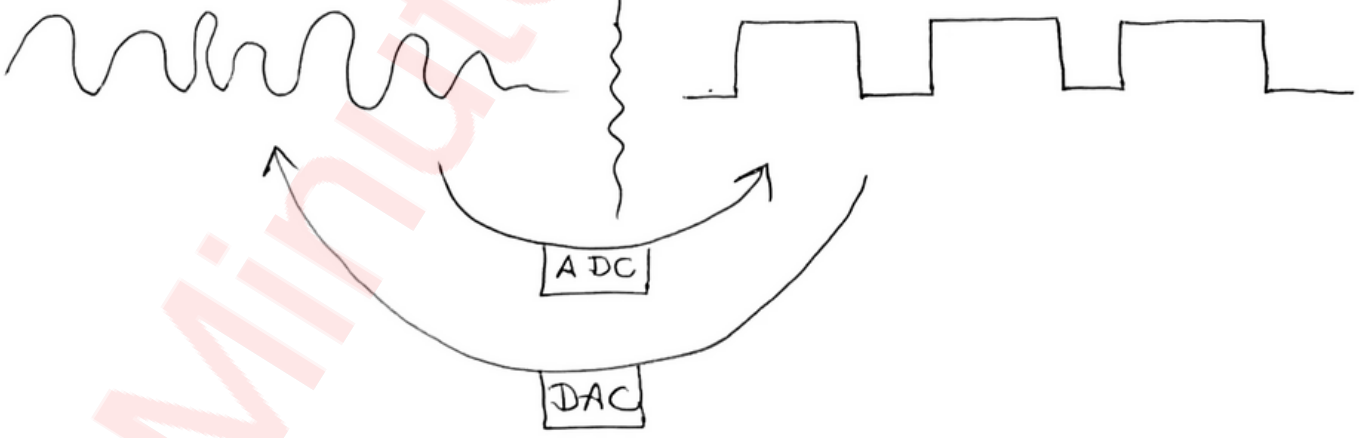
Introduction

Digital **Electronics**

Signal/ Discrete value — Branch

→ Analog Vs Digital

- Continuous Nature
- Discrete Nature [0 & 1]
- Sine wave
- Square wave
- Analog circuits [Resistors, capacitors, inductors]
- Digital circuits [Transistors, logic Gates, ICs]



- Number System → count
  → Representation

→ Binary (2), Octal (8), Decimal (10), Hexadecimal (16)

Base
[No. of symbols]

⇒ MSB & LSB

eg 0 1 1 0

Bit Bit Bit Bit Bit Bit Bit Bit

Byte

Binary	Octal	Decimal
0 0 0	0	0
0 0 1	1	1
0 1 0	2	2
0 1 1	3	3
1 0 0	4	4
1 0 1	5	5
1 1 0	6	6
1 1 1	7	7

Hexadecimal

0, 1, 2, 3, 4, 5, 6, 7, 8, 9,

A, B, C, D, E, F

↓ ↓ ↓ ↓ ↓ ↓

(10) (11) (12) (13) (14) (15) Total = (16)

Binary Arithmetic

↳ addition, substraction, division, multiplication.

⇒ Addition:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 0 [Carry = 1]

⇒ Substraction:

0 - 0 = 0

0 - 1 = 1 [Borrow = 1]

1 - 0 = 1

1 - 1 = 0

⇒ Multiplication

0 × 0 = 0 × 1 = 1 × 0 = 0

1 × 1 = 1

- Division

$0 \div 0 = ND$

$0 \div 1 = 0$

$1 \div 0 = ND$

$1 \div 1 = 1$

- 1's Complement

eg: 1010101 $\Rightarrow$ 0101010

- 2's complement = 1 + 1's complement

1010101

↓ 1's complement

0101010

+ 1

0101011 ← 2's complement.

Shortcut: from LSB complement bits as we see first '1'

$\Rightarrow$ 1010101

↓↓↓↓↓↓↓ ← first '1'

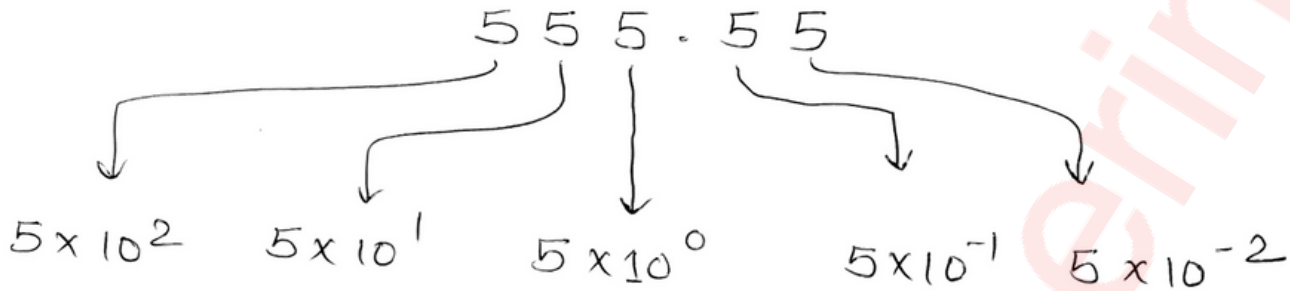
0101011

$\Rightarrow$ 1010000

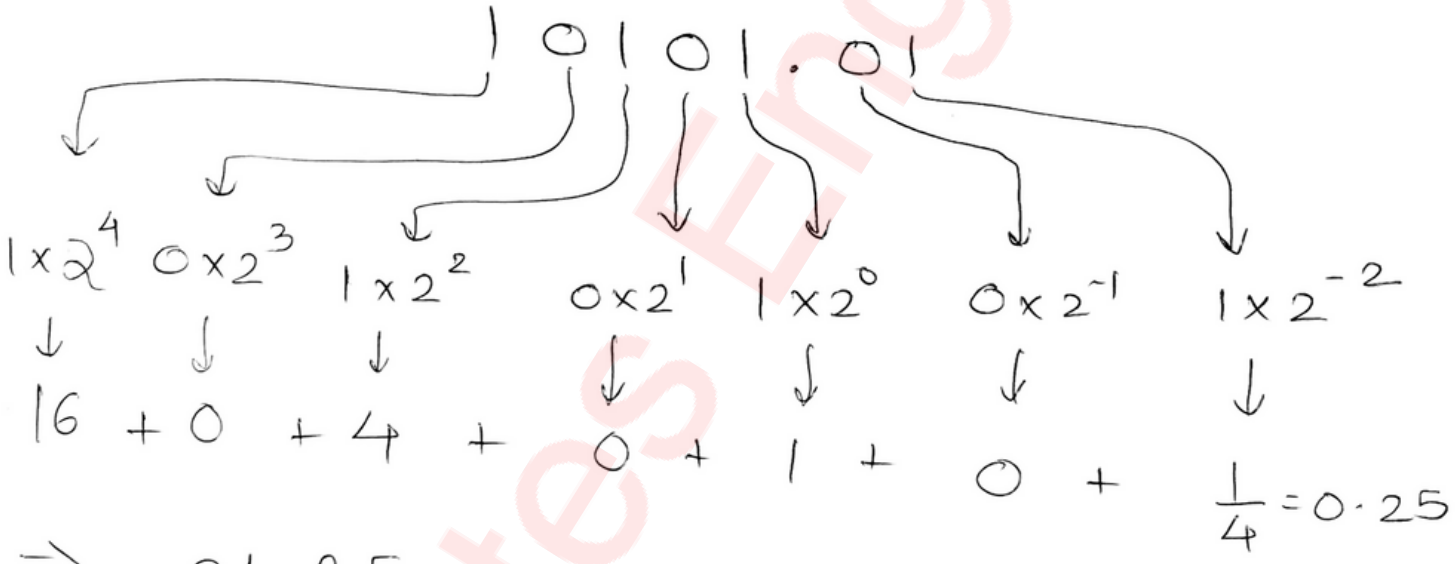
↓↓↓ ← first '1'

$\Rightarrow$ 011

Eg:- Decimal Number System

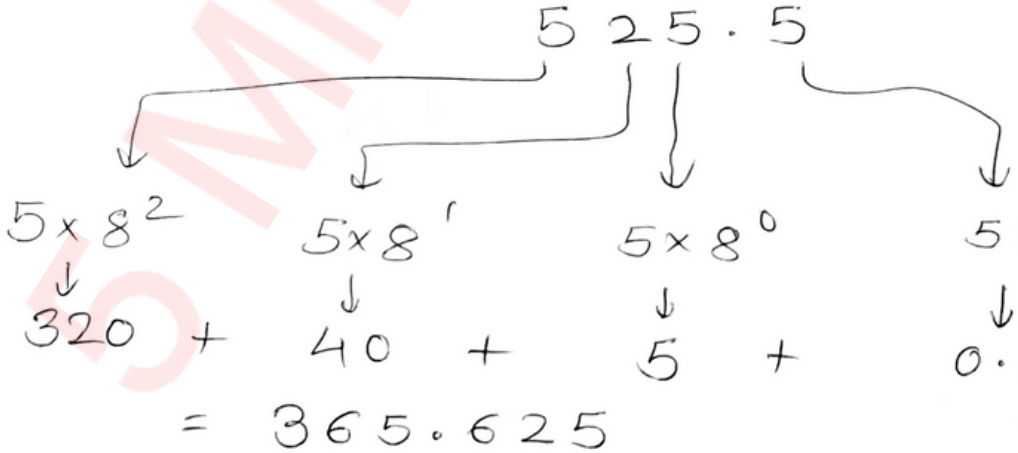


- Binary Number System



$\Rightarrow$ 21.25

- Octal Number System



= 365.625

$(1\ 0\ 1\ 0\ 1)_2$

$1\times2^4 \quad 0\times2^3 \quad 1\times2^2 \quad 0\times2^1 \quad 1\times2^0$

$16 + 0 + 4 + 0 + 1$

$(21)_{10}$

Convert to Octal.
[Divide by 8]

$$8\overline{)21}\quad 2,\ 16,\ 5$$

$(25)_8$

$= 2\times8^1 + 5\times8^0$

$= 16 + 5 = (21)_{10}$ ✓

$\Rightarrow (555.44)_{10}$

Convert to Octal.

$$8\overline{)555}\quad 69,\ 48,\ 75,\ 72,\ 3$$

$$8\overline{)69}\quad 8,\ 64,\ 5$$

$\Rightarrow 853.341$

$0.44\times8 = 3.52$

$0.52\times8 = 4.16$

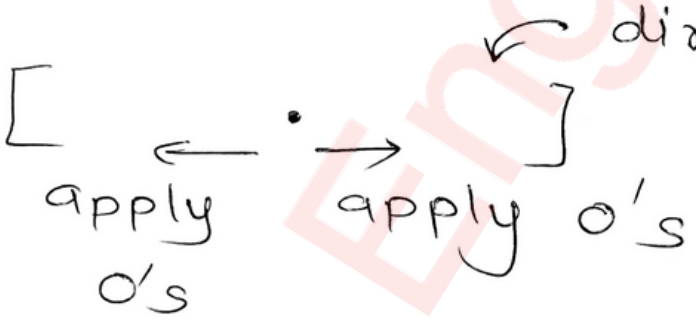
$0.16\times8 = 1.28$

- $(101010.10)_2$

  → $(222.2)_4$

  → $(52.4)_8$

  → $(2A.8)_{16}$



But to go from $B_4 \rightarrow B_8$, $B_8 \rightarrow B_{16}$

⇒ Try to first get it in B_2 then next.

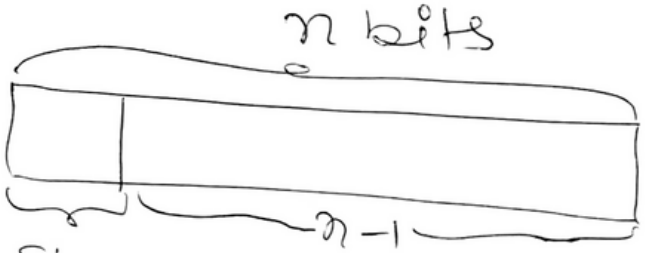
$B_4 \rightarrow B_2 \rightarrow B_8 / B_{16}$

Signed Numbers

↓

Magnitude + Sign

(+5, -5)



0 → +ve
1 → -ve

$[-(2^{n-1}-1)$ to $+(2^{n-1}-1)]$ (SM)

↓

Unsigned Numbers

↓

Only Magnitude

No sign (5, 55)

⇒ 5 bit

↳ $2^n - 1$

$= 2^5 - 1$

$= 31$ total/possible.

Decimal values	2's C	Signed Magnitude (MSB ↓)
+5	0 1 0 1	0 1 0 1
+3	0 0 1 1	0 0 1 1
+1	0 0 0 1	0 0 0 1
+0	0 0 0 0	0 0 0 0
-0	0 0 0 0	1 0 0 0
-1	1 1 1 1	1 0 0 1
-3	1 1 0 1	1 0 1 1
-5	1 0 1 1	1 1 0 1

Same

⇒ Eg Given 1 0 1 1 0

SB | mag.

1 → -ve
0 → +ve

1×2^4 0×2^3 1×2^2 1×2^1 0×2^0

$(-16) + 0 + 4 + 2 + 0$

$-16 + 6 = -10$

$(2^{1}s\,C)$ 1 0 1 1 0

(-ve) (10)

⇒ Substraction by Addition (2's C)

eg: 10 − 5

↓

10 + (−5)

0 1 0 1 0 + 1 1 0 1 1

(+ve) 10 −ve 5

↓

−16 + 8 + 0 + 2 + 1

= −5

0 1 0 1

↓ 2's C

1 0 1 1

1 1 1 1

1 0 1 0 1 0

\+ 1 1 0 1 1

0 0 1 0 1

↓ +ve 5

⇒ 10 − 11

↓

10 + (−11)

↓ ↓ 2's com

01010 1 0101

+ve 10 −ve

1011

0101

01010

10101

11111 → −ve no.

1×2^5 1×2^4 1×2^3 1×2^2 1×2^1 1×2^0

↓ ↓ ↓ ↓ ↓ ↓

$-32+16+8+4+2+1$

$-32+31 = -1$ Ans.

⇒ BCD code

→ 8421

→ 2421

} weighted code.

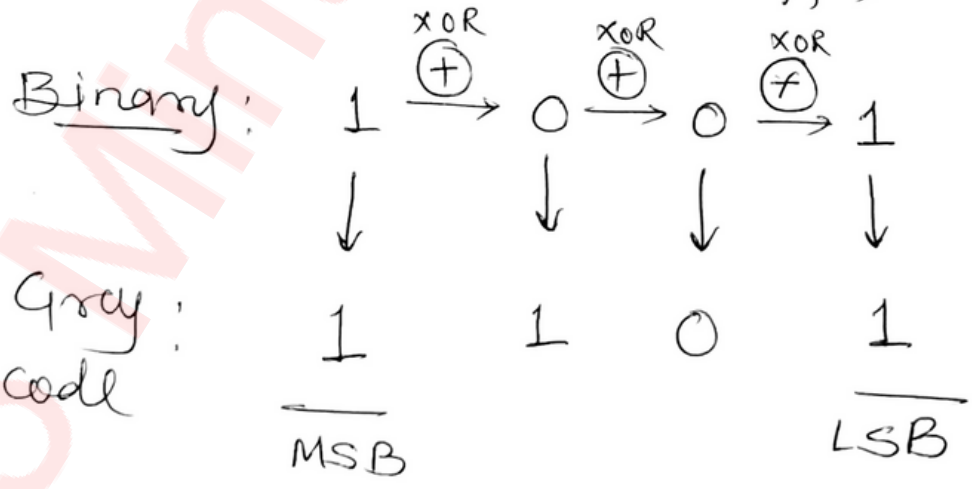
Decimal	8421	2421
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	1011
6	0110	1100
7	0111	1101
8	1000	1110

- Non weighted codes
  - Excess-3
  - Gray code

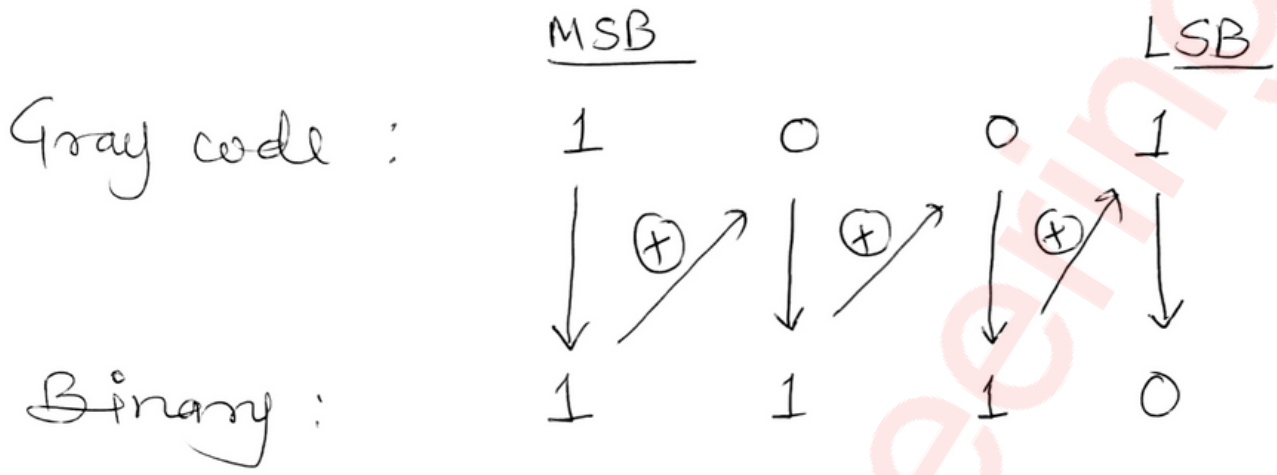
Decimal	BCD 8421	Excess-3 (+0011)
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Complement

- Gray Code : (A bit diff)



- Gray code → Binary



- Boolean Algebra:
  - ↓ Branch of Algebra
  - → True/false
  - → 1/0

: Laws

① Idempotent law:

$$x \cdot x = x \quad \& \quad x + x = x$$

② Associative law

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

③ Distributive law

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

④ Commutative law

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

⑤ De-Morgan law

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

⑥ Identity law

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

⑦ Complementation law

$$\overline{T} = F \qquad \overline{0} = 1$$
$$\overline{F} = T \qquad \overline{1} = 0$$

$$\boxed{x \cdot \overline{x} = 0} \qquad \boxed{x + \overline{x} = 1}$$

⑧ Involution law

$$\overline{(\overline{x})} = x$$

Logic Gates:

↳ 1 or more I/p

But

only 1 O/p.

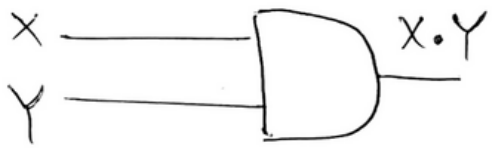
Basic Gates: AND, OR & NOT

Universal Gates: NAND, NOR

Special Gates: XOR, xNOR

- AND Gate [O/p is T/High if all I/p are True]

X	Y	X.Y
0	0	0
0	1	0
1	0	0
1	1	1

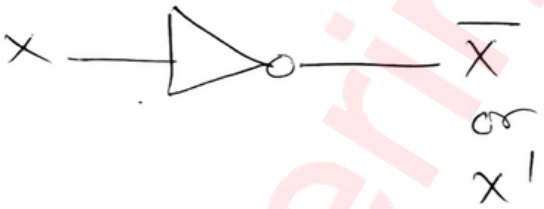


- OR Gate [O/p is 'T' if any one I/p is 'T']

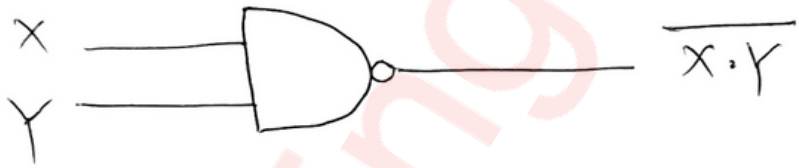
X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

- **NOT Gate** [0 → 1 & 1 → 0]

X	$\overline{X}$ or X'
0	1
1	0

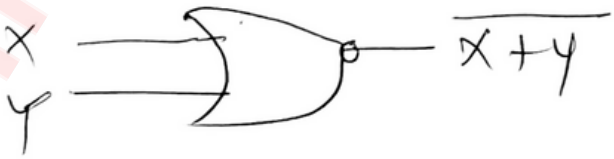


- **NAND Gate** [Complement of 'AND']



X	Y	$\overline{X \cdot Y}$
0	0	1
0	1	1
1	0	1
1	1	0

- **NOR Gate** [Complement of 'OR']



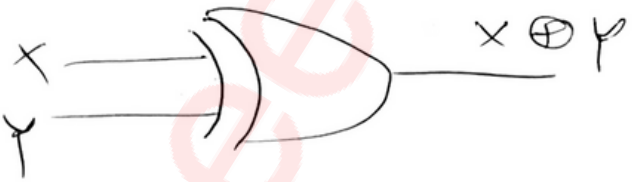
X	Y	$\overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	0

° XOR Gate [O/p is True if Different]

" 0 1 "

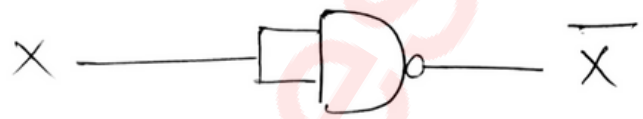
" 1 0 "

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

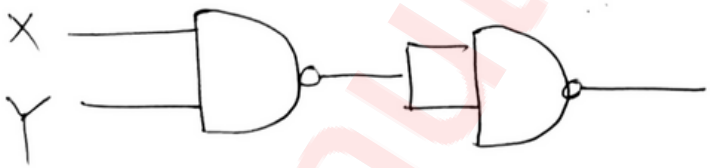


$$X \oplus Y = X.\overline{Y} + \overline{X}.Y$$

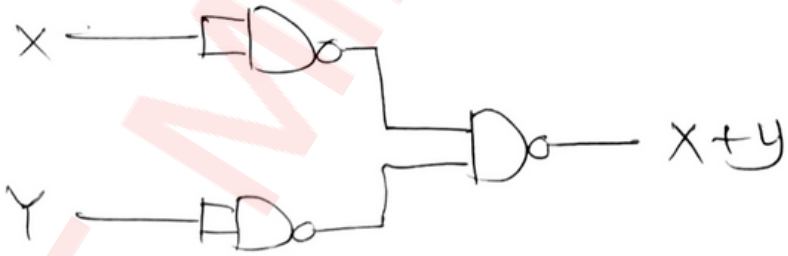
⇒ Realization of NOT, OR, AND & XOR using NAND gate.

