VII Combinational Circuits:

7.1 Combinational circuits: Half adder, full adder, Half subtractor, full subtractor:

In combinational circuits, current output is the function of current inputs.

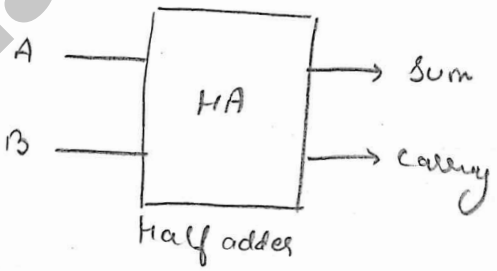
$$Y = F(I_1, I_2, I_3 \dots I_4)$$

Ex- Adders, subtractors, comparators, multiplexers etc.

- Adder
  - Half adder
  - Full adder
  - Parallel adder
  - BCD adder
- Subtractor
  - Half subtractor
  - Full subtractor
- Comparator
  - Single bit words
  - Multi bit words

Half adder: Adds two one-bit numbers and produce sum and carry.

Symbolic diagram:



Half adder

Truth table:

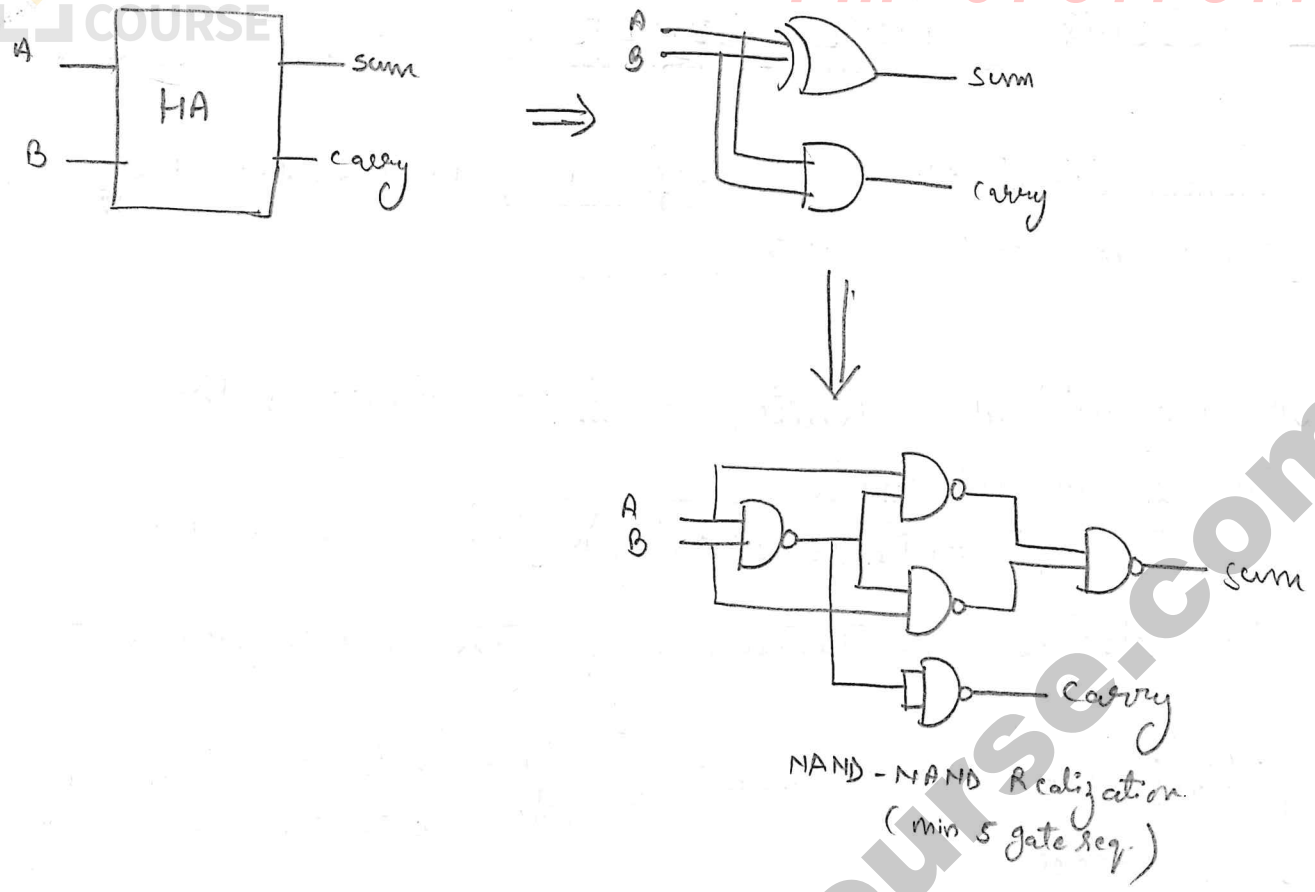
A	B	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Expression:

$$Sum = A'B + AB'$$

$$Carry = AB$$

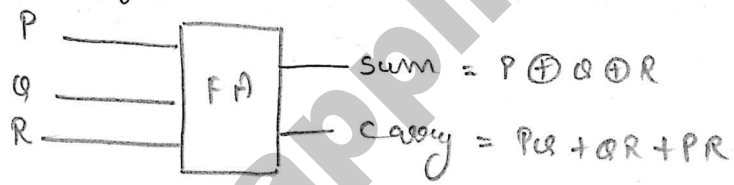
$$Sum = \underbrace{A'B + AB'}_{SOP} = \underbrace{(A+B)(A'+B')}_{POS}$$



NAND - NAND Realization
(min 5 gate req)

Full adder: It takes 3 external i/p and produces sum as well as carry.

Symbolic diagram:



Truth table:

	P	Q	R	sum	carry
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

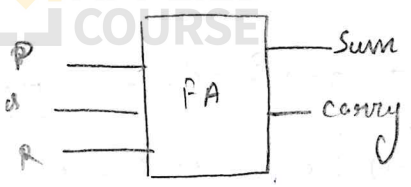
Expression:

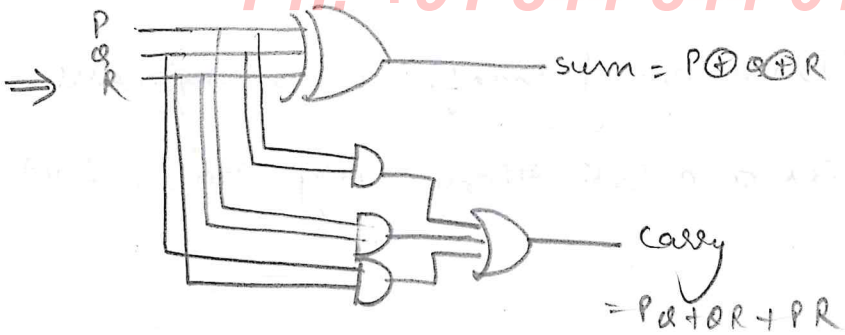
sum = $F(P,Q,R) = \Sigma(1,2,4,7)$

$= P'Q'R + P'QR' + PQ'R' + PQR = P \oplus Q \oplus R$

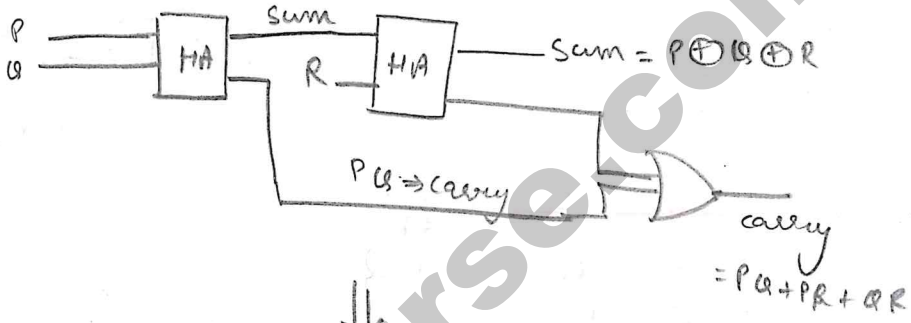
Carry = $F(P,Q,R) = \Sigma(3,5,6,7)$

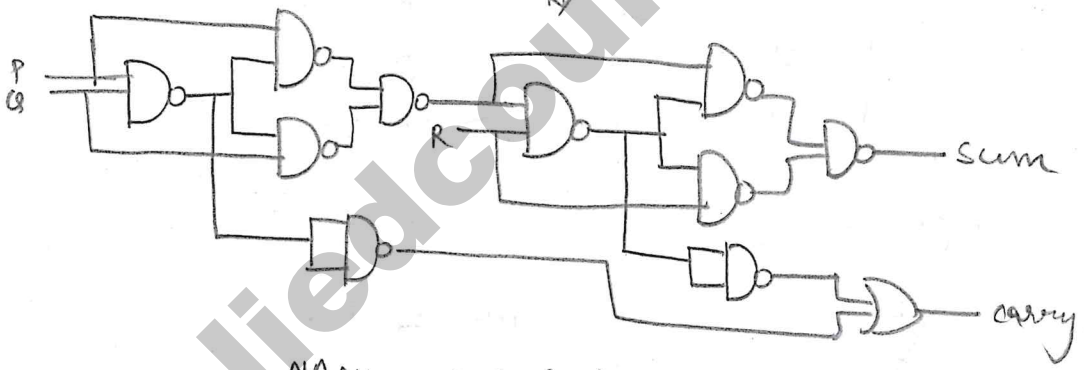
$= ~~P'QR + PQ'R' + PQR'~~ + PQ$





FA in terms of HA ⇒



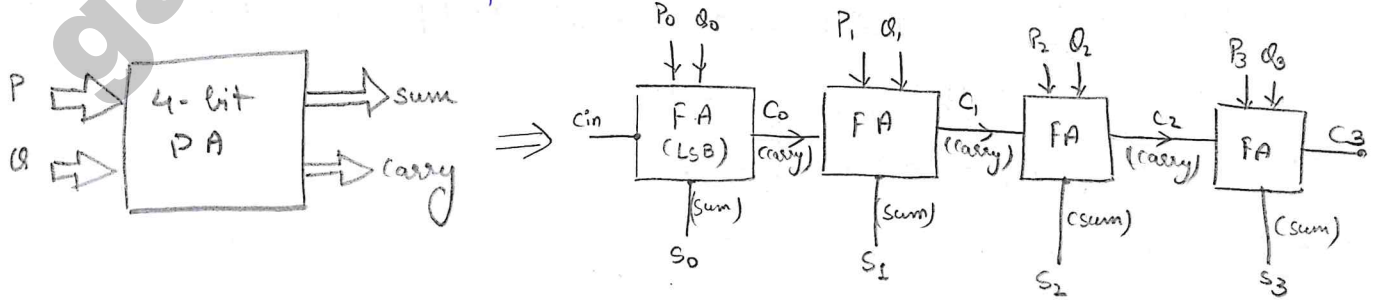


NAND - NAND Realization

(total 9 NAND gate are required)

Parallel adder: It is obtained by cascading full adders.

4 bit ~~full~~ parallel adder ⇒ 4 FA

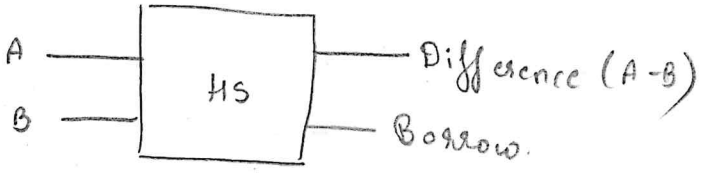


It is also called 4-bit ripple carry adder. It will have 4 FULL adder delay.

→ Total no. of half adders req. are 1 HA + 3·(2 HA) = 7 HA

→ For a n-bit ripple carry adder, 1 HA + 2·(n-1) HA = 2n-1 HA needed //

Half subtractor:

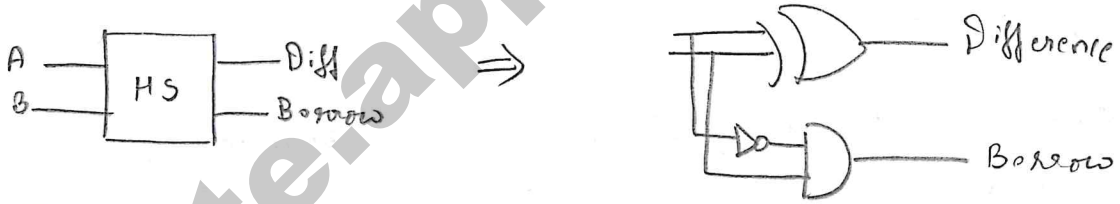


A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Expression -

Difference : $A'B + AB' = A \oplus B$

Borrow : $A'B$



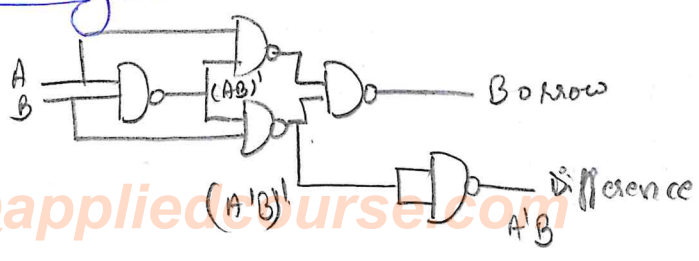
NOTE -

Relation between half adder and half subtractor is:

Sum = difference = $A \oplus B$

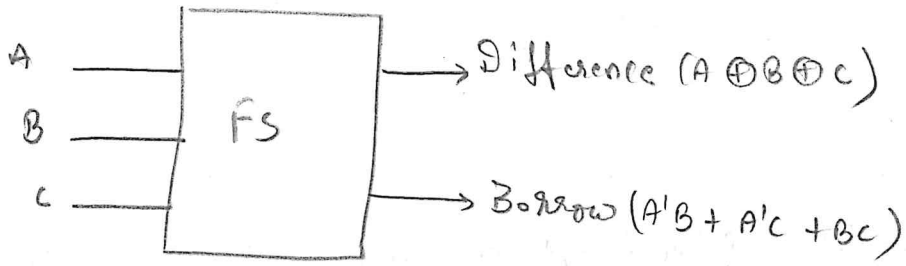
carry (A', B) = Borrow $(A, B) = A'B$

NAND-NAND realization -



∴ Total 5 NAND gates are required

Full Subtractor :



A	B	C	(A-B)-C Difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

(If A=0 and B=C≠0 Result Borrow
If A=1, and B=C=1 Result Borrow)

(A-B)
A=0 and B=1
(A-B)-C
A-B=0 and C=1

Difference = $\Sigma(1,2,4,7)$; Borrow = $\Sigma(1,2,3,7)$

= $A \oplus B \oplus C$ = $A'C + A'B + BC$

If first i/p of FA carry is complemented, it become borrow for FS

Borrow (A, B, C) = carry (A', B, C)

carry (A, B, C) = Borrow (A', B, C)

7.2 Combination Circuits: Parallel Adder / subtractor, BCD adder.

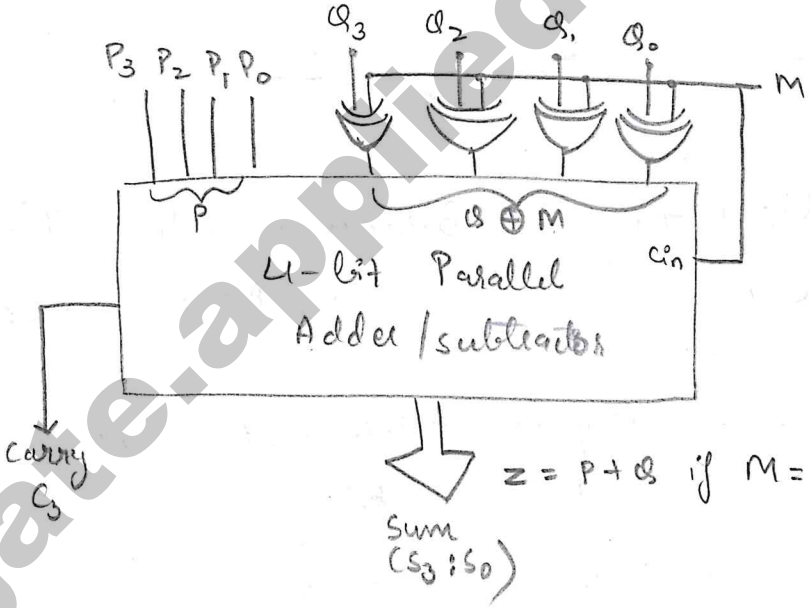
Parallel adder / subtractor: It is a controlled adder / subtractor

$P + Q$ = Addition = $P + Q + 0 = P + Q + M'$

$P - Q$ = Subtraction = $P + Q' + 1 = P + Q + M$

Control i/p

M	Operation
0	$P + Q$ = Addition
1	$P + Q' + 1$ = subtraction



$Q \oplus 0 = Q$

$Q \oplus 1 = Q'$

$Q + M = Q$ if $M = 0$

$Q + M = Q'$ if $M = 1$

$Z = P + Q$ if $M = 0$ and $Z = P - Q$ if $M = 1$

$P + \bar{Q} + 1 = P - Q$

BCD adder:

When 4 bit binary value > 9 then that 4-bit no. is invalid BCD no. Therefore, 6 combination of the 4 bits denote invalid BCD (1010, 1011, 1100, 1101, 1110, 1111)

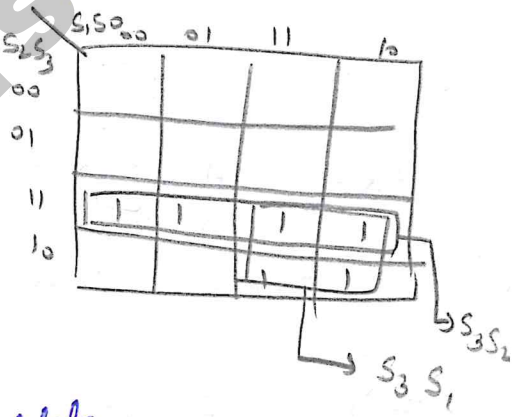
Invalid BCD result cases

(1) (A+B) sum > 9

(2) (A+B) result carry C_3 $\leftarrow f(S_3, S_2, S_1, S_0)$

Error indicator = $C_3 + \Sigma(10, 11, 12, 13, 14, 15)$

$= C_3 + S_3S_2 + S_3S_1$



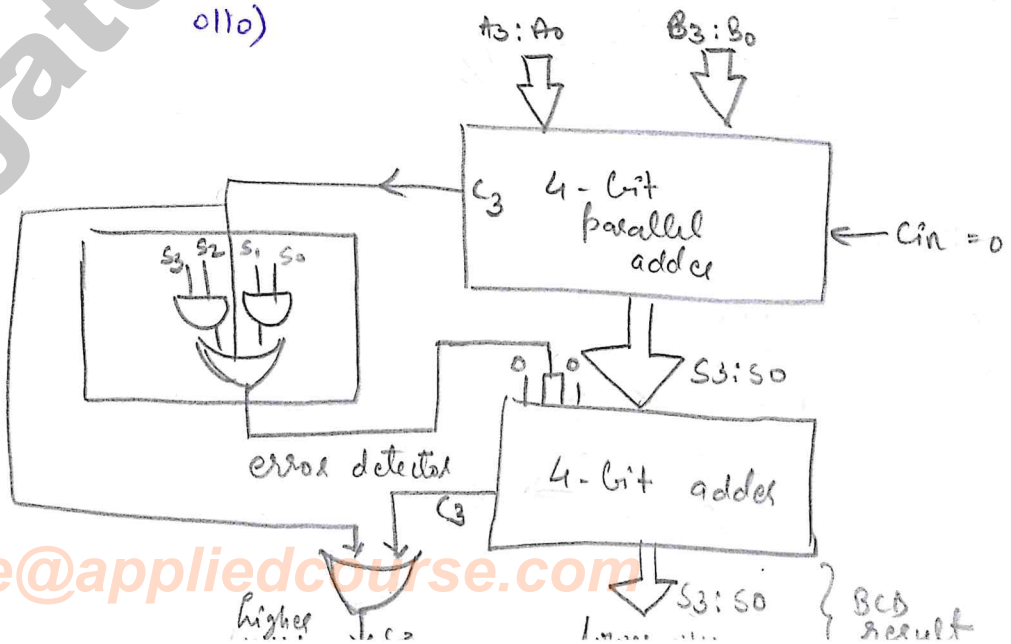
The implementation is having 2 stage addition

Stage 1: Add A, B

Stage 2: Add result with 0 or 6 based on error indicator.

$\Rightarrow$ (A+B) + 0 (0000) $\longrightarrow$ Error indicator = 0

(A+B) + 6 (0110) $\longrightarrow$ Error indicator = 1



Ex- A = 7
B = 6

We need to perform BCD addition.

⇒

A = 7 : 0 1 1 1
B = 6 : 0 1 1 0

1 1 0 1 > 9

0 1 1 0 +6

0 0 0 1 | 0 0 1 1

1 3

∴ 7 + 6 = 13

Ex- A = 8
B = 9

BCD addition?

⇒

A = 8 : 1 0 0 0
B = 9 : 1 0 0 1

0 0 0 1 > 9

0 1 1 0 +6

0 0 0 1 | 0 1 1 1

1 7 ⇒ 17

Ex- A = 6
B = 2

Bcd addition?

⇒

A = 6 : 0 1 1 0
B = 2 : 0 0 1 0

1 0 0 0 < 9

8 ⇒ 8

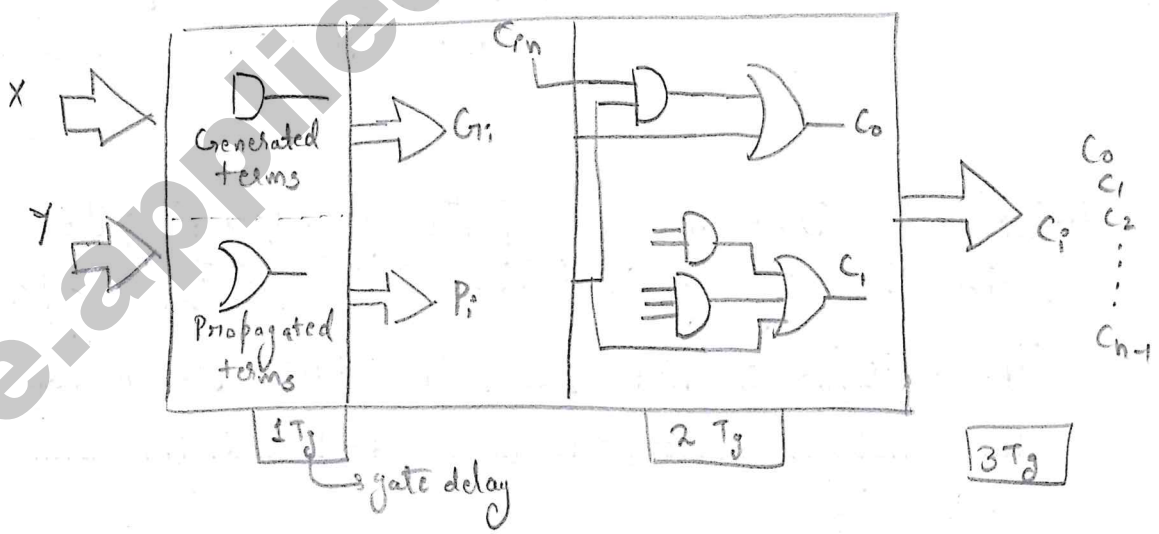
7.3 Carry lookahead adder:

→ It is aimed to provide constant time for addition.

→ It contains 2 stages
  (i) Carry generation stage
  (ii) Sum generation stage

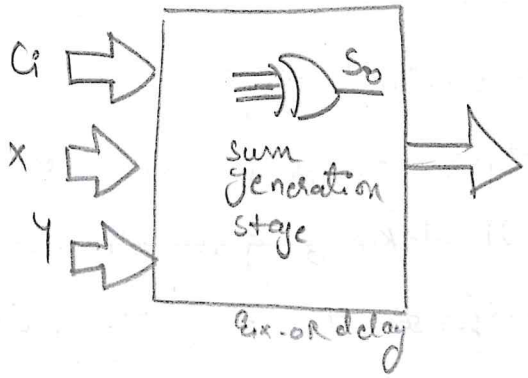
→ Carry generation stage generate all carries required for sum generation stage. It takes 3 gate delays. (1 gate delay is time required for response of AND/OR gate)

→ Carry generation stage is having 2 sub-stages
  (i) General/Propogated terms stage
  (ii) Carry generation stage



Carry generation stage

⇒ Sum generation stage: Once all carries are available, it provide sum generator $S_i = x_i \oplus y_i \oplus c_{i-1}$



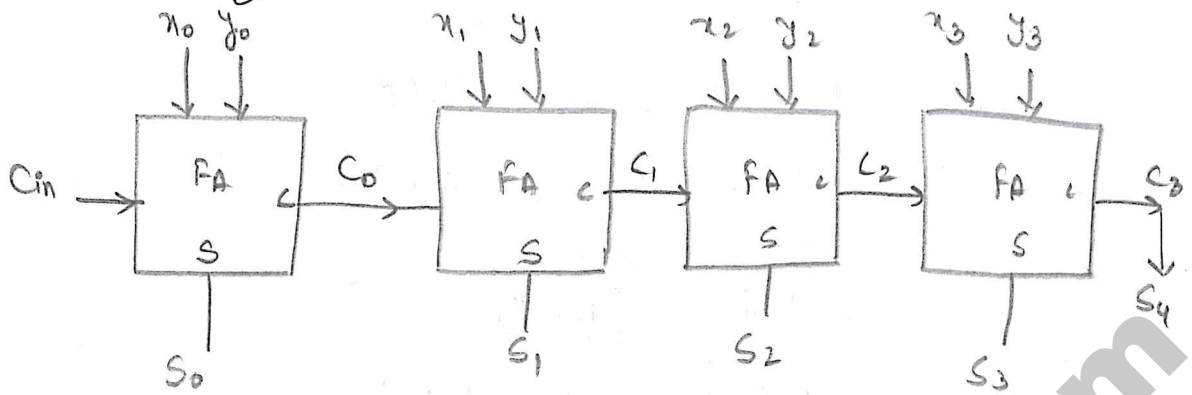
Features: It is expected to provide constant time for addition irrespective of size of operands X, Y

$T_{PD\,CLA} = 3T_g + T_{E\text{-}XOR}$

$= 4T_g / 5T_g / 6T_g$ (Based on Ex-OR behaviour)

limitation: The no. of inputs of internal gates increases with the size of addition. But getting single higher i/p AND/OR is not possible. Thus, fan-in capacity of gate imposes constraint on constant addition time achievement.

Why we need Carry lookahead adder?

