- AND & OR gate can't be created using XOR

- partially functionally complete (P.F.C) you need to take help of {0,1} to create all Boolean funct^n, without it you can't then it's called

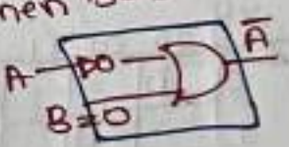
- A → B is PFC with false

ex: $\bar{A}+B$ is universal assuming V_{cc} & GND is available

V_{cc} = 1, GND = 0



when B=0



P.F.C

- Implication without false is P.F.C

- ex: $AB+\bar{A}\bar{B} \rightarrow A \odot B$ is also not P.F.C

ex: $A\bar{C}+BC$

Put A=0, then we get B.C [AND]

Put A=1 & B=0 then we get $\bar{C}$ [NOT]

- Functional Completeness

NAND, NOR, {$\overline{A+B}$ 0's, {$\bar{A}\cdot B$, 1}

Implication

$\overline{A\cdot B}$ ↕ $\bar{A}+\bar{B}$; $\overline{A+B}$ ↕ $\bar{A}\cdot\bar{B}$

ex: $\bar{A}+B\bar{C}$

when B=1

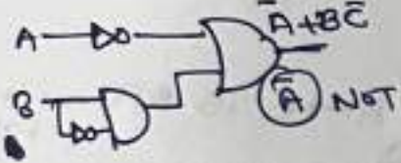
then; $\bar{A}+\bar{C}$ } NAND

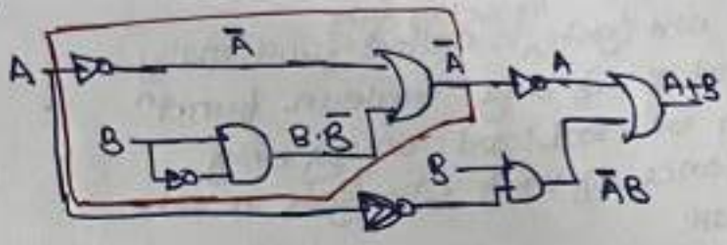
So, Funcn complete

ex: $\bar{A}+B\bar{C}$ is F.C??

V_{cc} & GND ↳ as OR

{$\bar{A}+B\bar{C}$} as NOT

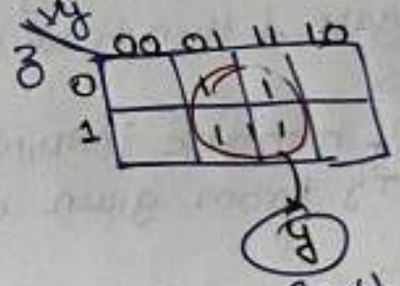




So, its Funcn complete.

gate

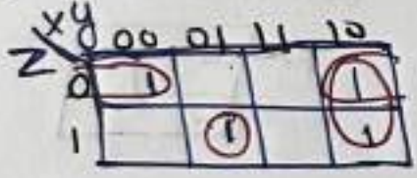
Q. $f(x,y,z)=x'yz+xy'+y'z'$ ⟶ using K-map



Can't prove anything so it's not functional complete

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

using Kmap



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

when Assuming ($x=y$)

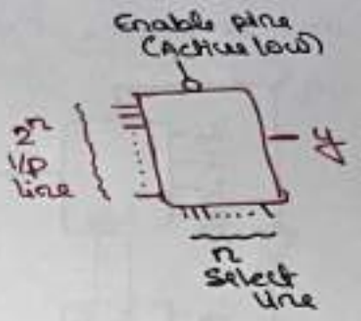
then

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

$\overline{x+z}$ NOR

So, it's functional complete

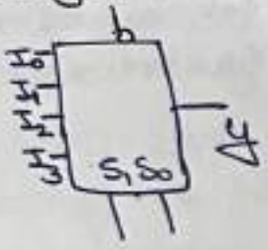
- Multiplexer / MUX / selector
- It is a C.C which has 2^n I/p lines, 1 o/p line, n select line & 1 enable pin