⇒ 

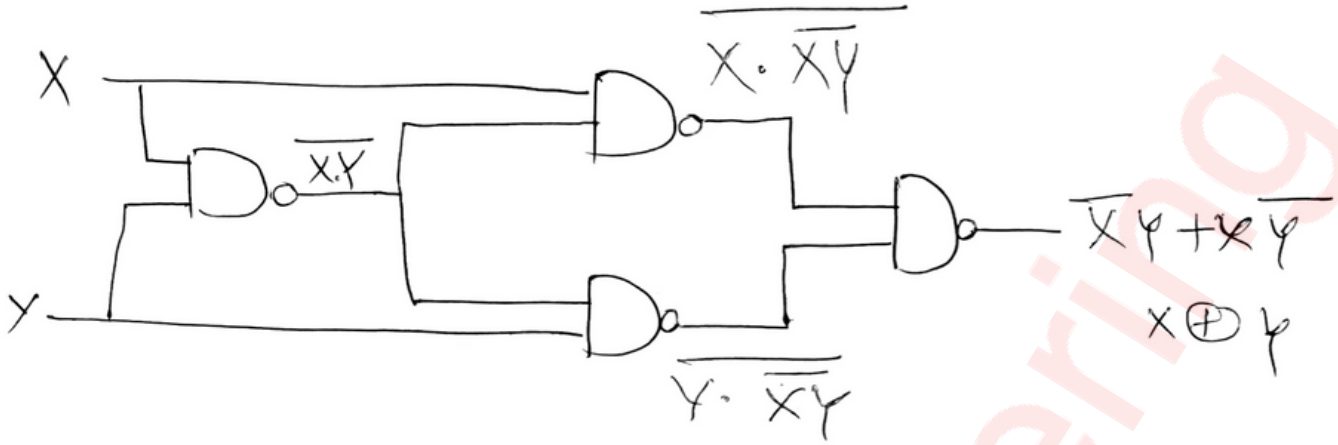
(NOT)



(AND)

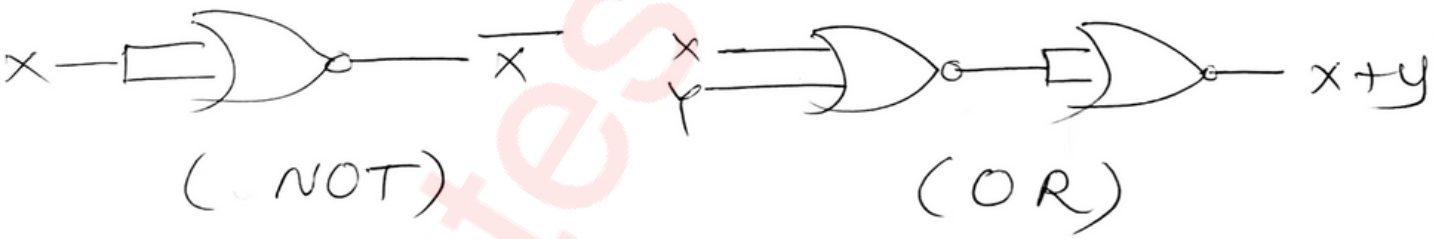


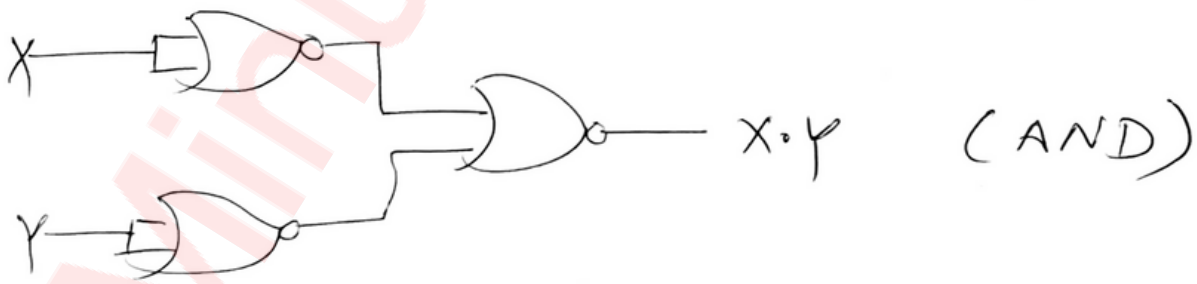
(OR)

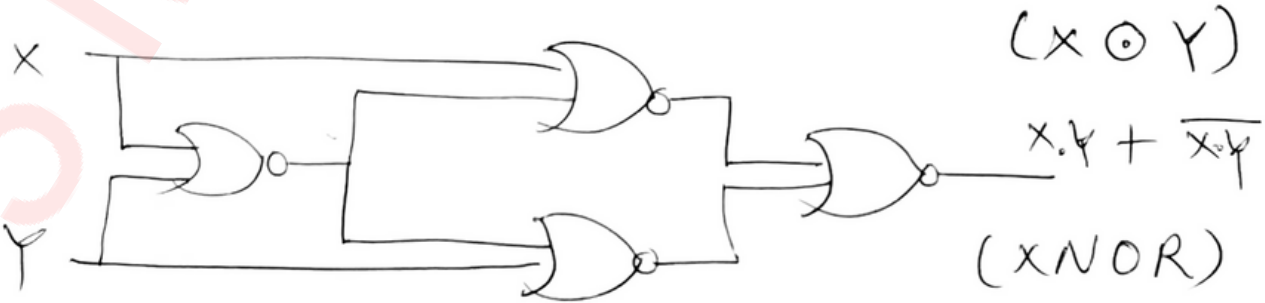


X	Y	X.Y	$\overline{X.Y}$	$\overline{X.\overline{XY}}$	$\overline{Y.\overline{XY}}$	$(X \oplus Y)$ O/P
0	0	0	1	1	1	0
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	1	1	0

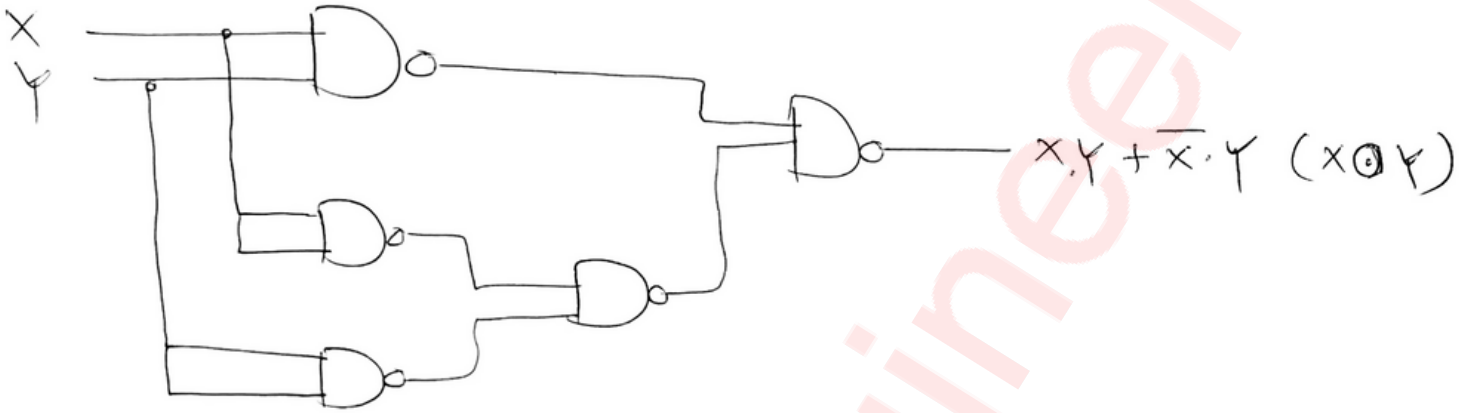
Similarly using NOR Gate.



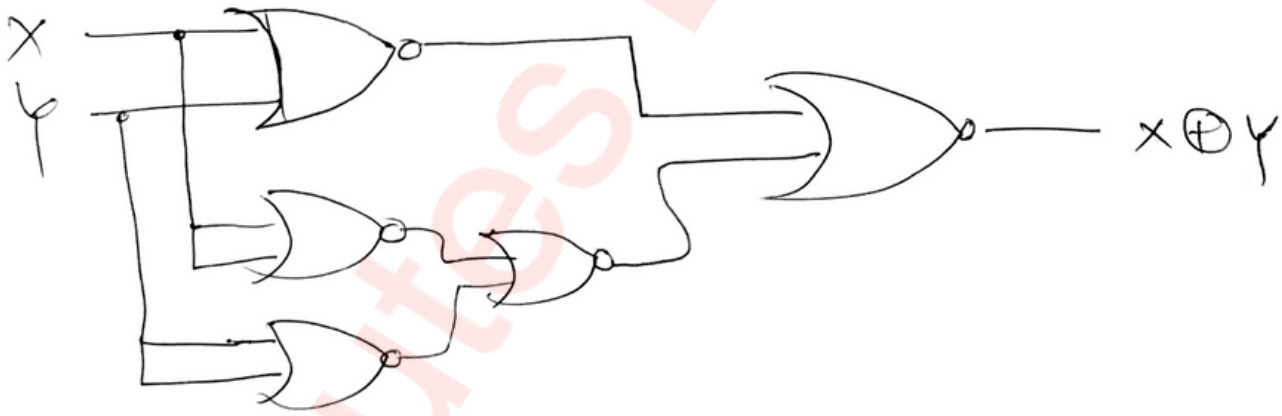




(XNOR using NAND)



(XOR using NOR)



Boolean Expression

↳ Expression that can tell when o/p → 0 or 1

→ SOP [Sum of Product] "1"

→ POS [Product of sum] "0"

X	Y	O/P
0	0	1
0	1	1
1	0	0
1	1	0

$O(X,Y) = \sum_m (0, 1)$ (m_0 m_1)

$O(X,Y) = \prod_M (2, 3)$ (M_2 M_3)

Eg:

X	Y	O/P = X+Y (POS)
0	0	0
0	1	1
1	0	1
1	1	1

$SOP = XY + \bar{X}y + x\bar{y}$

↑ Sum of Product

- Sum of Product

→ Expression have:

Product terms (AND) summed (OR)

Eg: $XY + YZ + \bar{X}Y\bar{Z}$

↑ 2 literals ↑ 3 literals

[complement or Non-complement allowed ✓]

⇒ minterm = $[XY, YZ, \bar{X}Y\bar{Z}]$

$f(x,y,z)$ ✗ ✗ ✓

Binary

0	0	0	$\bar{a}\bar{b}\bar{c} \rightarrow m_0$
0	0	1	$\bar{a}\bar{b}c \rightarrow m_1$
0	1	0	$\bar{a}b\bar{c} \rightarrow m_2$
0	1	1	$\bar{a}bc \rightarrow m_3$
1	0	0	$a\bar{b}\bar{c} \rightarrow m_4$
1	0	1	$a\bar{b}c \rightarrow m_5$
1	1	0	$ab\bar{c} \rightarrow m_6$
1	1	1	$abc \rightarrow m_7$

minterm

- Canonical logic forms

↳ CSOP: $xyz + x\bar{y}z + \bar{x}\bar{y}z$ (✓)

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

$y + x + z$ (x)

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

↓ ↓

$[xy\bar{z} + xyz + \bar{x}\bar{y}\bar{z}]$ ✓ CSOP

- POS [Product of Sum]

⇒ OR terms → product

$(x+y)(y+z)$

$(x+y)$, $(y+z)$ — OR

$(x+y)\cdot(y+z)$ — AND

⇒ Maxterm = $[\bar{x}, \bar{z}, \bar{x}\bar{z}]$

$f(x,z)$ (x) (x) (✓)

$0 \rightarrow x$

$1 \rightarrow \bar{x}$

(POS)

$0 \rightarrow \bar{x}$

$1 \rightarrow x$

(SOP)

- CPOS :- $(\bar{x}+y+z)\cdot(x+\bar{y}+\bar{z})$ (✓)

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

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

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

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

(✓)

Q → Given $n = 3$ ↑ (x, y, z) Boolean variables

① Total combinations $= 2^n$

i.e $n = 2\ (x, y)$

00, 01, 10, 11

$n = 3\ (x, y, z)$

000 --- 111

② Total Boolean fns $= 2^{2^n}$

$n = 2\ (x, y) \rightarrow 2^{2^2} = 16$

$n = 3\ (x, y, z) \rightarrow 2^{2^3} = 2^8 = 256$

Q → Complement of a fn.

$\Rightarrow f(x, y, z, 0, 1, \cdot, +)$

$\downarrow$

$f'(\bar{x}, \bar{y}, \bar{z}, 1, 0, +, \cdot)$

eg:- $\bar{x} + y + z \xrightarrow{C} x \cdot \bar{y} \cdot \bar{z}$

eg: $xy + \bar{x}\,\bar{y} \leftarrow$ [EXNOR]

$\downarrow$

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

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

$\downarrow \qquad\qquad \downarrow$

$(\bar{x} \cdot y) + (x \cdot \bar{y}) \leftarrow$ [EXOR]

- Duality Theorem

$\cdot \longleftrightarrow +$ (✓)

$\oplus \longleftrightarrow \odot$ (✓)

$0 \longleftrightarrow 1$ (✓)

$x \longleftrightarrow \bar{x}$ (x)

$\sim \longleftrightarrow \sim$ (✓)

eg: ① $XY\bar{Z} + \bar{X}YZ + XYZ$

$(X+Y+\bar{Z}) \cdot (\bar{X}+Y+Z) \cdot (X+Y+Z)$

② $XYZ + \bar{X}Y\bar{Z} = 1$

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

- Self dual fn:

→ If dual is equal to I/p fn.

eg: $(AB + BC + CA)$

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

$AB + AC + B + BC$

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

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

$(BC + BA + AC + AC)$

$\Rightarrow AB + BC + AC$

Q → No. of self dual f^ns -

→ $n = 2$

① Total combinations $(0, 1) = 2^n$

② Total f^ns possible $= 2^{2^n}$

③ Total self dual f^ns possible $= \left[2^{2^{n-1}}\right]$

Eg:

X	Y
0	0
0	1
1	0
1	1

$2^n = 2^2 = 4$

② Total $f^n = 2^{2^2} = 2^4 = \underline{16}$

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

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

③ Total self dual $f^n = 2^{2^{1-1}}$

$= 2^2$

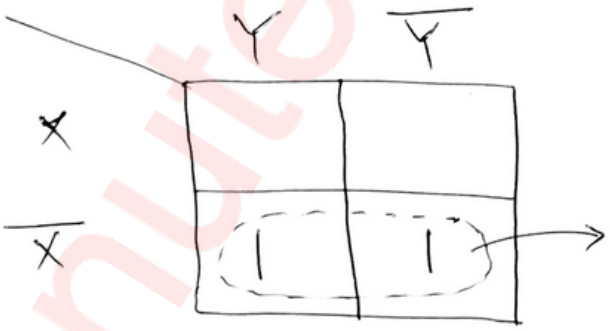
$\Rightarrow = 4$

- **Minimization** [Simplification]
  - → Boolean Laws
  - → Karnaugh map (kmap)

- **kmap** :- $f(X,Y) = \Sigma m(0,1)$

X	Y	f
0	0	1
0	1	1
1	0	0
1	1	0

$$\rightarrow \bar{X}\bar{Y} + \bar{X}Y$$
$$\Rightarrow \bar{X}(Y + \bar{Y})$$
$$\Rightarrow \bar{X}$$

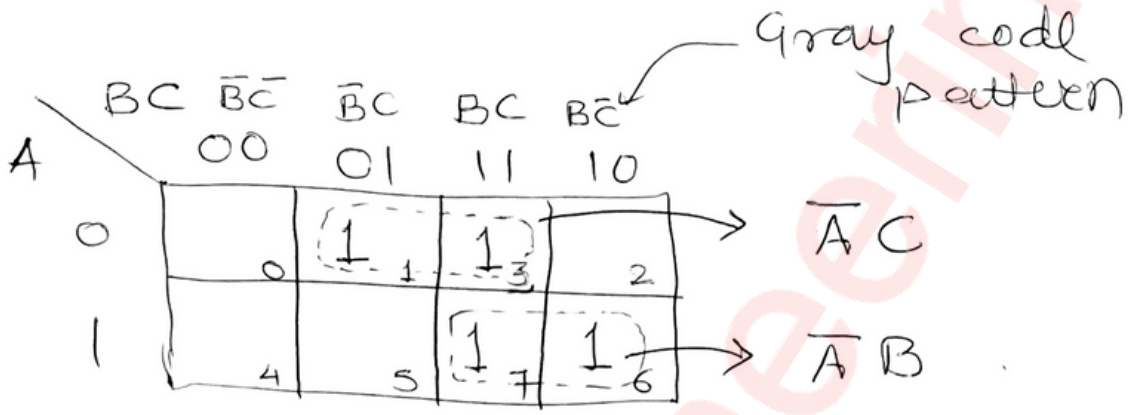


Y is changing $(Y, \bar{Y})$

But X is constant as $(\bar{X})$

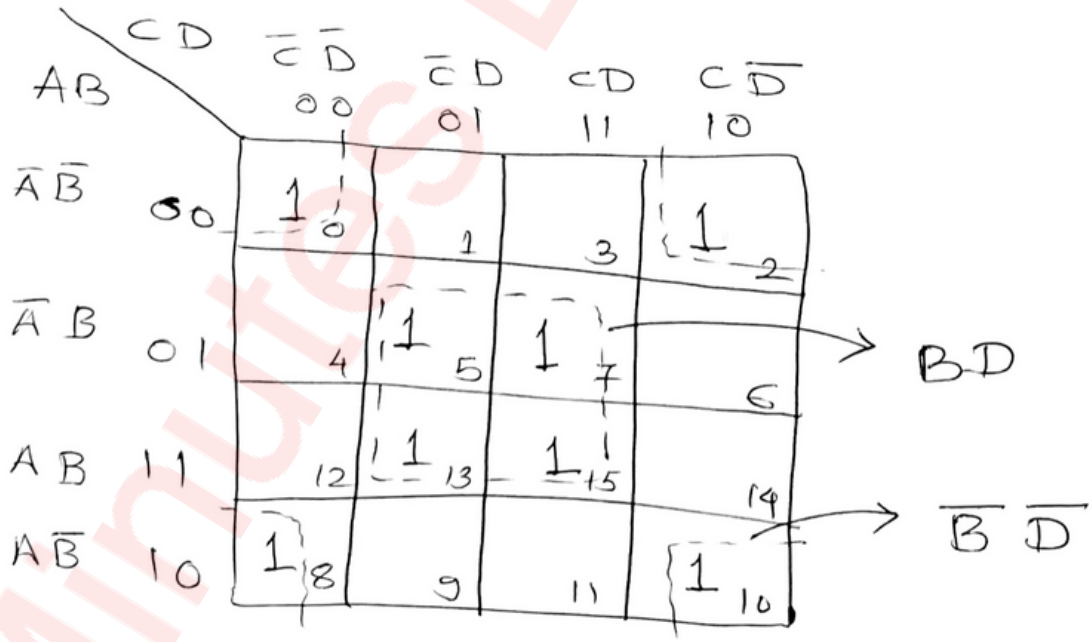
Eg: $n = 3$ $f(x, y, z)$

↳ $\Sigma m(1, 3, 6, 7)$



Eg: $n = 4$ $f(A, B, C, D)$

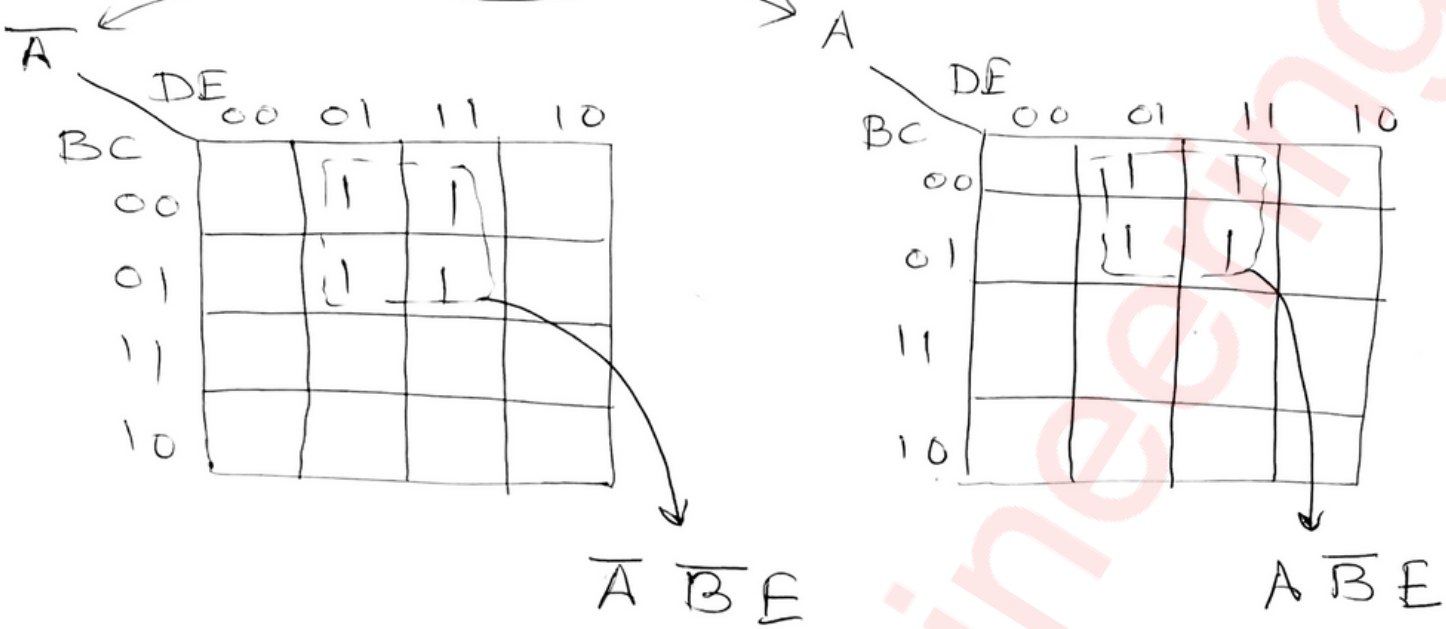
↳ $\Sigma m(0, 2, 5, 7, 8, 10, 13, 15)$

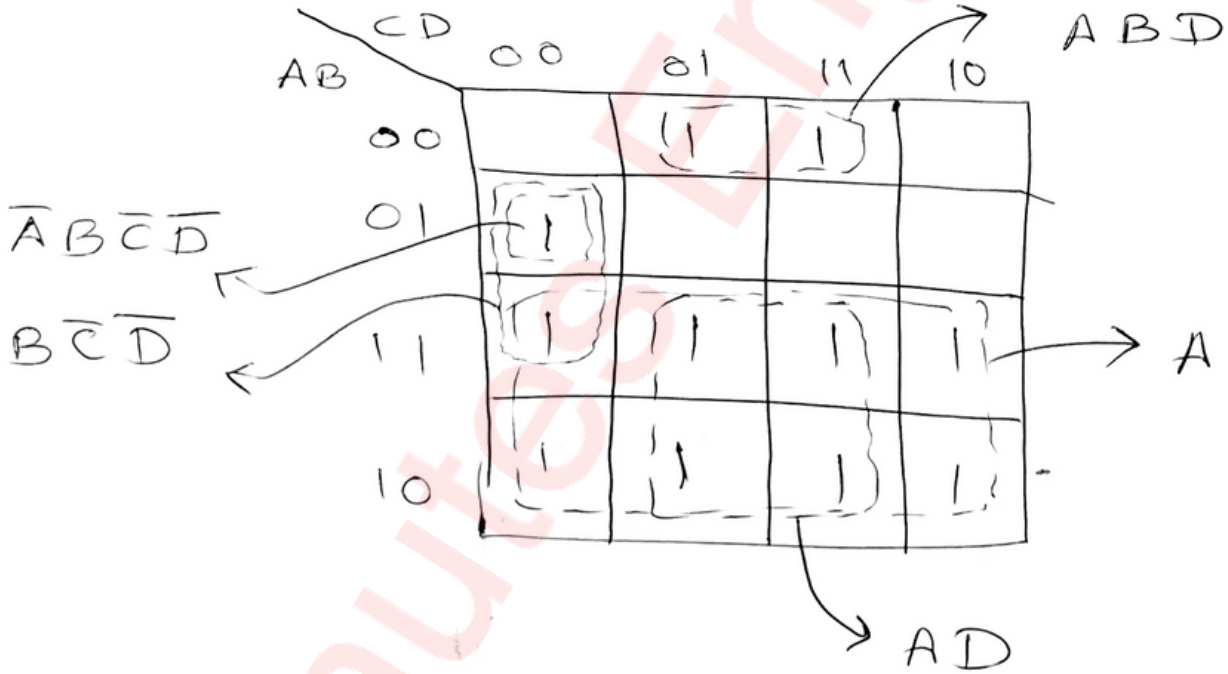


Don't care (x) → can be used to simplify

But we focus mostly on 1 | No separate group of (x) is formed.