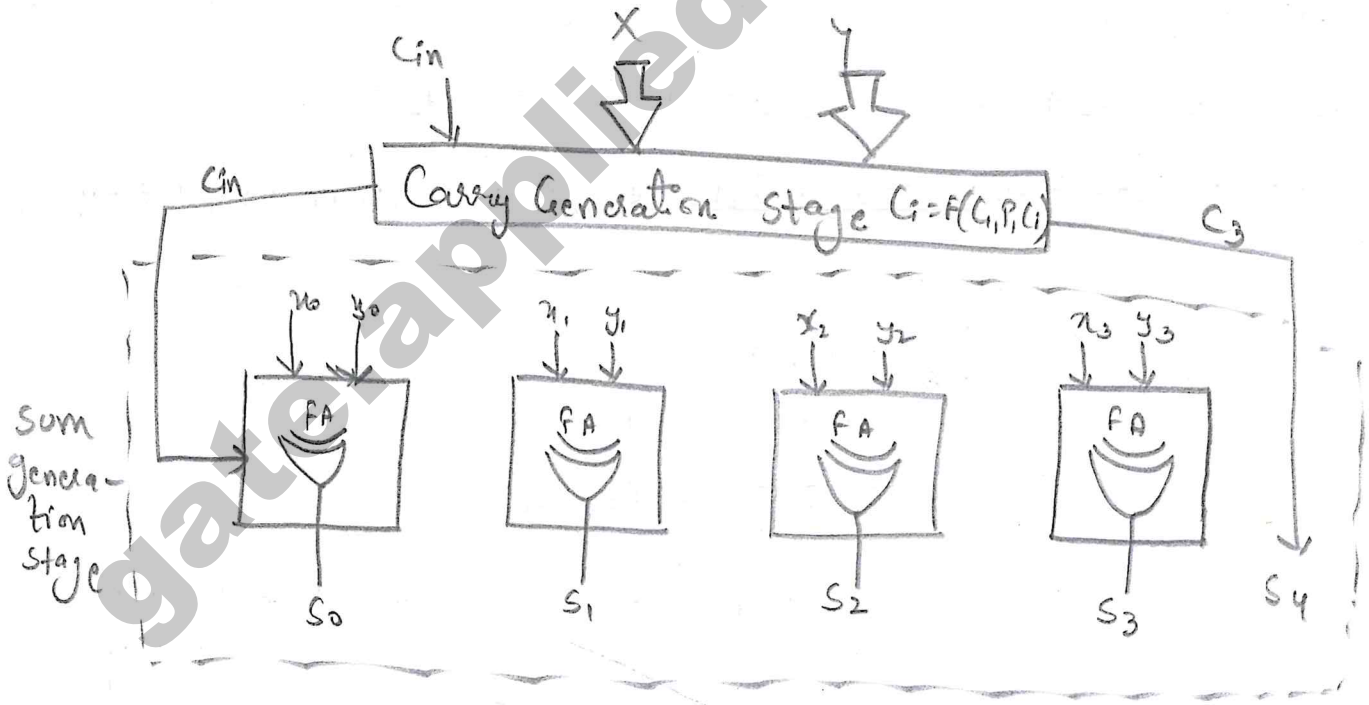


$$X + Y = S$$

$$X_3 X_2 X_1 X_0 + Y_3 Y_2 Y_1 Y_0 = S_4 S_3 S_2 S_1 S_0$$

$$T_{Pd\ RCA} = 4 * T_{FA}$$

T_{FA} → Full adder delay



$$T_{Pd_{CLA}} = T_{cg} + T_{sg}$$

The time taken by RCA is more than CLA as each FA is dependent on previous FA for carry, while this is not the case in CLA.

Expressions : $S_i = x_i \oplus y_i \oplus c_{i-1}$

$$C_i = x_i y_i + y_i C_{i-1} + x_i C_{i-1}$$
$$= x_i y_i + C_{i-1}(x_i + y_i)$$
$$= G_i + C_{i-1} P_i$$

G_i, generated term $= x_i y_i$

P_i, Propagated term $= x_i + y_i$

Carry expression interms $F(G, P, C_{in})$

$C_0 = G_0 + C_{in} P_0 \Rightarrow C_0 = F(G, P, C_{in}) \Rightarrow$  $\Rightarrow 2T_g$

$C_1 = G_1 + C_0 P_1 \Rightarrow G_1 + (G_0 + C_{in} P_0) P_1 \Rightarrow$  $\Rightarrow 2T_g$

$= G_1 + G_0 P_1 + C_{in} P_0 P_1 \Rightarrow C_1 = F(G, P, C_{in})$

Similarly,

$C_2 = G_2 + C_1 P_2 = G_2 + G_1 P_2 + G_0 P_1 P_2 + C_{in} P_2 P_1 P_0 \Rightarrow 2T_g$

$C_3 = G_3 + C_2 P_3 = G_3 + G_2 P_3 + G_1 P_2 P_3 + G_0 P_1 P_2 P_3 + C_{in} P_3 P_2 P_1 P_0 \Rightarrow 2T_g$

$\therefore T_{CLA} = T_{cg} + T_{sg}$
(carry) (sum)

$= 3T_g + T_{EX-OR}$

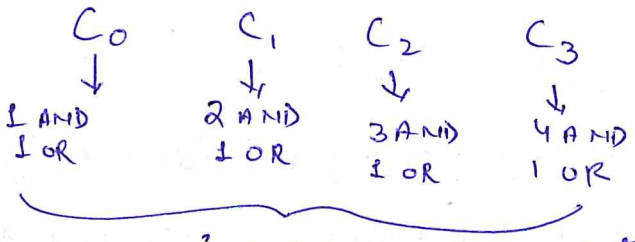
$= \max(\text{AND delay}, \text{OR delay}) + \text{AND delay} + \text{OR delay} + \text{EX-OR delay}$

if, $T_{AND} = T_{OR} = T_{EX-OR} = T_g$

$T_{CLA} = 4T_g$

(8) A carry generation stage of 4-bit CLA, having propagated term. How many AND, OR gates are required for the design of carry generation stage?

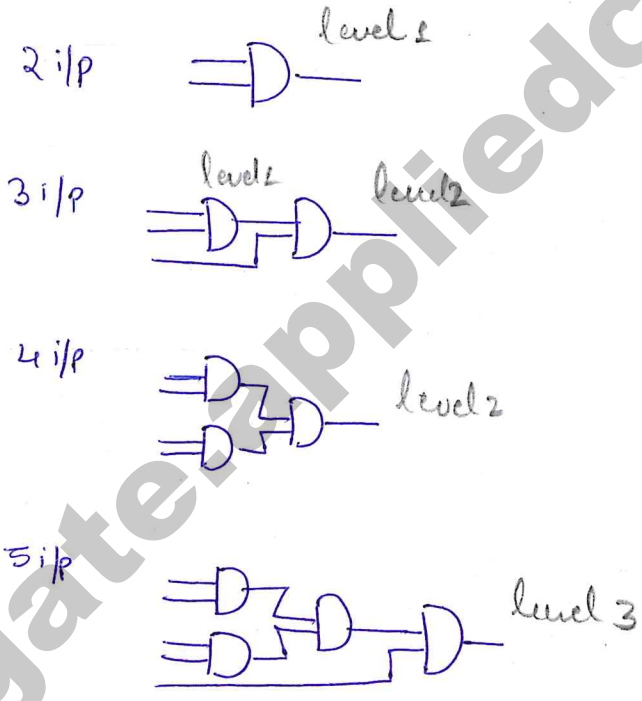
⇒



∴ in total 10 AND gates and 4-OR gates are required.

(Q) If carry generation stage has to use only 2 i/p gates then what will be the time increase order for addition?

⇒



like this, if we continue with n i/p AND gates, it will take $\log_2 n$ levels

∴ $O(\log_2 n)$ increase in T_{CLA} time.

(Q) Express C_1 in terms of operands? What are the no. of product terms?

$\Rightarrow$ $\because C_1 = g_1 + C_0 P_1$

$= g_1 + (g_0 + C_{in} P_0) P_1$

$= g_1 + g_0 P_1 + C_{in} P_0 P_1$

$= x_1 y_1 + x_0 y_0 (x_1 + y_1) + C_{in} (x_0 + y_0)(x_1 + y_1)$

$= x_1 y_1 + x_0 y_0 x_1 + x_0 y_0 y_1 + C_{in} x_0 x_1 + C_{in} x_0 y_1 + C_{in} y_0 x_1 + C_{in} y_0 y_1$

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

$\therefore$ # Product terms $= 7$ (if $C_{in} \neq 0$) $= 2^3 - 1$

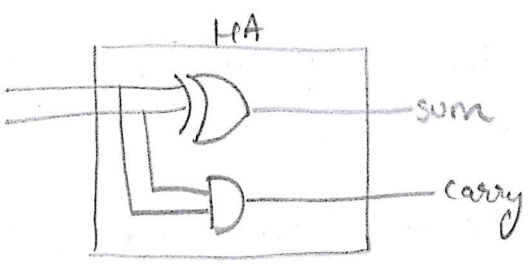
If $C_{in} = 0$, then # product terms $= 3 = 2^2 - 1$

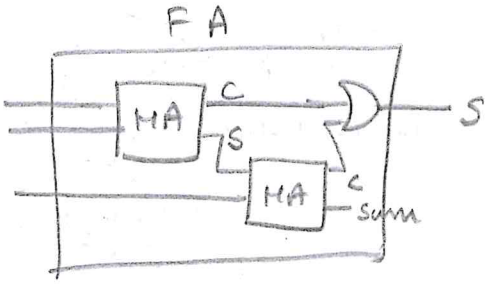
7.4 Previous Year gate questions: Adders, subtractor:

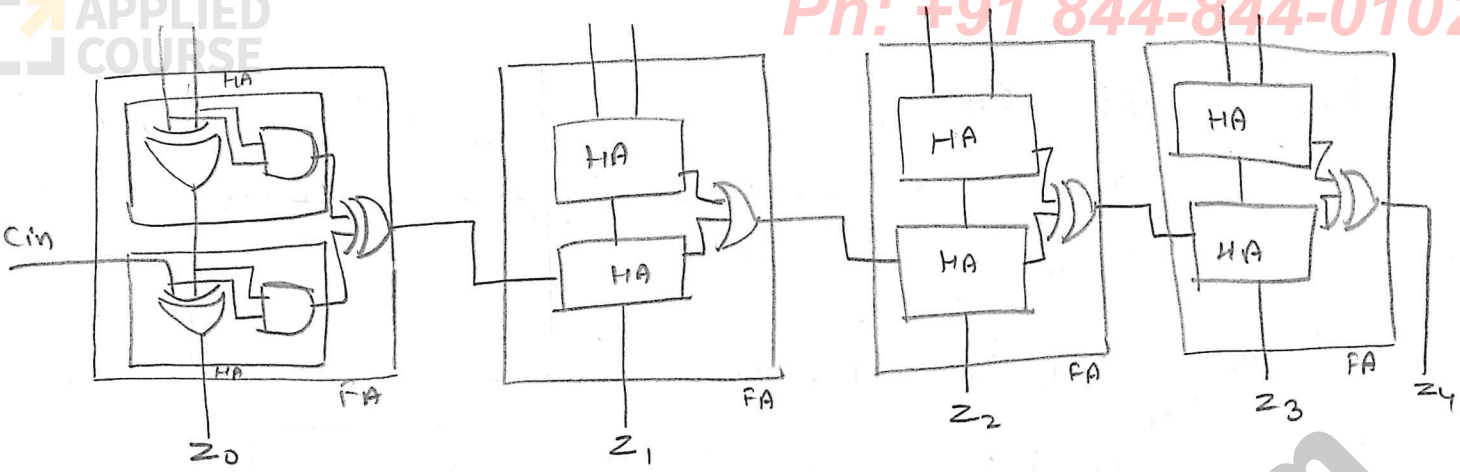
2015

(Q) Half adder is implemented with EX-OR and AND gates. A full adder with 2-half adder and one OR gate. If the propagation delay of EX-OR is twice to that of AND/OR gate. If Propagation delay of AND/OR gate is 1.2 ns. A bit 4-bit ripple carry adder is implemented with 4 full adders. Total propagation delay of 4-bit ripple carry adder is ____

$\Rightarrow$







Ex-OR delay = 2 × 1.2 MS
= 2.4 MS

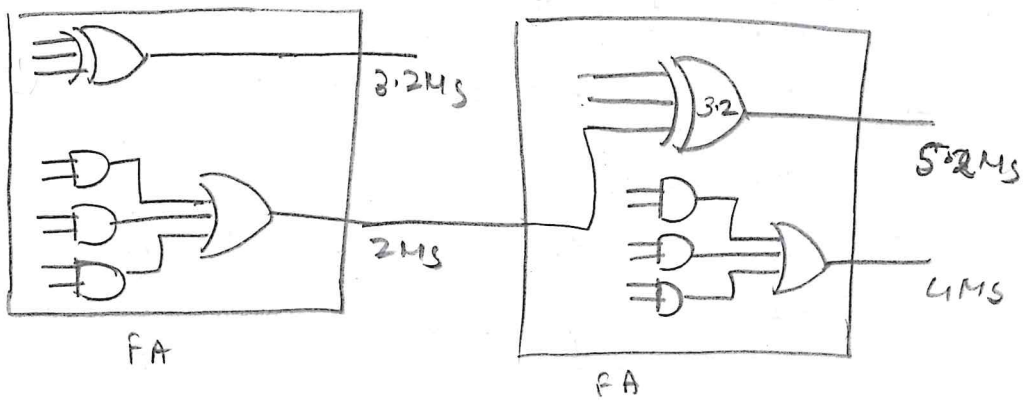
FA, sum o/p = 2 * Ex-OR delay
= 2 × 2.4
= 4.8 MS

Carry o/p = HA1 (1.2 MS), HA2 (3.6 MS) → 1.2 → 3.6 + 1.2 = 4.8 MS

∴ Total delay = 4 × 4.8 MS = 19.2 MS

(9) In the above question if FA is constructed with the help of Ex-OR and AND, OR gates. In this case the total ~~deg~~ delay will be ? (9f Ex-OR delay = 3.2 MS, AND/OR = 1 MS)

⇒



For a 4 bit RCA

$$\underbrace{3 \times 2}_{\text{Carry}}\text{ns} + \underbrace{3 \cdot 2\text{ns}}_{\text{Sum}} = 6 + 3 \cdot 2 = 9 \cdot 2\text{ns}$$

∴ In general it will take

$$(n-1)\text{ carry unit delay} + \text{Ex-OR delay}$$

NOTE-

In case, if Ex-OR is realized with 3 level gate (Not, and, OR gates) and consume 3g delays. Carry is constructed with 2-level gates (AND-OR) and consume 2g delays

Then for n-bit RCA, Propagation delay is

$$(n-1)\ 2\text{gate delay} + 3\text{ gate delays} = (2n+1)\text{ gate delays}$$

7.5 Comparator:

Comparator compares 2 words and produce 3-o/p based on their magnitude relation. Comparison will start from most significant bit.

A	B	A > B	A = B	A < B
A_0	B_0			
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

mutually exclusive o/p's

Single-bit Comparator

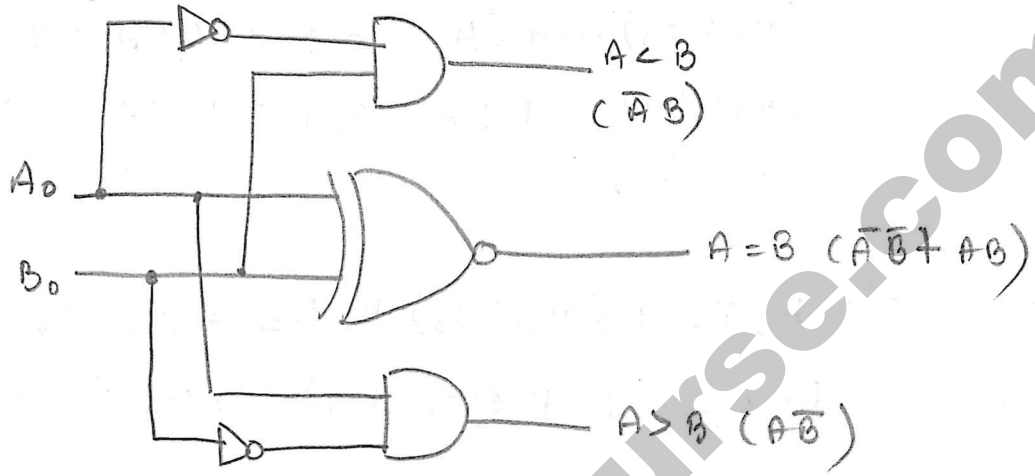
Expressions:

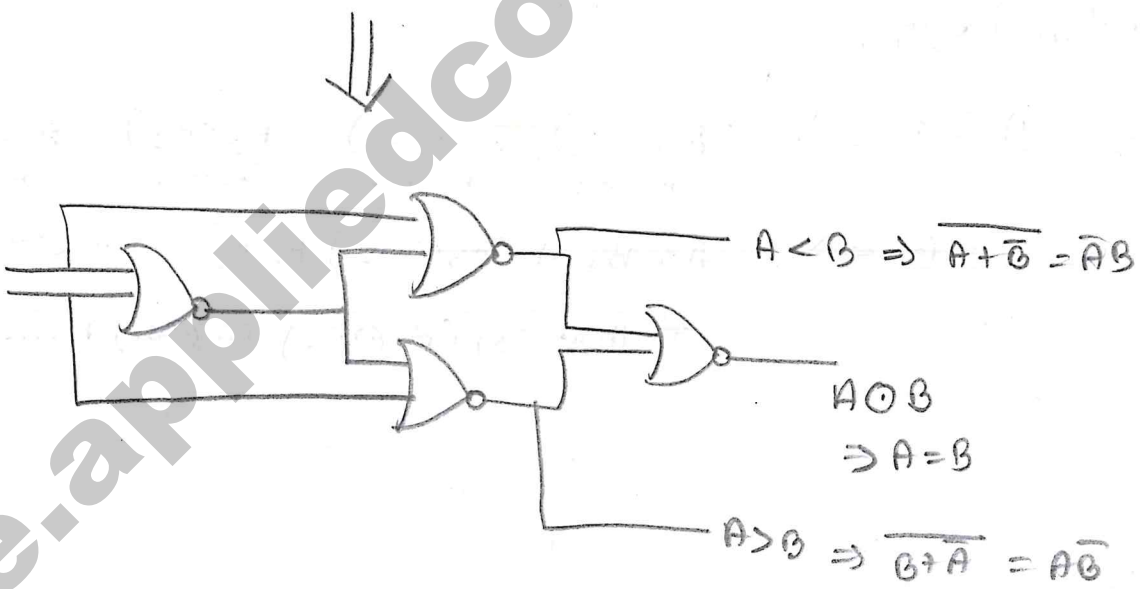
$A > B = A\bar{B}$

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

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

Circuit diagram:





NOR-NOR Realization

Multibit comparator: (4 bit word comparison)

$A = A_3\ A_2\ A_1\ A_0$

$B = B_3\ B_2\ B_1\ B_0$

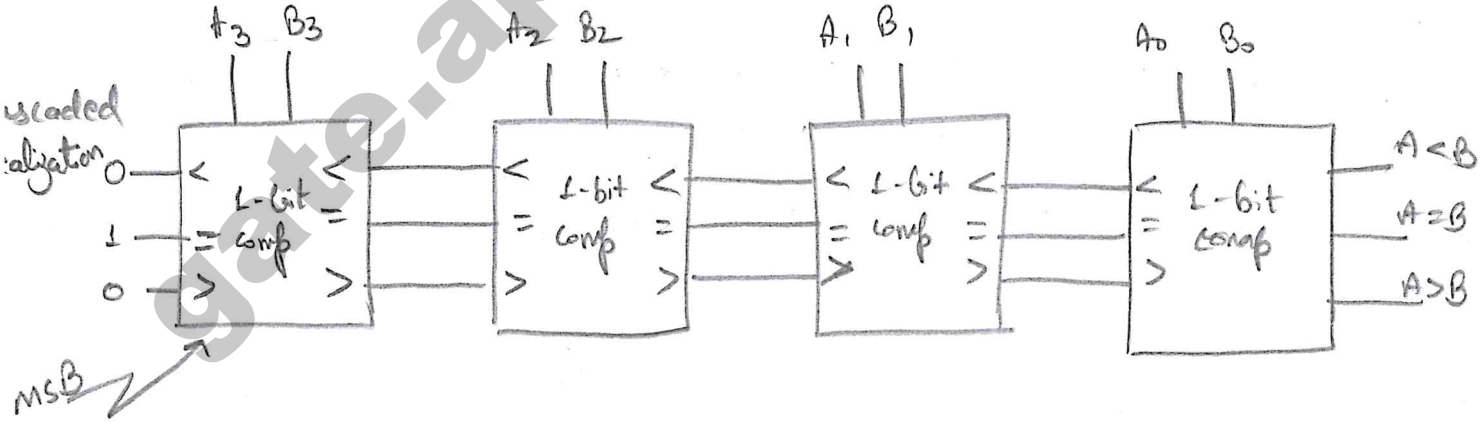
$A < B \iff A_3 < B_3$ or $(A_3 = B_3)$ and $(A_2 < B_2)$ or $(A_3 = B_3)$ and $(A_2 = B_2)$ and $(A_1 < B_1)$ or $(A_3 = B_3)$ and $(A_2 = B_2)$ and $(A_1 = B_1)$ and $(A_0 < B_0)$

$$\Rightarrow \overline{A_3} B_3 + (A_3 \odot B_3)\overline{A_2} B_2 + A_3 \odot B_3 (A_2 \odot B_2)\overline{A_1} B_1 + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)\overline{A_0} B_0$$

Similarly,

$$A = B \iff \underbrace{(A_3 \odot B_3)}_{(A_3 = B_3)} \underbrace{(A_2 \odot B_2)}_{(A_2 = B_2)} \underbrace{(A_1 \odot B_1)}_{(A_1 = B_1)} \underbrace{(A_0 \odot B_0)}_{(A_0 = B_0)} = \prod_{i=0}^{3} (A_i \odot B_i)$$

$$A > B \iff A_3 \overline{B_3} + (A_3 \odot B_3) A_2 \overline{B_2} + (A_3 \odot B_3)(A_2 \odot B_2) A_1 \overline{B_1} + (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1) A_0 \overline{B_0}$$



Cascaded realization of multibit comparator (4-bit i/p)

7.6 Introduction to MSI Circuits:

Data selector
↳ 1) Multiplexer (many to one)
Data distributor 2) De-multiplexer (one to many)
3) Encoder (many to some (compressors)) } MSI (Medium scale Integrated circuits)
4) Decoder (some to any decompressors)

NOTE –

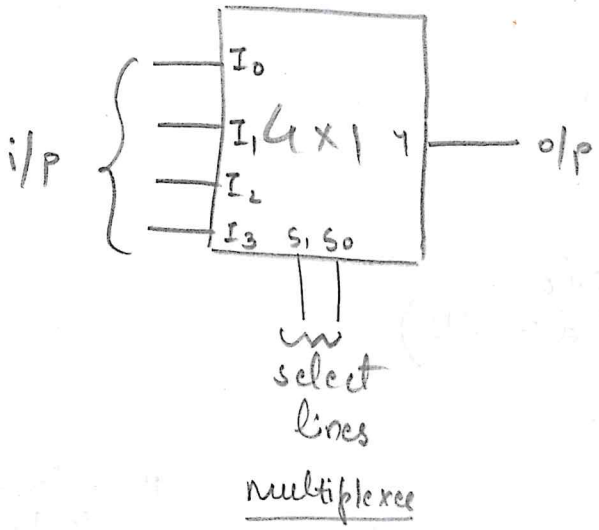
		# gates fabricated in IC
SSI	(small scale integrated circuits) →	< 10
MSI	(Medium " " ") →	> 10 and < 100
LSI	(Large " " ") →	> 100 and < 1000
VLSI	(Very large " " ") →	> 1000 and < 10k
VVLSI	(Very Very large " " ") →	> 10K

7.7 Functional realization using multiplexers:

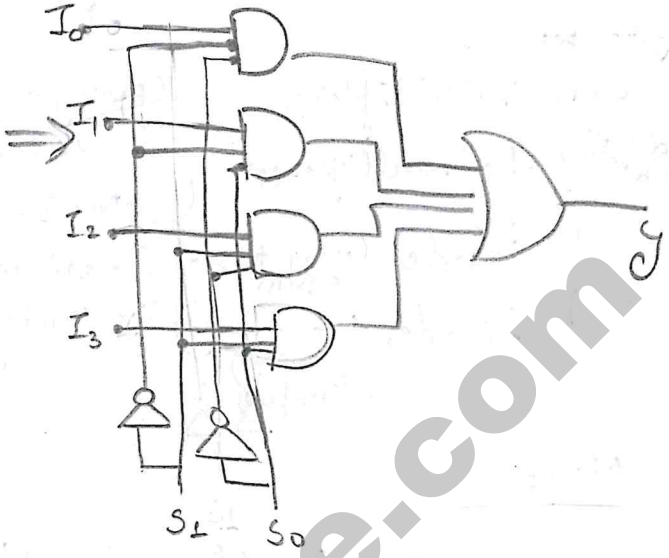
Multiplexer acts as a data selector. It selects one of many i/p's (using select lines) and pass to o/p. It provide functionally complete operations. It is characterized by i/p's, select lines and o/p.

Ex: 4x1 (4 by 1), 16x1 etc.

Symbolic representation :



multiplexer



Internal structure (AND-OR)

Characteristic equation:

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

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

NOTE -

For I i/p, the multiplexer will have selectlines, $S = \log_2 I$

OR $I = 2^S$ & No. of o/p's is always 1.

* Selection of MUX for realizing n-variable boolean function

functional independent realization

Case 1: n-select line multiplexer (2^n i/p)

Case 2: (n-1)-select line multiplexer (it requires 1-inverter)

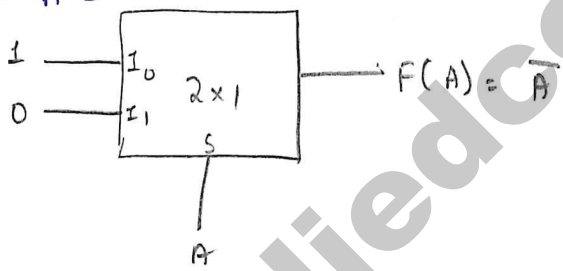
functional dependent realization.

Case 3: If <(n-1)-select line multiplexer, then we need multilevel realization.