- Select lines are used to select which I/p line will appears on o/p line
- Enable pin is used to enable or disable the ckt

Enable pin is optional
- Active High (1)
- Active Low (0)

ex: Design 4x1 MUX



Step1: I/p & o/p lines

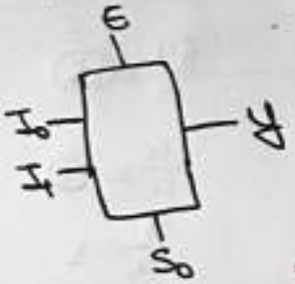
Step2: truth table

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

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

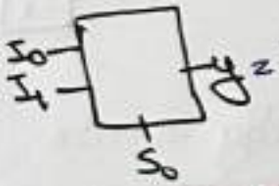
ex: A, $\bar{A}$, B, y $= A\bar{B} + \bar{A}B$ = A ⊕ B

ex: Explain 2x1 MUX



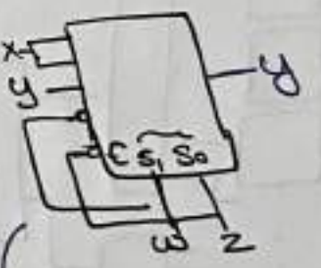
E	S_0	Y
1	0	I_0
1	1	I_1
0	X	0

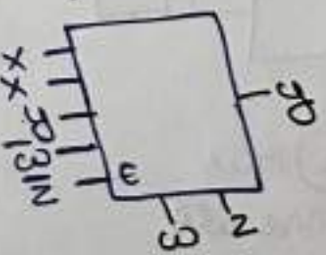
$y = \bar{E}\bar{S}_0 I_0 + \bar{E} S_0 I_1$



$y = \bar{S}_0 I_0 + S_0 I_1$

ex:

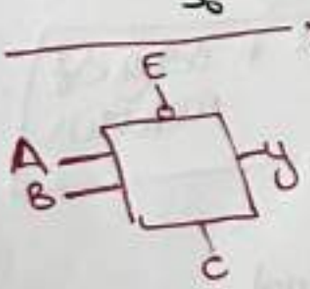




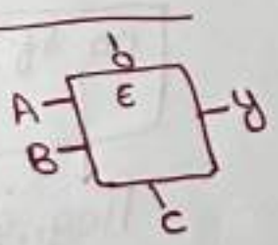
$y = (\bar{w}\bar{z}x + \bar{w}zx + w\bar{z}y + wz(\bar{w})\bar{z}$

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

$\Rightarrow \bar{w}\bar{z}x + w\bar{z}y$



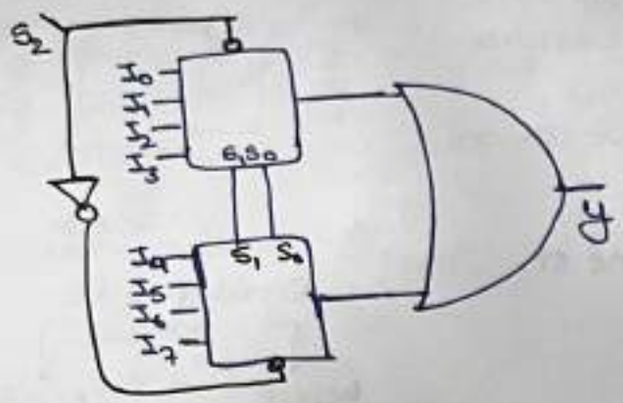
$y = \bar{E} \cdot (\bar{C} \cdot A + C \cdot B)$



$X = E(\bar{E}A + CB)$

$\bar{X}\ (\bar{C}A + CB)$

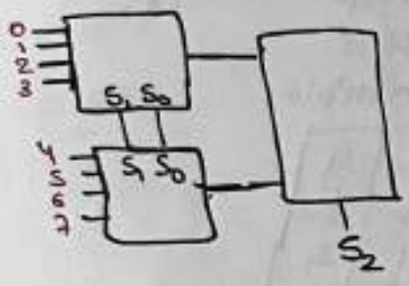
ex: How to realize 8x1 MUX from 4x1 MUX.