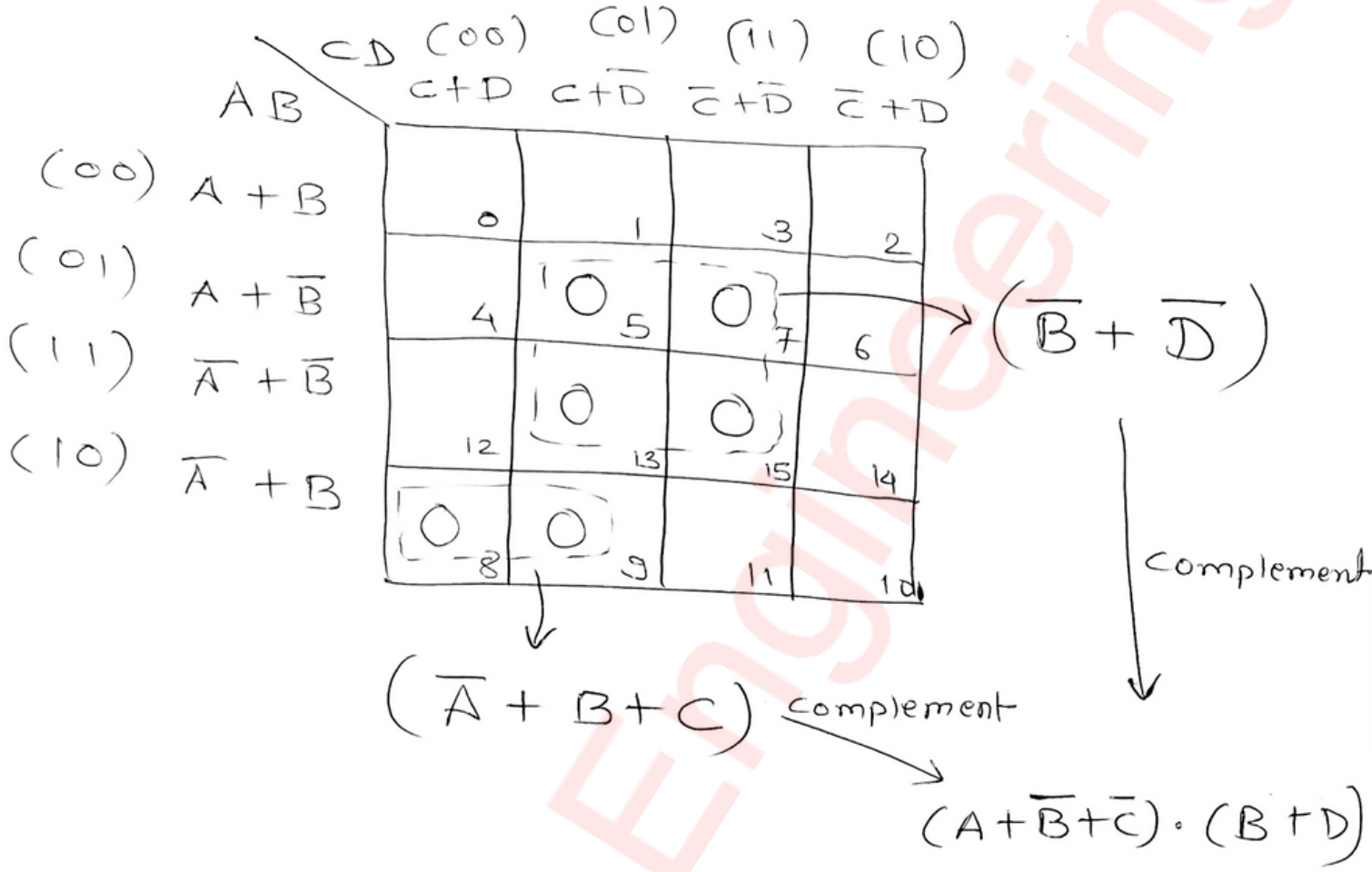
$n = 5$ (A, B, C, D, E)





Kmap for POS (A B C D)

n = 4



- Prime Implicants

(4) → BC̄D, ABD, ACD, ABC

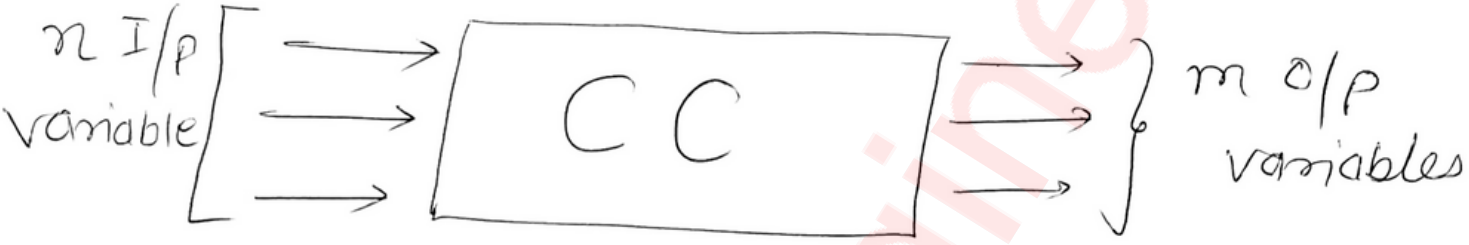
- Essential Prime Implicants

(3) → BC̄D, ACD, ABC

Σm(5, 13, 14, 15)

Combinational Circuit

→ (AND, OR, NOT, NAND, NOR)
→ (I/p & o/p variables)
→ Current combination of I/p
→ NO memory unit



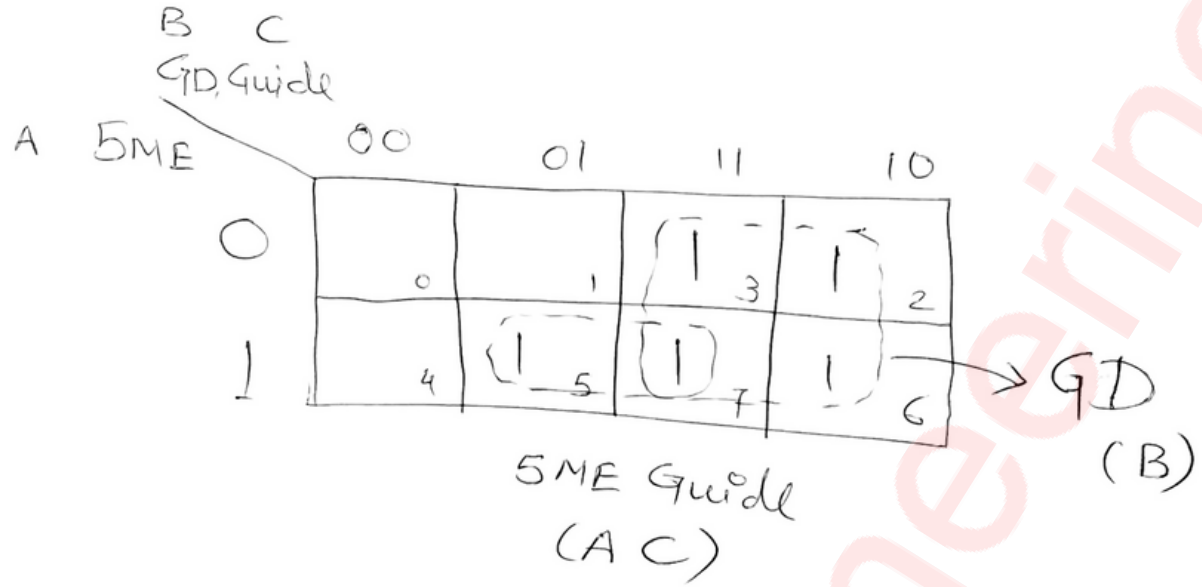
$$o/p = F(I/p)$$

Eg:- I/p: 5ME & Guide & GD
o/p: Result

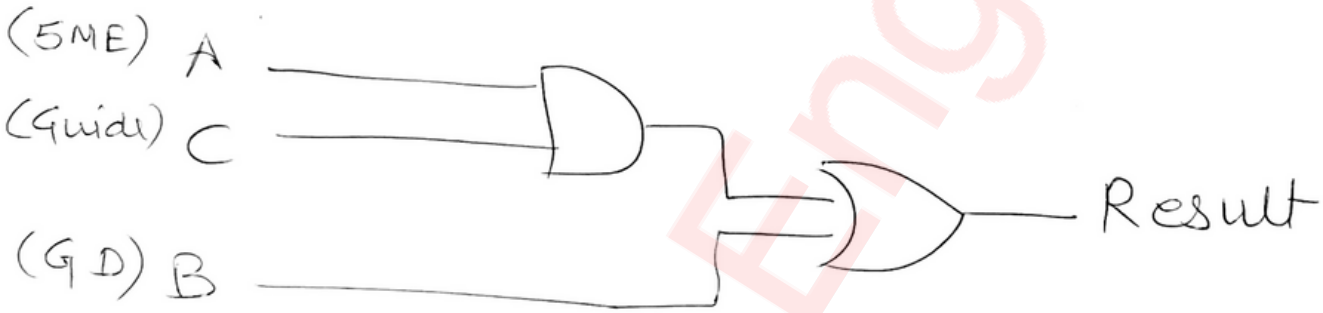
① Truth table

GD	5ME	Guide	Result
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

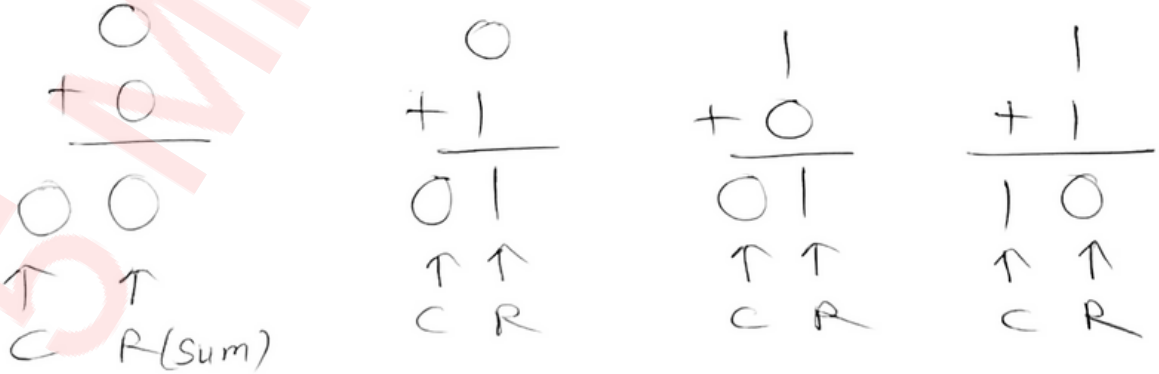
(2) Use k-map

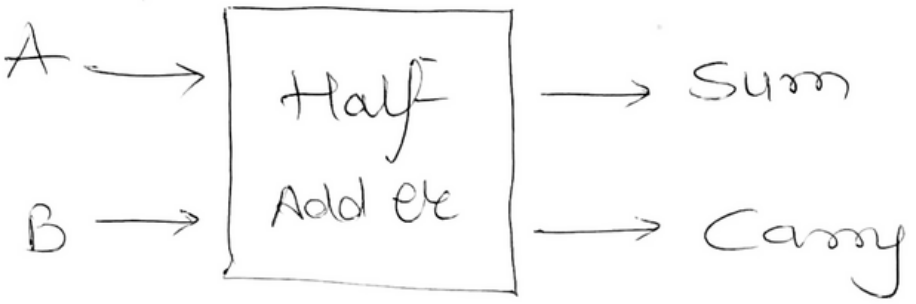


$R \Rightarrow AC + B$



- Adder : CC to perform addition of nos.
  - → Half adder (works on 2 bits addition)
  - → full adder (works on 3 bits additions)

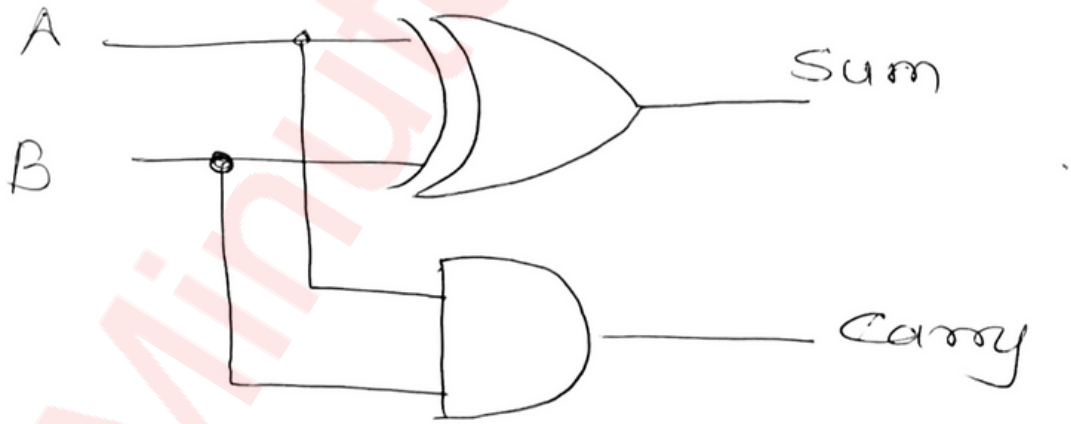




A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

XOR

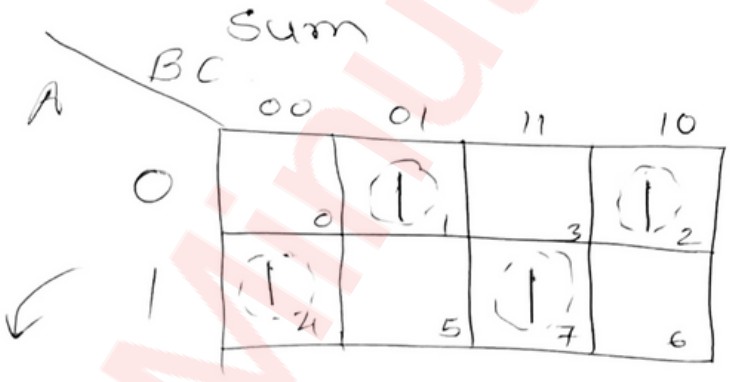
AND



∘full adder

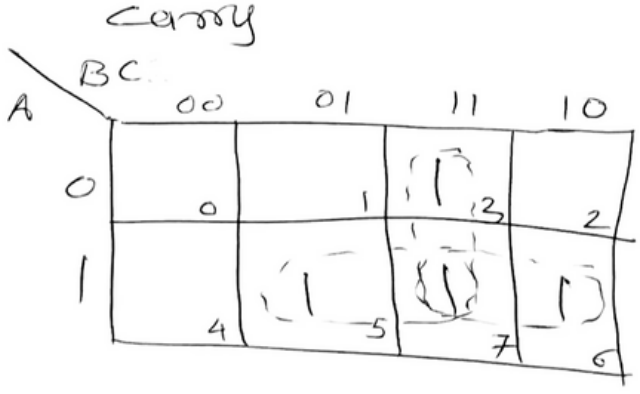
A	B	CI	Sum	CO
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Complement



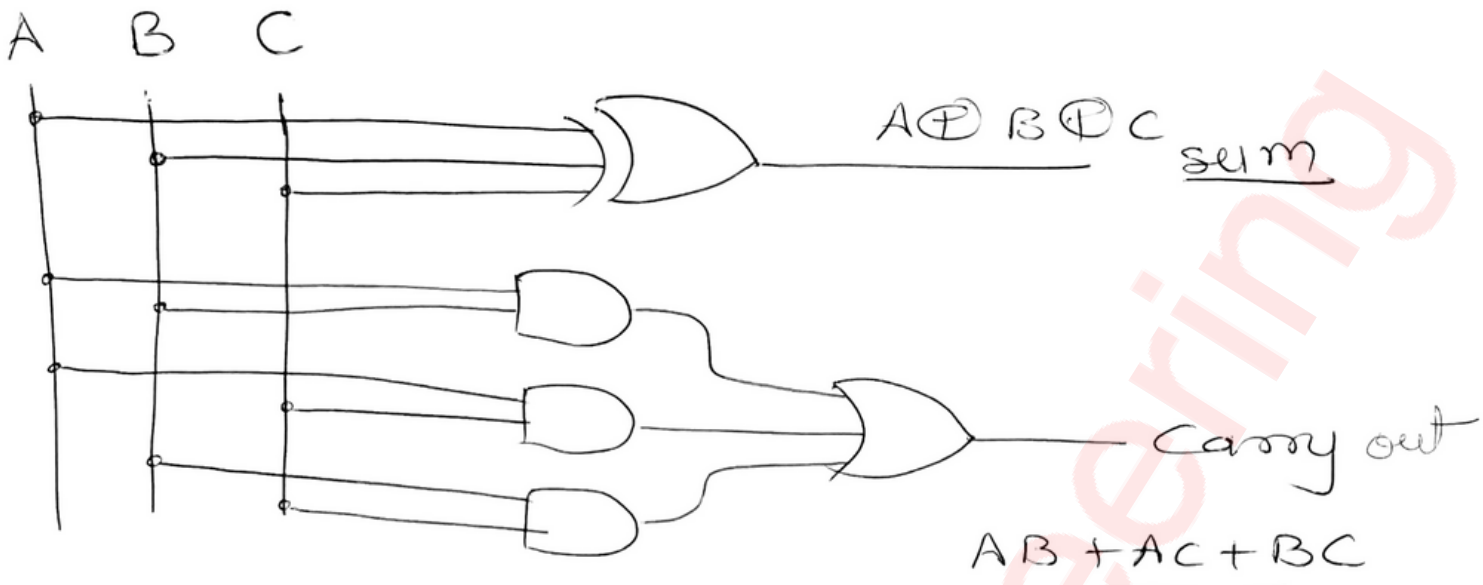
$A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$