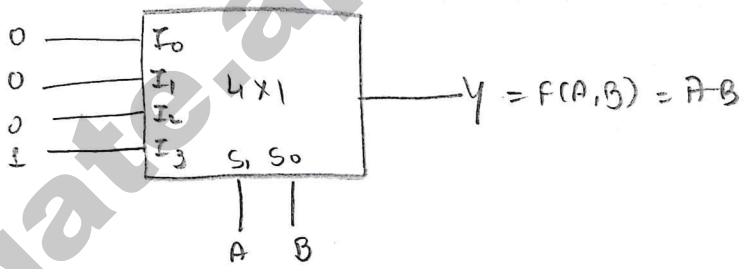
Case 4: If <(n-1)-select line multiplexer with external hardware. (functional dependent realization)

Function realization:

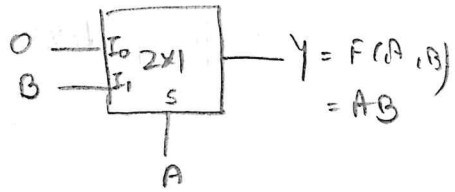
(1) NOT: $\Rightarrow \bar{A}\cdot 1 + A\cdot 0 = \bar{A}$



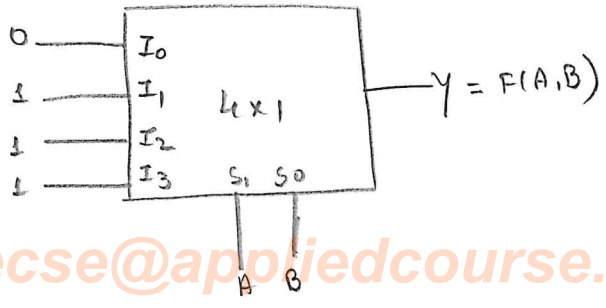
(2) AND: $\Rightarrow \bar{A}\bar{B}\cdot I_0 + \bar{A}B I_1 + A\bar{B} I_2 + AB I_3 \Rightarrow AB\cdot I_3 = AB\cdot 1 = AB$



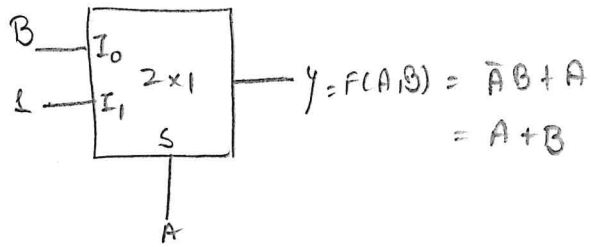
$\Rightarrow$



(3) OR: $\bar{A}B\cdot I_0 + \bar{A}B I_1 + A\bar{B} I_2 + AB I_3 = \bar{A}B + A\bar{B} + AB = A + B$



$\Rightarrow$



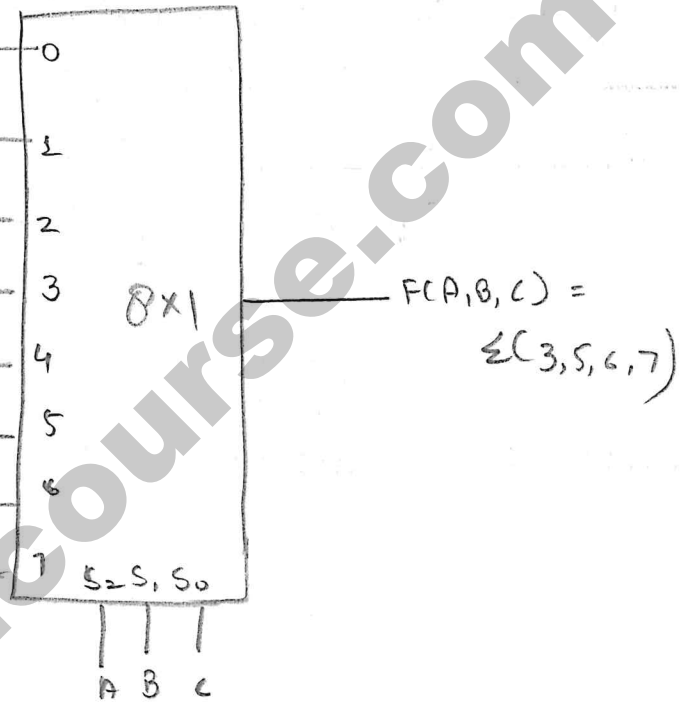
(9) Realize the function $F(A,B,C) = \Sigma(3,5,6,7)$ using 8x1, 4x1 and 2x1 mux.

$\Rightarrow$ $\because F(A,B,C) = \Sigma(3,5,6,7) = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

3x1 mux

The truth table will be

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



4x1 mux

Acc. to the expression: $\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

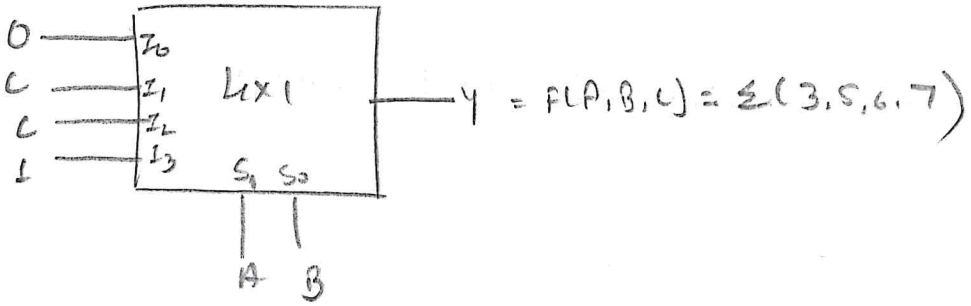
exp of 4x1 ~~selection~~: $\bar{A}\bar{B}I_0 + \bar{A}BI_1 + A\bar{B}I_2 + ABI_3$

$\bar{A}\bar{B}I_0$: No matching term $\therefore I_0 = 0$

$I_1 = C$

$I_2 = C$

$I_3 = \bar{C} + C = 1$



2x1 mux

The expression is : $\bar{A}BC + A\bar{B}C + ABC + AB\bar{C}$

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

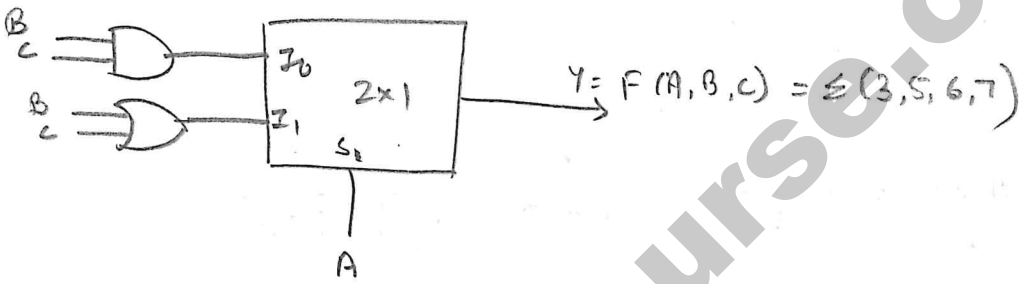
2x1 mux exp : $\bar{A}I_0 + AI_1$

$I_0 = BC$

$I_1 = \bar{B}C + BC + B\bar{C}$

$= \bar{B}C + B$

$= B + C$



7.8 Multiplexer Expansion :

It is a process of realizing higher capacity multiplexer with combination of lower capacity multiplexers arrange in the multiple levels.

Ex- 8x1 multiplexer is constructed with 7 2x1 multiplexer arranged in multiple levels

In general to realize Nx1 multiplexer with m x 1 multiplexers

then,

* No. of levels needed, $K = \lceil \log_m N \rceil$
* No. of multiplexers / level

level number :

$\lceil N/M \rceil^1 \quad \lceil N/M^2 \rceil^2 \quad \lceil N/M^3 \rceil^3 \quad \cdots \quad \lceil N/M^K \rceil^K$

* Total no. of multiplexers needed

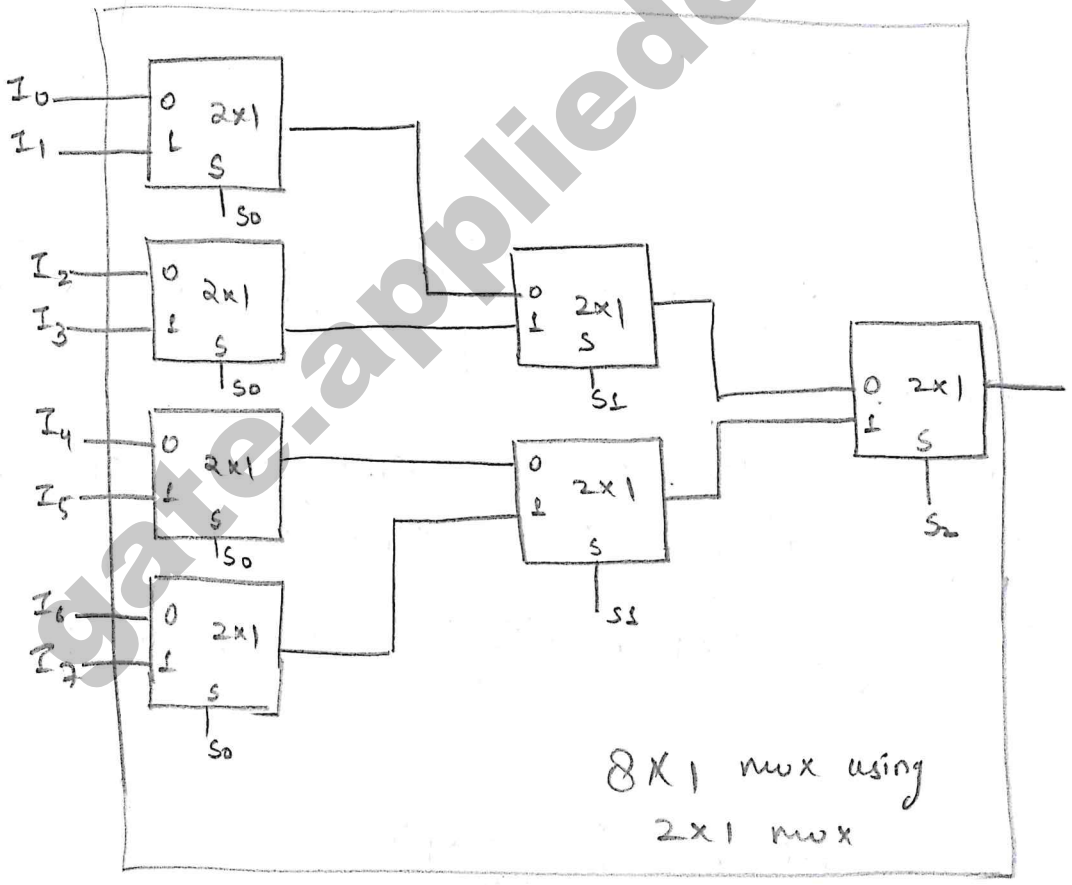
$$= \sum_{i=1}^{k} \lceil N/M^i \rceil$$

(8) # mux required to ~~Convert~~ implement 8×1 multiplexer using 2× multiplexer?

Here $N = 8$, $M = 2$, No. of levels $k = \lceil \log_2 8 \rceil = 3$

No. of 2×1 multiplexers each level

2×1 multiplexers per level	level i		
	1	2	3
	$\lceil 8/2 \rceil$	$\lceil 8/2^2 \rceil$	$\lceil 8/2^3 \rceil$
	4	2	1



8×1 mux using 2×1 mux

∴ Total multiplexers = 4 + 2 + 1 = 7 multiplexers arranged in 3-levels

(9) How many 4x1 mux needed to realize 128 x 1 mux

(a) 32
(b) 41
(c) 42
(d) 40

⇒ no. of levels = $\lceil \log_4 128 \rceil = \lceil 7/2 \rceil = 4$

now, in each level we need:

Level: 1 2 3 4

$\lceil \frac{128}{4} \rceil$ $\lceil \frac{128}{4^2} \rceil$ $\lceil \frac{128}{4^3} \rceil$ $\lceil \frac{128}{4^4} \rceil$

= 32 = 8 = 2 = 1

↳ we can replace 4x1 mux in level 4 with 2x1 mux, as 4x1 mux is not fully utilised

∴ Total mux = 32 + 8 + 2 + 1
= 43

(9) What is the optimal realization of above mux?

(a) 41, 4x1 mux (b) 40, 4x1 mux and one 2x1 mux
(c) 42, 4x1 mux and one 2x1 mux (d) 43, 4x1 mux and one 2x1 mux

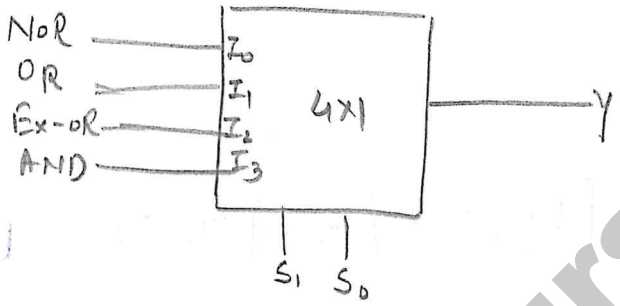
⇒ Since at last level we can not utilising 4x1 mux completely
∴ we can replace it with 2x1 mux. Thus, (c) is the correct option.

Multiplexer Applications

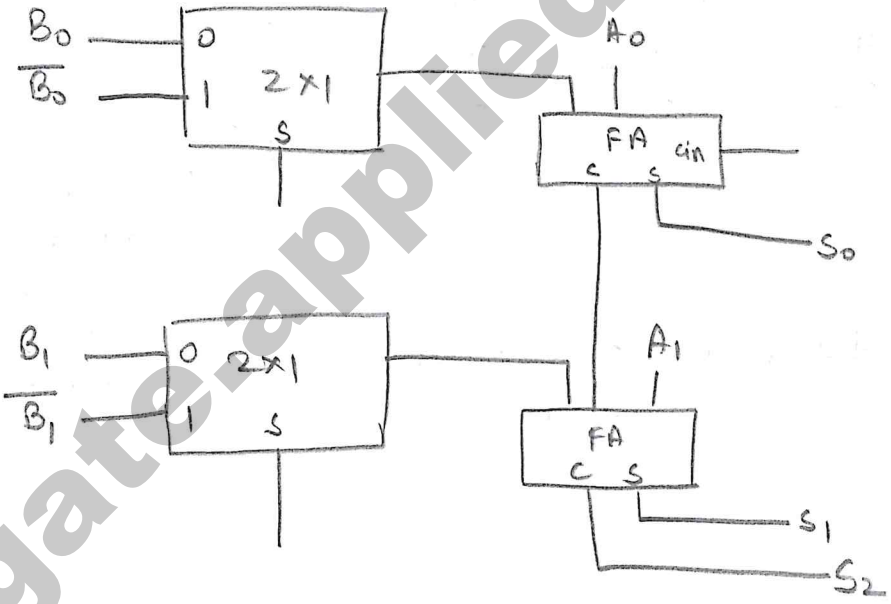
(1) Function selector:

Mux can be used as function selector along with the data selector.

Ex 1 –



Ex 2 –



2-bit nos

$A = A_1A_0$

$B = B_1B_0$

$A - B = A + \overline{B} + 1$

$= A$ + 2's comp of B

$= A - B$

S	Function
0	A + B = Adder
1	$A + \overline{B} + 1 = A - B$ = subtractor

here S is used to select function (addition / subtraction)

(2) Used for Realizing Sequential circuits:

Multiplexers along with D-FF, is used to realize Sequential circuits.

Here D-FF ($Q_n = Q$) i/p's are derived from multiplexers.

No. of multiplexers is equal to no. of varicbles in each state.

ex: Design Mod-4 binary up counter with state variable A and B.

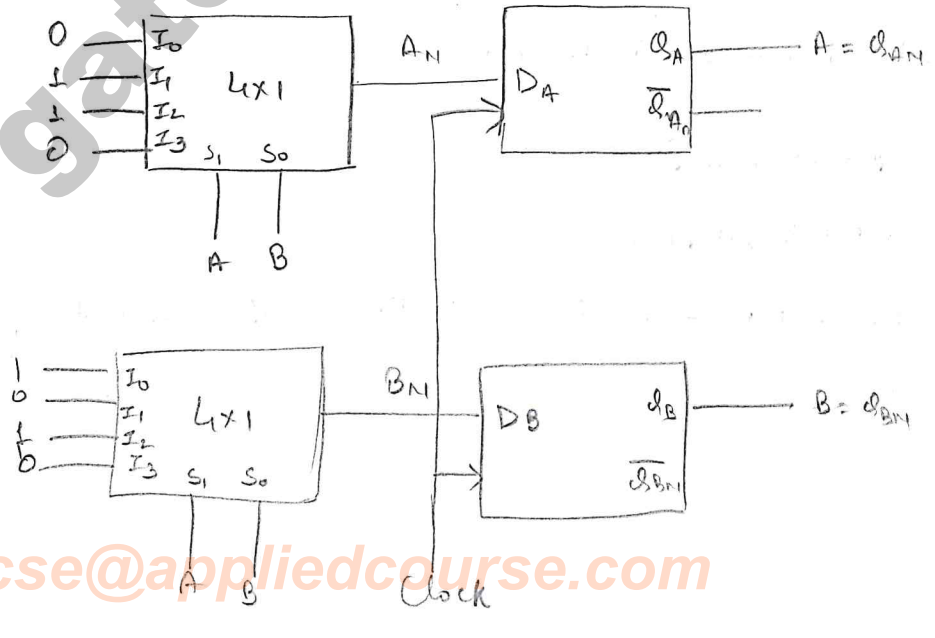
⇒ State table:

Present state		next state	
A	B	A_N	B_N
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

State exp:

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

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



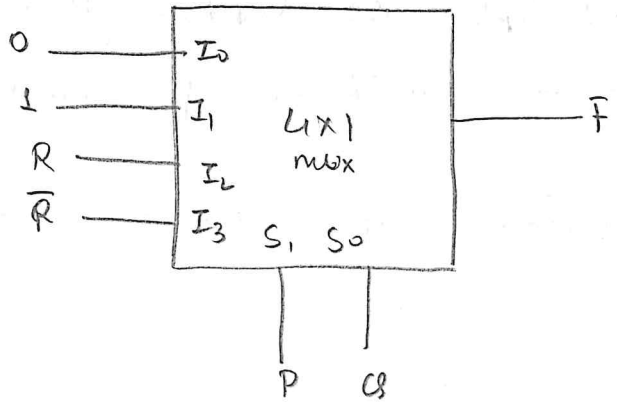
Note:

Each next state become present state for next clock.

7.9 Gate questions on Multiplexer - 1 :

2014

(Q) Minimal SOP for F = ?



(a) $\bar{P}Q + \bar{Q}R + PQ\bar{R}$

(b) $\bar{P}Q + \bar{P}Q\bar{R} + PQ\bar{R} + P\bar{Q}R$

(c) $\bar{P}QR + \bar{P}Q\bar{R} + Q\bar{R} + P\bar{Q}R$

(d) $PQ\bar{R}$

⇒ The characteristic equation of 4x1 mux ⇒

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

on substituting value of I_0, I_1, I_2 and I_3, we get;

⇒ $\bar{P}\bar{Q}\cdot 0 + \bar{P}Q\cdot 1 + P\bar{Q}\cdot R + PQ\bar{R}$

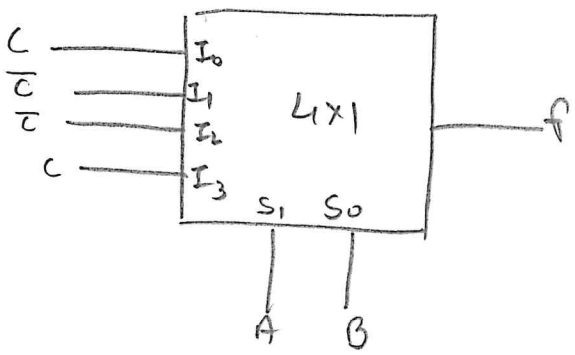
⇒ $0 + \bar{P}Q + P\bar{Q}R + PQ\bar{R}$

⇒ $Q(\bar{P} + P\bar{R}) + P\bar{Q}R$

⇒ $Q(\bar{P} + \bar{R}) + P\bar{Q}R$

$= \bar{P}Q + Q\bar{R} + P\bar{Q}R$ ⇒ (a) is the correct option.

(Q) F implements ?



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

(b) $A + B + C$

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

(d) $AB + BC + CA$

⇒ The characteristic eqn:

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

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

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

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

⇒ OR

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

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

if $\bar{B}C + B\bar{C} = X$

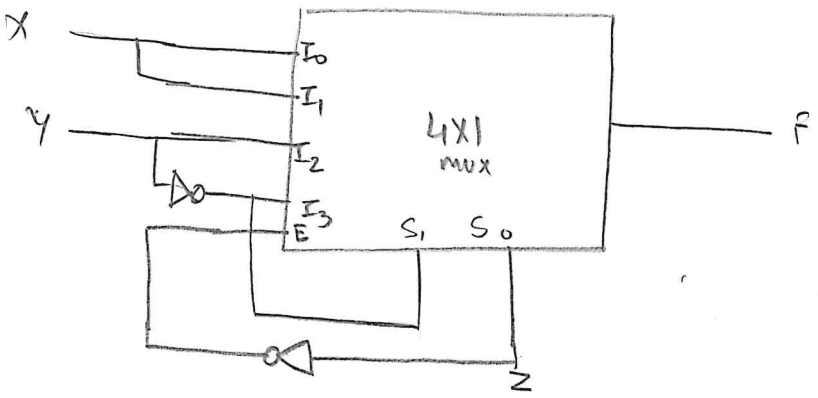
$B \oplus C$

⇒ $\bar{A}X + A\bar{X}$

⇒ $A \oplus X$ ⇒ $A \oplus B \oplus C$ ⇒ $A \odot B \odot C$

∴ (c) is the correct option.

(9) What does F denote?



(a) $xy\bar{z}$

(b) $xy+z$

(c) $x+y$

(d) None of these

$\Rightarrow$ This multiplexer will work only when Enable, E is high (1) and E=1 only when $z=0$

$\therefore$ we can eliminate (b) and (c) option.

The exp is:

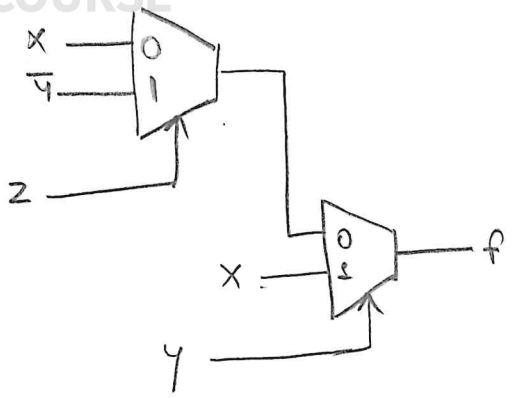
$\Rightarrow E(\bar{S}_1\bar{S}_0I_0 + \bar{S}_1S_0I_1 + S_1\bar{S}_0I_2 + S_1S_0I_3)$

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

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

$\Rightarrow xy\bar{z}$ //

(Q)



Which one correctly represent f ?

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

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

(c) $xz + xy + \bar{y}\bar{z}$

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

$\Rightarrow$ eqn of 1st mux $\Rightarrow$ $\bar{z}x + z\bar{y}$

eqn of 2nd mux $\Rightarrow$ $\bar{y}(\bar{z}x + z\bar{y}) + yz$

$\Rightarrow \bar{y}\bar{z}x + \bar{y}z + yx$

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

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

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

$\Rightarrow \bar{y}x + \bar{y}z + yx$ // $\longrightarrow$ 1 possible answer

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

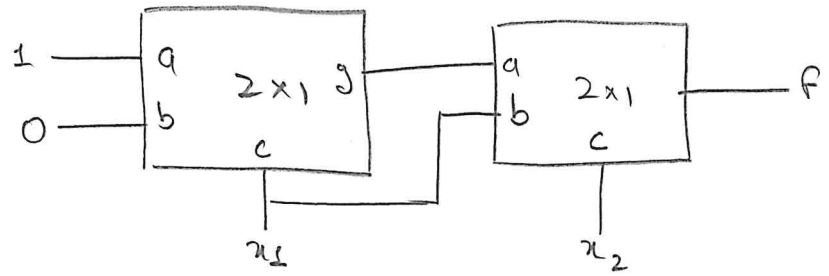
$\Rightarrow x(y + \bar{z}) + \bar{y}z$

$\Rightarrow xy + x\bar{z} + \bar{y}z$ $\longrightarrow$ 2nd possible answer.

$\therefore$ (a) is correct option

2001

(9) What is f = ?



(a) $x_1' + x_2$

(b) $x_1 x_2 + \overline{x_1}\,\overline{x_2}$

(c) $x_1' x_2 + x_1 x_2'$

(d) $x_1 + x_2'$

$\Rightarrow$

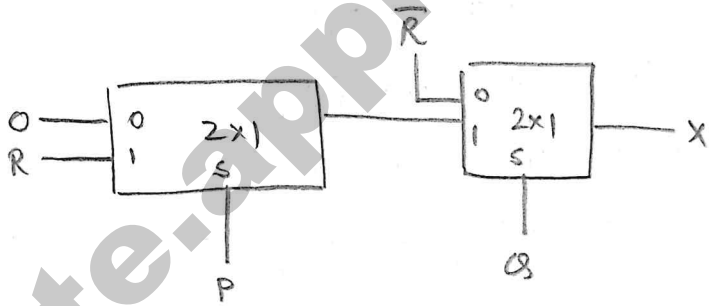
here exp of 1st mux will be $\overline{x_1}\cdot a + x_1 \cdot b = \overline{x_1}\cdot 1 + x_1 \cdot 0 = \overline{x_1}$

2nd exp will be: $\overline{x_2}\,a + x_2 b \Rightarrow \overline{x_2}\,a + x_2 x_1 \Rightarrow \overline{x_2}\,\overline{x_1} + x_2 x_1$

hence (b) is the correct option.

2016

(8)



Minimal SOP at X = ?

(a) $\overline{P}\,\overline{Q} + PQR$

(b) $PQ + \overline{P}\,\overline{Q}\,R$

(c) $\overline{P}Q + QR$

(d) $\overline{Q}\,\overline{R} + PQR$

$\Rightarrow$ exp at 1st mux $\Rightarrow$ $\overline{P}\cdot 0 + PR = PR$

exp at 2nd mux $\Rightarrow$ $\overline{Q}\cdot\overline{R} + Q\cdot P\overline{R}$

$\therefore X = \overline{Q}\,\overline{R} + PQR$ $\therefore$ (d) is the correct option.

7.10 Gate questions on multiplexer-2:

(Q) 2007
Let only one multiplexer and only one inverter is allowed to use to realize n-variable boolean function. What is the minimum size of the multiplexer is needed.

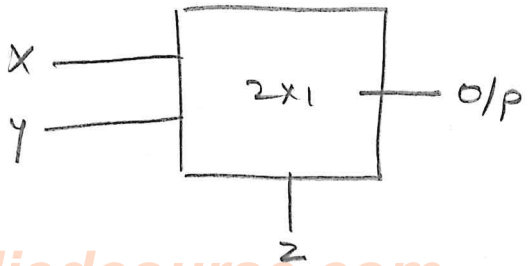
(a) 2^n line to 1 line
(b) 2^{n-1} line to 1 line
(c) 2^{n+1} line to 1 line
(d) 2^{n-2} line to 1 line

⇒ We can eliminate option (a) and (c) as they are not minimum.