1 NOT
1 OR
2, 4x1 MUX

method (2)

(2) 4x1 MUX
(1) 2x1 MUX



ex. How many 4x1 MUX are needed to realize 64x1 MUX

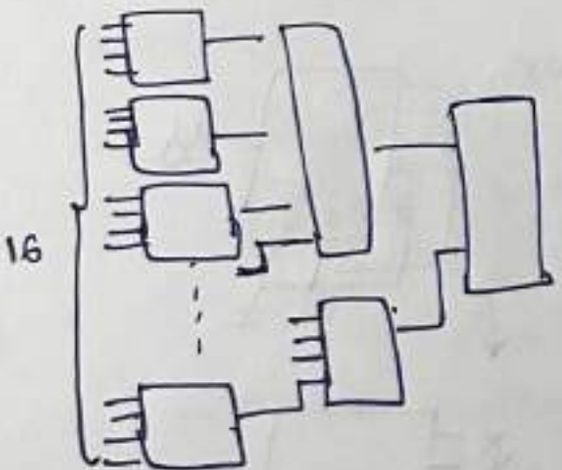
$\frac{64}{4} = 16$

$\frac{16}{4} = 4$

$\frac{4}{4} = 1$

no. of MUX → 21 4x1 MUX needed

ex: Consider realization of 64x1 MUX from 4x1 MUX if I/P given at select lines is 100110. find MUX no. & I/P line no. at stage (1) which is shown as o/p?



$\frac{64}{4} = 16$

$\frac{16}{4} = 4$

$\frac{4}{4} = 1$

21

100110 → 10 (2+1) 3rd line

(9+1) 10th MUX

(4x1) MUX

I/P line will be

00 — 1st
01 — 2nd
10 — 3rd
11 — 4th line

the, overall delay in MUX

no. of stage * delay of each MUX

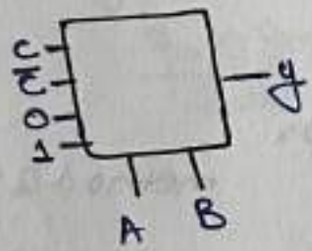
$\lceil \log_4 64 \rceil$ → given, desired

- Generalized [same for DeMux]
  $x \times 1$ MUX from $y \times 1$ MUX
- no. of MUX → $\sum_{k=1}^{\lceil \log_y x \rceil} \frac{x}{y^k}$
- no. of Stage → $\lceil \log_y x \rceil$

- Application of MUX
→ we can realize any boolean Functn of 'n' variable w/o minimizing it using $2^n \times 1$ MUX

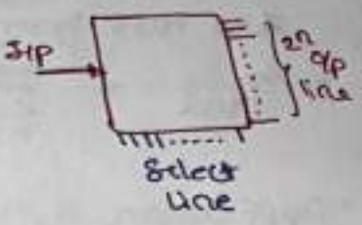
ex: can we realize any Boolean functn of degree 'n' by $2^{n-1} \times 1$ MUX? yes
ie $f(A,B,C) = \Sigma m(1,2,6,7)$
Realize this functn using 4×1 MUX

A B C	f(A,B,C)
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	0
1 1 1	1



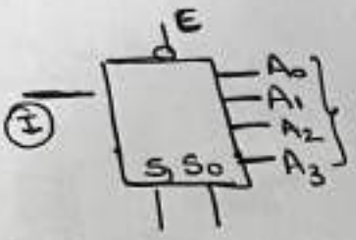
$y = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}0 + AB1$

- Demultiplexer / DeMUX:
- it is c.c which has 1 i/p line, 2^n o/p line, n select line & 1 enable pin (optional)



- select line are used to select o/p line on which i/p will appear
- enable pin is used to enable or disable the ckt.

ex: Step 1:



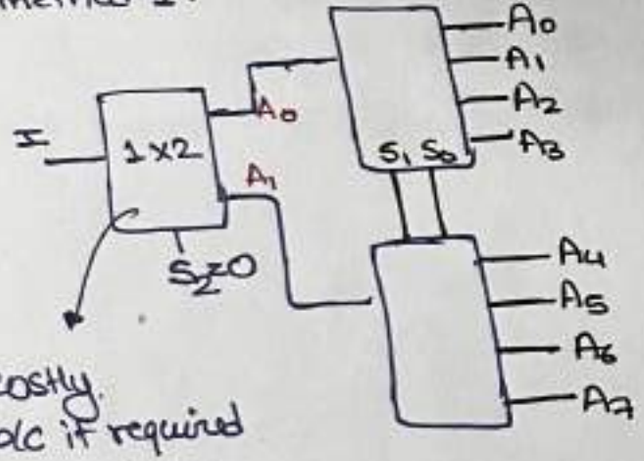
Step 2: truth table

E	S_1	S_0	A_0	A_1	A_2	A_3
0	0	0	I	0	0	0
0	0	1	0	I	0	0
0	1	0	0	0	I	0
0	1	1	0	0	0	I
1	x	x	0	0	0	0

$A_0 = \bar{E}\bar{S}_1\bar{S}_0 I$
$A_1 = \bar{E}\bar{S}_1 S_0 I$
$A_2 = \bar{E} S_1 \bar{S}_0 I$
$A_3 = \bar{E} S_1 S_0 I$

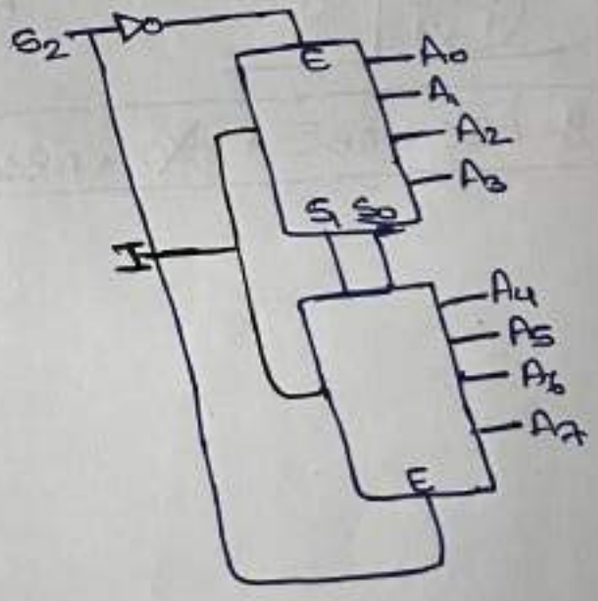
ex: 1x8 DeMUX with 1x4 DeMUX

method 1:



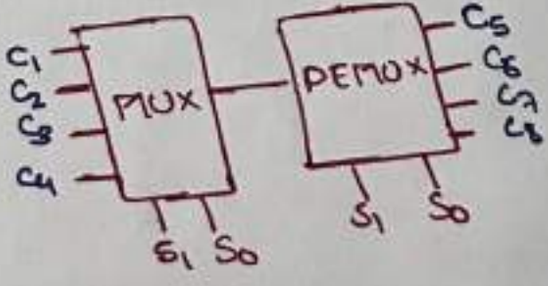
costly.
b/c if required
1 NOT gate
& 2 AND gate

method 2: Active high Enable pin

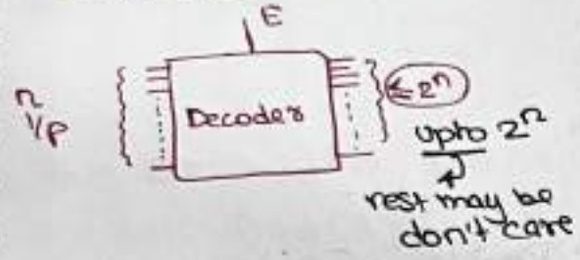


- Application of DeMUX

Multiple computers can communicate via single link with the help of MUX & DeMUX.