$Sum = A \oplus B \oplus C$

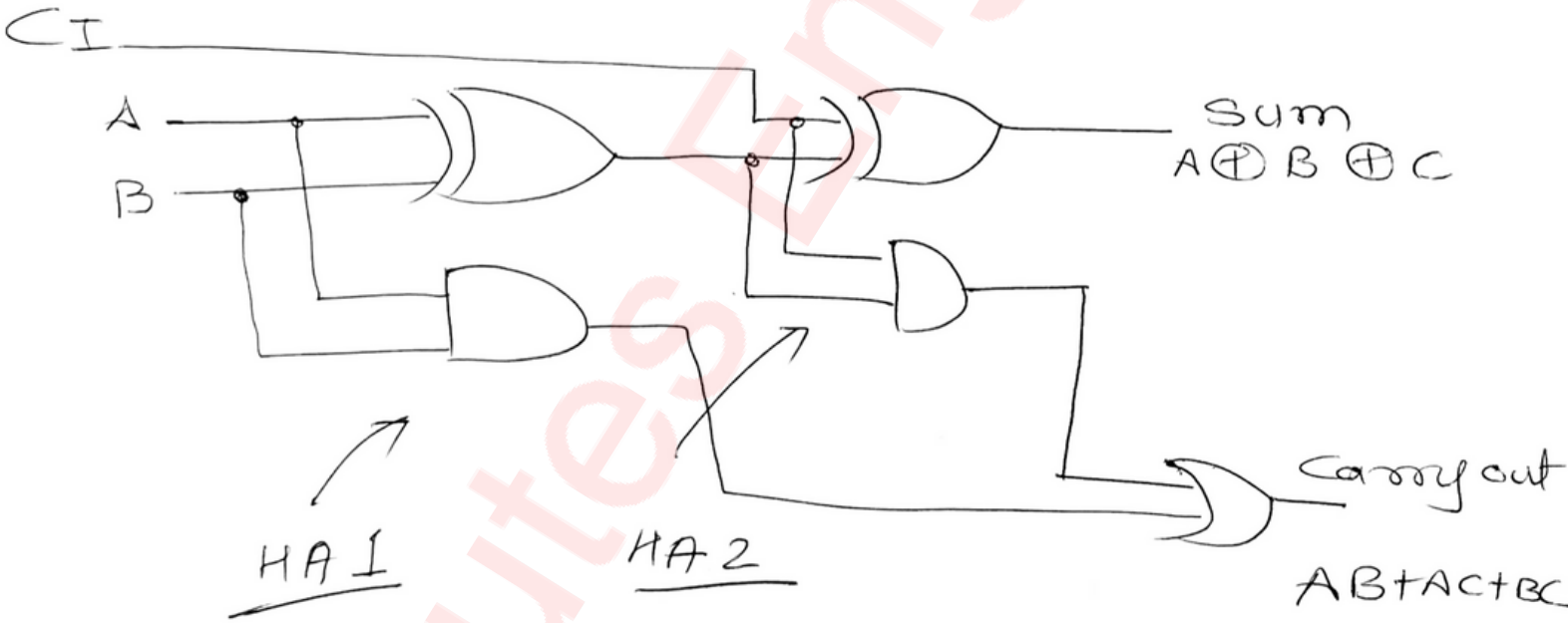


$AC + AB + BC$

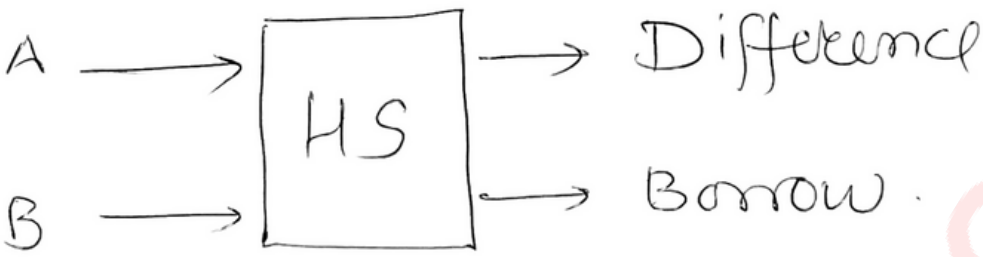
carry =



- NOW implementing the behaviour of Full adder with 2 half adder



Half Subtractor

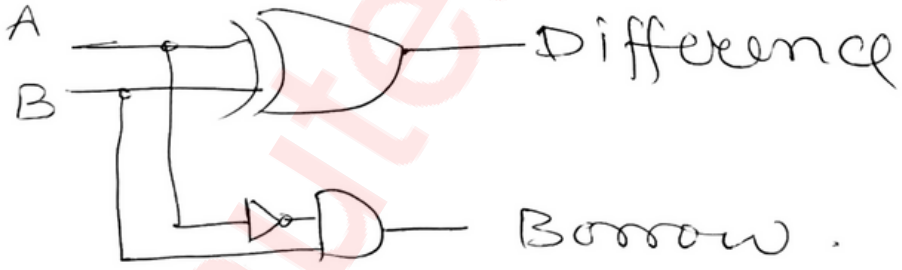


A	B	D	BO
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

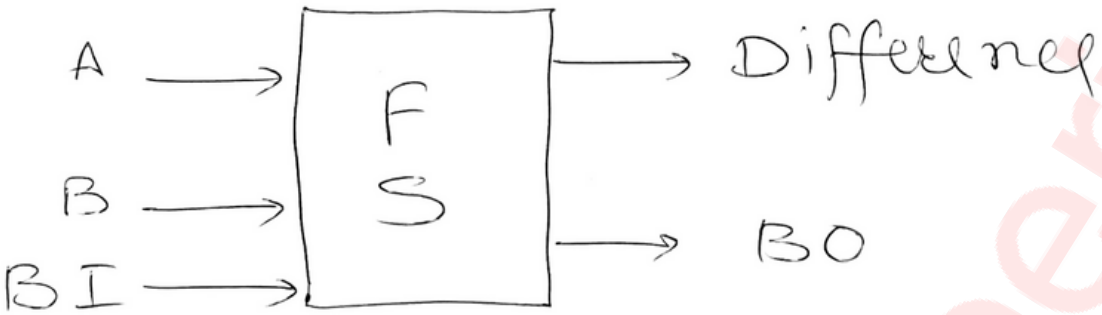
XOR

$\overline{A}$ B

0 1

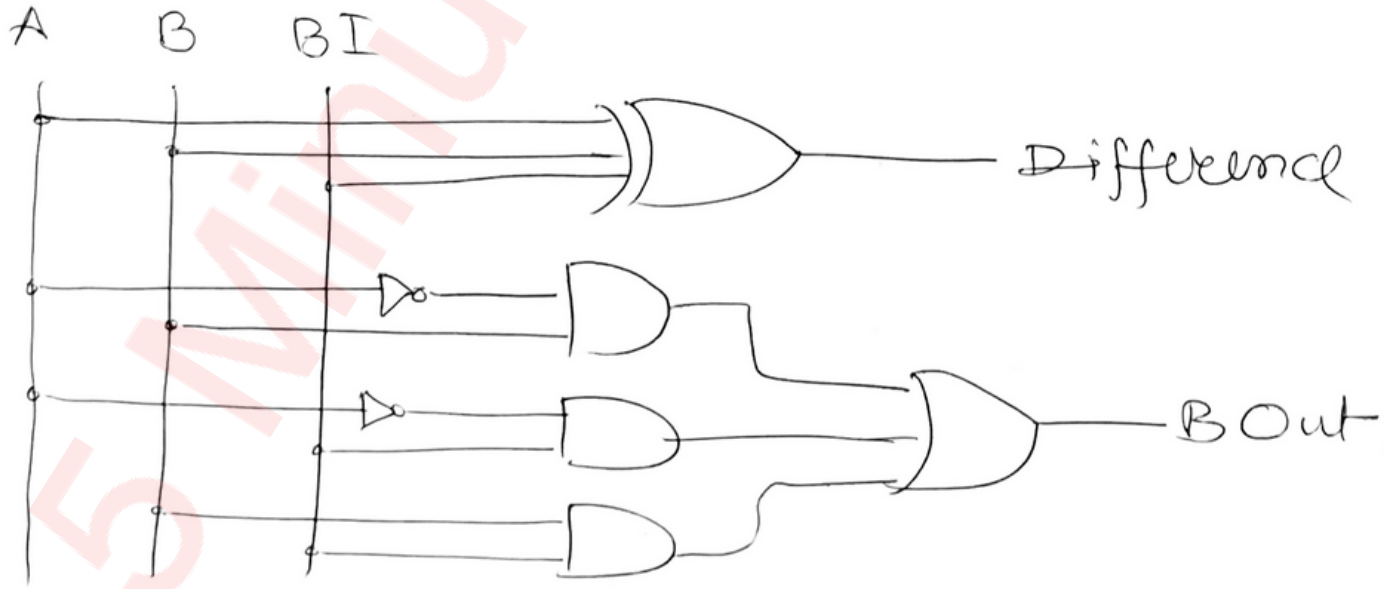


° full Subtractor

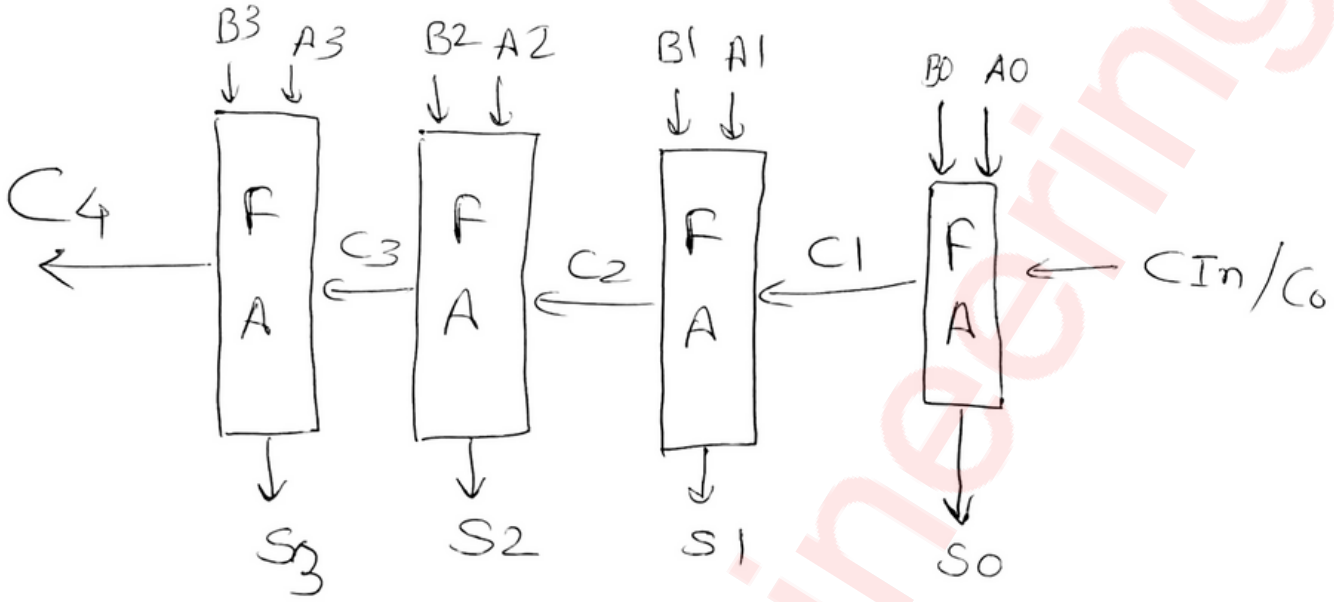


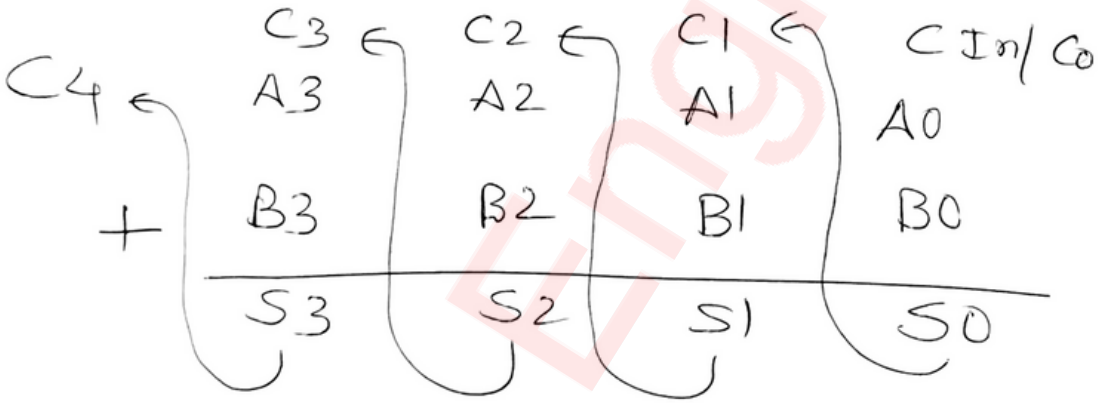
A	B	BI	D	BO
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

compliment



◦ Ripple Adder (4 bit adder)

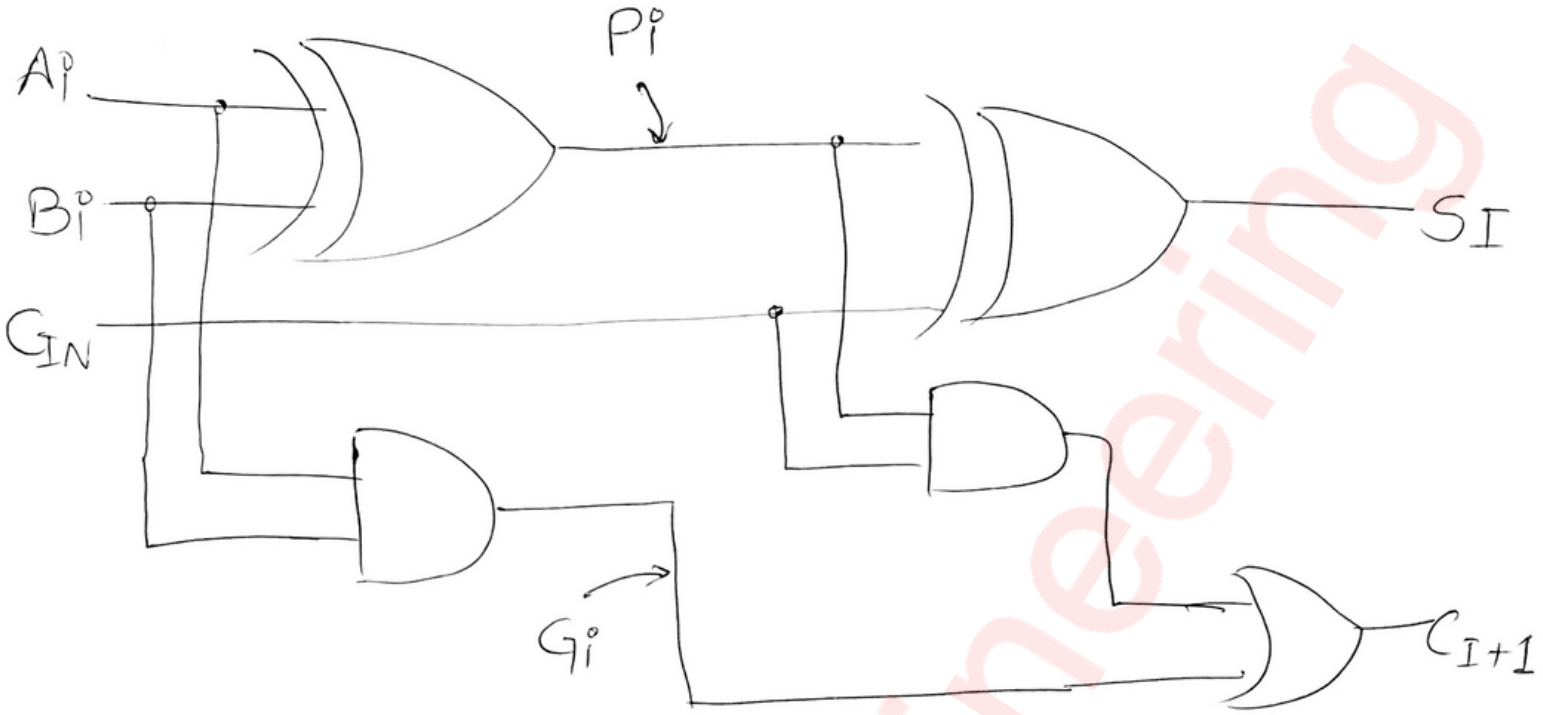




◦ Carry propagation Lag / Delay problem

So we use Look ahead carry adder

Idea is to get C1, C2, C3 from C0 itself.



$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

$$S_i = P_i \oplus C_i \;/\; A_i \oplus B_i \oplus C_i$$

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

$$C_0 = 0$$

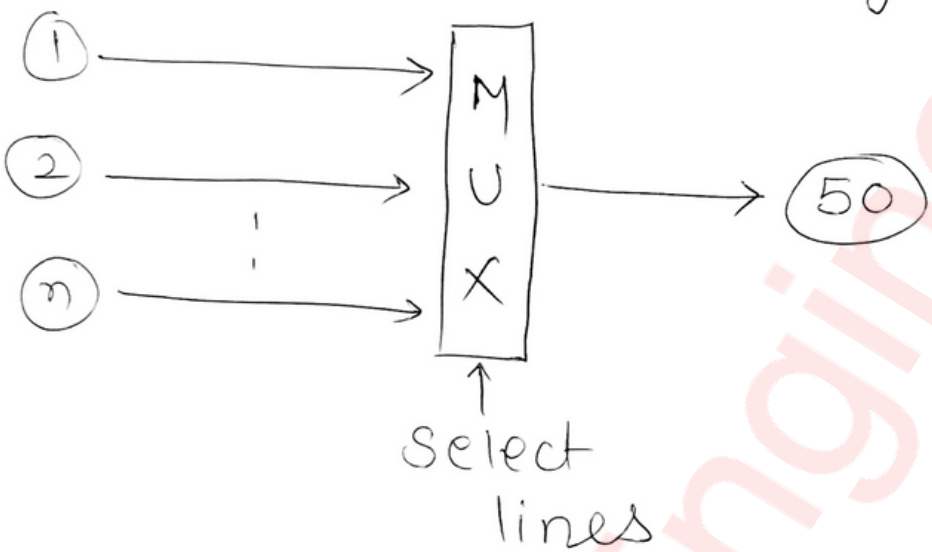
$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$$

$$C_3 = G_2 + P_2 G_1 + P_2 P_1 P_0 C_0$$

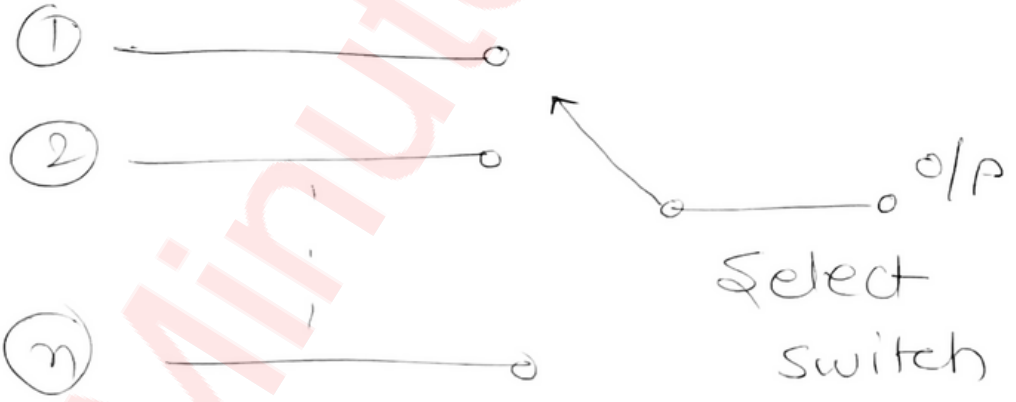
$$C_4 = G_3 + P_3 G_1 + P_3 P_2 G_0 + P_3 P_2 P_1 P_0 C_0$$

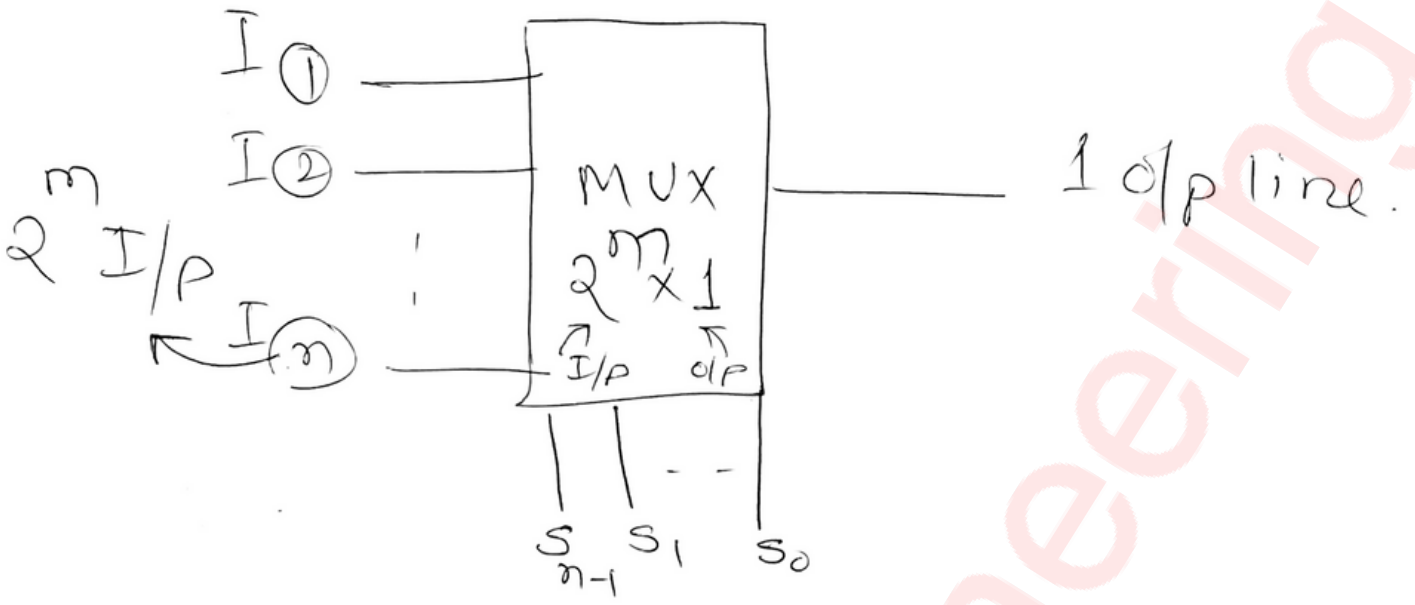
Multiplexer [Trending CC]



Multiple I/p lines — Single o/p Line

[Multiple option select one]

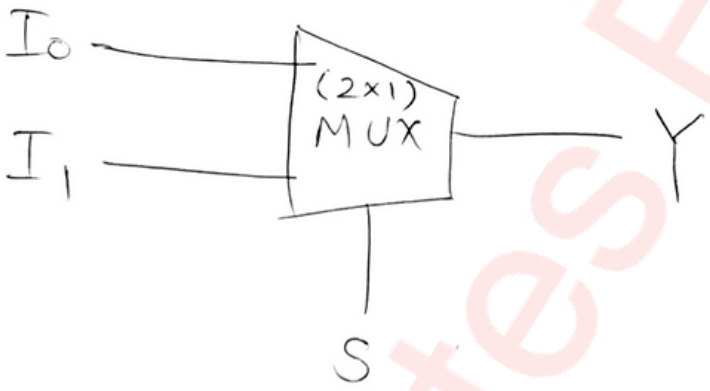




n select lines

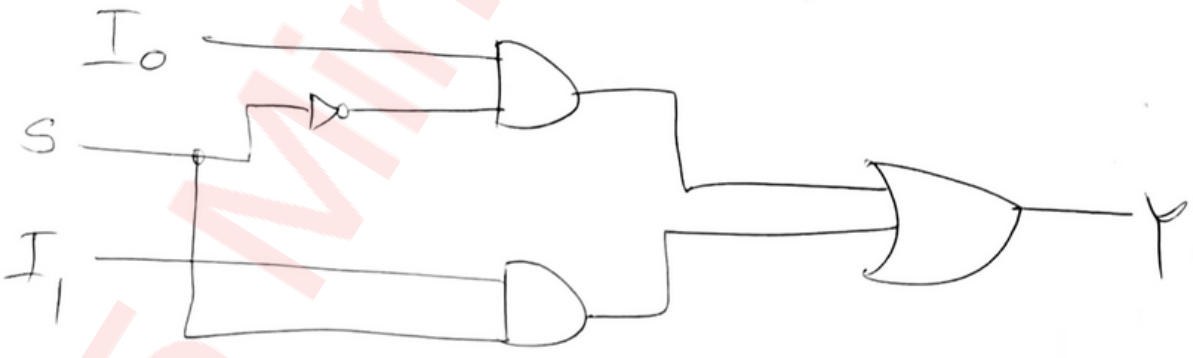
- Two-to-One Multiplexer

I/P → (2×1) → O/P

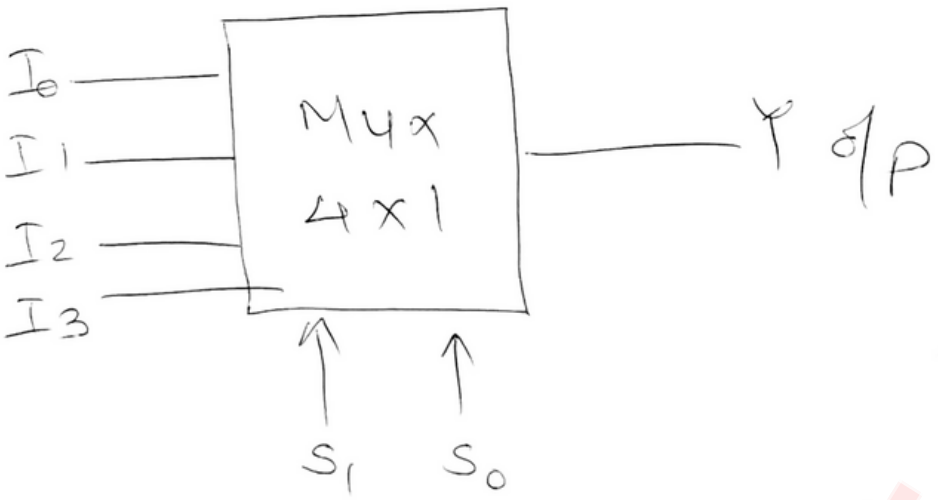


I/P	O/P
S	Y
0 →	I0
1 →	I1

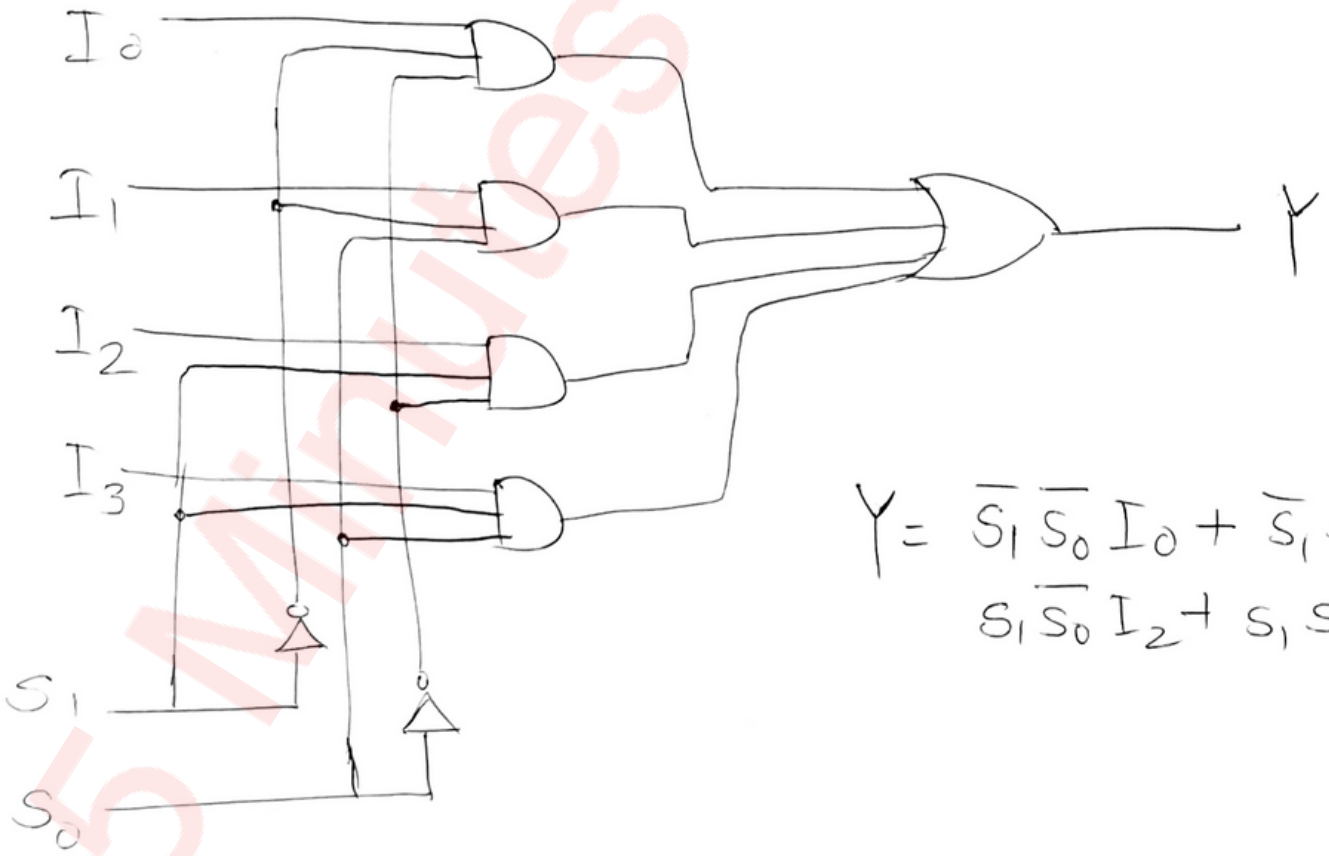
eqn = $Y = \overline{S_0} I_0 + S_0 I_1$