If (n-1) select lines are given than we can realize n-variable boolean function by using an inverter

∴ (b) is the correct option.

(Q) 2004
Let X, Y are data i/p's of multiplexer and Z as control i/p, if Z=0 o/p get X, else o/p get Y. What are the connections for X, Y, Z, if f = T + R, a 2-variable function is to be realized.



	X	Y	Z
(a)	R	1	T
(b)	T	R	T
(c)	T	R	0
(d)	R	0	T

⇒ here, the exp will be: $\bar{z}x + zy \Rightarrow \bar{T}R + T$

$\Rightarrow R + T$

for (a) $x = R, y = 1, z = T$

we will get exp $\Rightarrow \bar{T} \cdot R + T \cdot 1 = T + R$

for (b) $x = T, y = R, z = T$

we will get exp $\Rightarrow \bar{T} \cdot T + TR = TR$

for (c) $x = T, y = R, z = 0$

We will get exp $\Rightarrow \bar{0} \cdot T + 0R = T + 0 = T$

for (d) $x = R, y = 0, z = T$

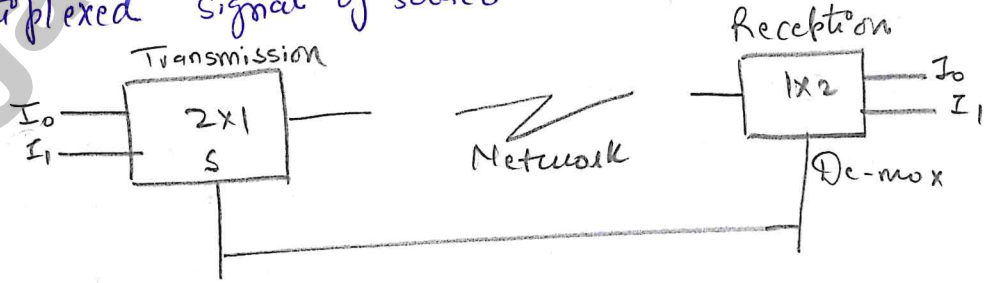
We will get exp $\Rightarrow \bar{T} \cdot R + T \cdot 0 = \bar{T}R$

∴ (a) is the correct option.

7.11 Demultiplexer and Decoder:

Demultiplexer: (one to many)

Demultiplexer is used at destination to extract i/p from multiplexed signal of source.



De-mux performs one-to-many mapping. It is termed as data distributor.

ex: 1×4 (one i/p, 4 o/p's and 2 select lines)

Select lines		o/p			
S_1	S_0	O_0	O_1	O_2	O_3
0	0	I	0	0	0
0	1	0	I	0	0
1	0	0	0	I	0
1	1	0	0	0	I

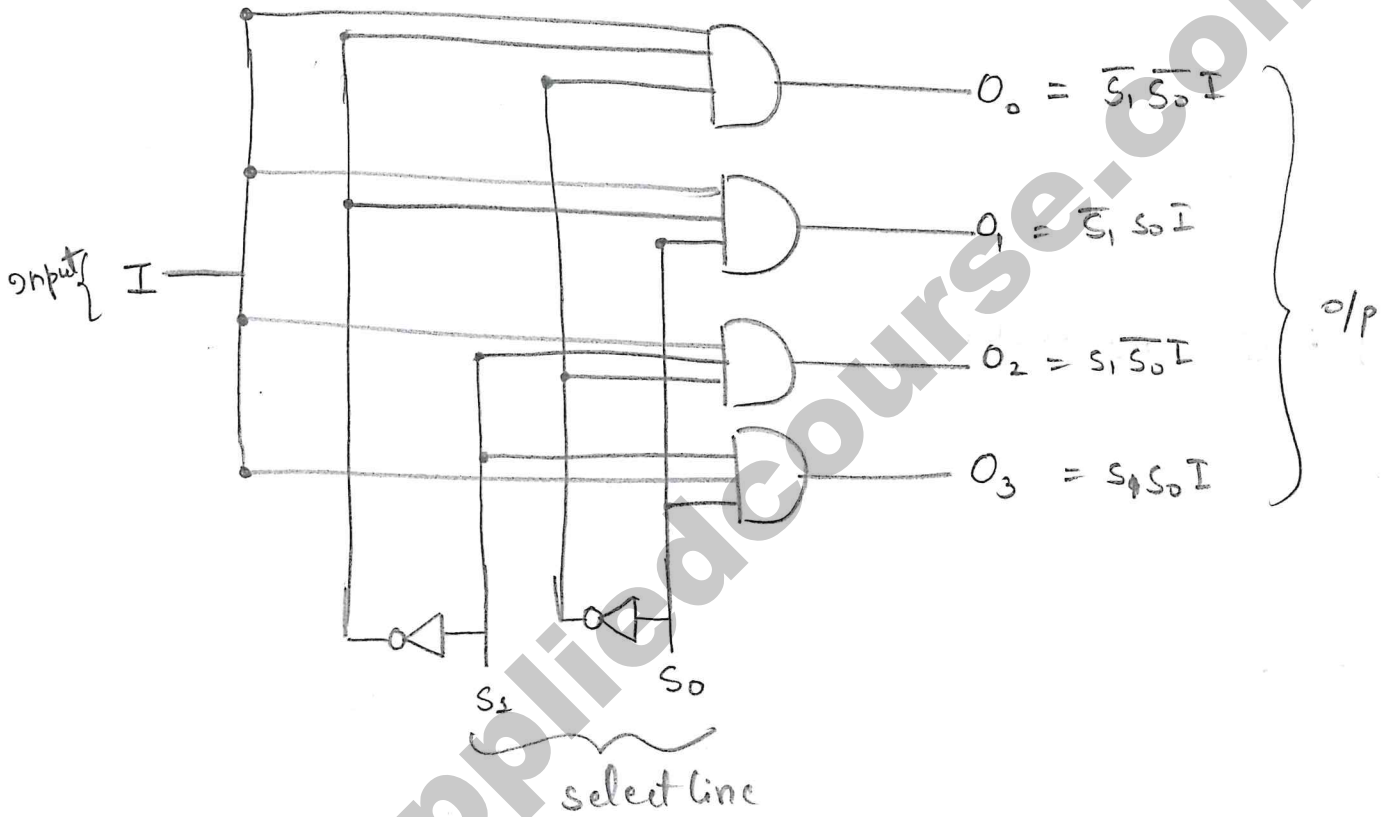
Expressions for o/p:

$O_0 = \bar{S_1}\,\bar{S_0}\,I$

$O_1 = \bar{S_1}\,S_0\,I$

$O_2 = S_1\,\bar{S_0}\,I$

$O_3 = S_1 S_0 I$



Decoder from Demultiplexer:

Inactivate i/p of De-mux (I = 1) and convert select lines as i/p's. Example : 74LS138 De-mux

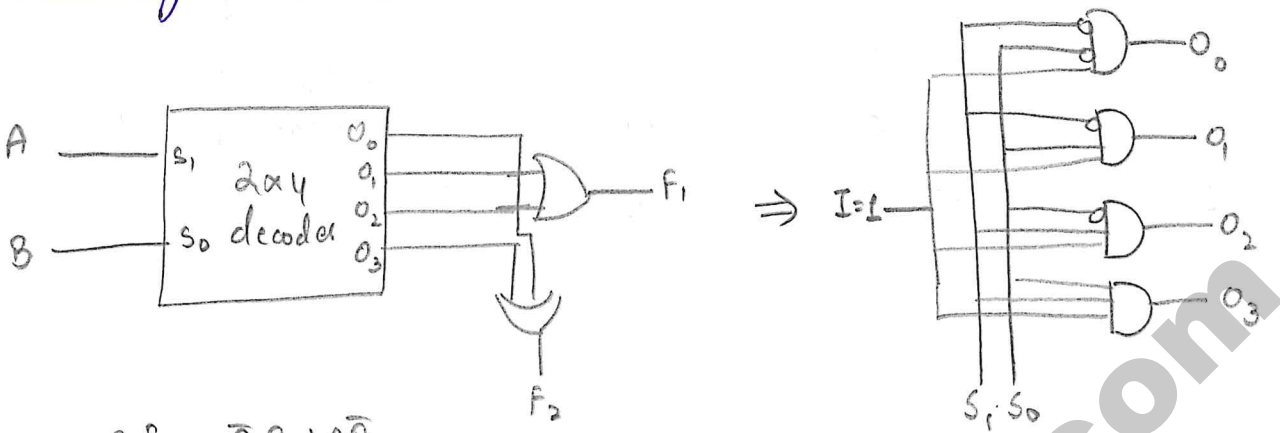
Types of Decoder

- Selected o/p is high ⇒ Active high decoders.
- Selected o/p is low ⇒ Active low decoders.

Application: Realizing boolean functions, code conversions, memory selection (chip selection)

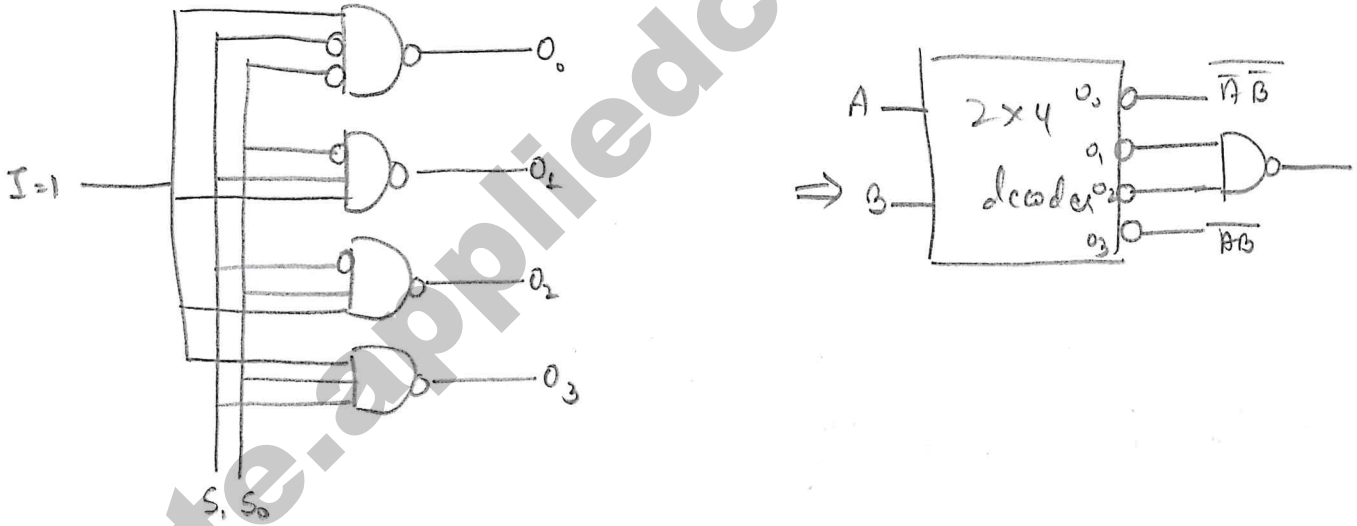
Example:

(i) Active high decoder:



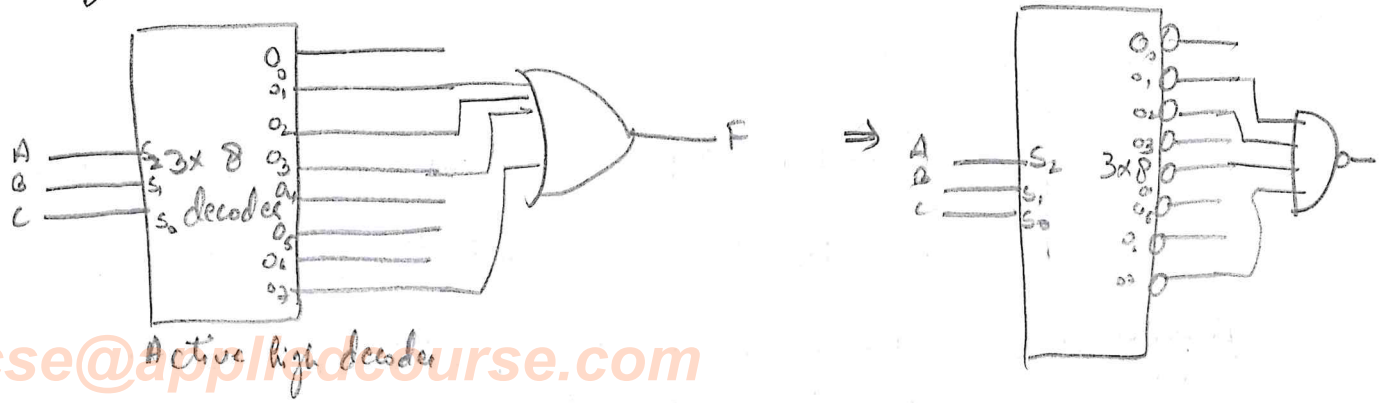
SOP $\begin{cases} F_1 = \bar{A}B + A\bar{B} \\ F_2 = \bar{A}\bar{B} + AB \end{cases}$

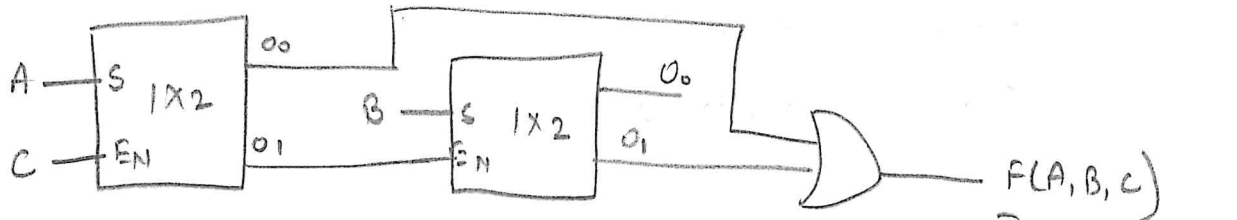
(ii) Active low decoder:



Ex- $F(A,B,C) = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC$ (1) (4) (2) (7). Realize using active high decoder.

⇒





$O_0 = \bar{S} \cdot E_N = \bar{A}C$ | $O_0 = \bar{S} \cdot AC = \bar{B}AC$

$O_1 = S \cdot E_N = AC$ | $O_1 = S \cdot AC = BAC$

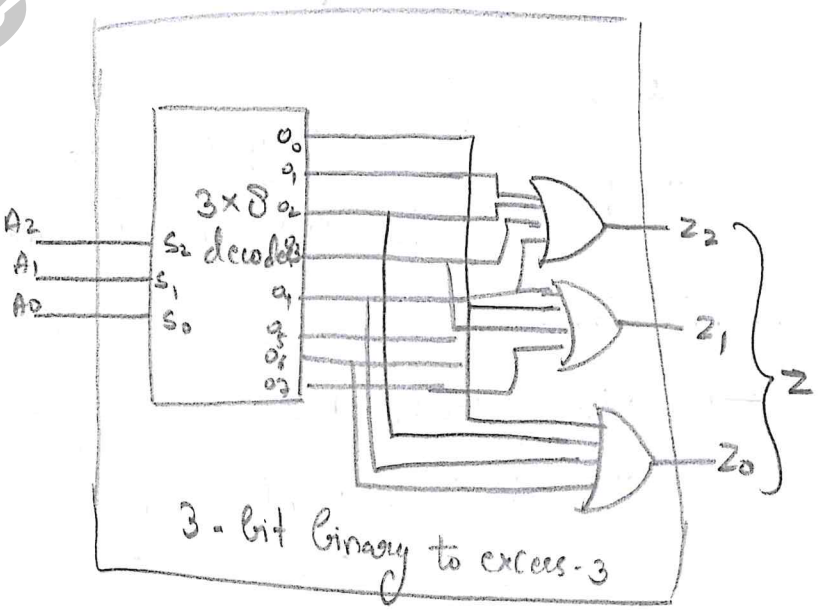
$\bar{A}C + ABC$
$= C(\bar{A} + AB)$
$= C(\bar{A} + B)$

7.12 Decoder Applications:

Decoder: code conversion:

ex - 3-bit binary to excess-3

A			Z		
A_2	A_1	A_0	Z_2	Z_1	Z_0
0	0	0	0	1	1
0	0	1	1	0	0
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	0	0	1
1	1	1	0	1	0



here, $z_2 = \Sigma(1,2,3,4)$

$z_1 = \Sigma(0,3,4,7)$

$z_0 = \Sigma(0,2,4,6)$

Decoder expansion:

Devise a high capacity of decoder using small capacity decoders with multiple levels.

Q x N Decoder using P x M decoders ($N = 2^Q$, $M = 2^P$)

Number of level $k = \lceil \frac{Q}{P} \rceil = \lceil \log_m N \rceil$

Number of decoders in i^{th} level $= \frac{N}{M^{k-i+1}}$

Total decoders $= \sum_{i=1}^{k} \left({}^{N}/_{m^{k-i+1}} \right)$

* First level only one decoder
* Decoders of level 'i' provide enable signals for level 'i+1'

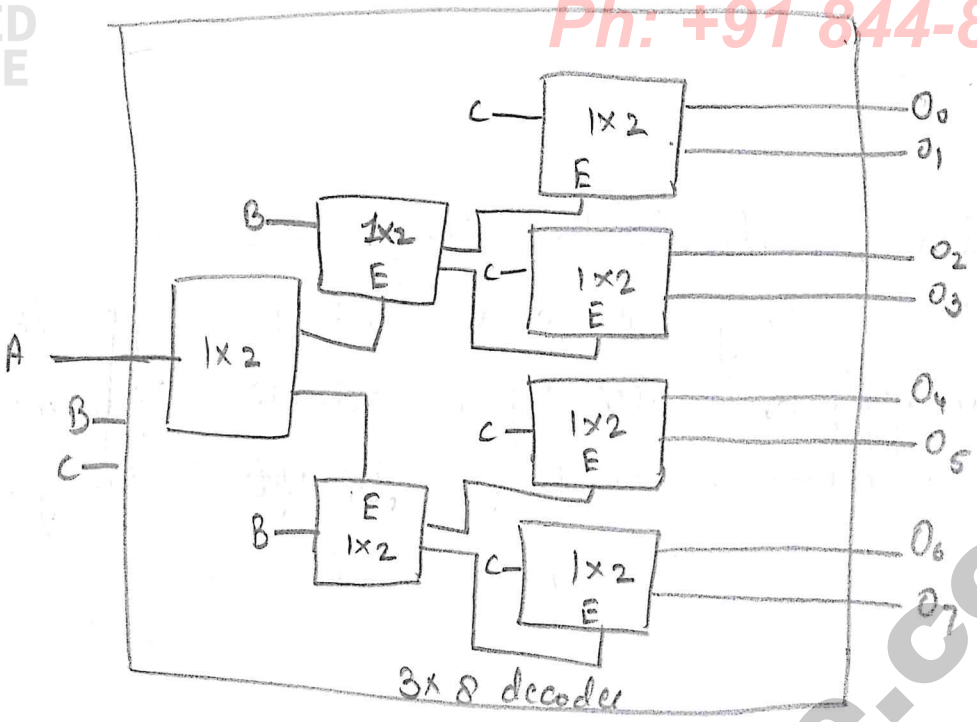
(Q) Construct 3x8 decoder using 1x2 decoder.

⇒ No. of levels, $k = \frac{3}{1} = \log_2 8 = 3$

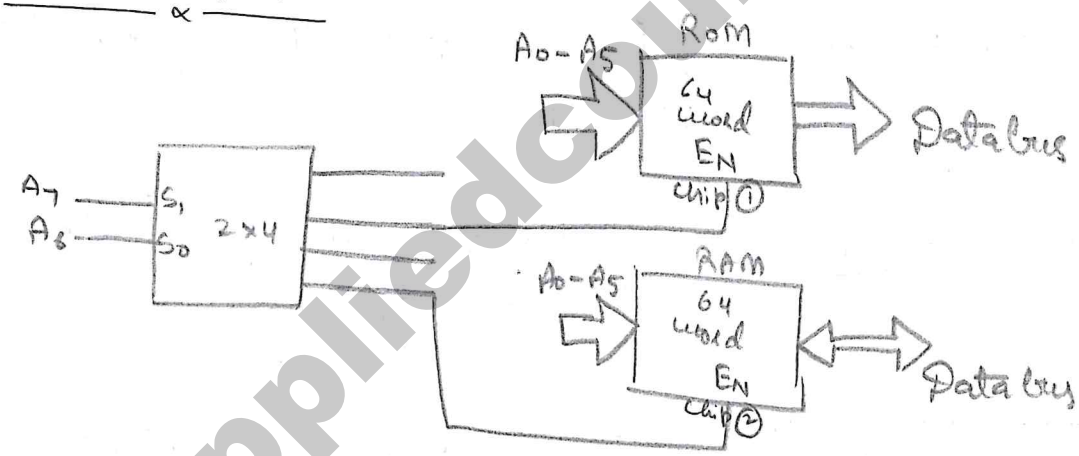
No. of 1x2 decoders in each level, i =

1	2	3
$= {}^{8}/_{2^{3-1+1}}$	$= {}^{8}/_{2^{3-2+1}}$	$= {}^{8}/_{2^{3-3+1}}$
$= {}^{8}/_{2^3}$	$= {}^{8}/_{2^2}$	$= {}^{8}/_{2}$
$= 1$	2	4

$= 7$



Decoder – Address selection:



8-bit address bus (A_7:A_0)

Address range of chip ① and chip ②

	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
chip ①	fixed 0	1	0	0	0	0	0	0	40H to 7FH (64 word)
			1	1	1	1	1	1	$(64)_{10}$ to $(127)_{10}$
chip ②	1	fixed 1	0	0	0	0	0	0	C0H to FFH (64 word)
			1	1	1	1	1	1	$(192)_{10}$ to $(255)_{10}$

O_1 → EN signal for ROM

O_3 → EN signal for RAM

7.13 Encoder:

Encoder provide lossless compression. Encoders work based on encoding rule and support single i/p activation (non-priority) or multi-input activation (priority). ~~Prior~~ Priority encoders are used for interrupt servicing.

Ex:- 4×2 encoder, 4 i/p and 2 o/p.

Encoding rule: Higher suffix i/p is encoded with highest binary value.

Priority Assignment: Higher suffix i/p is given with highest priority.

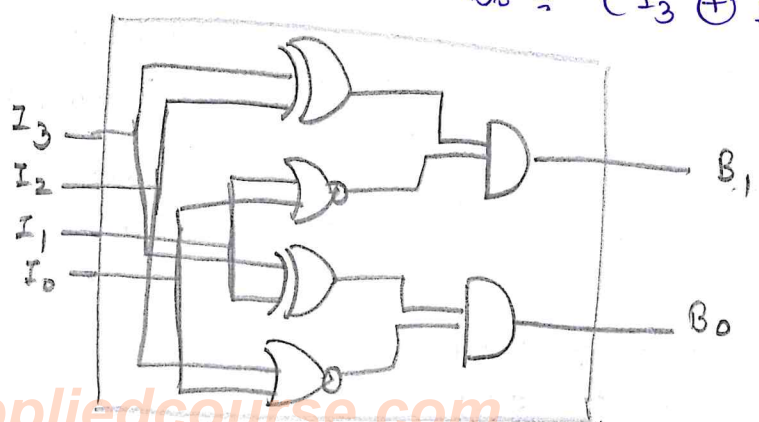
Hardware requirement: less for priority encoder compared to non-priority encoder.

4×2 non-priority encoder

I_3	I_2	I_1	I_0	B_1	B_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

$$B_1 = I_3' I_2 I_1' I_0' + I_3 I_2' I_1' I_0$$
$$= (I_3' I_2 + I_3 I_2') I_1' I_0'$$
$$= (I_3 \oplus I_2)(I_1 + I_0)'$$

Similarly,
$$B_0 = (I_3 \oplus I_1)(I_2 + I_0)'$$



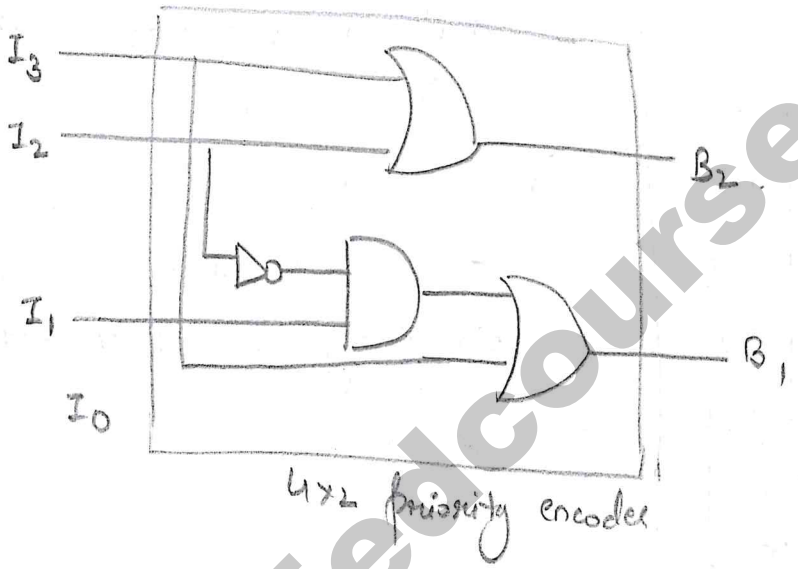
4×2 Non-priority encoder

Priority encoder: Allow simultaneous activation of i/p's

I_3	I_2	I_1	I_0	B_2	B_1
0	0	0	1	0	0
0	0	1	x	0	1
0	1	x	x	1	0
1	x	x	x	1	1

$B_2 = I_3' I_2 + I_3 = I_3 + I_2$

$B_1 = I_3' I_2' I_1 + I_3 = I_2' I_1 + I_3$



4×2 priority encoder

7.14 ROM, PAL and PLA:

ROM: Read only memory (fixed AND, Programmable OR)

PAL: Programmable Array logic (Programmable AND, fixed OR)

PLA: Programmable logic Array (Programmable AND, OR)

These LSI devices are used for realizing combinational functions in sum of Product (SOP) form. (AND, OR gate)

ROM: Read only memory capable of realizing SOP expressions by storing appropriate bits at appropriate words.

Example: 8x2 Rom associate 8 words and each word contain 2-bits. This effectively represents 8-i/p combinations for 2-functions. Thus it realizes two 3-variable boolean functions.