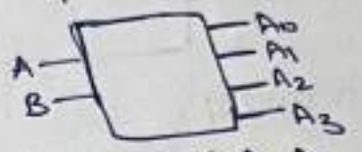
- Decoder
- It is a C.C which has n I/p lines, upto 2^n O/p line & 1 enable pin (optional)
- Enable pin is used to enable (or) disable the ckt.



- If we consider 2^n O/p lines then decoder is a special case of demux in which select line become I/p lines & I/p of demux is assumed to be 1//

ex: 2x4 Decoder

Step 1: I/p & O/p line



Step 2:

AB	A_0	A_1	A_2	A_3
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

$A_0 = \bar{A}\bar{B}$, $A_1 = \bar{A}B$

$A_2 = A\bar{B}$, $A_3 = AB$

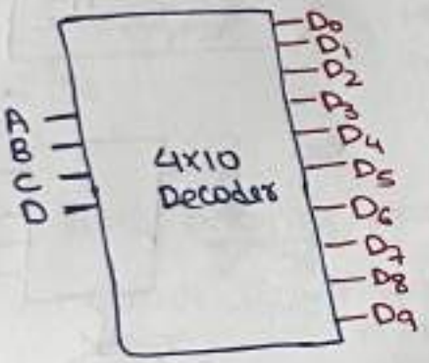
ex: 4n BCD to decimal decoder

what is the literal count of D_9 → 2 literal count assuming no enable pin for the circuit.

D_{10} to D_{15} → 6 don't care

BCD to Decimal

Step 1:



Step 2: truth table

ABCD	D_0	D_1	D_2	.....	D_9
0000	1	0	0	.....	0
0001	0	1	0	.....	0
0010	0	0	1	.....	0
⋮					
1001	0	0	0	.....	1
1010	x	x	x	.....	x
⋮					
1111	x	x	x	.....	x

CD \ AB	00	01	11	10
00	D_0	D_4	D_{12}	D_8
01	D_1	D_5	D_{13}	D_9
11	D_3	D_7	D_{15}	D_{11}
10	D_2	D_6	D_{14}	D_{10}

$D_0 = \bar{A}\bar{B}\bar{C}\bar{D}$

$D_1 = \bar{A}\bar{B}\bar{C}D$

$D_2 = \bar{B}C\bar{D}$

$D_3 = \bar{B}CD$

$D_4 = B\bar{C}\bar{D}$

$D_5 = B\bar{C}D$

$D_6 = BC\bar{D}$

$D_7 = BCD$

$D_8 = A\bar{D}$

$D_9 = AD$

(n) variable ⇒ Total possible (2^n) minterm

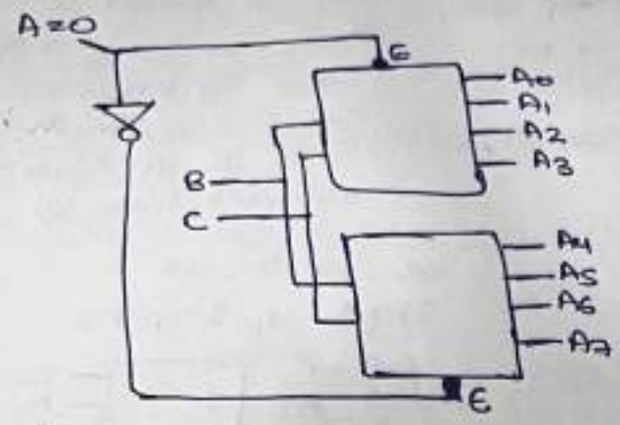
$\frac{2^n}{2} = 2^{n-1}$

less than 2^{n-1} minterm present = OR

more than 2^{n-1} minterm present = NOR

exactly 2^{n-1} minterm present = NOR/OR

ex: How to realize 3x8 decoder from 2x4 decoder



2, 2x4 decoder
1, 1 NOT gate

generalize:

$m \times 2^m$ realize from 2x4

I/p $\log_2 m \times m$ O/p

$$\sum_{k=1}^{\lceil m/2 \rceil} \frac{2^m}{2^{2k}}$$

- Realize $n \times 2^n$ decoder from $m \times 2^m$ decoder

$$\sum_{k=1}^{\lceil n/m \rceil} \frac{2^n}{(2^m)^k}$$

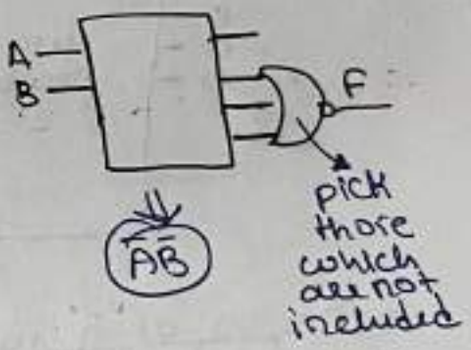
OR

$$\sum_{k=1}^{\lceil \log_n m \rceil} \frac{m}{n^k}$$

Application of Decoders

- we can realize any boolean funct^n of 'n' variable w/o minimizing it using $n \times 2^n$ decoder & 1 'm' I/p OR gate where 'm' is No. of minterms present in funct^n
- In place of OR gate we can use NOR gate of $(2^n - m)$ I/p.

ex:



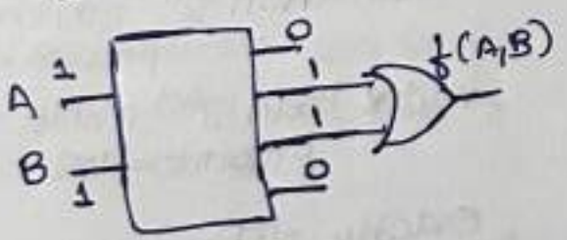
'n' variable funct^n = $n \times 2^n$ decoder

& 'm' minterm = 'm' I/p OR gate

↓

$(2^n - m)$ I/p NOR gate

ex: $f(A,B) = A\bar{B} + \bar{A}B$

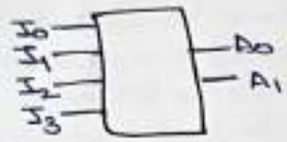


generalize: To realize 'm' funct^n of degree 'n' we need 1, $n \times 2^n$ decoder, & m OR gate

- Encoders
- It is a c.c which has upto 2^n I/p lines, n o/p lines & 1 enable pin [optional].
- enable pin is used to enable or disable the ckt

ex: Design 4x2 encoder

$S_1 \rightarrow$



S_2 (Step-2)

	I_0	I_1	I_2	I_3
m_0	1	0	0	0
m_4	0	1	0	0
m_2	0	0	1	0
m_1	0	0	0	1
m_0	0	0	0	0

A_0	A_1	V (valid)
0	0	1
0	1	1
1	0	1
1	1	1
0	0	0

(No I/P is same as VP)

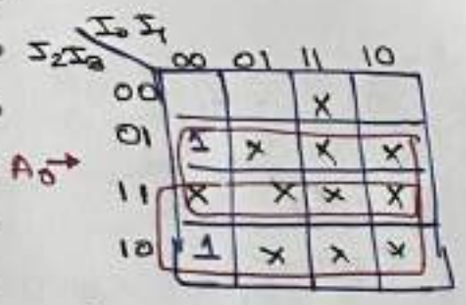
to distinguish we need a valid bit

$A_0 = \Sigma m(1,2) + d(3,5,6,7,9,10,11,12,13,14,15)$

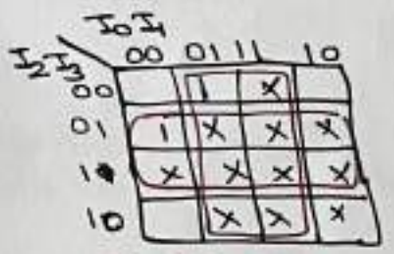
$A_1 = \Sigma m(1,4) + d(3,5,\ldots\text{"}\ldots\text{"}\ldots)$