⇒ 4 x 1 Mux

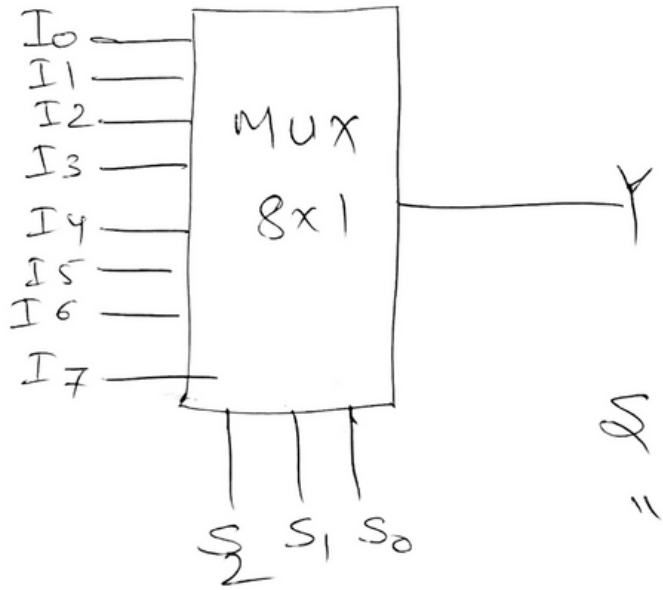


S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



$$Y = \bar{S_1}\bar{S_0}I_0 + \bar{S_1}S_0I_1 + S_1\bar{S_0}I_2 + S_1S_0I_3$$

Eg: 8x1 Mux



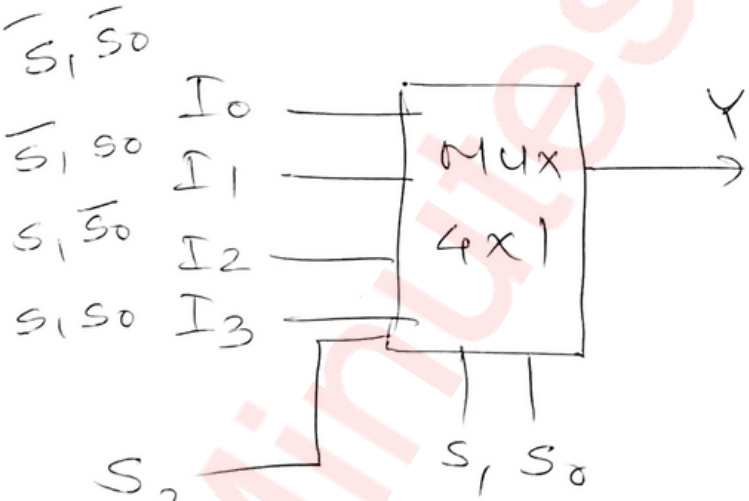
$\Sigma_m(0,1,2,3)$

Simply make.

" $I_0, I_1, I_2 I_3$ " o/p

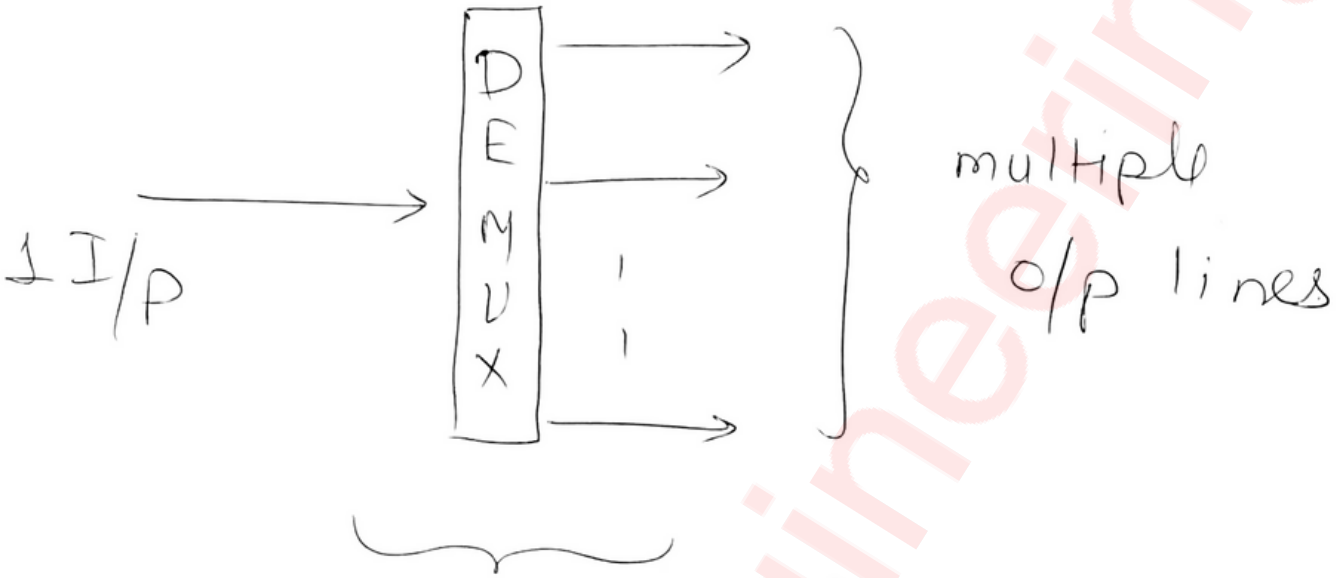
S_0	S_2	S_1	S_0
✓	0	0	0
✓	0	0	1
✓	0	1	0
✓	0	1	1

'OR' we can Implement the same thing using (4x1) Mux.

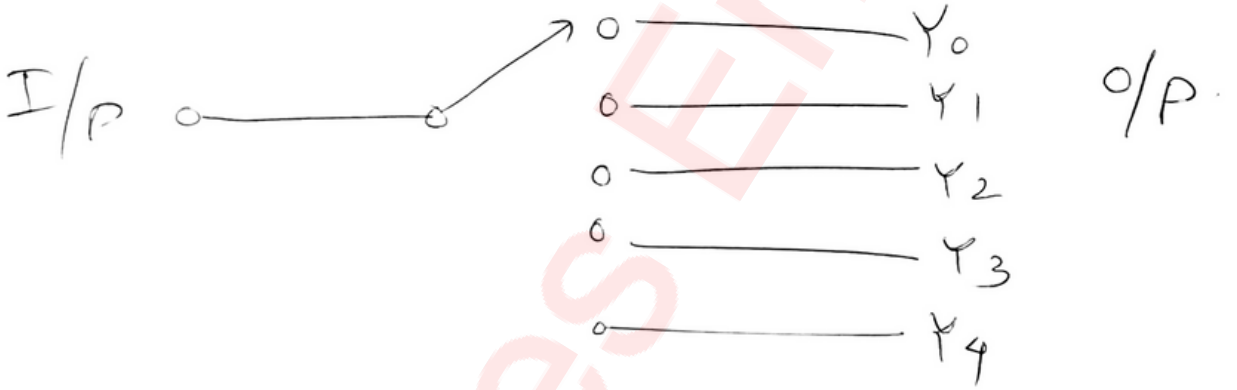


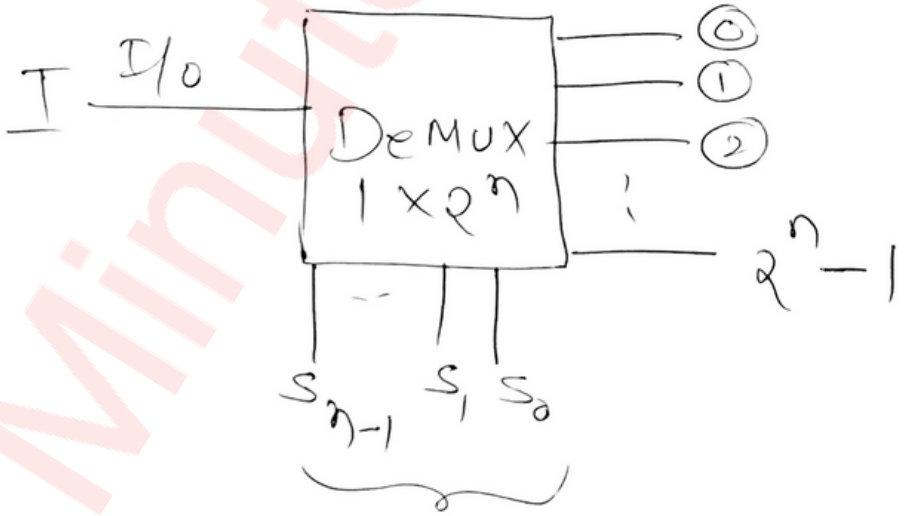
" 0/1 "

Demultiplexer [opposite of Mux]



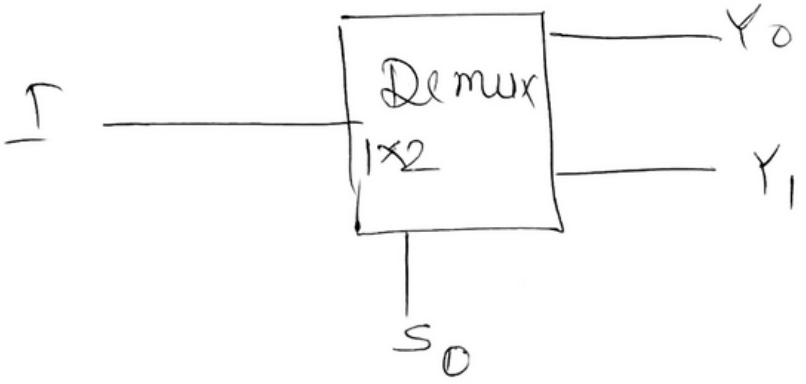
data distributor



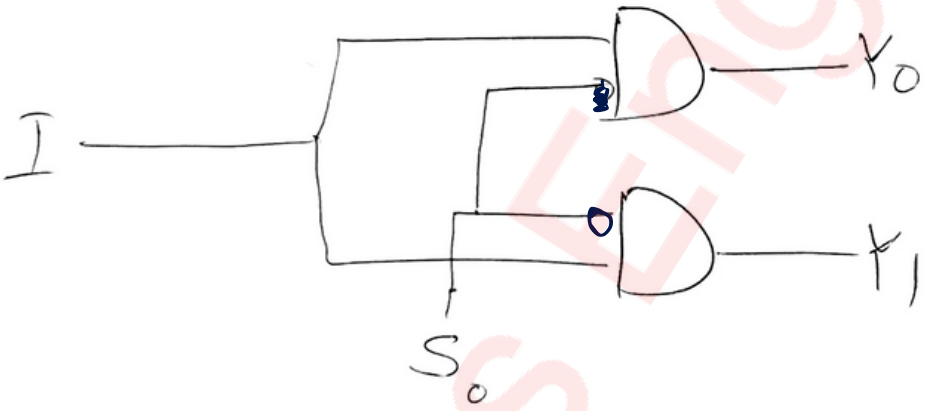


n select lines.

⇒ 1 x 2 Demultiplexer

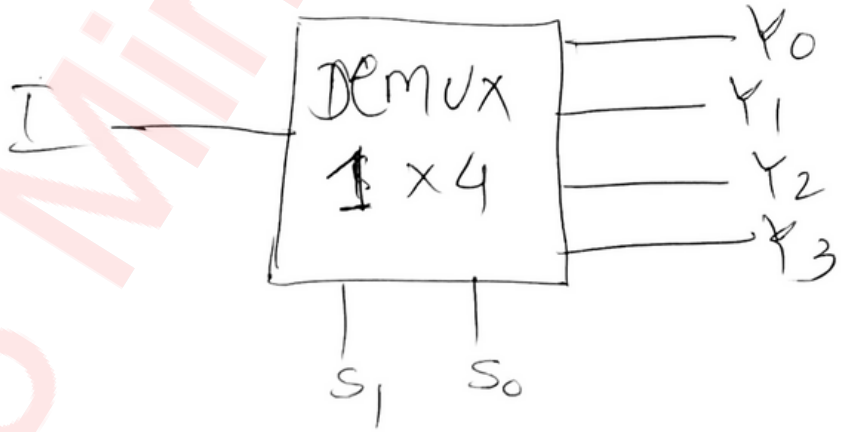


S_0	Y_0	Y_1
0	0	I
1	I	0

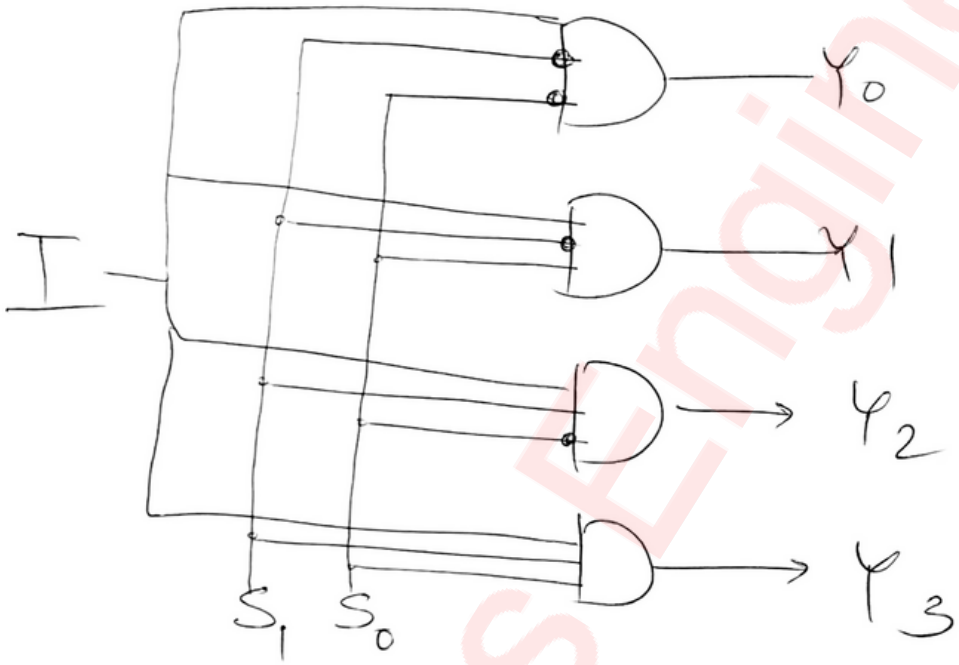


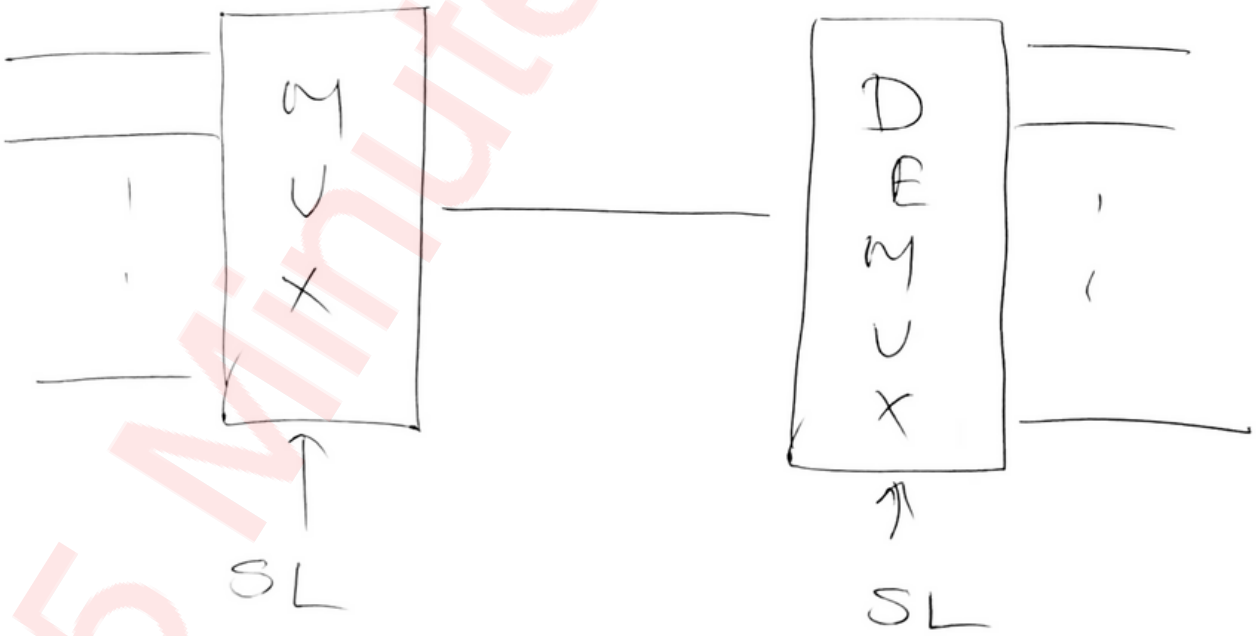
$Y_0 = S_0 \quad Y_1 = \overline{S_0}$

⇒ 1 x 4 Demultiplexer

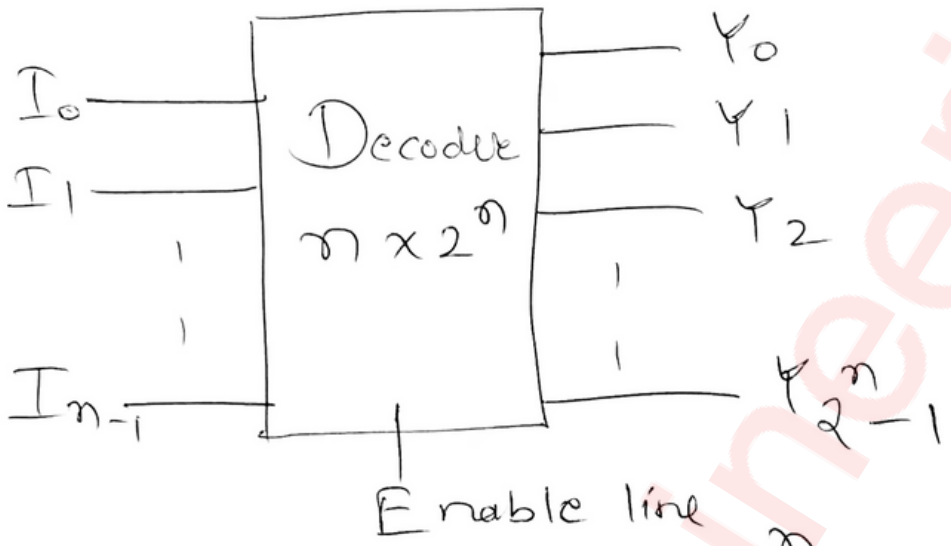


S_0	S_1	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0





Decoder



n bits $\longrightarrow$ 2^n options

Hence $n \times 2^n$

Eg: $n = 2$

I_0	I_1	
0	0	Y_0
0	1	Y_1
1	0	Y_2
1	1	Y_3

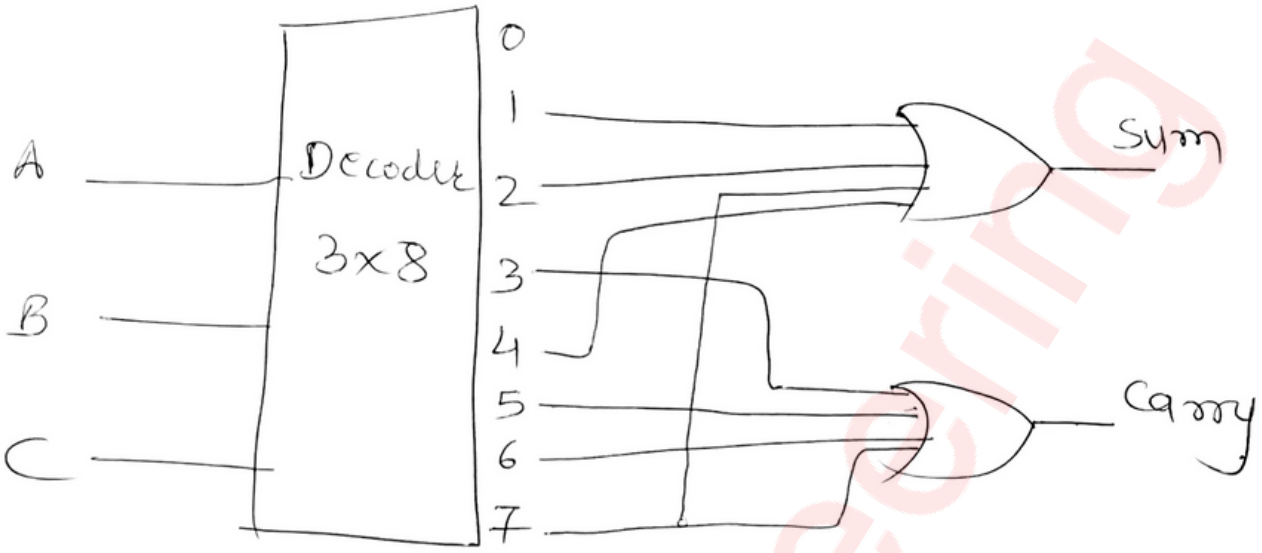
$2^n = 2^2 = 4$

(Decoder) " SL $\leftrightarrow$ I " (DeMux)