Word:

abc	Word
0	00
1	10
2	10
3	01
4	10
5	01
6	01
7	11

B_1 B_0

F_1 F_2

Sum, Carry

001 010 100 111

$F_1(a,b,c) = \Sigma(1,2,4,7)$

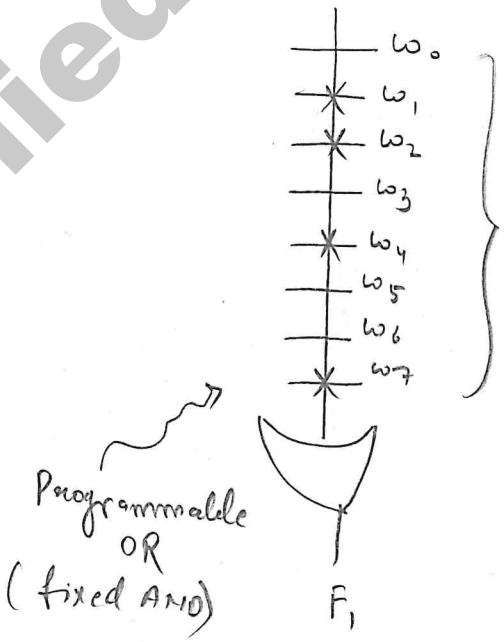
$a'b'c + a'bc' + ab'c' + abc$

⇒ sum of a full adder

$F_2(a,b,c) = \Sigma(3,5,6,7)$

$a'bc + ab'c + abc' + abc$

⇒ carry of a full adder

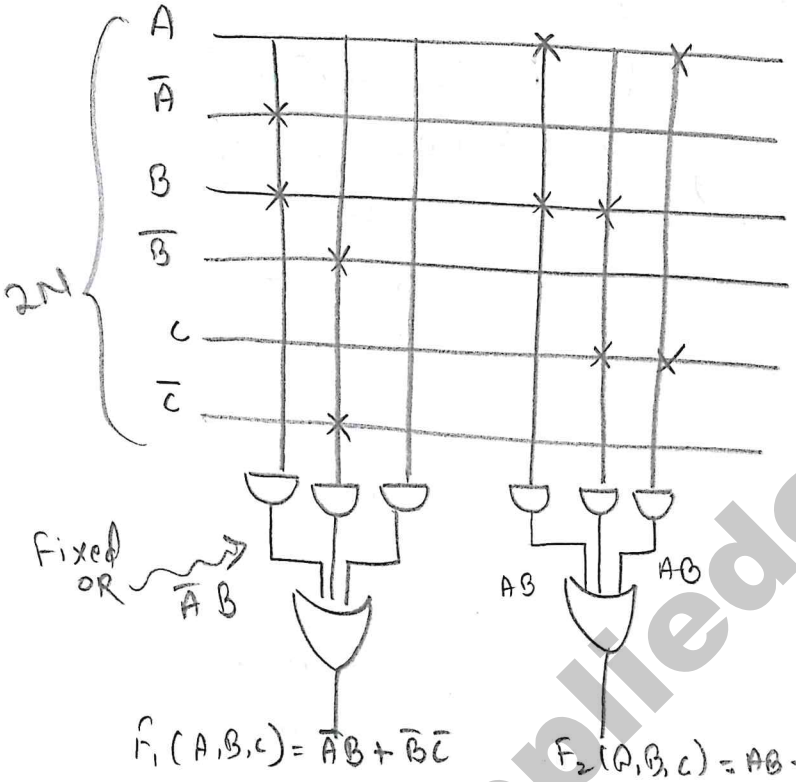


Application:- Rom is used to represent look up Tables

Features: Fixed AND & Programmable OR.

PAL: Programmable logic array, it used to realize combinational functions. Here programmable AND and fixed OR are used. (2n i/p's are sufficient generate all products)

PAL supports minimization.

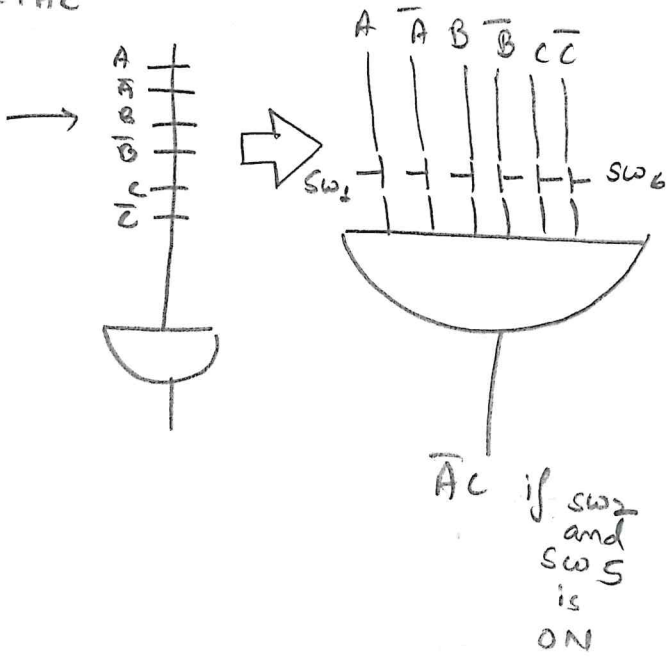


$F_1(A,B,C) = \bar{A}B + \bar{B}\bar{C}$

$F_2(A,B,C) = AB + BC + AC$

OR: i/p are fixed

AND: i/p are programmable



PLA: Programmable Logic array, also support minimization and allow programming both AND and OR gates. Here any product is generated by 2N i/p's and these products are shared across several functions.

$F_1(A,B,C) = B + AC$

$F_2(A,B,C) = AB + BC + CA$

$F_3(A,B,C) = \bar{A}\bar{B} + C$

$F_4(A,B,C) = \Sigma(3,4,5,6,7) \Rightarrow A + BC$

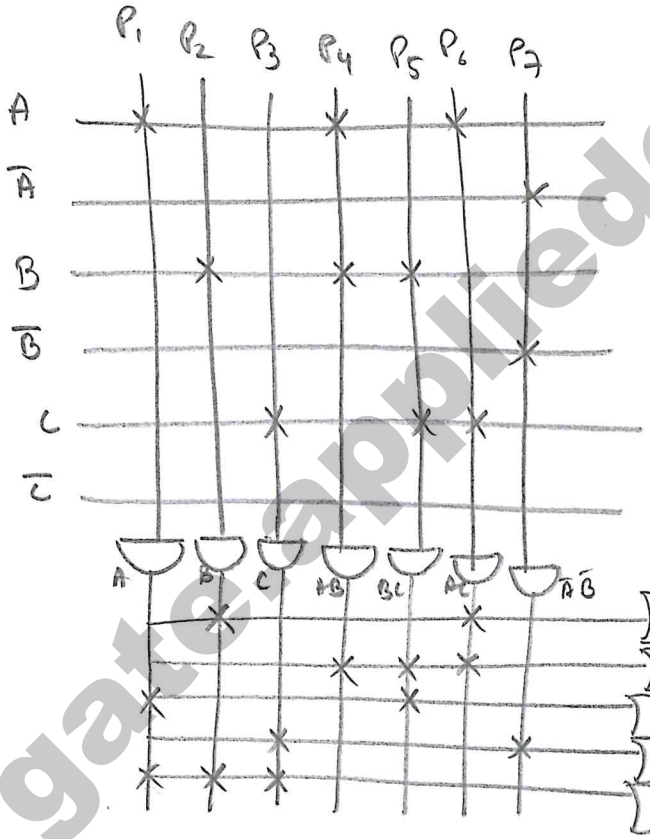
$F_5(A,B,C) = A + B + C$

5 - 3 variable functions

$F_3(A,B,C)$ after simplifying

$= A + BC$

A	B	C	
0	1	1	- 3
1	0	0	- 4
1	0	1	- 5
1	1	0	- 6
1	1	1	- 7



$P_1 = A$; $P_2 = B$; $P_3 = C$

$P_4 = AB$; $P_5 = BC$;

$P_6 = AC$; $P_7 = \bar{A}\bar{B}$

$F_1 = B + AC = P_2 + P_6$

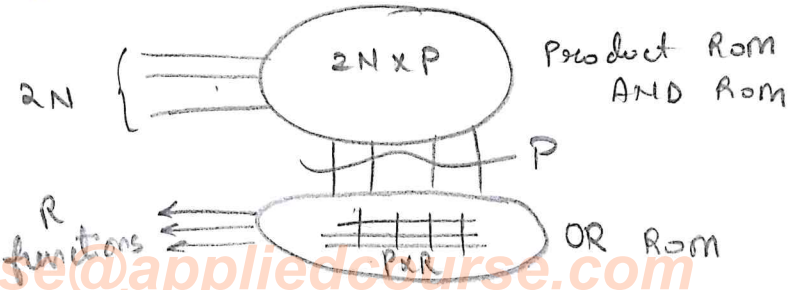
$F_2 = AB + BC + CA = P_4 + P_5$

$F_3 = A + BC = P_1 + P_5$

$F_4 = \bar{A}\bar{B} + C = P_3 +$

$F_5 = A + B + C = P_1 + P_2$

Size of PLA:



∴ $2N \times P + P \times R$ connections are required.

Example:

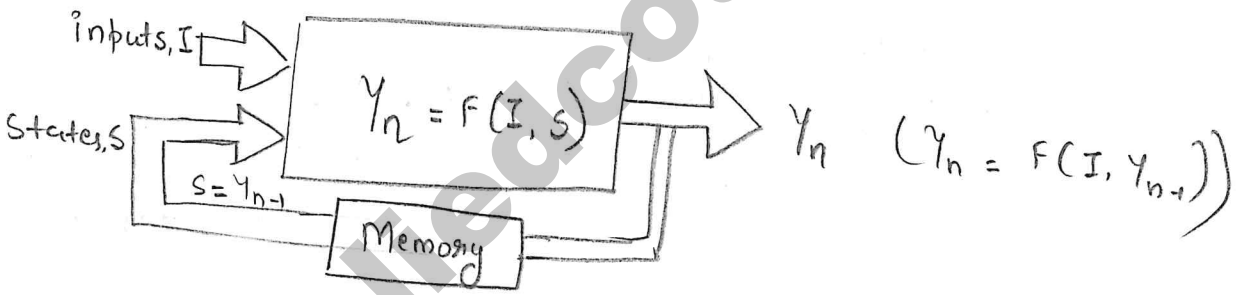
5, 3-variable functions using PLA, There are 8 ~~inf~~ independent products

$6 \times 8 + 8 \times 5 = 48 + 40 = 88$ links or 88 connection.

(6 × 8: AND Rom; 8 × 5: OR Rom)

8. SEQUENTIAL CIRCUITS:

8.1 Sequential circuits:



In Sequential circuits current o/p Y_n is depend not only on current i/p's I, but also on previous ~~out~~ o/p's Y_{n-1}.

The previous o/p's termed as state, S

$$Y_n = F(I, Y_{n-1}) = F(I, S)$$

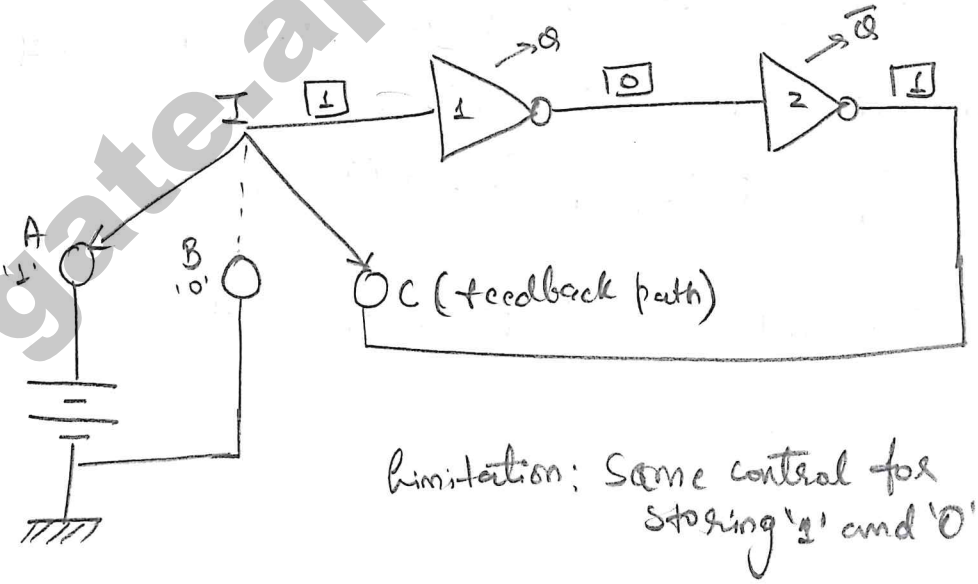
Sequential circuit = Combinational circuit + memory element

Flip Flop: It is a basic memory element capable of one bit storage ('1' or '0'). It contain 2-states (Bi-stable multivibrators). Flip-flop maintain 2-outputs always have complementary relation for valid operation.

FF's are classified on
- Function they perform (Delay, set, reset, toggle)
- Synchronization (Latch, clocked FF etc)
- Construction (NOR, NAND FF)

Each flip-flop is described by ($Q_n = F(I, S)$)
- Characteristic expression
- state table
- state diagram
- Excitation table (what values are used for exciting FF, so that it is having desired response)

Analysis of a Sample 2-inverter back-to-back connected circuits

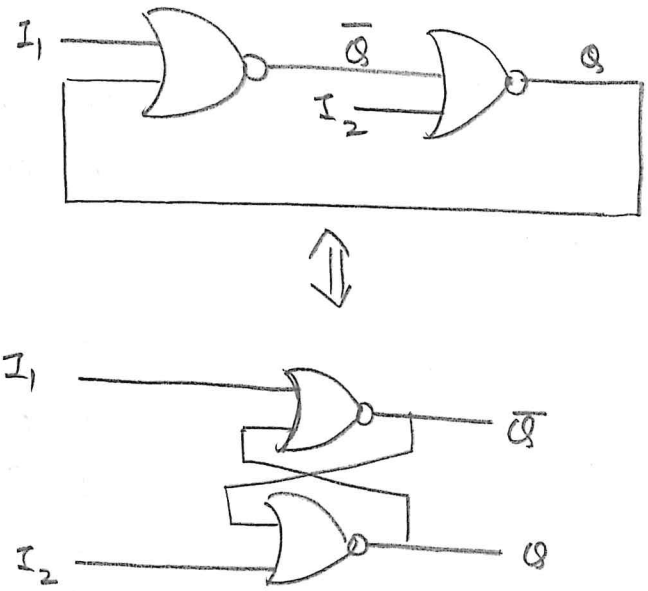


The contact is initially at 'A' and moved to 'C' after 1ons

Limitation: Same control for storing '1' and '0'

This inverter act as memory element due to back to back connection.

OR



18.2 Introduction to Flipflops : SR Flipflop :

SR Flip flop is called as set-reset flip-flop. SR flip-flop is constructed with NOR gates it is termed as NOR FF

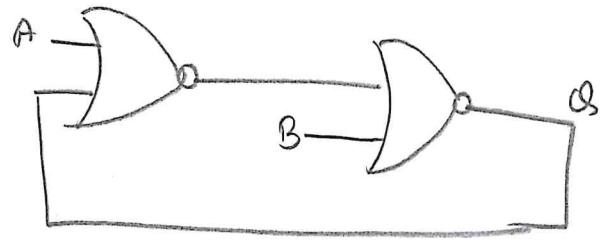
NOR gate analysis :

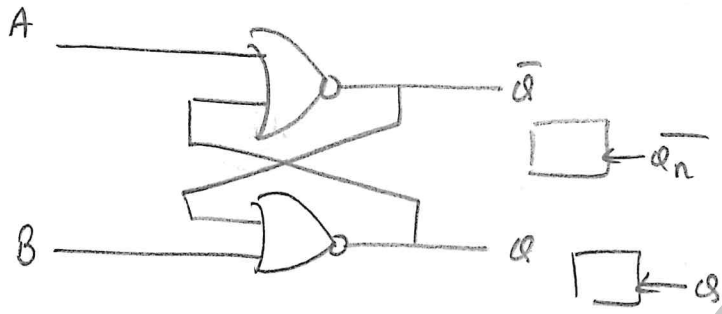
$\Rightarrow$

P	Q	NOR
0	0	1
0	1	0
1	0	0
1	1	0

NOR o/p $\Rightarrow$ $1\phi = \phi 1 = 0$

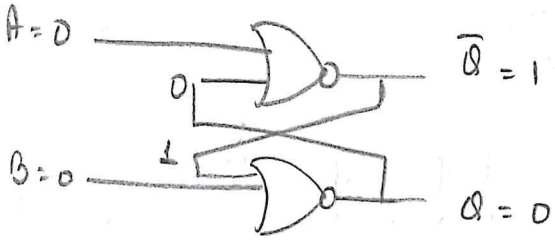
* If any one of NOR i/p is high, o/p is low.





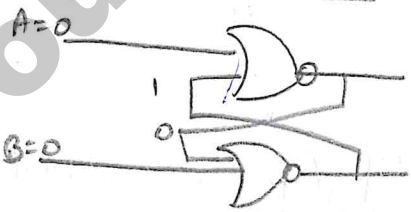
A	B	Q	Q_n	
0	0	0	0	latch or retain
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	Q	invalid
1	1	1	$\bar{Q}$	

For A=0, B=0, Q=0

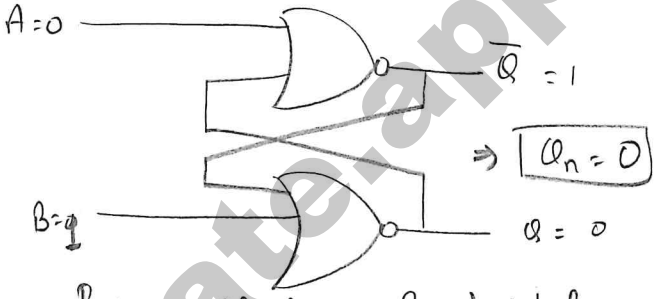


$\Rightarrow Q_n = Q = 0$

For A=0, B=0, Q=1



For A=0, B=1, Q=0



B is called as Reset input

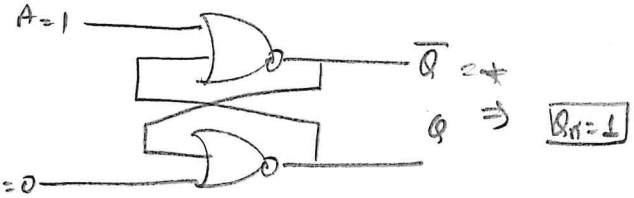
A=0, B=1, whatever be Q

$Q_n = 0$

For A=0, B=1, Q=1

$\Rightarrow$ same.

For A=1, B=0, Q=0



A is called set i/p

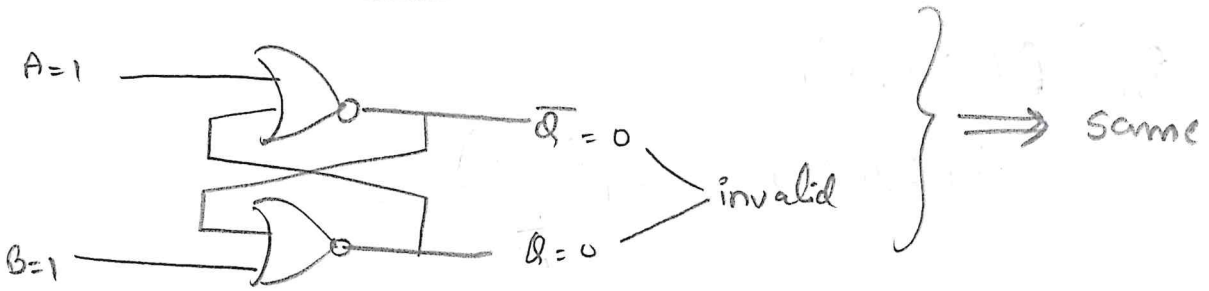
A=1, B=0, whatever value of Q, $Q_n = 1$

For A=1, B=0, Q=1

$\Rightarrow$ same.

for A=1, B=1, Q=0

for A=1, B=1, Q=1



Stable o/p are not satisfying complementary relation.

NOR SR-FF give invalid operation when both S=R=1 and invalid is denoted as don't care.

P.3 SR Flip Flop: Characteristic expression, Excitation table, State table and state diagram:

As we know, for SR-FF:

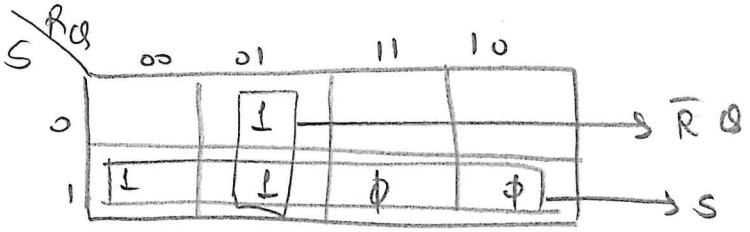
S	R	Q_n	
0	0	Q	} function table
0	1	0	
1	0	1	
1	1	ϕ	

Characteristic table:

S	R	Q	Q_n	
0	0	0	0	} Retain
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	ϕ	} invalid
1	1	1	ϕ	

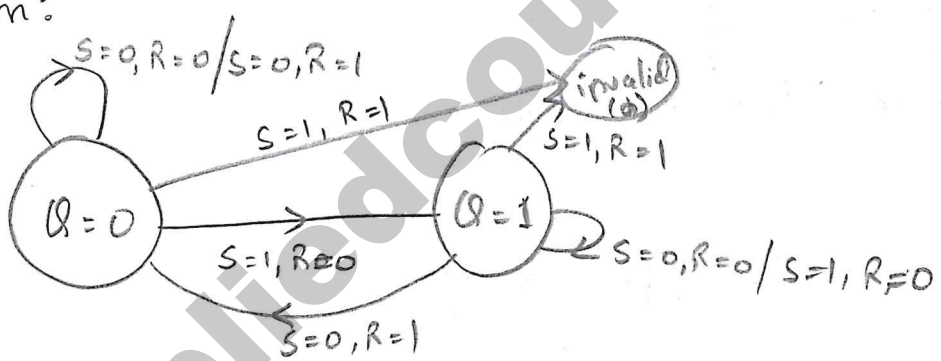
Characteristic Expression:

$$Q_n = f(S, R, Q)$$
$$= \Sigma(1,4,5) + \Sigma_\phi(6,7)$$



∴ expression = $S + \bar{R}Q$

State diagram:



Every state has 4 transitions (SR = 00, 01, 11, 10)

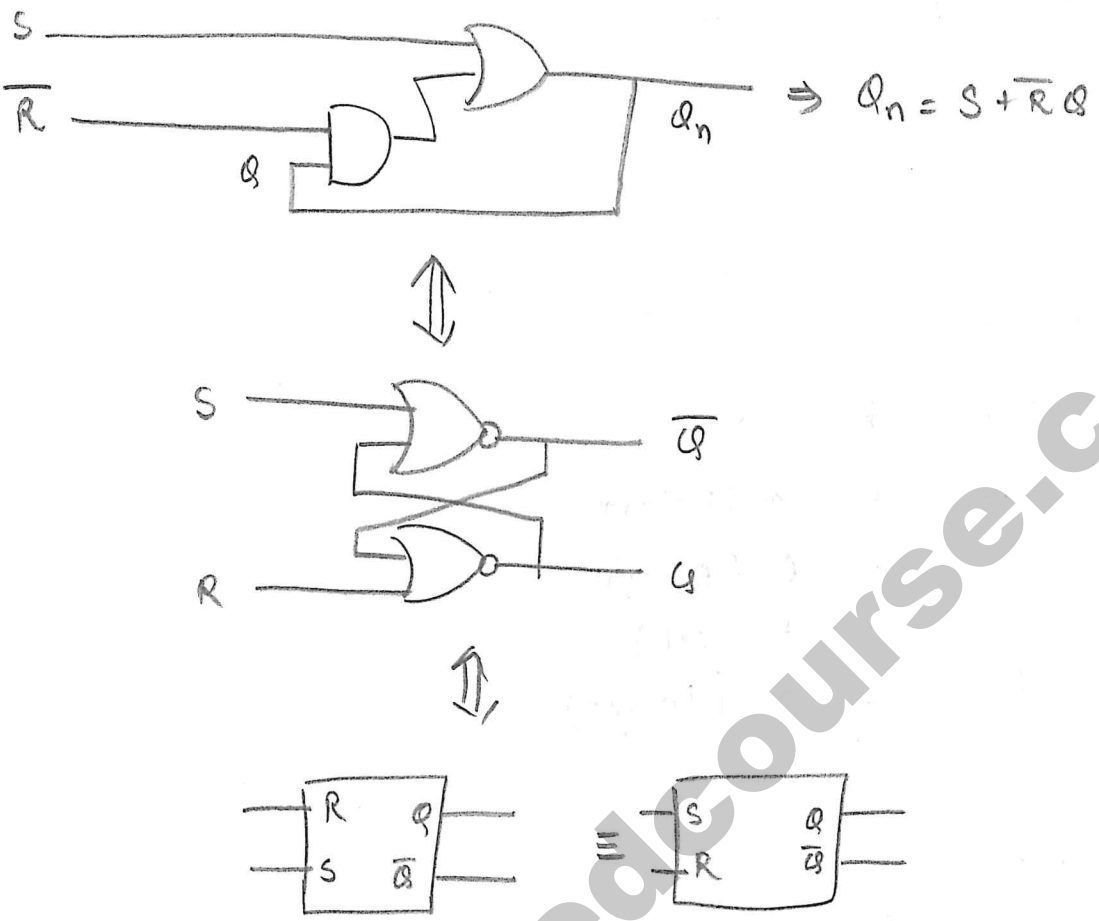
Excitation table:

To get desired change at o/p ($Q \to Q_n$) what values are to be applied at S, R variable

Q	Q_n	S	R
0	0	0	ϕ
0	1	1	0
1	0	0	1
1	1	ϕ	0

(Built with the help of state diagram)

8.4 SR-Flip Flop Part 3:



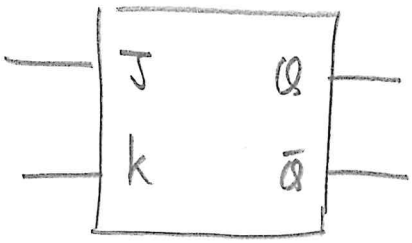
Drawback: It won't allow S=R=1 for valid operation

This operation is solved with new FF called as JK-FF.