$V = I_0 + I_1 + I_2 + I_3$

	I_0I_1	$I_0\bar{I}_1$	$\bar{I}_0\bar{I}_1$	$\bar{I}_0I_1$	
m_3	0	0	1	1	X X 1
m_5	0	1	0	1	X X 1
m_6	0	1	1	0	X X 1
m_7	0	1	1	1	X X 1



$A_0 \rightarrow I_3 + I_2$



$A_1 \rightarrow I_3 + I_1$

ex: Design 4x2 priority encoders.

Case1: lower I/p more priority

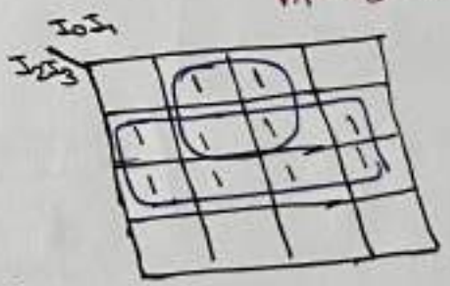
i.e $I_0 I_1 I_2 I_3$

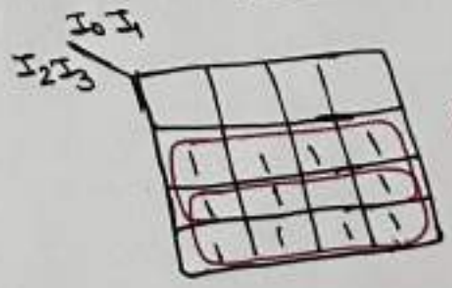
Case2: higher I/p more priority

S_2:

I_0	I_1	I_2	I_3	A_0	A_1	V
0	0	0	0	0	0	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1

$A_1 = I_3 + I_2\bar{I}_3$





$A_0 = I_3 + I_2$

ex. 4x2 encoder (priority) & priority is given to the smaller no.

truth table

I_0	I_1	I_2	I_3	A_0	A_1	V
0	0	0	0	0	0	0
1	x	x	x	0	0	1
0	1	x	x	0	1	1
0	0	1	x	(1)	0	1
0	0	0	1	1	1	1

$V = I_0 + I_1 + I_2 + I_3$

$A_0 = I_2 \bar{I}_0 \bar{I}_1 + I_3 \bar{I}_0 \bar{I}_1$

$A_1 = I_1 \bar{I}_0 + I_3 \bar{I}_0 \bar{I}_2$

I_2 should be (1) & all the no. which are smaller & are '0' should be included

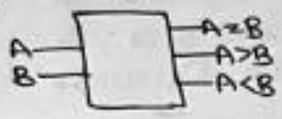
- Comparator

It is a cc which is used to compare 2 binary no.

1bit 2bit 3bit ....

- 1 bit Comparator

Step 1:



Step 2:

A	B	A=B	A>B	A<B
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

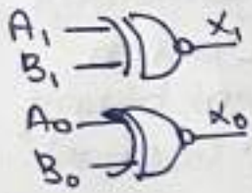
$f_1(A,B) = A \odot B$

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

$f_3(A,B) = \bar{A}B$

Same with 2bit comparator

$A_1A_0B_1B_0$	A=B	A<B	A>B
0 0 0 0	1	0	0
0 0 0 1	0	1	0
0 0 1 0	0	1	0
⋮	⋮	⋮	⋮



A=B

$(X_1 \cdot X_0)$

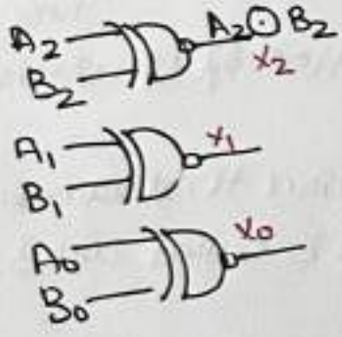
A>B

$A_1\bar{B}_1 + X_1 \cdot A_0\bar{B}_0$

A<B

$\bar{A}_1B_1 + X_1\bar{A}_0B_0$

- 3bit comparator



A=B

$X_2 \cdot X_1 \cdot X_0$

A>B

$A_2\bar{B}_2 + X_2A_1\bar{B}_1 + X_2X_1A_0\bar{B}_0$

A<B

$\bar{A}_2B_2 + X_2\bar{A}_1B_1 + X_2X_1\bar{X}_0B_0$

- no. of minterm in 'n' bit comparator.

Total no. of minterm 2^{2n}

A=B ⇒ 2^n

A>B } Brothuse

A<B } (A=B) possibility is removed from total & then split into half

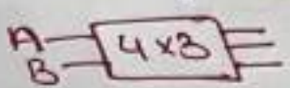
$\frac{2^{2n}-2^n}{2}$ A>B

$\frac{2^{2n}-2^n}{2}$ A<B

- **ROM**

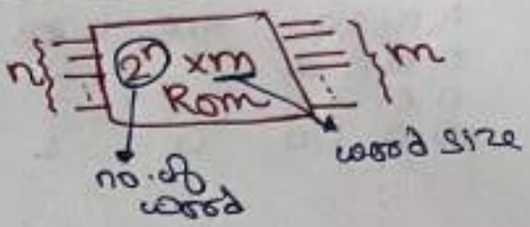
It is a combinatorial ckt which consist of $n \times 2^n$ decoders & each o/p line of decoder with fuse is given as I/p to 'm' 2^n I/p OR gate

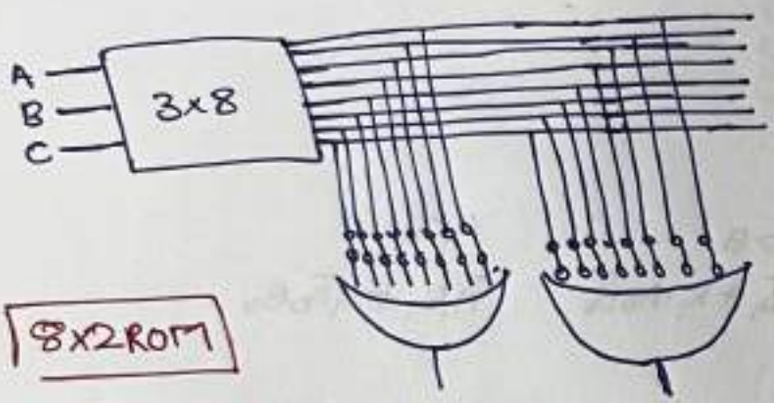
word size : 3
no. of words : 4



ex. 8x2 ROM
- no. of cells
- cell size

Generalize





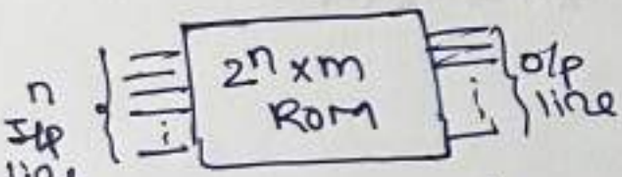
- ROM is a memory device in which permanent binary information is stored

8x2 ROM

3x8 Decoder, 2 OR gate & 16 fuse

3 NOT & 8 AND

- It has n I/p line & m o/p line
- Each bit combination of I/p variable is called address
- each bit combination that comes out of the o/p line is called word.



$n \times 2^n$ decoder & m OR gate & $2^n * m$ fuse

- n NOT
- 2^n AND
- n I/p

of 2^n I/p

ex: To realize 'n' function of 'm' variable
ROM needed will be $2^m \times n$

ex: to realize any funct^n of (3) variable with 4x2 Rom?
we need extra 2x1 MUX

1 function # 3 variable

8x1 ROM