I $\rightarrow$ EL
SL $\rightarrow$ I

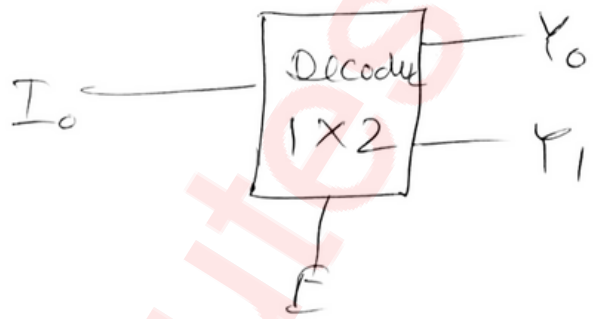
EL $\rightarrow$ I
I $\rightarrow$ SL

Eg

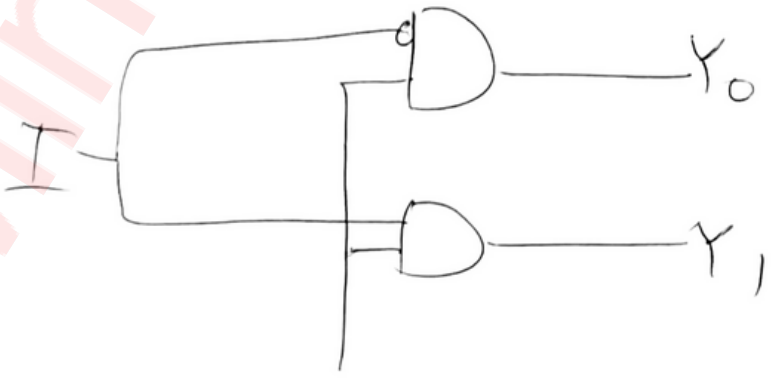


0	1	2	3	4	5	6	7
$\bar{A}\bar{B}\bar{C}$	$\bar{A}\bar{B}C$	$\bar{A}B\bar{C}$	$\bar{A}BC$	$A\bar{B}\bar{C}$	$A\bar{B}C$	$AB\bar{C}$	ABC
0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1

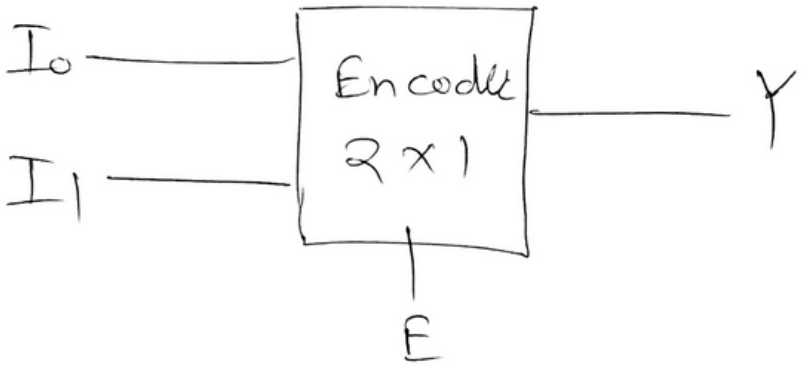
Eg :- 1 x 2 Decoder

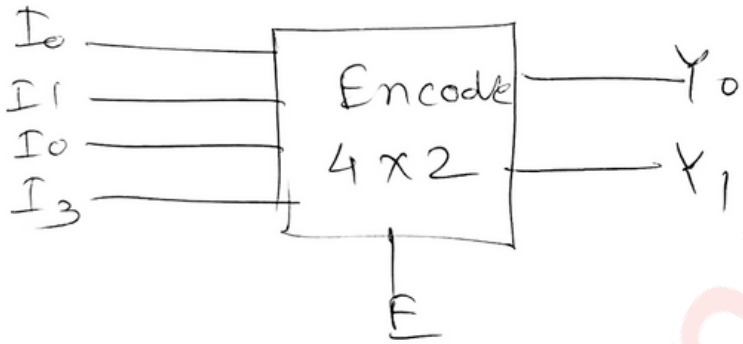


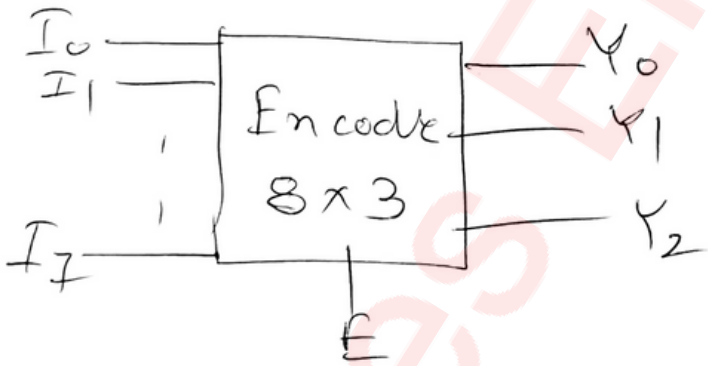
I	Y_0	Y_1
0	0	1
1	1	0



◦ Encoder







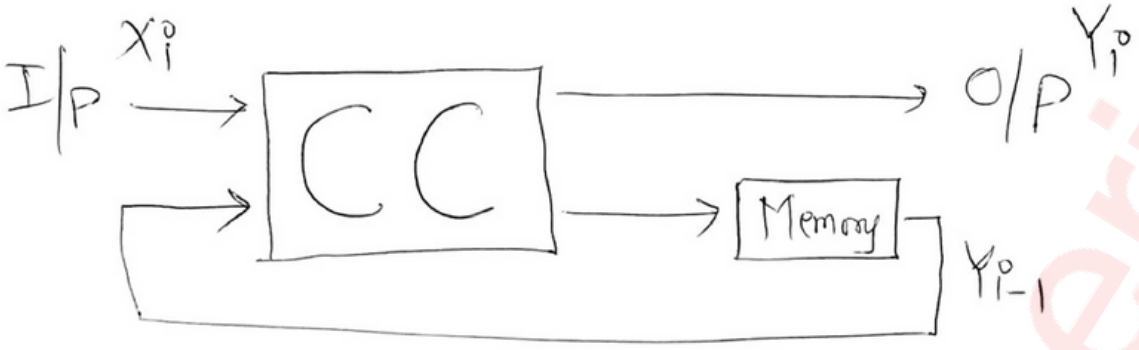
2^n I/P

n O/P

◦ Priority Encoder [Assign the priority to multiple I]

I_3	I_2	I_1	I_0	Y_1	Y_0
0	0	0	1	0	0
0	0	1	DC	0	1
0	1	DC	DC	1	0
1	DC	DC	DC	1	1

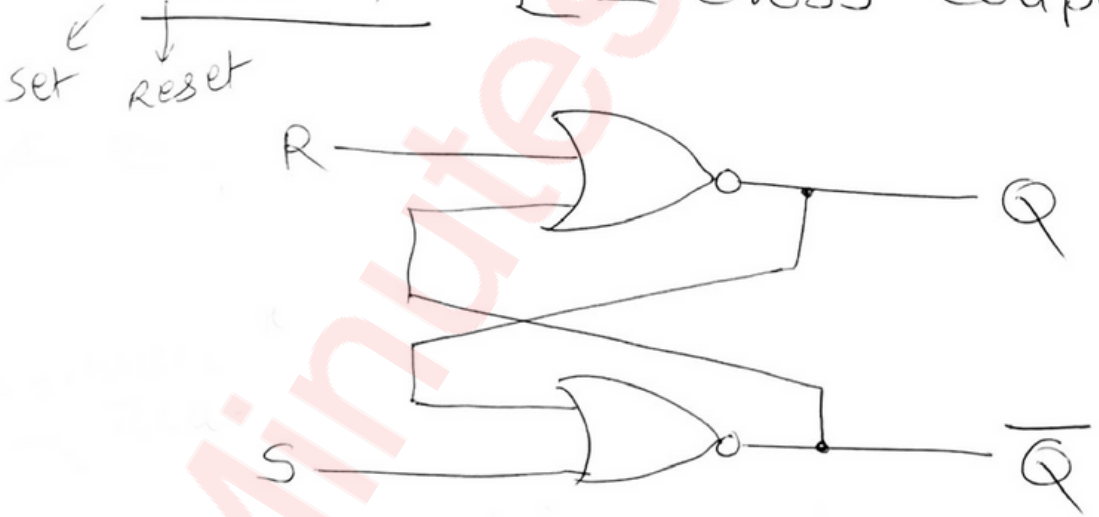
Sequential Circuits = CC + Memory Element



$$F(x_i, Y_{i-1}) = Y_i$$

→ Latch (used to hold a bit)
↳ used in making of FFs.

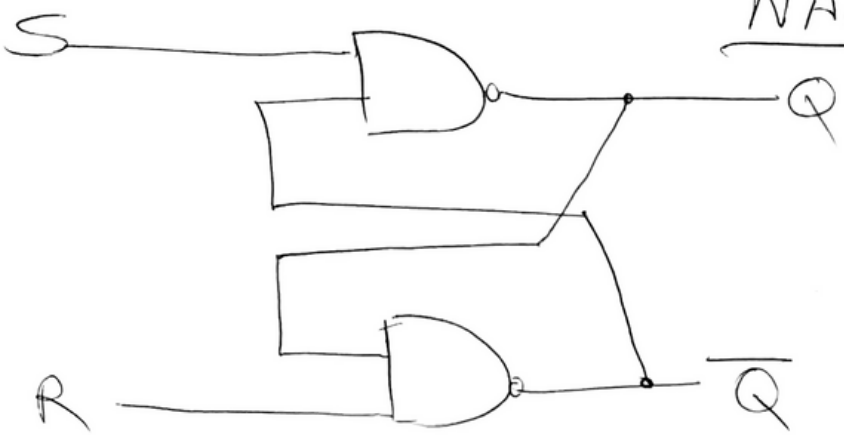
• SR Latch [2 cross coupled NOR gates, NAND]



SR Latch → SR FF → SR TT → SR CT → SR ET

S	R	Q_n	Q_{n+1}	
0	0	0	0	same/NO change.
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	Random
1	1	1	X	

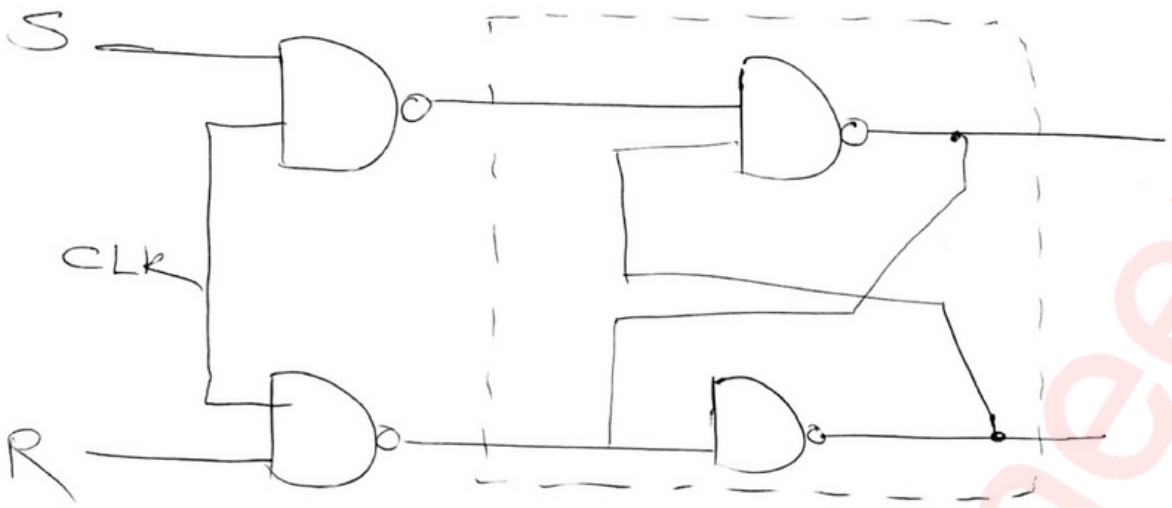
NAND (S ⟷ R)
NOR



S	R	Q_n	Q_{n+1}	
0	0	0	X	Invalid / Random
0	0	1	X	
0	1	0	1	set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	0	Same / NO change
1	1	1	1	

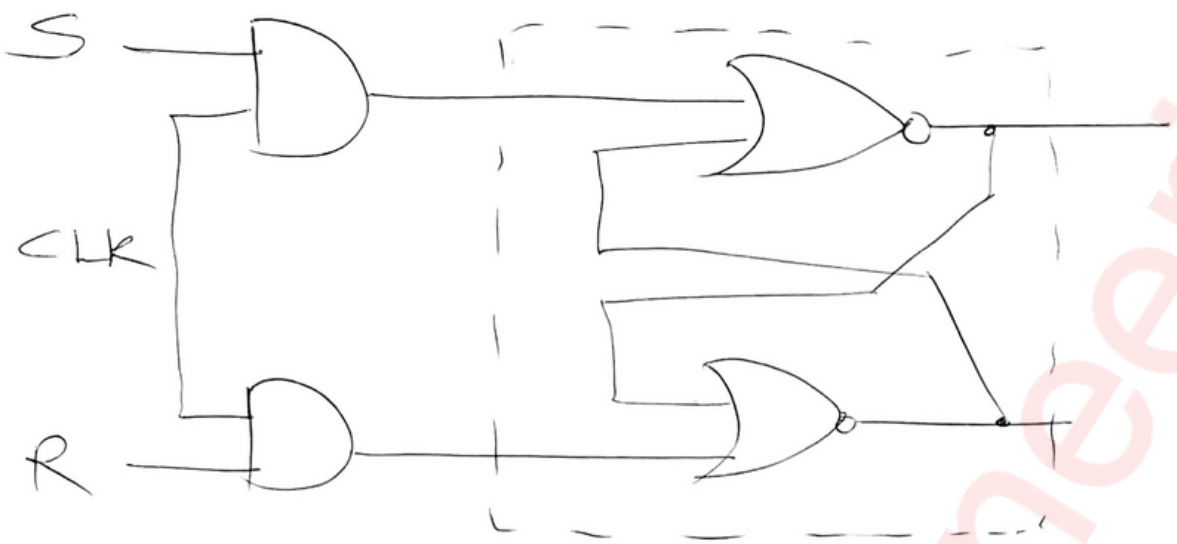
S	R	Q_{n+1}
0	0	Invalid (X)
0	1	set
1	0	Reset
1	1	NO change

SR Flip-Flop (NAND)



S	R	CLK	Q_{n+1}
0	0	T/1	Q_n [Same / Hold]
0	1	T/1	0 [Reset]
1	0	T/1	1 [set]
1	1	T/1	X [Invalid]
X	X	NT (0)	Q_n

(NOR Gate) SR FF



S	R	CLK	Q_{n+1}
0	0	T/↓	[Same/Hold] Q_n
0	1	T/↓	0 [Reset]
1	0	T/↓	1 [set]
1	1	T/↓	x [Invalid]
x	x	NT	Q_n

Same as NOR latch situation.

SR FF Excitation Table

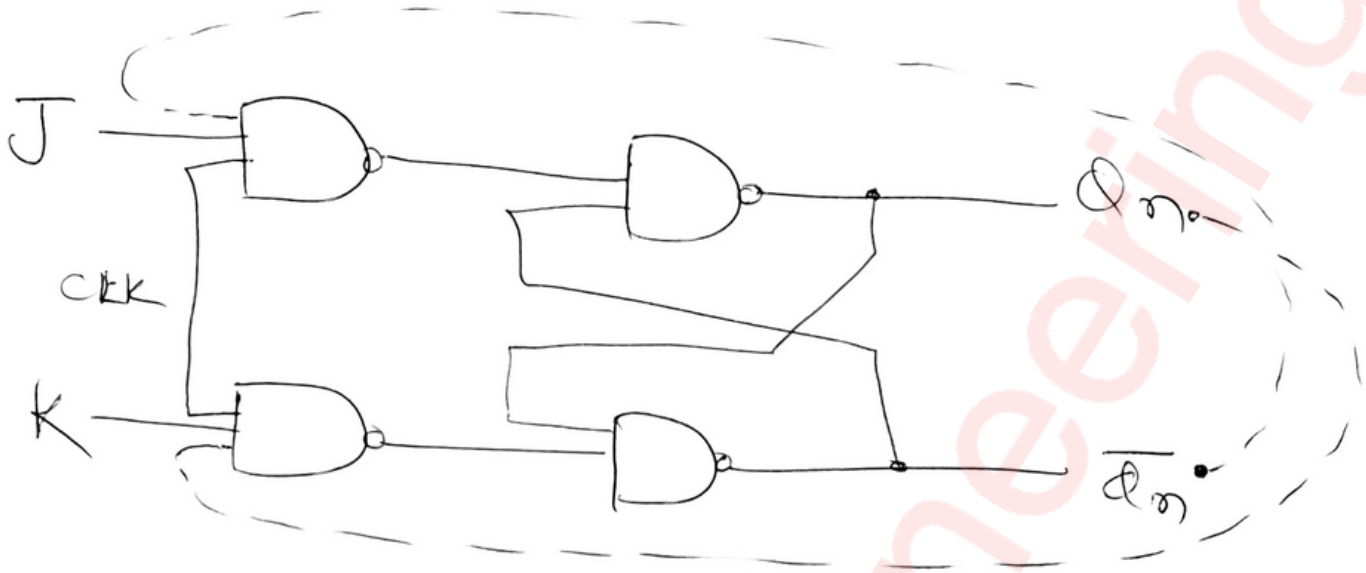
S	R	Qn	Qn+1	
0	0	0	0	Same/hold
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	x	Invalid.
1	1	1	x	

$Q_{n+1} = S + \bar{R}Q_n$ → characteristic eqn.

Qn	Qn+1
0	0
0	1
1	0
1	1

S	R
0	x
1	0
0	1
x	0

J K flip flop



S	R	Q_{n+1}
0	0	Hold
0	1	Reset
1	0	set
1	1	Invalid ← focuses / fix this.

Now when

J = 1, K = 1

→ $Q_n = 1$; [1, 0] → Q_{n+1} Reset (0)

→ $Q_n = 0$; [0, 1] → Q_{n+1} set (1)

finally ↓ $\overline{Q_n}$

Toggle

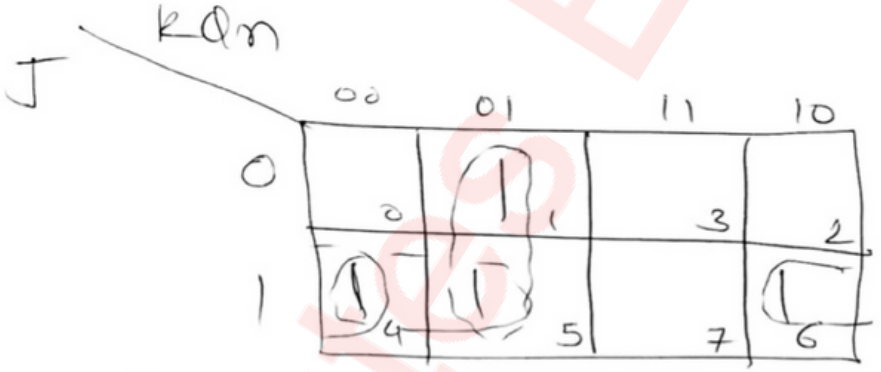
Q_n : 0 → Q_{n+1} : 1

1 → 0

JK flip flop Characteristic Table

⇒

J	k	Q_n	Q_{n+1}	
0	0	0	0	Hold
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	1	$\bar{Q}$
1	1	1	0	

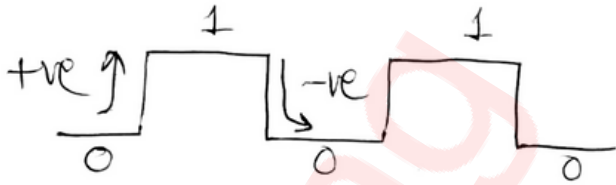


⇒ $Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$ → char. eqn

- Excitment table

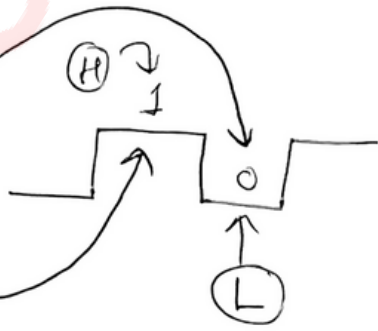
Q_n	Q_{n+1}	J	k
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

→ Triggering

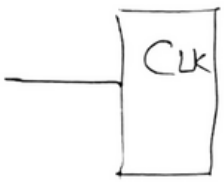


→ Edge → Negative
→ Positive

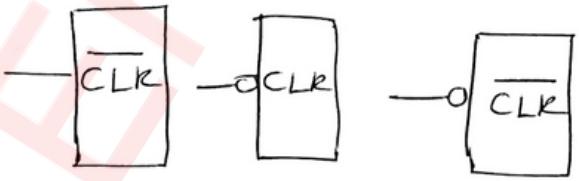
→ Level → Low
→ High



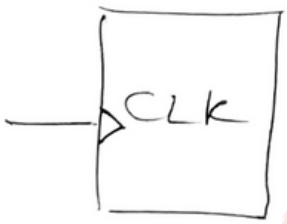
+ve level



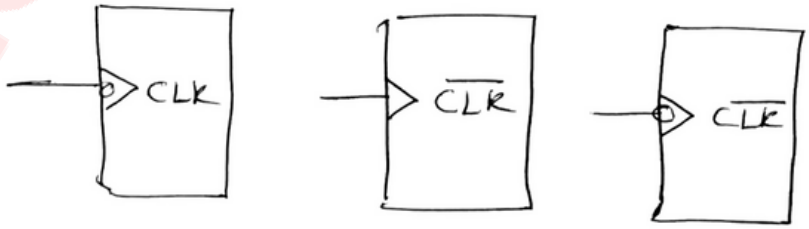
-ve level



+ve edge



-ve Edge



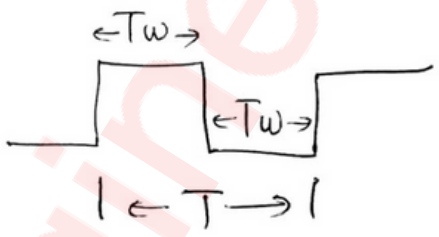
◦ Race condition In JK ff

① K = J = 1 [Toggle]