8.5 JK-Flip Flop:

JK Flip flop is nothing but SR FF but it have valid operation if J=K=1. When both i/p's are 'one' JK Flip flop complement its o/p $\boxed{Q_n = \bar{Q}}$

Symbolic diagram:



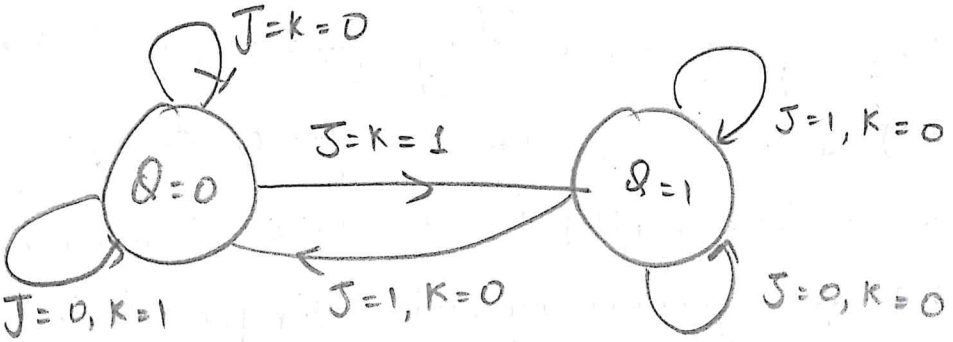
Function table:

J	K	Q_n
0	0	Q (Retain)
0	1	0 (Reset)
1	0	1 (set)
1	1	$\bar{Q}$ (toggle)

State table:

J	K	Q	Q_n	
0	0	0	0	} Retain $Q_n = Q$
0	0	1	1	
0	1	0	0	} Reset $Q_n = 0$
0	1	1	0	
1	0	0	1	} set $Q_n = 1$
1	0	1	1	
1	1	0	1	} toggle $Q_n = \bar{Q}$
1	1	1	0	

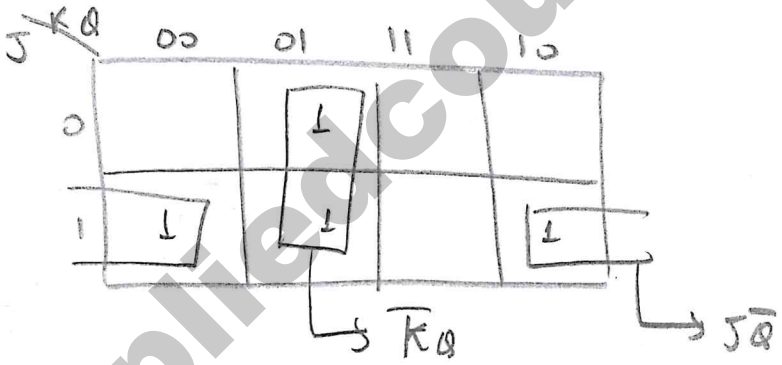
State Diagram:

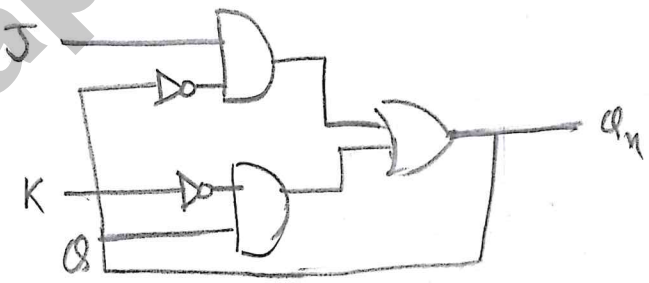


Characteristic expression:

$$Q_n = F(Q, J, K) = \Sigma(1, 4, 5, 6)$$

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





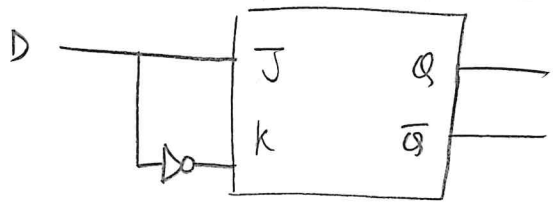
Excitation table:

Q	Q_n	J	K
0	0	0	ϕ
0	1	1	ϕ
1	0	ϕ	1
1	1	ϕ	0

8.6 D-Flip Flop: (J ≠ K)

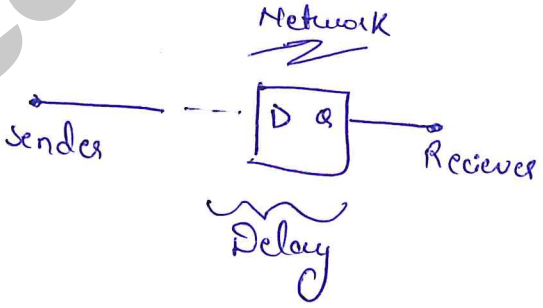
JK Flip Flop provide delay operation, if J ≠ K. Delay FF (D-FF) is the basic FF in shift counters / shift registers. It is the versatile flipflop in the data communication. It is used for synchronising data transfer between fastest sender to slowest reciever

Symbolic diagram:



$D = 0 \Rightarrow J = 0, K = 1 \Rightarrow Q_n = 0$ (Reset)

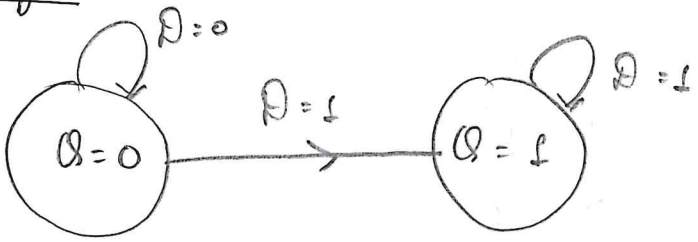
$D = 1 \Rightarrow J = 1, K = 0 \Rightarrow Q_n = 1$ (set)



State table:

D	Q	Q_n	
0	0	0	$Q_n = 0$ (Reset)
0	1	0	
1	0	1	$Q_n = 1$ (set)
1	1	1	

State Diagram:



Characteristic equation:

$$Q_n = f(D, Q)$$
$$= D\overline{Q} + DQ$$
$$Q_n = D$$

D-FF exhibit True combinational behaviour

Anything and everything act as D-FF, a piece of wire, buffer or literally anything that offer delay

Ex -

① D — write or wire — Q_n

② D — Buffer — Q_n

⑤ D — inverter cascading — $Q_n = D$

Produce D-FF operation

Excitation table:

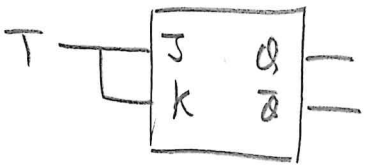
Q	Q_n	D ($D = Q_n$)
0	0	0
0	1	1
1	0	0
1	1	1

whatever Q_n is needed it must be applied at D.

8.7 T - Flip Flop: (J = K)

JK FF provide toggle operation if J=K. T-FF is the basic element in asynchronous counters. If T=1, it provide modulo-2 operation (MOD-2 counter)

Symbolic Diagram:



$T=0 \Rightarrow J=K=0 \Rightarrow Q_n = Q$

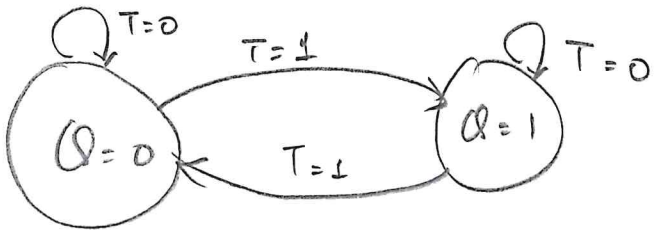
$T=1 \Rightarrow J=K=1 \Rightarrow Q_n = \overline{Q}$

T	Q_n
0	Retain
1	Toggle

State table:

T	Q	Q_n	
0	0	0	$Q_n = Q$
0	1	1	
1	0	1	$Q_n = \overline{Q}$
1	1	0	

State Diagram:



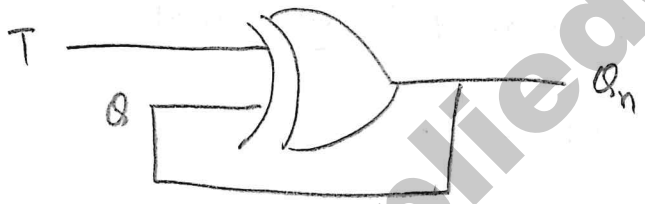
Excitation table:

Q	Q_n	T
0	0	0
0	1	1
1	0	1
1	1	0

if $\Delta Q \neq 0 \Rightarrow T = 1$
else, $\not\Rightarrow T = 0$

Characteristic Expression:

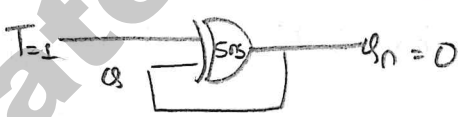
$$Q_n = F(T, Q) = \Sigma(1, 2)$$
$$= \bar{T}Q + T\bar{Q}$$
$$= T \oplus Q$$

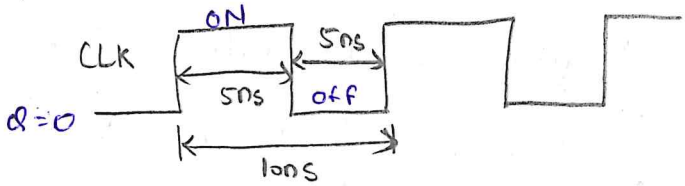


(Q) Find clock period if $T = 1$ in T-FF

$Q_n = \bar{Q}$ for $T = 1$

$\Rightarrow$





$T_{CLK} = 10\,ns$, $f_{CLK} = 1/10\ ns/cycle = \frac{10^9}{10}$ cycles/sec

$= \frac{1000 \times 10^6}{10}$ cycles/sec

$= 100MHz$

9t produce a square wave clock with 50% duty cycle.

Application: Clock signal generation

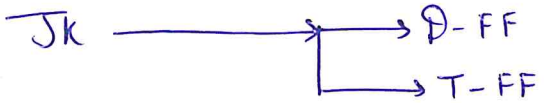
Note:

Clock period = Time taken for 1 cycle

9.8 FF-interconversion:

FF interconversion allows converting any other flip flop.

for ex, 2-i/p FF into 1-i/p FF



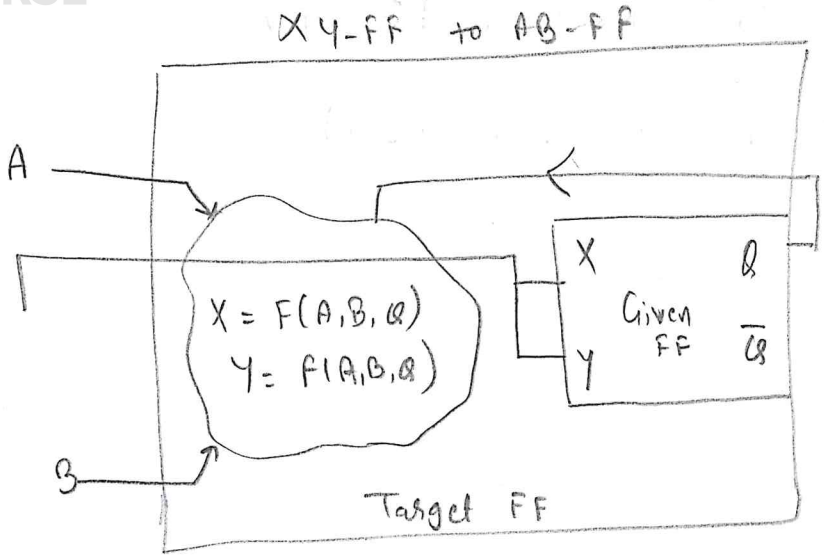
While 1-i/p FF is converted into 2-i/p FF by simulation.

FF used in the design is called as Given FF (Basic) and FF designed is called Target FF (Desired)

Ex- Convert T-FF to JK-FF.

Interconversion Procedure:

1. Get state table of target FF.
2. Replace next state (Q_n) using excitation table of given FF. This step ensure given FF simulating behaviour of target FF.
3. Get expression for i/p's of given FF in terms of target FF and realize.



Example ① Convert T-FF into JK-FF

① state table of JK-FF

J	K	Q	Q_n	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

⇓ step 2

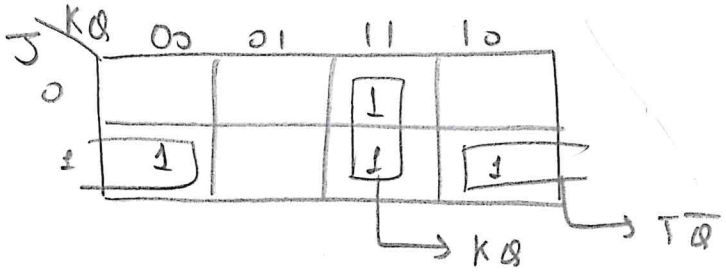
② Replace Q_n with excitation table of T-FF

$\Delta Q \neq 0 \Rightarrow T = 1$
else $T = 0$

$Q \longrightarrow Q_n$	
0 →	1 } T = 1
1 →	0 }

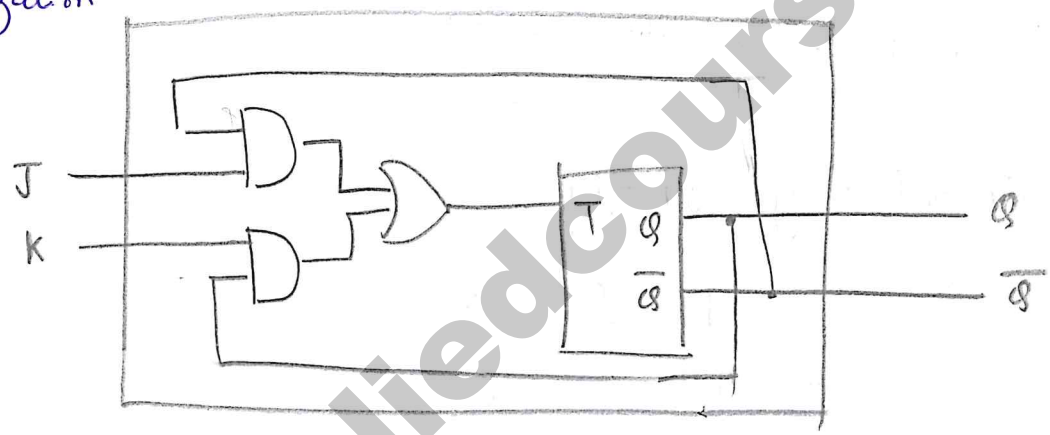
(3) Get expression for T (Given FF i/p)

$T = F(J, K, Q) = \Sigma(3,4,6,7)$



$T = J\bar{Q} + KQ$

(4) Realization



Example (2) Convert D-FF into T-FF

(1) State table of T-FF

(2) Replace next state (Q_n) with excitation of D-FF

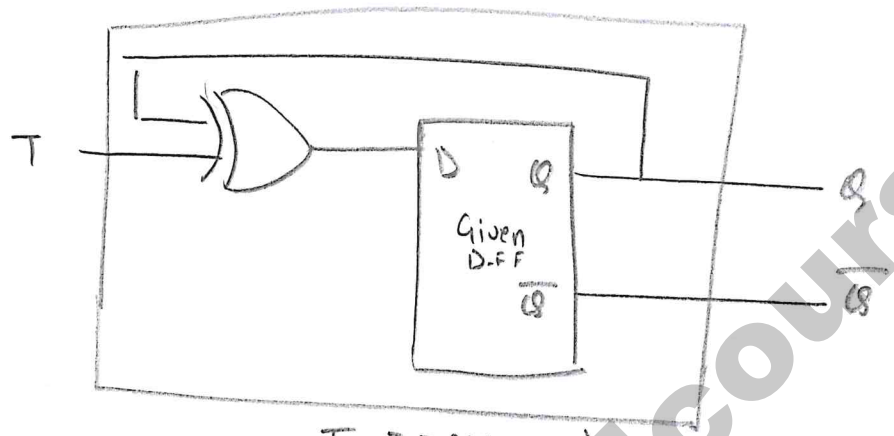
$D = Q_n$

T	Q	Q_n	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

③ Get expression for D

$$D = F(T, Q) = \Sigma(1, 2)$$
$$= \bar{T}Q + T\bar{Q}$$
$$= T \oplus Q$$

④ Realization:



T-FF (Target)

$D = T \oplus Q$

Example ③ T-FF into D-FF:

① State table for target FF

D	Q	Q_n	T_{Req}
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

② Replace Q_n using excitation table of T-FF.

i.e $A \neq 0 \Rightarrow T = 1 \ (Q_n \neq Q)$

else $T = 0$

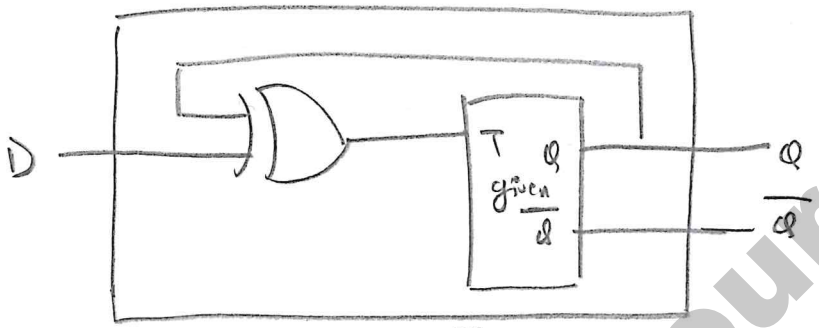
③ Expression of T as a function of D, Q

$$T = F(D, Q) = \Sigma(1, 2)$$

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

$$= D \oplus Q$$

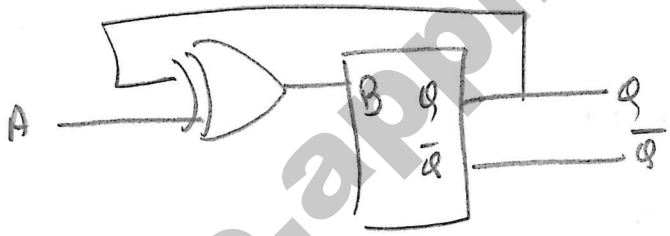
④ Realization



D-FF Target

$T = D \oplus Q$

NOTE-



If A = T-FF, then B = D-FF or, A = D-FF then B = T-FF

∴ D and T-FF are interchangeable.

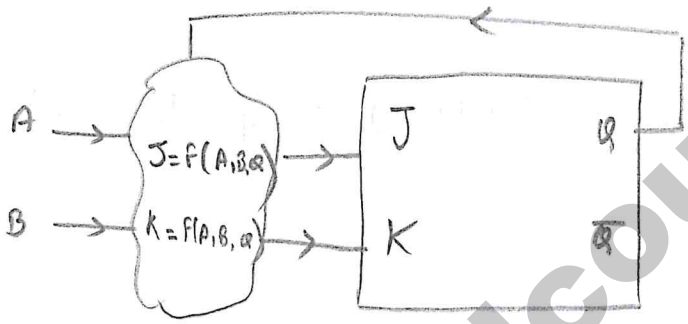
NOTE-

Given	Target-FF: SR	JK	T	D
f(S,R,Q) SR-FF	—	J= K=	T=	D=
f(J,K,Q) JK-FF	S= R=	—	T=	D=
f(T,Q) T-FF	S= R=	J=, K=	—	D=
f(D,Q) D-F	S=	J=	T=	—

8.9 Design a new AB flip flop using JK flip flop:

Let a new AB-FF behaviour is given below::

A	B	Q_n
0	0	0 (Reset)
0	1	Q (Retain)
1	0	$\overline{Q}$ (Toggle)
1	1	1 (set)



Solution:

① State table of AB-FF

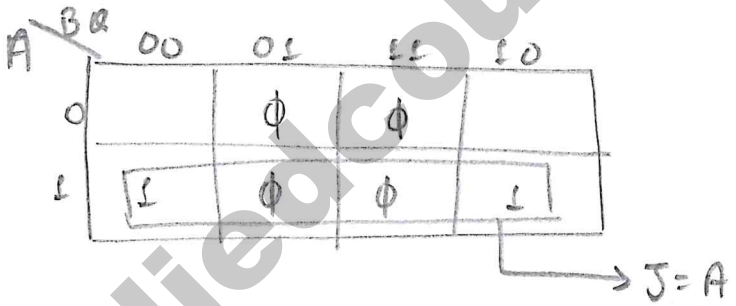
A	B	Q	Q_n	J	K
0	0	0	0	0	ϕ
0	0	1	0	ϕ	1
0	1	0	0	0	ϕ
0	1	1	1	ϕ	0
1	0	0	1	1	ϕ
1	0	1	0	ϕ	1
1	1	0	1	1	ϕ
1	1	1	1	ϕ	0

(2) Replace Q_n using JK excitation table

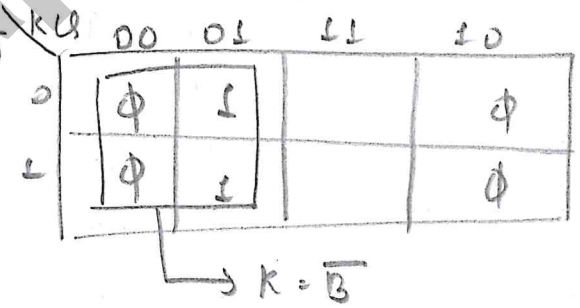
Q	Q_n	J	K
0	0	0	ϕ
0	1	1	ϕ
1	0	ϕ	1
1	1	ϕ	0

(3) Expression for J, K

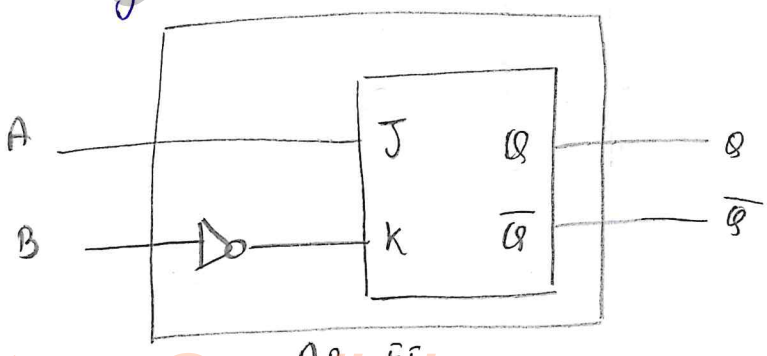
$$J = F(A, B, Q) = \Sigma(4, 6) + \Sigma_\phi(1, 3, 5, 7)$$



$$K = F(A, B, Q) = \Sigma(1, 5) + \Sigma_\phi(0, 2, 4, 6)$$



(4) Realization



8.10 FF interconversion GATE Questions:

(Q1) SR Flipflop is constructed with NOR gates, S=0, R=0, Q=0. If clp 0 is changed to 1, SR values

(a) 0, 1
(b) 1, 0
(c) 1, ϕ
(d) ϕ, 1

⇒ ∵ S = R = 1 ⇒ invalid

The function table is:

S	R	Q_n
0	0	Q
0	1	0
1	0	1
1	1	invalid

∴ S = 1 and R = 0 in order to have $Q_n = 1$

∴ b is the answer.

(Q2) In SR-FF, if S = R = 1 then Q_n value is

(a) 0 (b) 1 (c) Q (d) Invalid

⇒ for S = 1, R = 1

Q_n = invalid

(Q3) A state table of 2-FF is Given below

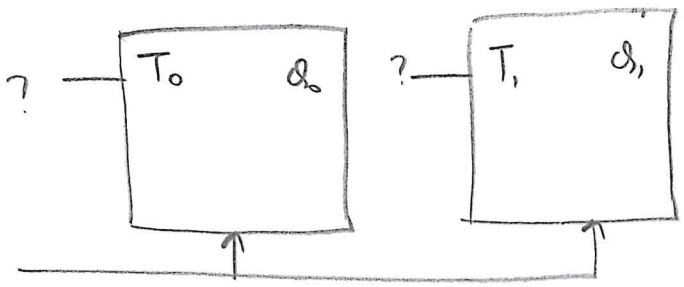
Present state		Next state	
Q_1	Q_0	Q_{1n}	Q_{0n}
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	1

If flip flop T_1 realizes state variable Q_1 and T_0 realizes state variable Q_0

$T_1 (Q_1, Q_0) = ?$

$T_0 (Q_1, Q_0) = ?$

$\Rightarrow$



$\therefore \Delta Q \neq 0 \Rightarrow T = 1$

else $T = 0$

$Q_{1n} \neq Q_1 \Rightarrow T = 1$

Q_1	Q_0	Q_{1n}	T_1
0	0	0	0
0	1	1	1
1	0	1	0
1	1	1	0

$T_1 = \overline{Q_1} Q_0$

$Q_{1n} \neq Q_1 \Rightarrow T_1 = 1$

$Q_{1n} = Q_1 \Rightarrow T_1 = 0$

Q_1	Q_0	Q_{0n}	T_0
0	0	1	1
0	1	0	1
1	0	1	1
1	1	1	0

$T_0 = \overline{Q_1} + \overline{Q_0}$ or

$T_0 = \overline{Q_1}\,\overline{Q_0} + \overline{Q_1} Q_0 + Q_1 \overline{Q_0}$

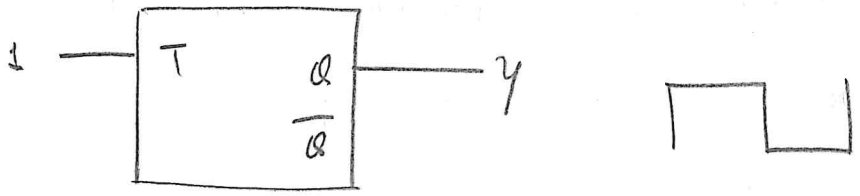
$Q_{0n} \neq Q_0 \Rightarrow T_0 = 1$

$Q_{0n} = Q_0 \Rightarrow T_0 = 0$

$\therefore T_1 = \overline{Q_1} Q_0$

$T_2 = \overline{Q_1} + \overline{Q_0}$

(Q4)

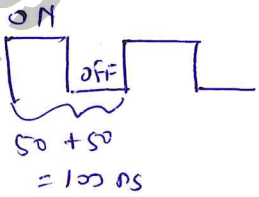


If propogation delay of T-FF is 50 ns then clock frequency at y

(a) 50 GHz
(b) 20 GHz
(c) 10 GHz
(d) 0.01 GHz

⇒ Clock period at $y = 2 * 50$ ns

$= 100$ ns



$$\text{frequency} = \frac{1}{100 \text{ ns/cycle}} = \frac{1}{100 \times 10^{-9} \text{ sec/cycle}}$$

$$= \frac{10^9}{100} \text{ cycles/sec} = \frac{1000}{100} \times 10^6 \text{ Hz}$$

$= 10$ MHz $= \frac{10}{1000}$ GHz $= 0.01$ GHz //

(Q5) XY-FF behaviour is given below:

X	Y	Q_n
0	0	1
0	1	Q
1	0	$\overline{Q}$
1	1	0

If XY-FF is realized by JK-FF expression for J and K are

(a) $J = X$, $K = \overline{Y}$ (b) $J = \overline{Y}$, $K = X$

(c) $J = \overline{X}$, $K = \overline{Y}$ (d) $J = Y$, $K = X$

⇒ (b) State table and excitation table will be:

X	Y	Q_n	J	K
0	0	1	1	0
0	1	Q	0	0
1	0	$\overline{Q}$	1	1
1	1	0	0	1

$J = F(X, Y) = \overline{X}\,\overline{Y} + X\overline{Y} = \overline{Y}(\overline{X} + X)$

$= \overline{Y}$

∴ only one option have $J = \overline{Y}$

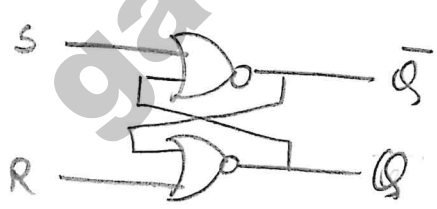
∴ (b) is the correct answer.

Similarly, $K = F(X, Y) = X\overline{Y} + XY = X(\overline{Y} + Y) = X$

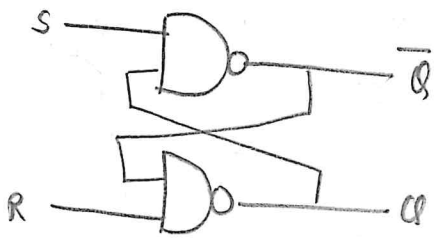
∴ $K = X$, $J = \overline{Y}$

8.11 FF Clocks Terminology:

SR-NAND gate FF:



SR-NOR gate FF

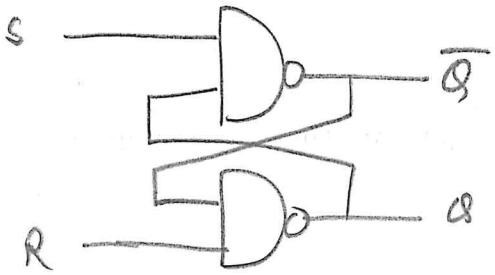


SR-NAND gate FF

In a NAND gate,

$\Rightarrow$ $0\phi = \phi 0 = 1$ (i/p, o/p)

$\Rightarrow$ $11 = 0$ (i/p, o/p)



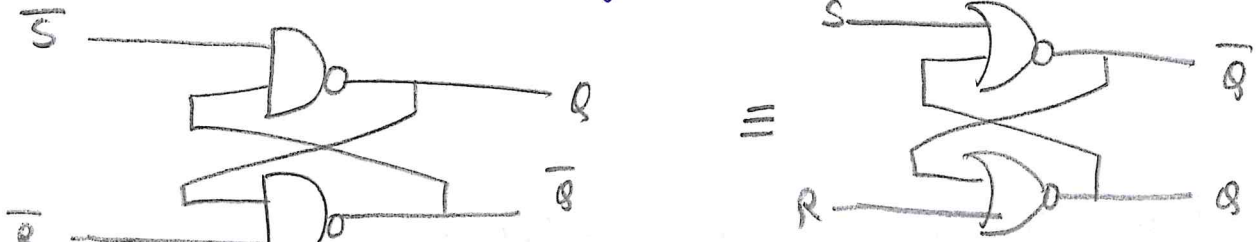
$\Rightarrow$

S	R	Q_n
0	0	Invalid
0	1	0 (Reset)
1	0	1 (Set)
1	1	Q (Retain)

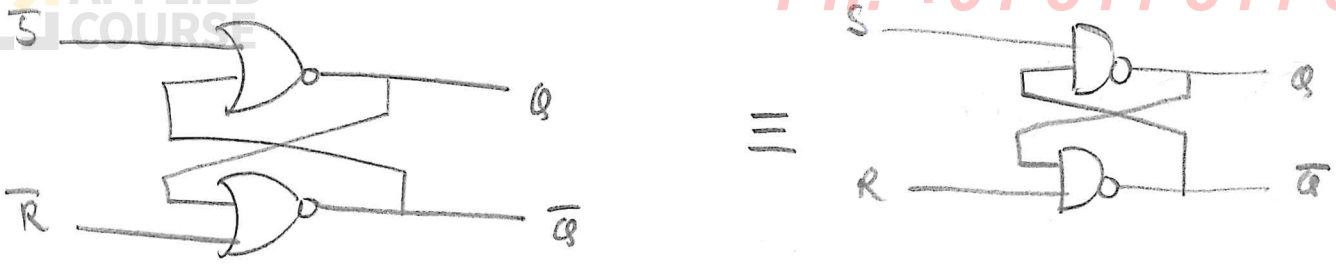
NAND SR behave as NOR SR if $S \neq R$. If $S = R$ NAND SR-FF and NOR SR-FF behave oppositely.

		Q_n	
S	R	NOR SR	NAND SR
0	0	Q	ϕ (Invalid)
0	1	0	0
1	0	1	1
1	1	ϕ (Invalid)	Q

NAND SR-FF $\equiv$ NOR SR-FF, if i/p's are complemented and o/p positions are shifted.



NAND SR-FF $\equiv$ NOR SR-FF

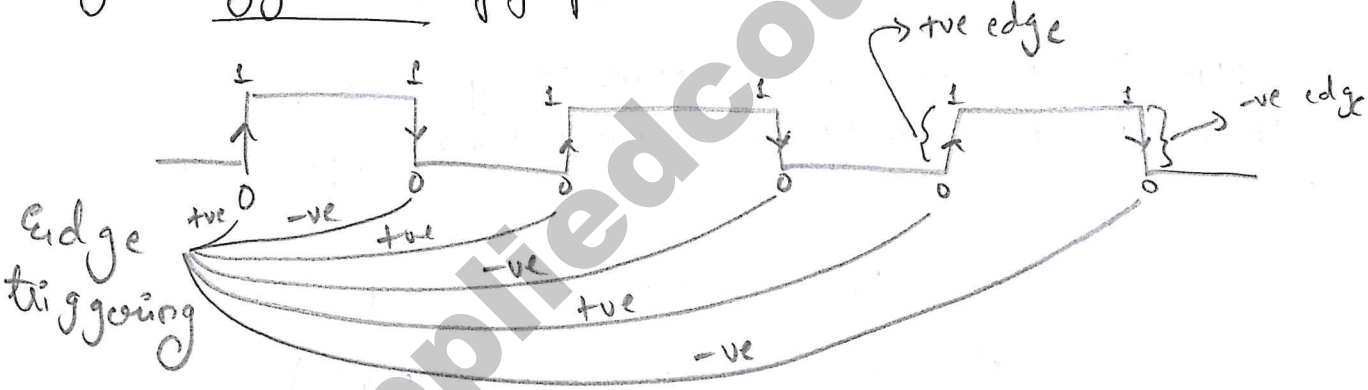


NOR SR-FF ≡ NAND SR-FF

Clocked FF :

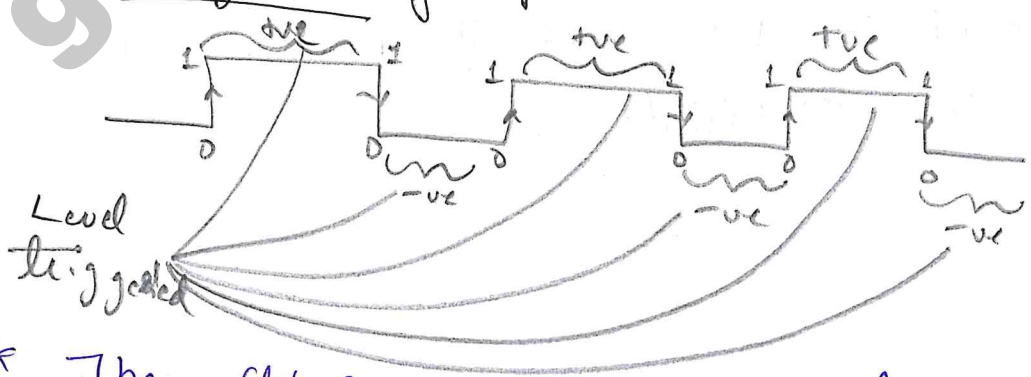
Flip flop operation is made to synchronize with respect to external signal clock.

* Edge triggered flip flops :



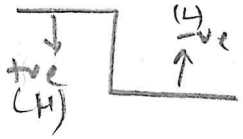
* Edge triggered FF are called synchronous FF.
* It is better solution for noisy signal
* -ve edge triggering is mostly used in practical applications

* Level triggered Flip Flops :

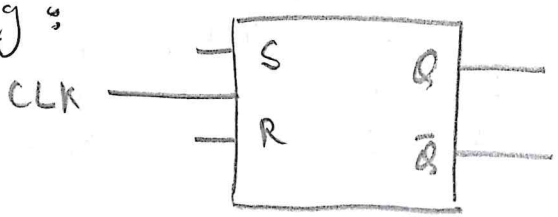


* These flip flops are also called latches and Asynchronous flip flops.

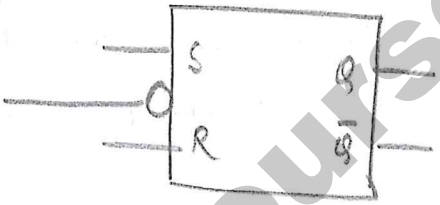
Symbolic representation of clocked flip flops:

Level triggering :— 

+ve level triggering:

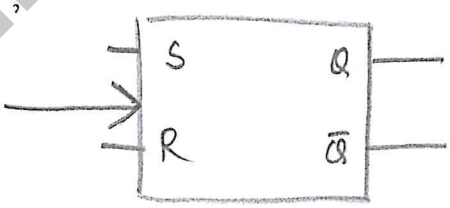


-ve level triggering:

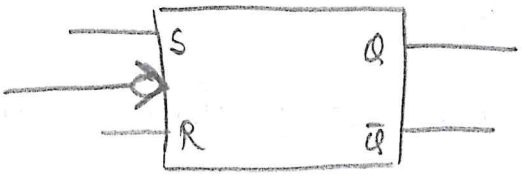


Edge triggering :— (more insensitive to noise) 

+ve edge triggering:



mostly used -ve edge triggering:

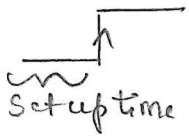


Clocked SR-FF :

S	R	CLK	Q_n	
ϕ	ϕ	Inactive	Q	→ latched ($Q_n = Q$)
0	0	active	Q	→ Follow external i/p's S, R
0	1		0	
1	0		1	
1	1		Invalid	

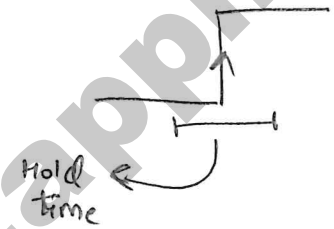
Set up time:

It is the minimum amount of time stable input is needed before active clock edge present



Hold time :

It is the minimum amount of time input must available (hold) after active clock edge present.



Propagation delay:

It is the amount of time required to produce valid outputs after accepting input.

Pulse width:

It is the active "high duration" of the clock.

duty cycle = $(t_{pw}/T) * 100\%$ ⇒ $T_{on} = T_{OFF}$ then duty cycle = 50%

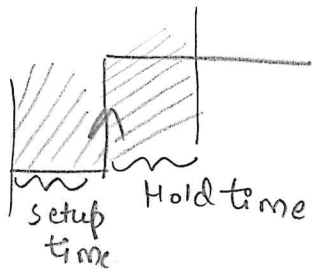
and clock is called square wave signal.

(Q) If T = 100 ns and duty cycles = 30% then what is t_{PW}?

⇒ $\because t_{PW}$ = 30% of T

= 0.03 × 100 = 30 ns

NOTE:

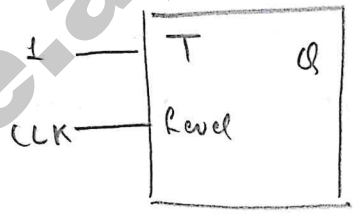


+ve edge triggering.

Race Condition:

For a level triggered T-FF, with T = 1 and pulse width (t_{PW}) greater than propagation delay (t_{Pd}). Then FF produces unpredictable output as it undergoes unpredicatable no. of toggles.

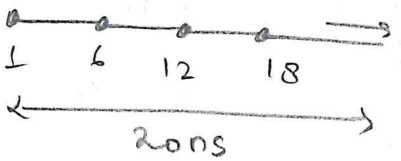
unpredictable o/p from JK FF if J=K=1 with level triggering



case1: t_{PW} = 20ns

t_{Pd} = 6ns

(complements after every 6ns)



(in 20ns it went for 4 times Complementation ∴ $Q_n = \bar{Q}$)

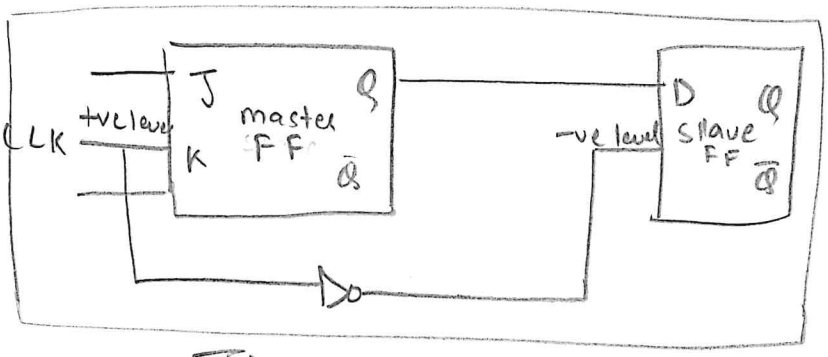
Case2: t_{PW} = 20ns

t_{Pd} = 8ns

in 20ns, it is complementing 3 time and thus $Q_n = \bar{Q}$

One of the solution to overcome race condition is JK master slave Flip flop.

It is composed of JK and D-FF cascaded such that the o/p of JK is given as i/p to D-FF. Complementary clocks are connected to flip flops.



JK- master slave FF

J	K	Q_n
0	0	Q
0	1	0
1	0	1
1	1	$\bar{Q}$

At +ve level only master FF is active and ∴ Slave FF will copy the o/p of master FF ($Q_n = Q$) while at -ve level slave FF is active and ∴ master remain same (it maintain $Q_n = Q$).

8.12 Memory Construction:

Memory is divided into words and bits. Addressable unit is word. All bits are accessed at a time.

Ex – Let $2^m \times n$ bit memory

This memory contains 2^m words and each word is having n-bits.

NOTE: Address bus size (in bits) = $\lceil \log_2 2^m \rceil = m$ bits

Data bus size n-bits.

(Q) Consider 4GB memory is organized into 4- designs having word size 8-bits, 16-bits, 32-bits and 64-bits. Compute Address bus and Data bus sizes each case

⇒ Designed : 4G x 8 bits = 4GB

$= \log_2 4 \times 2^{30} = \log_2 2^{32}$

⇒ 32-bits (A_{31} to A_0)

Data bus size = 8 bits

} 4 Giga words ⇒ 8 bits/word, 4G words = 4×2^{30} words

Design 2: 16-bit words

4GB = 4G x 8 = 2G x 16

address bus size = $\log_2 2G$

$= \log_2 2 \times 2^{30}$

$= \log_2 2^{31}$

= 31 bits

Data bus size = 16 bits

Design 3: 32-bit words

4GB = 4G x 8 bits

= 1G x 32 bits

Address bus size ⇒ $\log_2 1G = \log_2 2^{30}$

= 30 bits

Data bus size = 32 bits

Design 4: 64-bit words

Address bus size

$64GB = 64 \times 8$ bits

$= 1G \times 32$ bits

$= \frac{1}{2}G \times 64$ bits

$\log_2 \frac{1}{2}G = \log_2 2^{30}/2$

$= \log_2 2^{29}$

$= 29$ bits

Data bus size = 64 bits //

8.13 Memory Expansion:

Memory expansion is a process of constructing higher capacity memory with smaller size memory chips arranged in multiple levels.

Let $P \times Q$ ($\rightarrow$ P words x Q bits) memory is used/need and basic memory chips of each size $M \times N$ (M words x N bits)

$$\# \text{ Total chips} = \frac{\text{Desired size}}{\text{Basic chip size}}$$

$$= \frac{P \times Q}{M \times N}$$

$M < P$: Expansion

Ex- How many 1K x 4 memory chips are required to Construct 2KB memory?

⇒ # chips = $\frac{2KB}{1K \times 4} = \frac{2K \times 8}{1K \times 4}$ = 4 chips

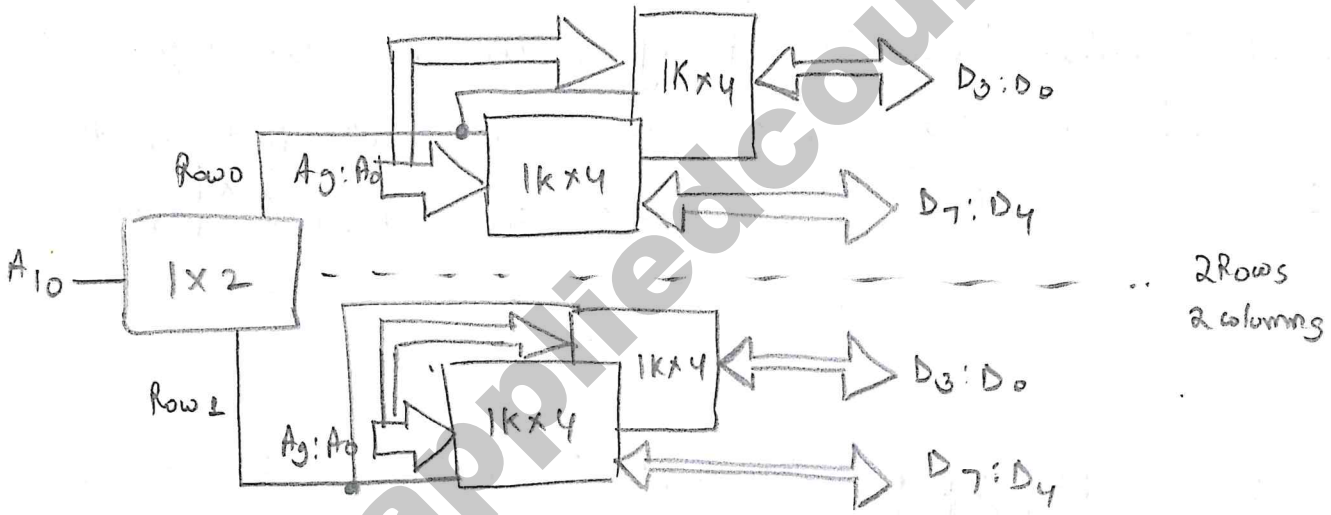
4 chips are arranged in 2-rows and 2-columns

Design with interfacing diagram

2K x 8 memory

Address lits = $Log_2 2^K$ = 11 bits (A_{10}-A_0)

Data bits = 8 bits.



(here we need 1 → 1x2 decoder and 4 → 1K x 4 chips)

(Q) 1K x 4 memories organised in 8 rows and 4 columns what is the size address bus for the target memory?

⇒ # words = 8K (~~8K x 1~~ 8 x 1K)

Address bus size = $Log_2 8K = log_2 2^{13}$ = 13 bit

Data bus size = 4 x 4 = 16 bits/words

∴ Target memory = 8K x 16 //. ⇒ #chips = $\frac{8K \times 16}{1K \times 4}$

These 32 chips are arranged in 8 rows and 4 columns
Hence, 3X8 decoder is needed.

$A_1 : A_0$ → given to chips

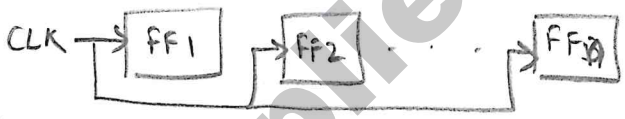
$A_{12} - A_{10}$ → given to decoder.

9 COUNTERS :

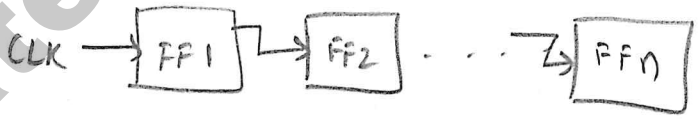
9.1 Basics of counter :-

Counters are used to provide accurate timing or control signals. Basic element of counter is Flip flop.

* All flip flops responds to the same clock instance then counters are called Synchronous Counters. These are faster.



* If one flip flop provide Clock Signal to another flipflop then counters are called as asynchronous Counters. ("UP/Down" Counter)



* In a counter, the states in the counting path (loop) are called as Counting states. While the states which are outside the counting path (loop) is called non-counting state.

* In a counter, the number of states in the counting path (loop) is called modulous of counter.

* For a given counter, if all possible states are in counting path then it is called free running counter.

* For a given counter, irrespective of initial value if counter progresses such counter is called self-starting counter.

* For a counter, if modulous is K, then clock period of counter is given as

$$T_{clock} = \frac{T_{counter}}{K}$$

NOTE -

Clock design $\Rightarrow T_{clock} \geq T_{pd}$

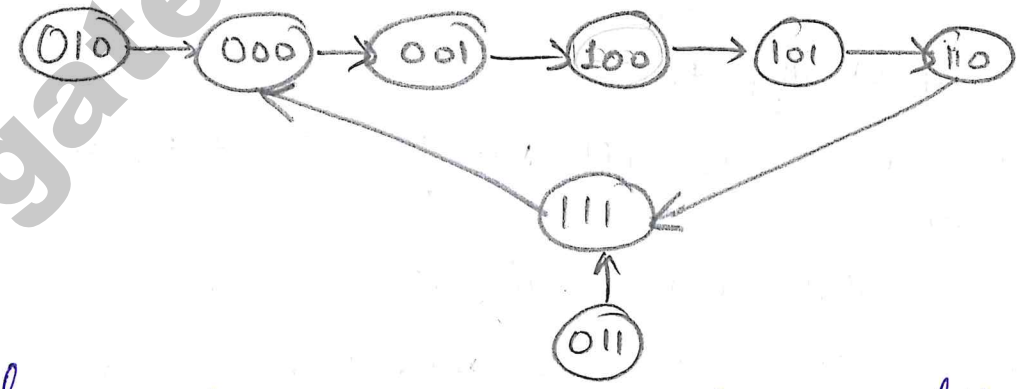
Synchronous counter: (faster than asynchronous counter)

$$T_{clock} = T_{FF} + T_{comb}$$

Asynchronous counter:

$$T_{clock} = n\,T_{FF} + T_{comb}$$

Ex -

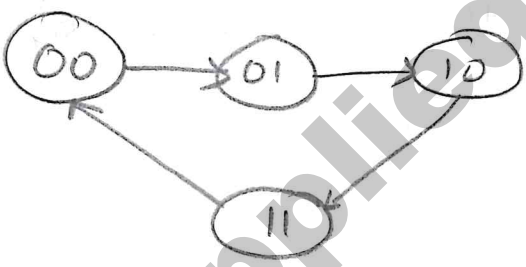


- Since 3 bits are associated ∴ it is a 3-bit counter
- Since in a loop, there are only 6 states ∴ it is a MOD-6 counter. ie modulas = 6

and, these ~~are~~ states are counting states.

- Since there are 6 states in the counting path (loop)
  Therefore, Counting states = 6
  non-counting states = 2 (010, 011)
- Since all states are not the part of counting path.
  Therefore, it is not a free running counter
- It is self starting as with any initial state, we can enter into the counting path.
  ∴ minimal clock delay = 1 clock
  (max time req to move (path length) from non-counting state to counting state)

Ex 2 -



⇒ The given counter is modulus 4 counter as there are 4 states in the counting path.

→ It is self starting

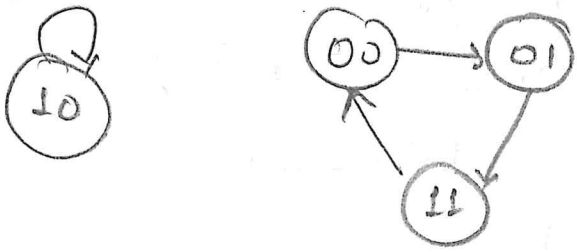
→ It is free running as all the possible states are in counting path.

→ Sequence generated with initial state (00) are

(i) 00 → 11 → 00 → 11 ---→ max length of seq MLS = 4

(ii) 01 01 01 01 ----

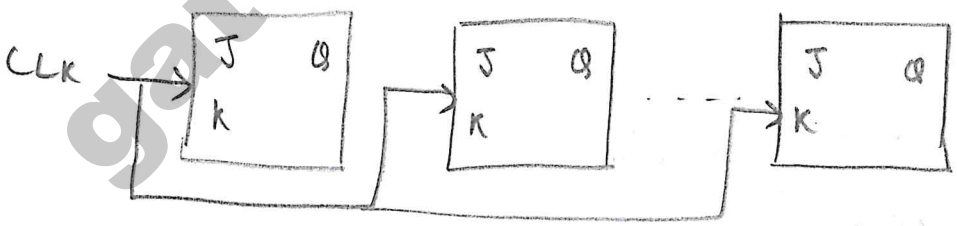
Ex 3 -



- here there are 3 states in counting path, therefore, it is Mod-3 counter
- It is not free running
- It is not self starting as we cannot reach to counting state from the non-counting state.

9.2 Synchronous Counters - Design issues and Simplified version (Shift counter, Ring counter, Jhonson counter):

Synchronous counter provides fastest response and used for realtime applications. The design complexity is more than asynchronous counters.



Each i/p is to be expressed as a function of outputs of all FF's

$$I_i = F(Q_{n-1}, Q_{n-2} \cdots Q_1, Q_0)$$

$$0 \le i \le n-1$$

Simplified versions:

(i) Shift counter: (Data is shifted from previous FF to current FF)

$$I_0 = f(Q_{n-1}, Q_{n-2}, \cdots Q_1, Q_0)$$

$$I_i = Q_{i-1} \qquad 1 \le i \le n-1$$

$(I_1 = Q_0,\ I_2 = Q_1)$

(ii) Ring counter: (The relation of i/p and o/p forms ring)

$$I_0 = Q_{n-1} \ ; \ I_i = Q_{i-1} \quad 1 \le i \le n-1$$

(iii) Jhonson or twisted ring counter (Complement of last o/p is i/p to first FF)

$$I_0 = \overline{Q_{n-1}} \ ; \ I_i = Q_{i-1} \quad 1 \le i \le n-1$$

→ With n-FF Shift counter provide 2^n maximum modulas with 'Ex-OR' maximum modulas is $2^n - 1$

→ With n-FF Ring counter can provide modulas as n.

→ With n-FF Jhonson or Twisted ring counter provide max modulas as 2n. Jhonson counter provide grey counting.

Ex- for, 8-FF ⇒

Counter	max modulas
Shift	$2^8 - 1 = 255$
Ring	8
Jhonson	$2 \times 8 = 16$

Drawback: Main problem of synchronous counter are clock skew, more design complexity.

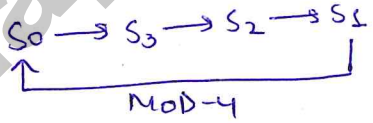
Procedure:

1. For a given state diagram, identify no. and type of FF's to be used.
2. Construct state table and replace the associate next state using excitation table of given FF.
3. Get i/p expressions for all FF i/p's and realize them.