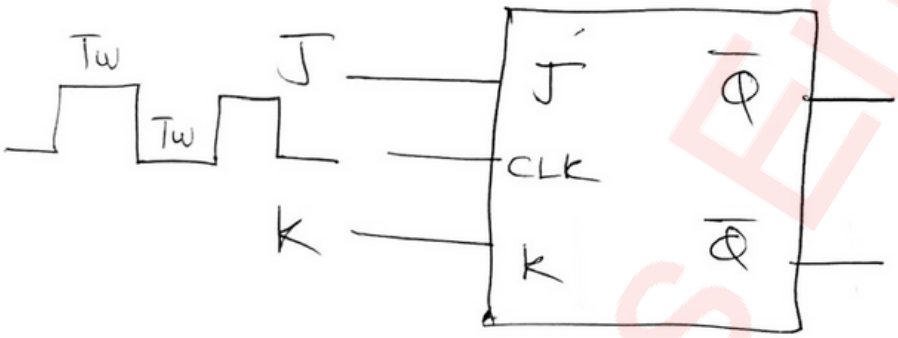
② Level Triggered

③ Td < Tw

↓ delay



$T_w = \frac{1}{2}(T)$



0 → 1 → 0

1 → 0 → 1

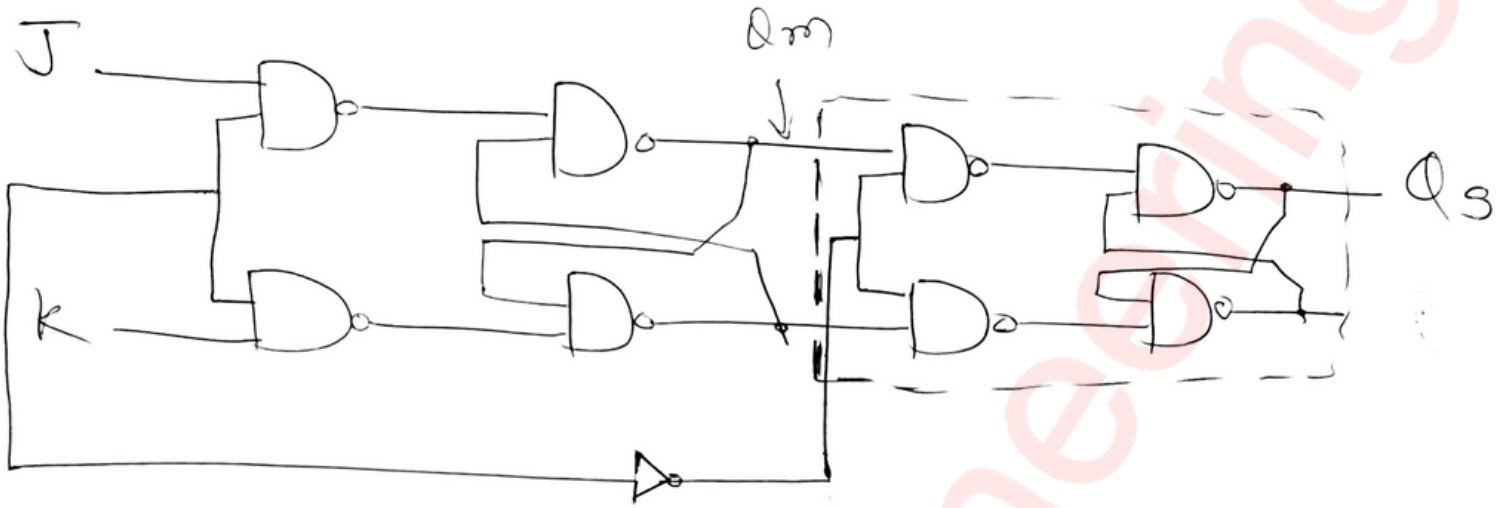
↑ Initially

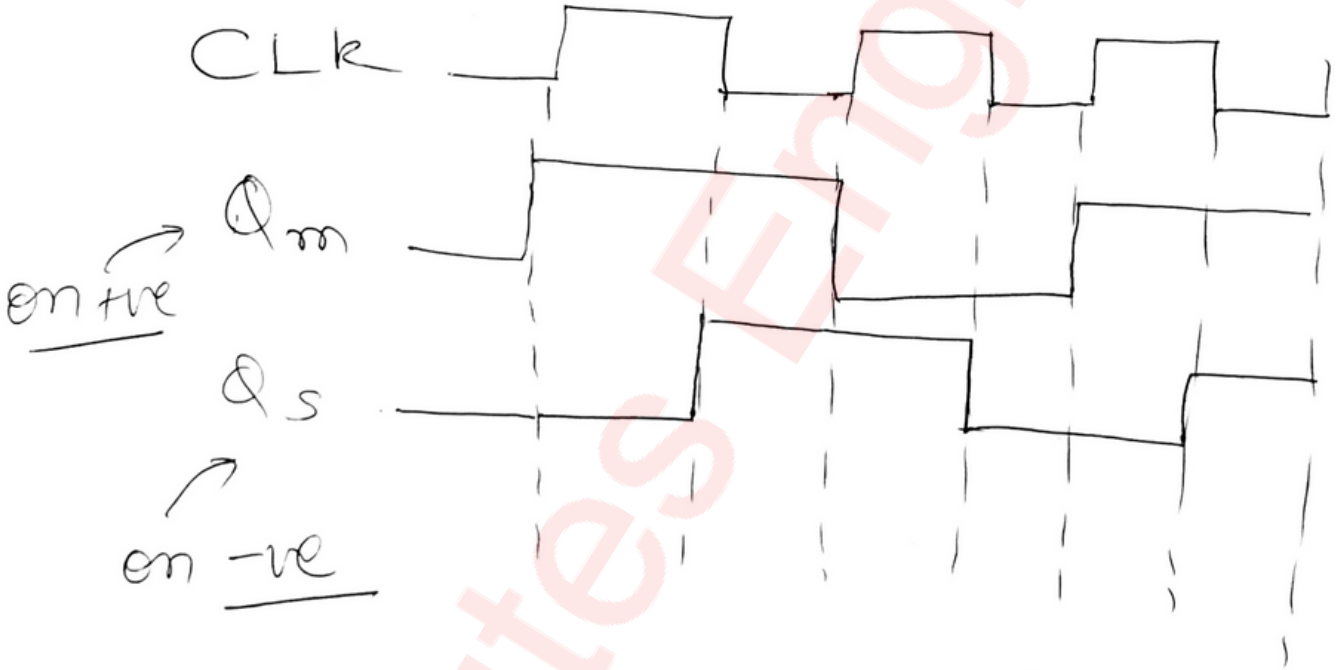
↑ Toggle

↑ Again Toggle

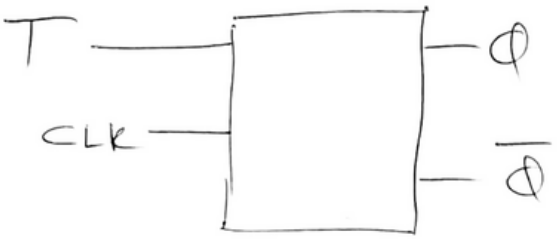
Due to Tw > Td

◦ Master slave JK FF





T flip flop



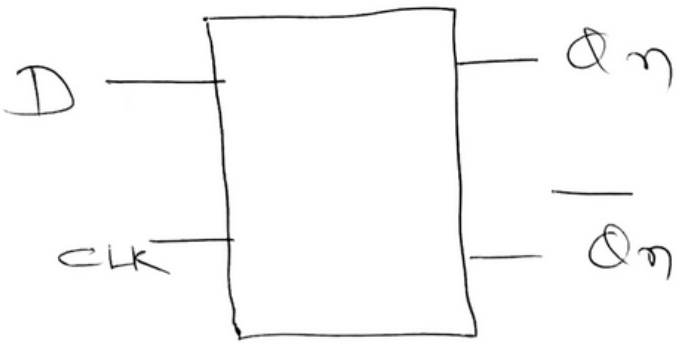
T	Q_{n+1}	
0 →	Q_n	} Toggle behaviour
1 →	$\overline{Q_n}$	

T	Q_n	Q_{n+1}	
0	0	0	] as T = 0 [NO Toggle]
0	1	1	
1	0	1	} as T = 1 [Toggle]
1	1	0	

$$Q_{n+1} = T \oplus Q_n$$

Q_n	Q_{n+1}	T	
0	0	0	
0	1	1	} mirror
1	0	1	
1	1	0	

D flip flop



D	Q_{n+1}	
0 →	0	Same as D
1 →	1	

D	Q_n	Q_{n+1}	
0	0	0	Same as D
0	1	0	
1	0	1	
1	1	1	

$$Q_{n+1} = D$$ characteristic eqn.

Q	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

SR → D ff conversion

O/P → Dff characteristic Table

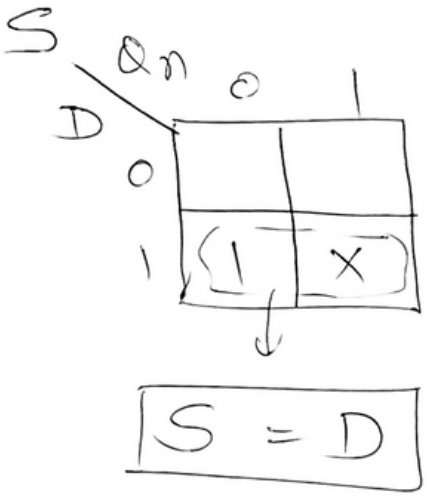
I/P → SR Excitation Table

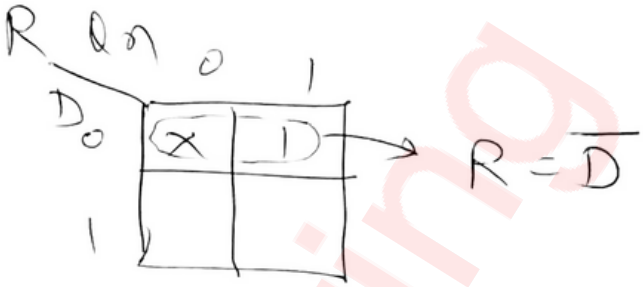
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

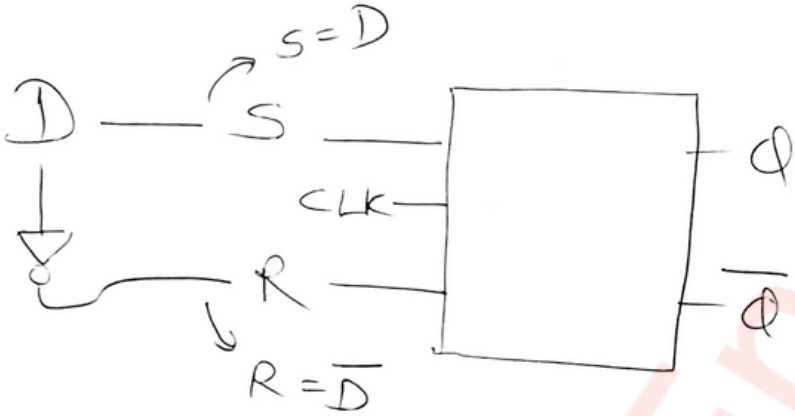
D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

D	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

Refer to ET (SR)







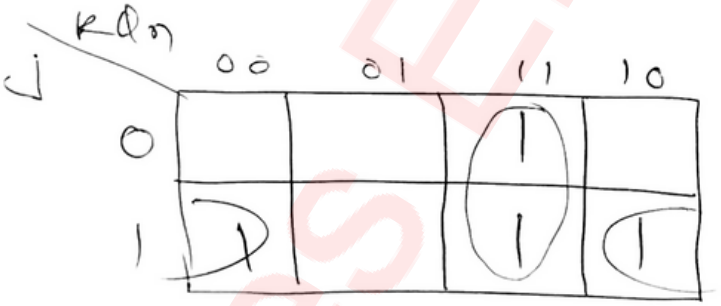
⇒ T → JK ff conversion

↓(ET) (CT)

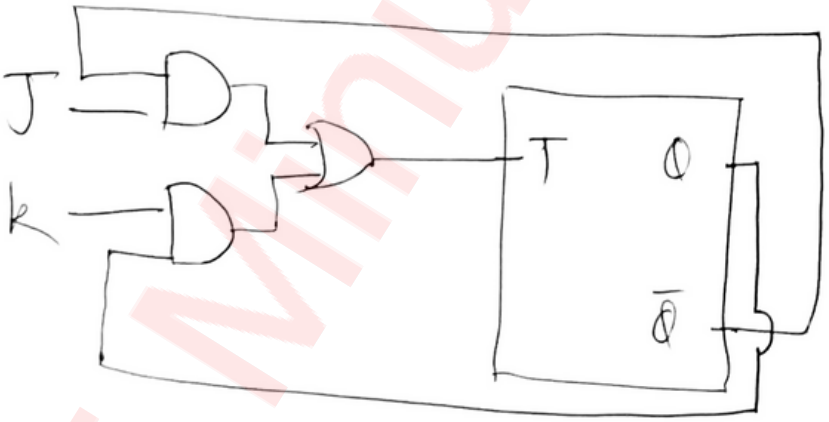
Qn	Qn+1	T
0	0	0
0	1	1
1	0	1
1	1	0

J	k	Qn	Qn+1	
0	0	0	0	] samp.
0	0	1	1	
0	1	0	0	] Reset
0	1	1	0	
1	0	0	1	] Set
1	0	1	1	
1	1	0	1	] Toggle
1	1	1	0	

J	k	Q_n	Q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1



$$T = J\bar{Q}_n + kQ_n$$

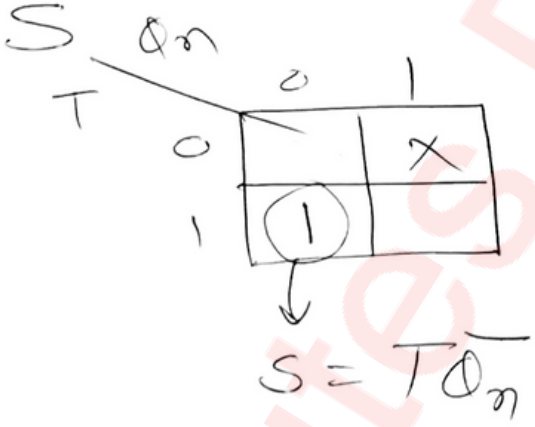


SR — T ff Conversion

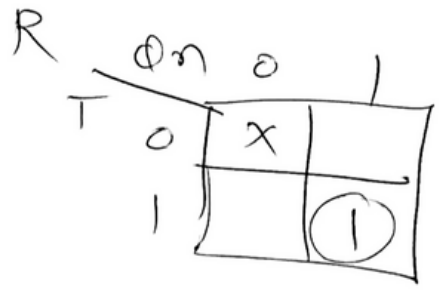
(ET) (CT)

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

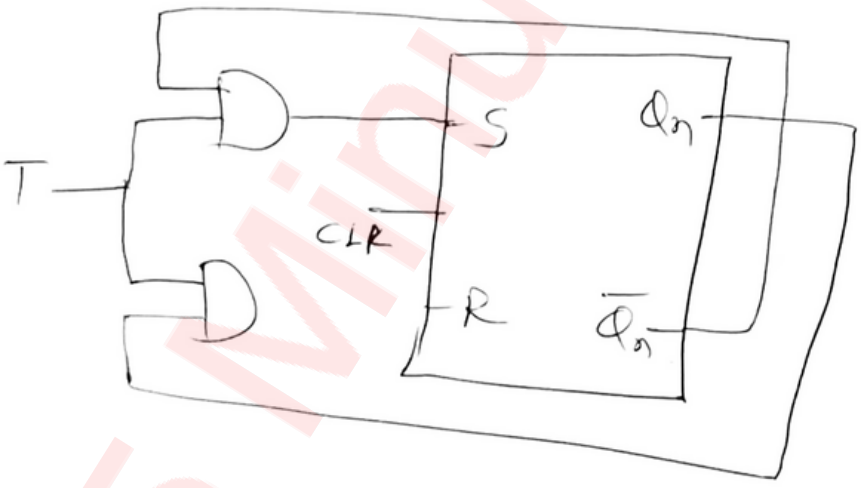
S	R
0	X
X	0
1	0
0	1



$S = T\bar{Q}_n$



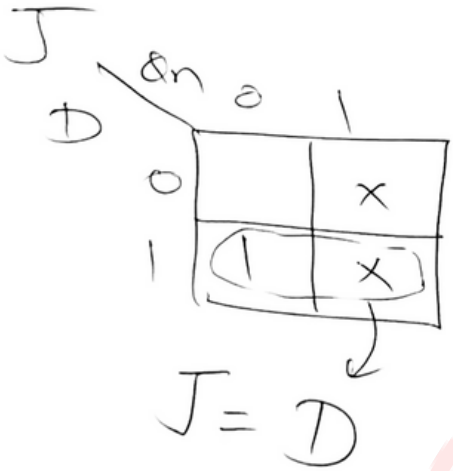
$R = TQ_n$

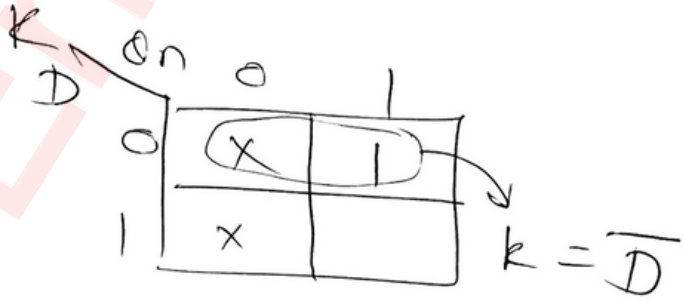


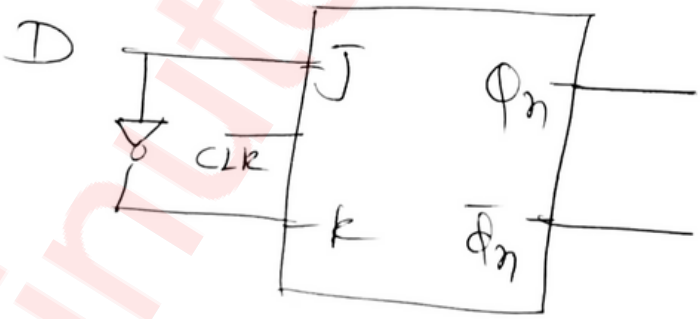
Jk — D ff conversion

(ET) (CT)

D	Q_n	Q_{n+1}	J	k
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	1	X	0







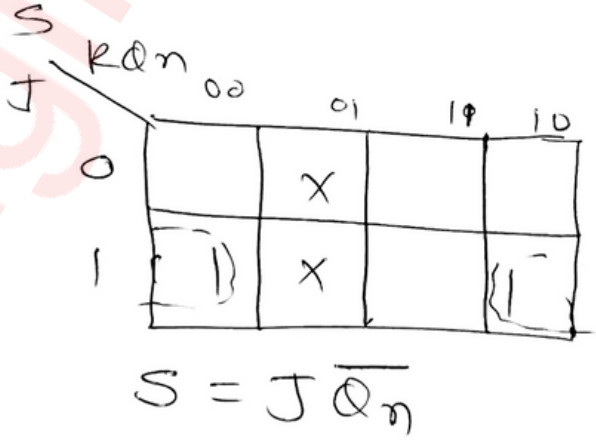
SR → JK Conversion

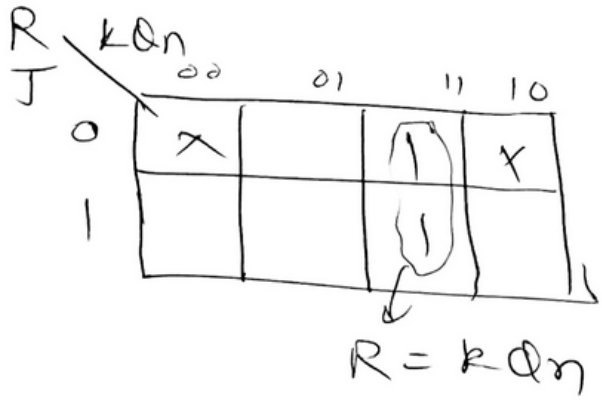
(ET) (CT)

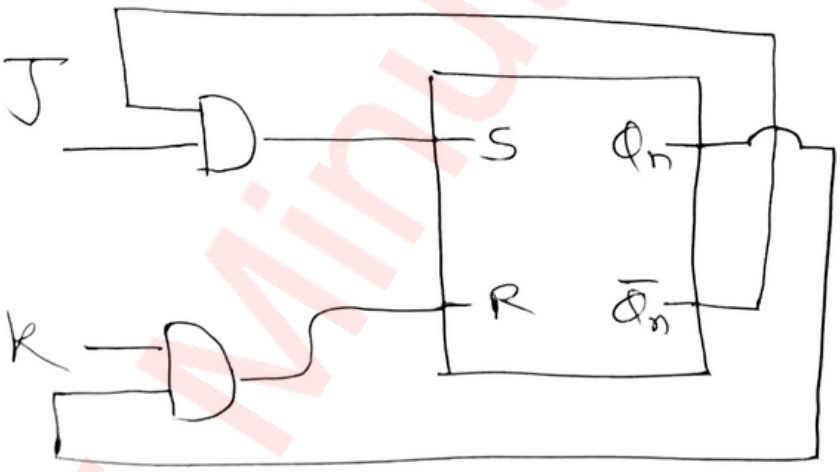
SR (ET)

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

J	K	Q_n	Q_{n+1}	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







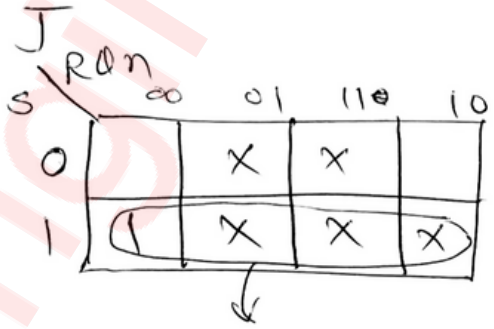
JK → SR Conversion

(ET) (CT)

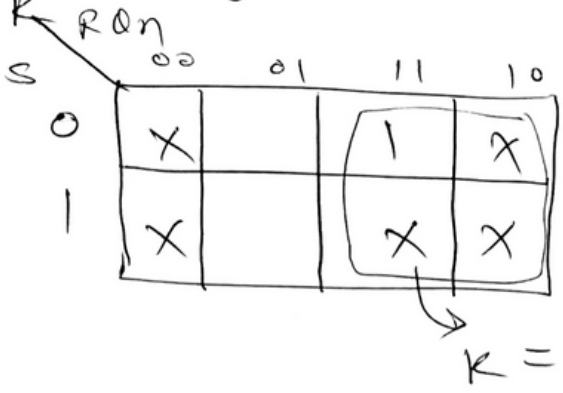
JK (ET)

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

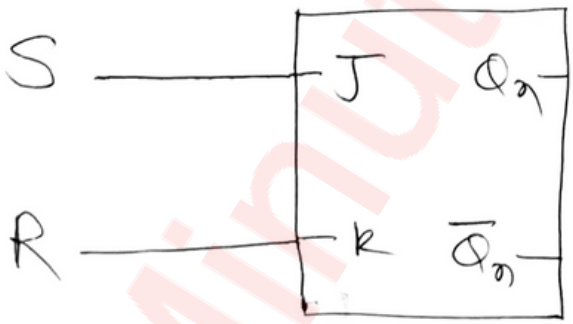
S	R	Q_n	Q_{n+1}	J	K
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	X	1
1	0	0	1	1	X
1	0	1	1	X	0
1	1	0	X	X	X
1	1	1	X	X	X



$J = S$



$K = R$



- **Counters**: count / frequency / occurrence no. of times.
  [store & Display]

Eg: ◦ Washing maching
◦ Microwave oven
◦ Traffic lights

→ n-bit counter
↓ have
n flip flops
[0 to $2^n - 1$]

- Types
  - → Synchronous (parallel counter)
  - → Asynchronous (Ripple counter)

Eg: 0 → 1 → 2 → 3 → 4 → 5 (↺)

↓
Mod 6 Counter
↑
unique states

Q: at 10 clk pulse whats the state.

0 → 1 → 2 → 3 → 4 → 5 → 0 → 1 → 2 → 3
↓
4 ✓

Simply multiple of '6' will take you to state '0' or value 0

Like: 12, 18, 24, 30, 36. etc CLK pulses

for '1' value 13, 19, 25, 31, 37 CLK pulses

for '2' value 14, 20, 26, 32, 38 CLK pulses

Short cut

m CLK pulse for n mod counter

Simply: "'m' mod 'n'" [If it starts from '0']

eg: 12 mod 6 = 0

13 mod 6 = 1

14 mod 6 = 2

Q: for mod n counter [no. of ffs]

use: $2^k \geq n$ mod 6 ← n

$2^k \geq 6$

$2^1 \geq 6$ x $2^2 \geq 6$ x

$2^3 \geq 6$ ✓

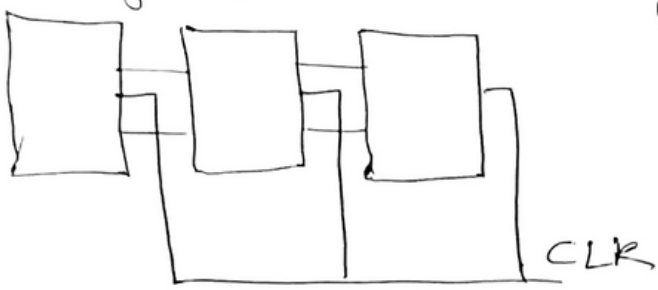
k = 3

eg:- 0 → 2 → 9 → 4 (so, consider the max bits for no. of ffs)

0, 2, 4: 3 bits but 9: 1001 (4 bits)

0, 10, 100

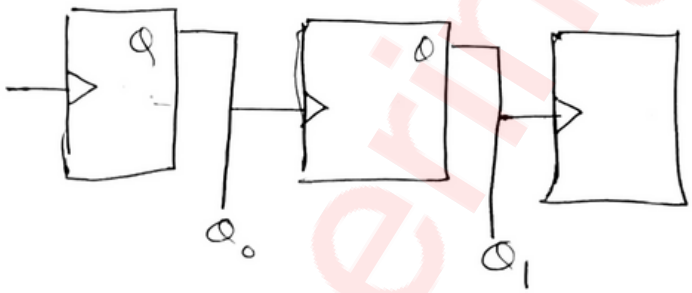
- Synchronous Design



- Parallel nature
- Random sequence ($0 \to 2 \to 4 \to 7$)
- faster
- But complex
- High cost
- delay factor ↓ (PD)
- Eg: Ring & Johnson counters

Vs

- Asynchronous Design

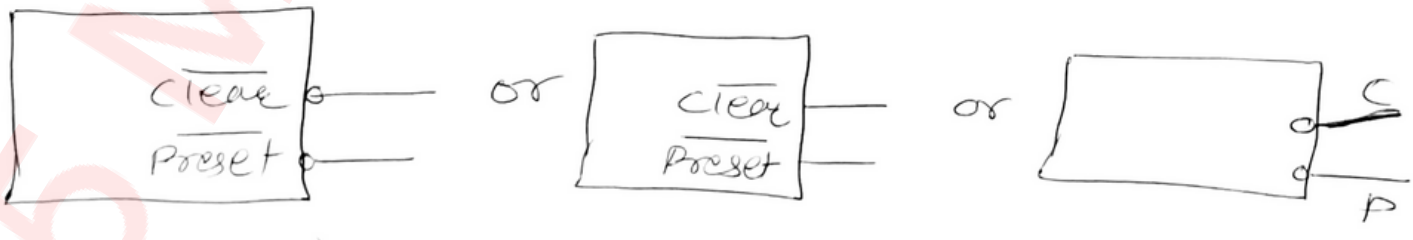


- Ripple nature / serial nature
- Up or down sequence ($0 \to 1 \to 2 \to 3$) or ($3 \to 2 \to 1 \to 0$)
- slower
- But simple & easy
- low cost
- delay factor ↑ (PD)
- Ripple counter.

⇒ Q (o/p of ff)

→ Present (set → 1)

→ clear (Reset → 0)

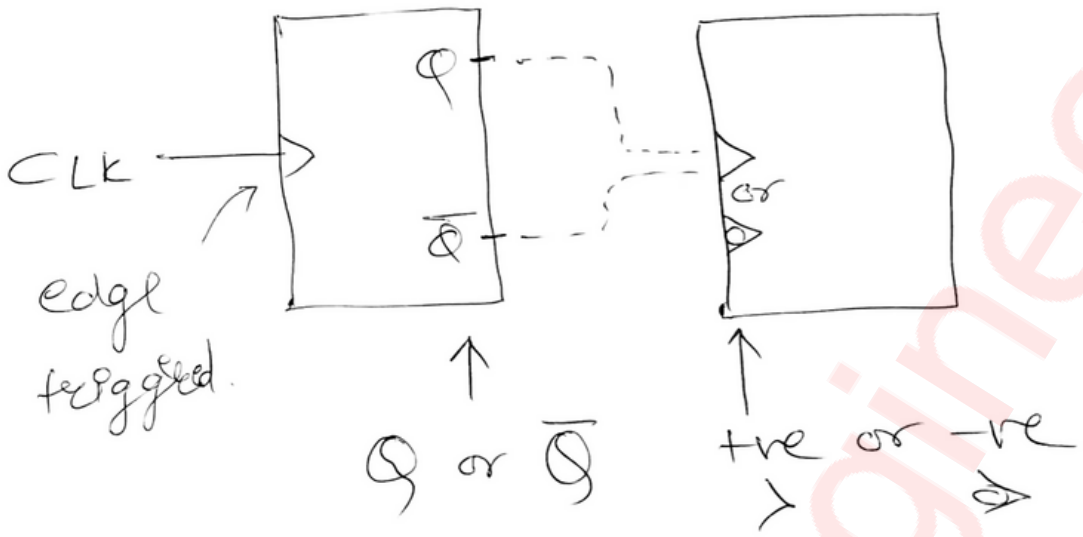


Low enabled.

- Up & down counters

  → (0 → 1 → 2 → 3)

  (3 → 2 → 1 → 0)



	+ve	-ve
+ve Q	Down	Up
-ve Q̄	Up	down

- Steps to design synchronous counter

(1) Let's say Given :- 0 → 3 → 2 → 1 (seq.)

(2) no. of ff's : max no = 3 (bits required = 2)
hence 2 ff's [can use any ff]

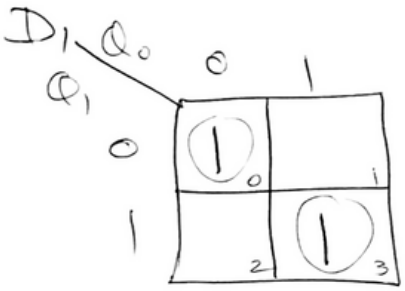
(3) Let's say we use 'D' ff.

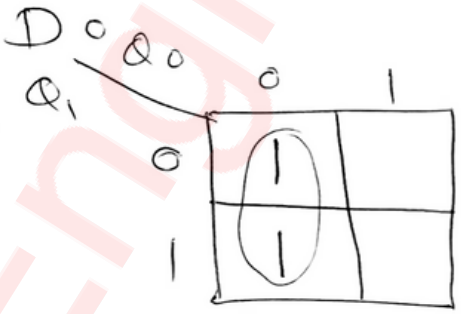
(4) will require excitation table of D ff

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

⑤ Sequence (Present state & next state)

	PS		NS		↓ for D values refer (ET)	
	Q_1	Q_0	Q_1	Q_0	D_1	D_0
0)	0	0	3) 1	1	1	1
1)	0	1	0) 0	0	0	0
2)	1	0	1) 0	1	0	1
3)	1	1	2) 1	0	1	0

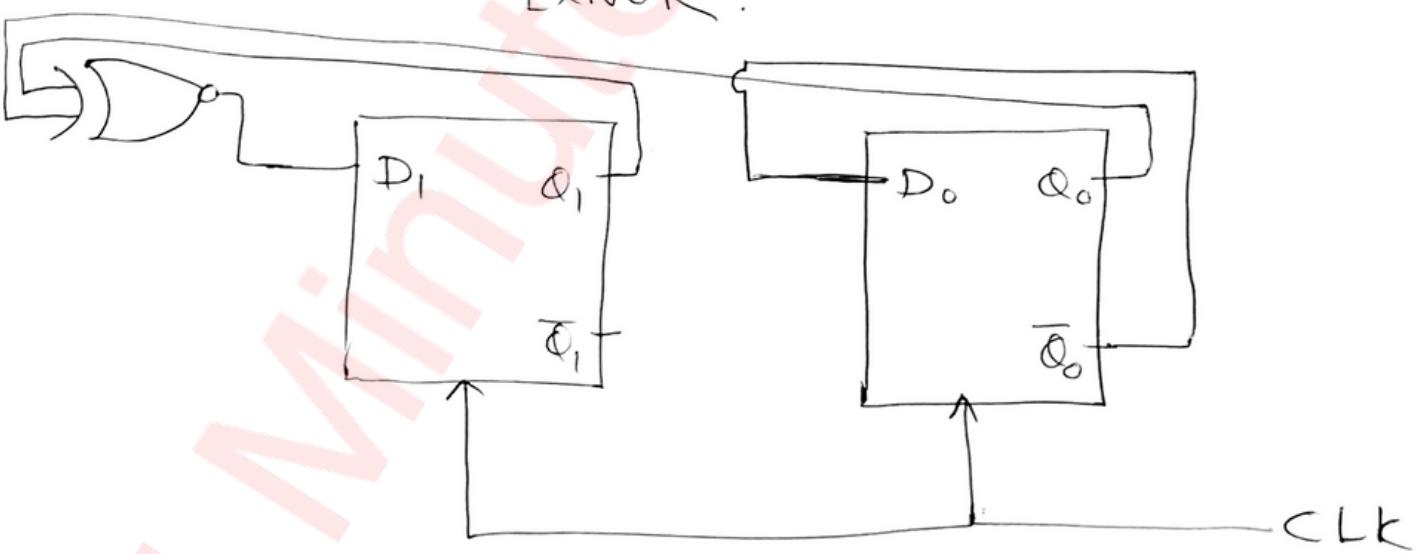




$D_1 = \overline{Q_1}\,\overline{Q_0} + Q_1 Q_0$ $\quad D_0 = \overline{Q_0}$

$= Q_1 \odot Q_0$

↑ EXNOR



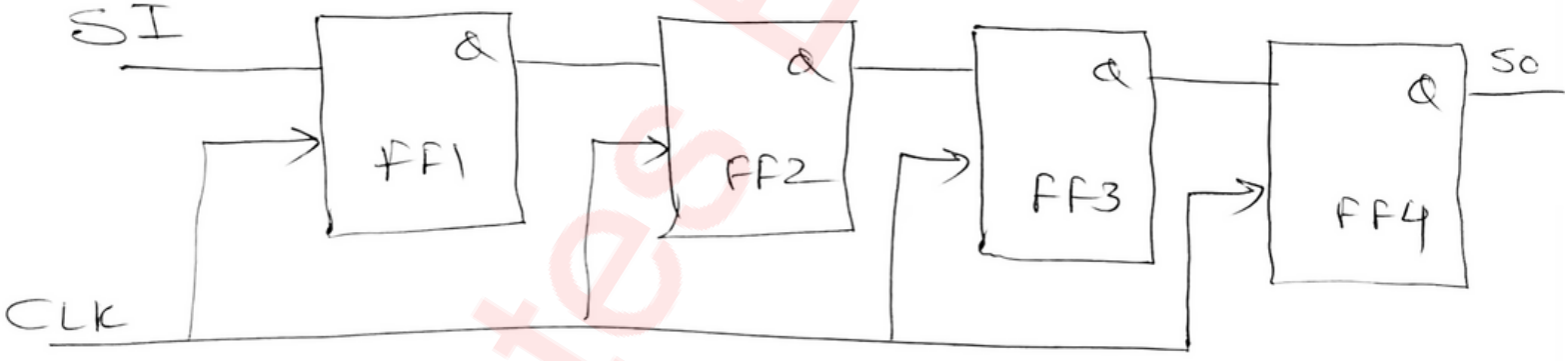
- **Registers**: Storage primary unit using ff.

(D - ff) acts just as buffer memory.

Modes [Shift Registers]

- Serial In Serial Out
- Serial In parallel out
- Parallel In parallel out
- Parallel In serial out

① **SISO**



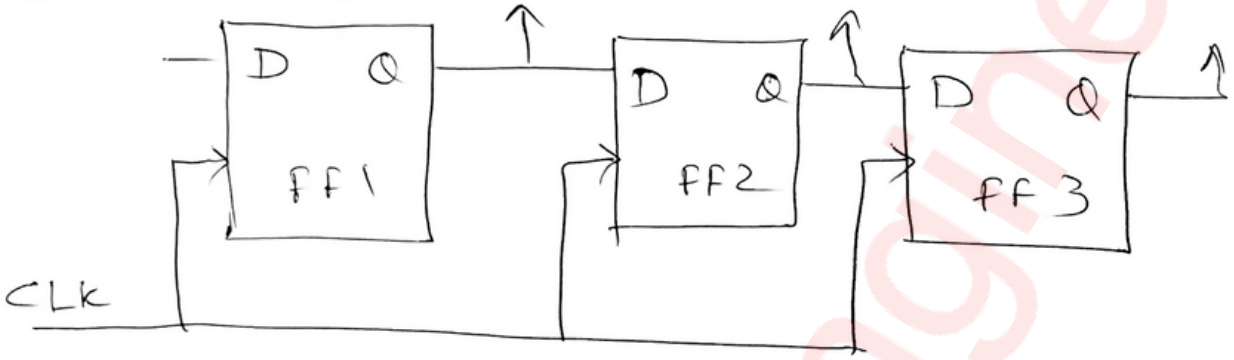
I/P	Q_3	Q_2	Q_1	Q_0	O/P
0	x	x	x	x	x
1	0	x	x	x	x
0	1	0	x	x	x
1	0	1	0	x	x
0	1	0	1	0	0
1	0	1	0	1	1

↑ clock

- No. of clocks to write : 4 for 4 ff
  i.e 'n' for n ff
- No. of clock to Read : 'n-1'

Total = n + n-1
= 2n-1

② SIPO

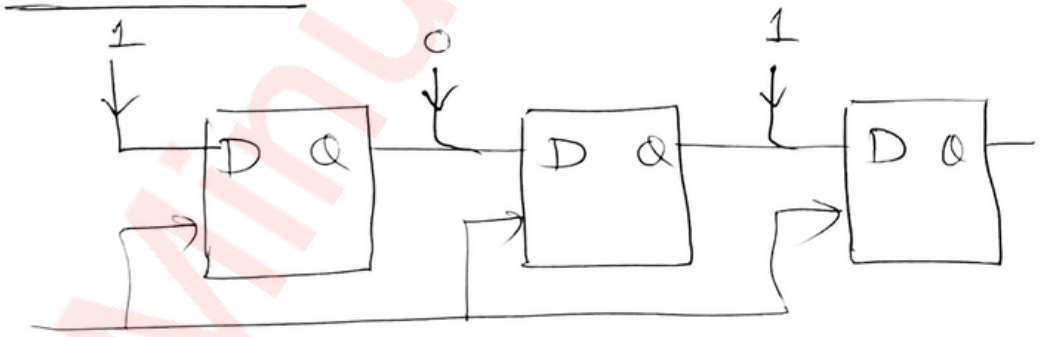


here I/p behaviour is same
so 'n' clock cycle.

But O/p is parallel so '0' CLK

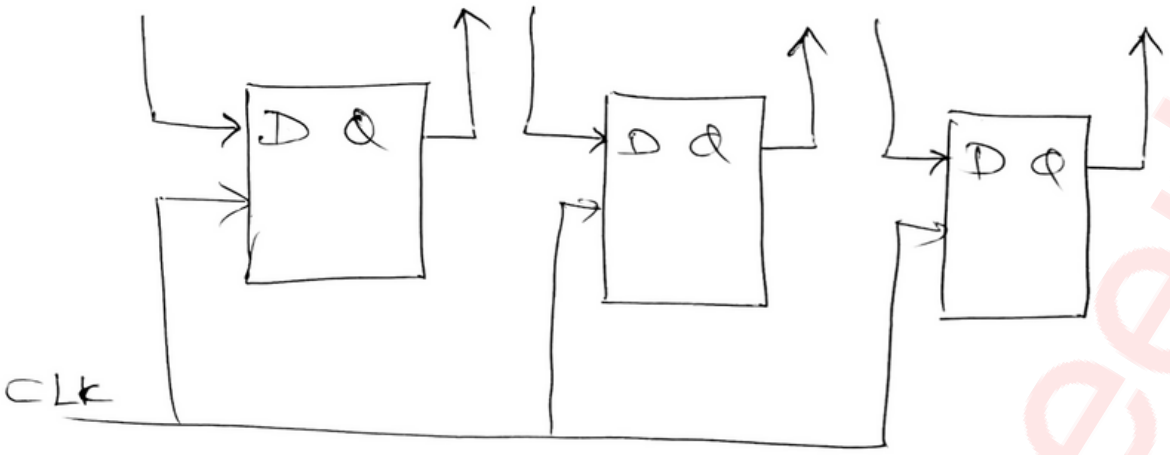
∴ Total = 'n'

③ PISO



I/p behaviour parallel → 1 clk
O/p behaviour serial → n-1 clk
Total = 'n'

④ PIPO



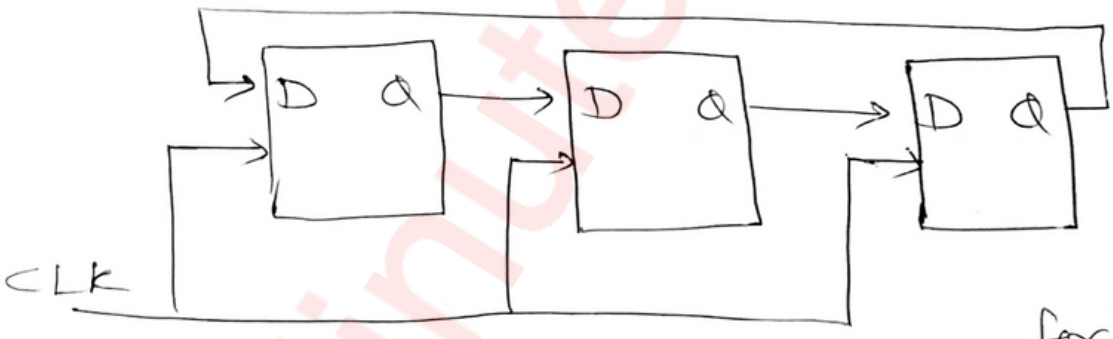
I/p & o/p both parallel behaviour

20 for write → 1 clk

— " — Read → 0 clk

Total 1 ✓

- Ring counter [circular shift Reg]



CLK	Q_2	Q_1	Q_0
0	1	0	0
1	0	1	0
2	0	0	1
0	1	0	0

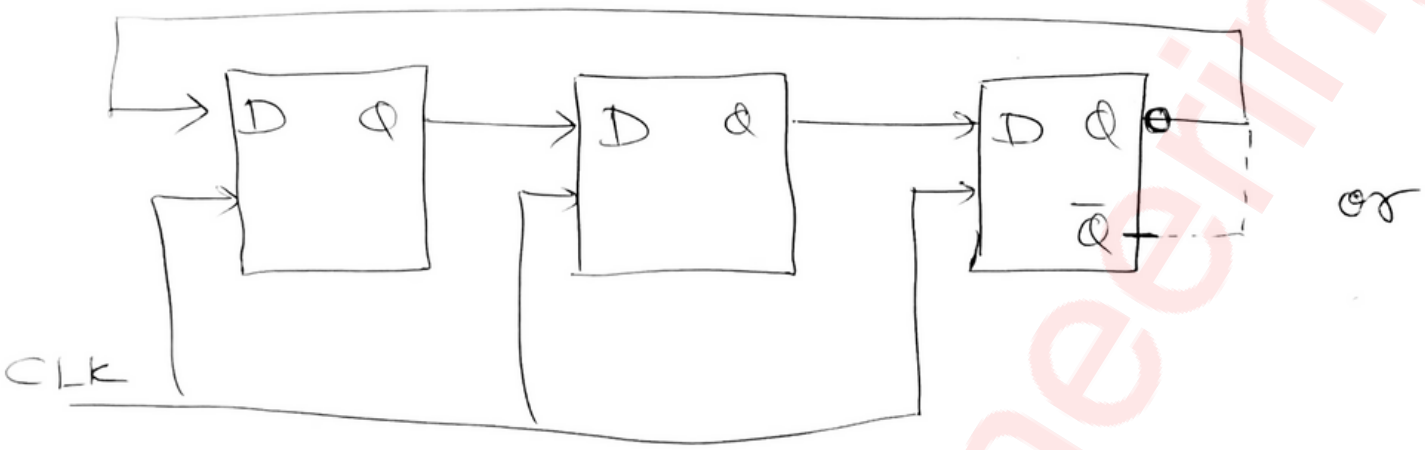
for 'n' no. of ff

→ 2^n (0 to $2^n - 1$)

→ But here we used only 'n' i.e 3 state

→ So $2^n - n$ unused state.

◦ Johnson Counter



CLK	Q_2	Q_1	Q_0
0	0	0	0
1	1	0	0
2	1	1	0
3	1	1	1
4	0	1	1
5	0	0	1
0	0	0	0

so here we get '6' states previously it was just '3' i.e 2(n)

So for n bits →
- Ideally (2^n count)
- Ring (n state count)
- Johnson (2n state count)