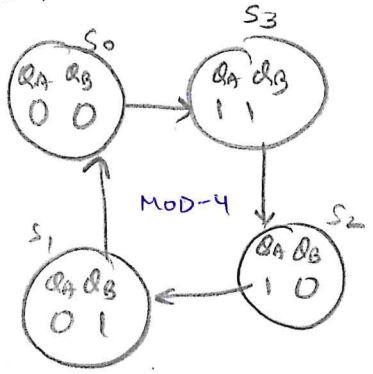
9.3 Synchronous Counters: Design procedure:

Example: Let the two state variables as Q_A Q_B and decimal value of state $S_i = 2*Q_A + Q_B$.

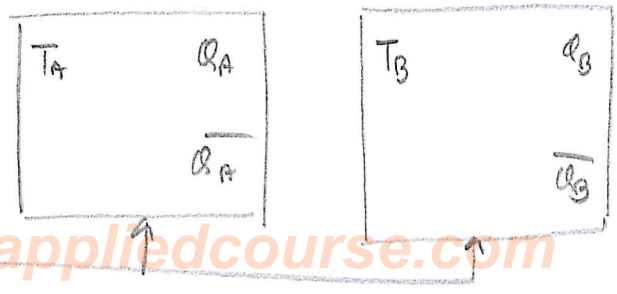
⇒ The given counter is MOD-4 counter



⇒



The Mod-4 is realized with T-FF's



$T_A = F(Q_A, Q_B)$

$T_B = F(Q_A, Q_B)$

Present state		Next state			
Q_A	Q_B	Q_{AN}	Q_{BN}	T_A	T_B
0	0	1	1	1	1
0	1	0	0	0	1
1	0	0	1	1	1
1	1	1	0	0	1

Q_{AN} is replaced with T_A using its excitation table

$\therefore \Delta Q_{AN} \neq 0 \Rightarrow T_A = 1$

else $T_A = 0$

$\therefore T_A = F(Q_A, Q_B) = \overline{Q_A}\,\overline{Q_B} + Q_A\overline{Q_B}$

$= \overline{Q_B}(Q_A + \overline{Q_A})$

$= \overline{Q_B}$

Similarly, Q_{BN} is replaced with T_B using its excitation table.

ie $\Delta Q_{BN} \neq 0 \Rightarrow T_B = 1$

else $T_B = 0$

$T_B = F(Q_A, Q_B) = \overline{Q_A}\,\overline{Q_B} + \overline{Q_A}Q_B + Q_A\overline{Q_B} + Q_AQ_B$

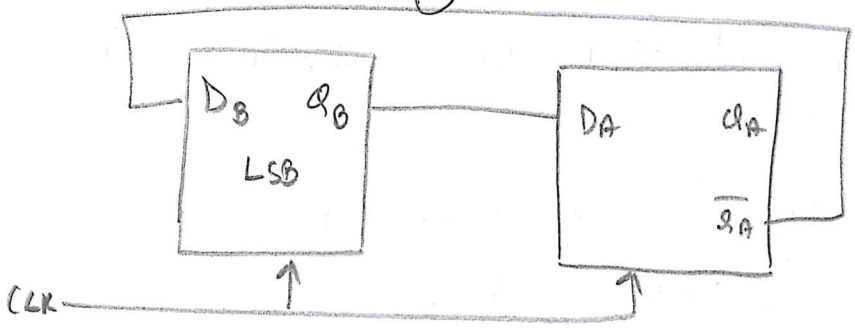
$= 1$

$\therefore T_A = \overline{Q_B}$

$T_B = 1$

Example 2:

Let the following synchronous counter



(i) What is the counting sequence ($Q_A Q_B$)?

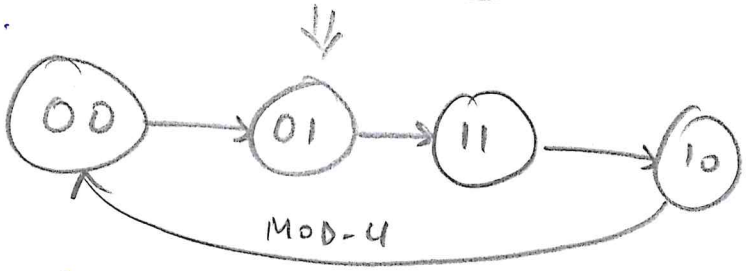
(ii) What is the sequence generated by FF Q_B with initial state $Q_A Q_B$ as 01 is?

$\Rightarrow$ From the ~~deg~~ design we can get state table and state diagram.

here, $Q_{BN} = D_B = \overline{Q_A}$

$Q_{AN} = D_A = Q_B$

Present state		Next state	
Q_A	Q_B	Q_{AN}	Q_{BN}
0	0	0	1
0	1	1'	1
1	0	0	0
1	1	1	0



(i) ∴ Counting seq is 00 → 01 → 11 → 10 → 00

(ii) at 01, the seq will be 1100 1100 1100 ...

(Q) For the previous obtained state diagram, if initial state is 01 after 183 clocks what is the state of counter.

⇒ Since after every 4th clock, the counter is at its initial state

∴ 183 ÷ 4 = 3 Clock (remainder)

∴ State will be 00

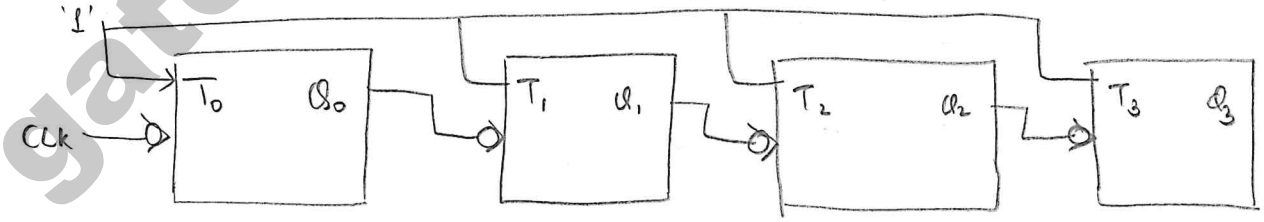
9.4 Asynchronous counters : Ripple counter:

→ In asynchronous counter, O/P of one of the flipflop give the clock i/p to next FF.

→ The clock move as a ripple from one FF to another. Hence called as Ripple counter

→ Natural Counting is binary up counting and basic flipflop is T-FF (operated in toggle mode)

→ Design complexity is less than Synchronous counter but provide slow response.

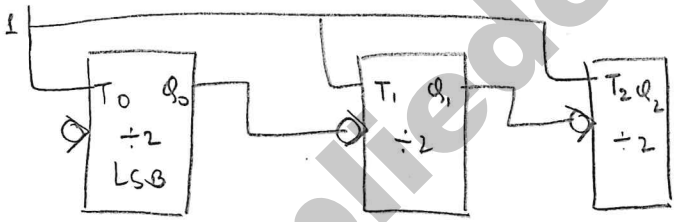


$T_{clock} \geq 4 * FF\ delay$

Clock edge	FF o/p used for clock	Nature of counting
-ve	True (Q)	Binary UP
+ve	False ($\bar{Q}$)	Binary UP
-ve	False ($\bar{Q}$)	Binary Down
+ve	True (Q)	Binary Down

if, Edge and output clock position matches - Down
else - up counting.

Analysis:



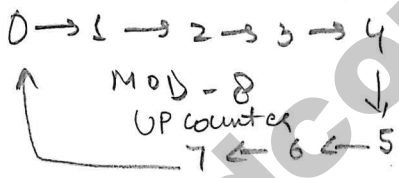
$2^3 = 8$, Mod-8 Counter.

LSB flipflop assumed to have valid clock always.
Hence $Q_n = Q_0$

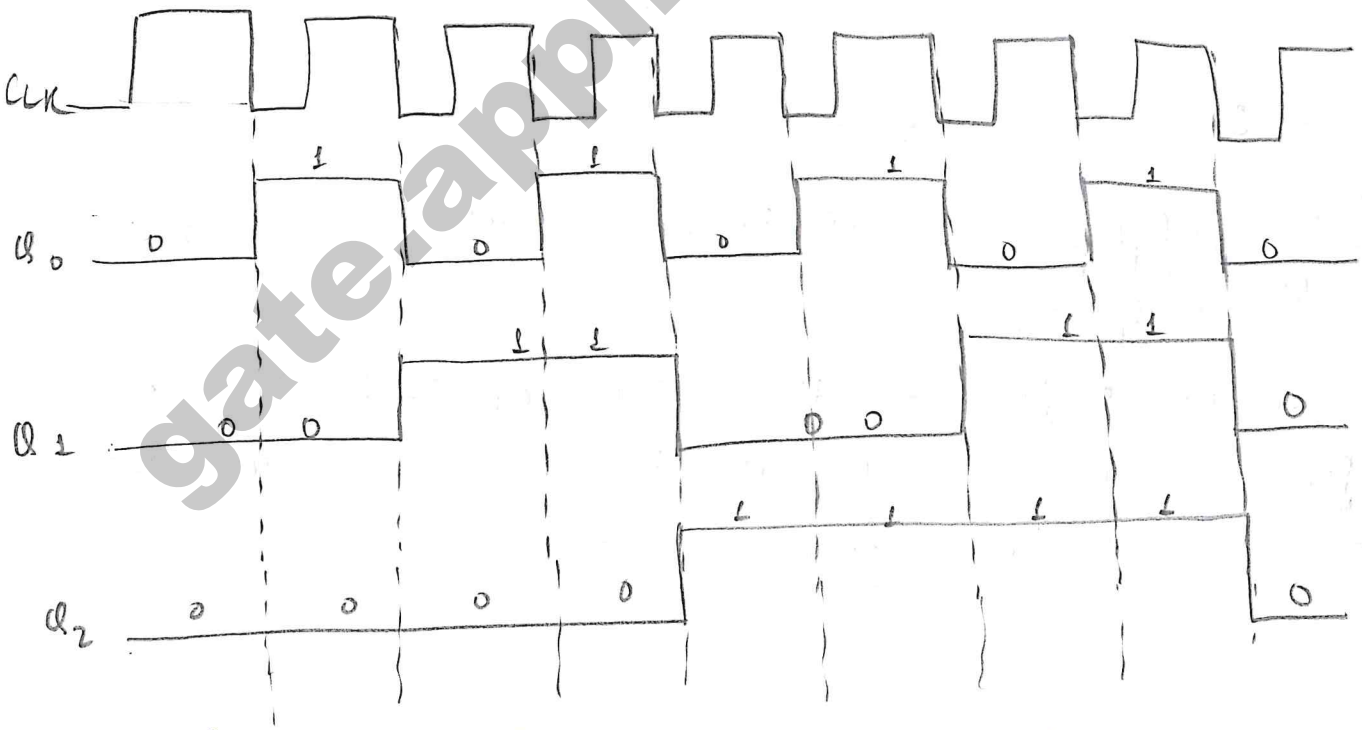
Other FF complement their o/p iff clock changes from $1 \rightarrow 0$ ($Q_{i-1} : 1 \rightarrow 0$). For all other values of clock FF retain it previous o/p

Previous state			Next state		
Q_2	Q_1	Q_0	Q_{2n}	Q_{1n}	Q_{0n}
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

→ valid change



Timing signals:



$t_0 = 2t_{CLK}$; $t_1 = 2t_0$; $t_2 = 2t_1$

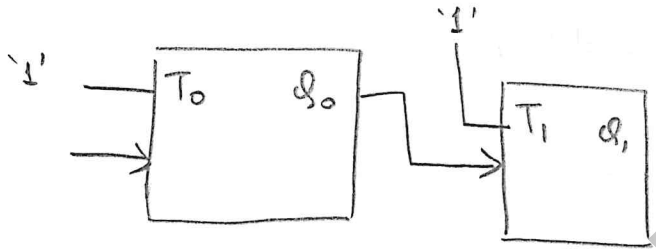
LSB FF acts as $\div 2$ (MOD-2), middle one $\div 4$ (MOD-4) and last FF acts as $\div 8$ (MOD-8)

$f_0 = f_{clk}/2$; $t_0 = 2t_{clk}$

$f_1 = f_0/2$; $t_1 = 2t_0 = 4t_{clk}$

$f_2 = f_1/2$; $t_2 = 2t_1 = 8t_{clk}$

(8) If $Q_1Q_0 = 01$, the state of ~~sss~~ counter for next state 3 clocks is



(a) 01, 10, 11

(b) 10, 11, 00

(c) 00, 11, 10

(d) 01, 00, 11

⇒ ∵ +ve edge and true o/p for CLK ⇒ Down counting the given counter is MOD-4 Down counter.

$11 \longrightarrow 10 \longrightarrow 01 \longrightarrow 00$

∴ answer is (c)

(8) Given the counting seq below, if initial state is 11, what will be the state after 113 clocks.

$01 \longrightarrow 00 \longrightarrow 11 \longrightarrow 10$ ↑

⇒ 113 ÷ 4 = 1 as remainder

∴ after 1 clock of 11 we will get 10

(Q) If each FF delay is 25 ns. What is the minimum clock delay?

⇒ ∵ $T_{CLK} >= 3 * T_{FF}$ delay

$T_{CLK} \geq 3 * 25$ ns

≥ 75 ns

∴ minimum delay = 75 ns

(Q) What is the max usable clock frequency? (for above question)

⇒ ∵ $F_{CLK\ max} = \frac{1}{T_{CLK\ min}}$

$= \frac{1}{75 \text{ ns/clock}} = \frac{10^9 \text{ sec}}{75}$ clocks/sec

$= \frac{1000}{75} \times 10^6$ Hz

$= \frac{40}{3}$ MHz

$= 13.33$ MHz //

9.5 Gate Questions on Counters:

2017

(Q) The next state table of 2-bit saturating up-counter is given below

Q_1	Q_0	Q_{1n}	Q_{0n}
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	1

The counter is built as a synchronous sequential circuit using T-FF. The expression for T_1 and T_0 are

(a) $T_1 = Q_1 Q_0$, $T_0 = \overline{Q_1}\,\overline{Q_0}$

(b) $T_1 = \overline{Q_1} Q_0$, $T_0 = \overline{Q_1} + \overline{Q_0}$

(c) $T_1 = Q_1 + Q_0$, $T_0 = \overline{Q_1}\,\overline{Q_0}$

(d) $T_1 = \overline{Q_1}\, Q_0$, $T_0 = Q_1 + Q_0$

⇒

$T_1 : Q_1 \rightarrow Q_1^+$

$T_0 : Q_0 \rightarrow Q_0^+$

$\therefore \Delta Q \neq 0 \Rightarrow T = 1$

else $T = 0$

∴ excitation table will be:

Q_1	Q_0	Q_{1n}	Q_{0n}	T_1	T_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	0	0

$T_1 = F(Q_1, Q_0) = (\overline{Q_1} Q_0)$

$T_0 = F(Q_1, Q_0) = (\overline{Q_1} + \overline{Q_0})$

∴ (b) is the correct option.

2016
(Q) We want to design a synchronous counter that counts the sequence 0–1–0–2–0–3 and then repeats. The minimum no. of JK-FF req to implement this counter is ____

⇒ ∵ there are 3 zero which ref. different states

ie 000 → 1st zero
010 → 2nd zero ← it represent state 2
100 → 3rd zero

∴ we need 4 bits (3 bits will not be enough)

0000 → 1st 0
0100 → 2nd 0
1000 → 3rd 0
0001 → 1
0010 → 2
0011 → 3

Hence, 4 bits are sufficient (No clash)

Thus 4, flipflops are required.

2015
(Q) The minimum no. of JK-FF required to construct a Synchronous counter with the count sequence (0, 0, 1, 1, 2, 2, 3, 3, 0, 0 . . .) is ____

⇒

1st 0, 2nd 0	000 → 1st zero
1st 1, 2nd 1	100 → 2nd zero
1st 2, 2nd 2	001 → 1st one
1st 3, 2nd 3	101 → 2nd one
	010 → 1st two
	110 → 2nd two
	011 → 1st three
	111 → 2nd three

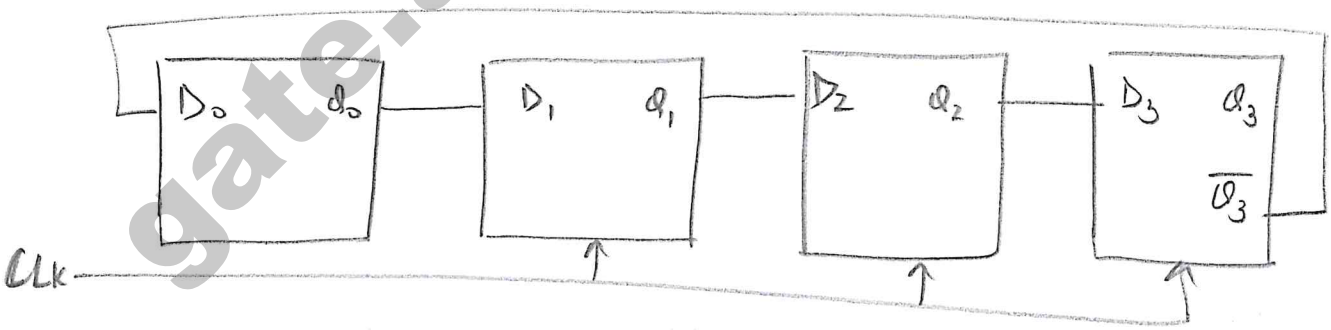
∴ no clash

∴ we need 3 bits thus 3 FF are required.

2015
(Q) Consider a 4-bit Johnson counter with an initial value as 0000. The counting seq of this counter is

(A) 0, 1, 3, 7, 15, 14, 12, 8, 0
(B) 0, 1, 3, 5, 7, 9, 11, 13, 15, 0
(C) 0, 2, 4, 6, 8, 10, 12, 14, 0
(D) 0, 8, 12, 14, 15, 7, 3, 1, 0

→ Given $Q_3 Q_2 Q_1 Q_0 = 0000$



here $Q_{0N} = D_0 = \overline{Q_3}$, $Q_{1N} = D_1 = Q_0$

$Q_{2N} = D_2 = Q_1$; $Q_{3N} = D_3 = Q_2$

∴ if $Q_3 Q_2 Q_1 Q_0$ ⇒ $Q_{3n} Q_{2n} Q_{1n} Q_{0n}$ ⇒ 0 → 3 → 7 ...

(a) 0 0 0 0 ⇒ 0 0 0 1 (1)
(1) 0 0 0 1 ⇒ 0 0 1 1 (3)

∴ (a) is correct //

9.6 Sequence Detectors

Design of a sequence detector,

→ A circuit accepts a serial bit stream X as i/p and produces a serial bit stream Z as o/p

→ Whenever the bit pattern 0110 appears in the i/p stream, it o/p's Z = 1; at all other times, Z = 0.

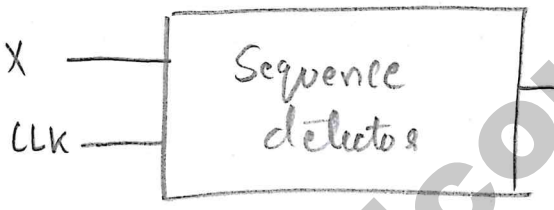
→ Overlapping occurances of the pattern are also detected.

X = 0 1 1 0 1

X = 0110
X = 1101

while non overlapping i/p

X = 0110 1

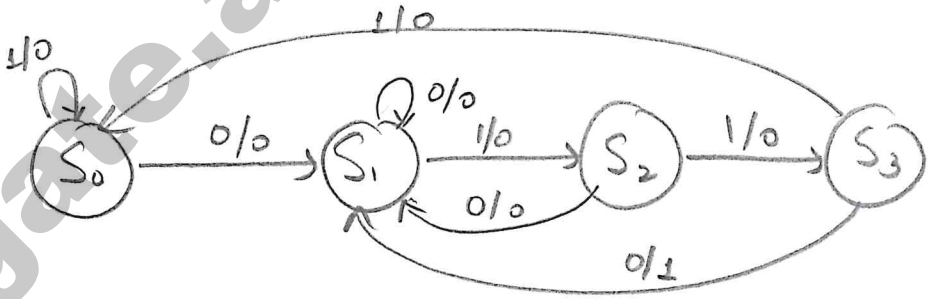


Z = 1 for valid seq.
+ve seq detector

NOTE – For seq of length n, sequence detector takes n-states.

Example:

X:	0	1	0	1	0	1	1	0	1	1	0	0	1	0	1	1	0	1	1	0
Z:	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0



PS	NS, Z	
	X=0	X=1
S0	S1, 0	S0, 0
S1	S1, 0	S2, 0
S2	S1, 0	S3, 0
S3	S1, 1	S0, 0

state assignment →

PS	NS, Z	
	X=0	X=1
00	01, 0	00, 0
01	01, 0	10, 0
10	01, 0	11, 0
11	01, 1	00, 0

split NS, Z →

PS	NS		O/P Z	
	X=0	X=1	X=0	X=1
00	01	00	0	0
01	01	10	0	0
10	01	11	0	0
11	01	00	1	0

Suppose we use T-FF

— Two FF's i/p T_1 and T_2

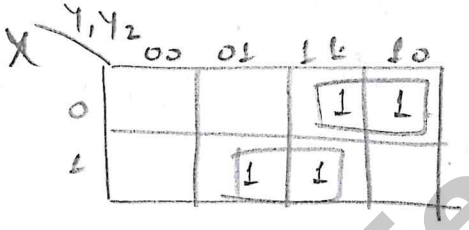
$T_1 \rightarrow y_1$, $T_2 \rightarrow y_2$ } $\Delta \neq 0$ & $T=1$ $CP \rightarrow \Phi_0$ else $T_1 = 0$

PS	NS		Output Z	
	X=0	X=1	X=0	X=1
00	01	00	0	0
01	01	10	0	0
10	01	11	0	0
11	01	00	1	0

Y_1Y_2	T_1 T_2		Z	
	X=0	X=1	X=0	X=1
00	01	00	0	0
01	00	11	0	0
10	11	01	0	0
11	10	11	0	0

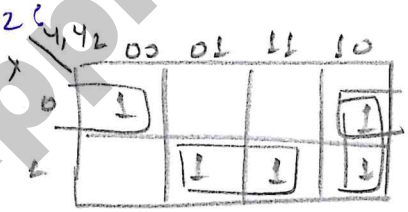
$T_1(Y_1, Y_2, X) = \Sigma(4,6,3,7)$

T_1:



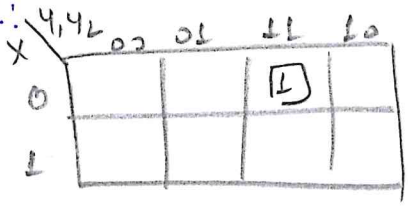
$\Rightarrow T_1 = \bar{X}Y_1 + XY_2$

Similarly for T_2:



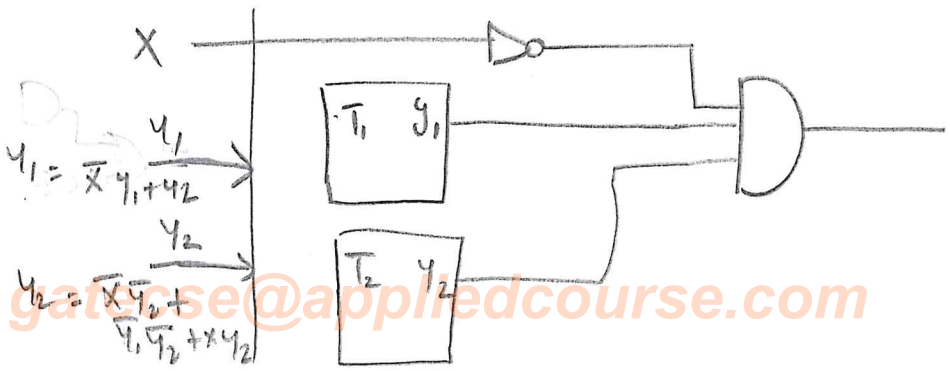
$\Rightarrow T_2 = \bar{X}\bar{Y_2} + Y_1\bar{Y_2} + XY_2$

Similarly for Z:



$\Rightarrow Z = \bar{X}Y_1Y_2$

Design will be:



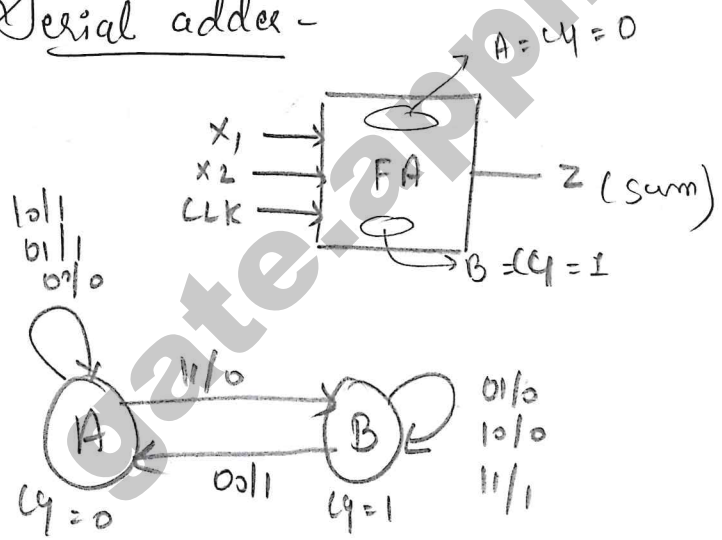
9.7 Synthesis of serial adder:

FSM Synthesis:

To construct the circuit the following steps are required

1. Assign unique binary code to each state : state assignment
2. Construct the transition / output table
3. Select type of memory elements : SR or JK or D or T and construct excitation table
4. Obtain the excitation and outputs, and minimise them.
5. Realise the functions using gates or any other combinational circuit modules.

Serial adder -



State transition diagram

x_1 A	x_2 B	PS C	O/P sum	NS ($C4_N$) carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

PS	NS, z (sum)			
	X=00	X=01	X=10	X=11
A	A, 0	A, 1	A, 1	B, 0
B	A, 1	B, 0	B, 0	B, 1

State table

Since, there are only 2 states ∴ 1 FF is sufficient

State assignment:

- Two states, and so one bit is sufficient
- Suppose we assign 0 for state A, and 1 for state B

PS	NS, Z			
	X=00	X=01	X=10	X=11
0	0,0	0,1	0,1	1,0
1	0,1	1,0	1,0	1,1

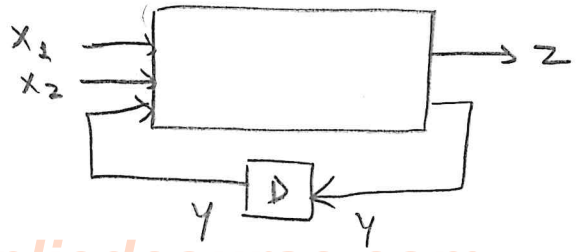
After state assignment

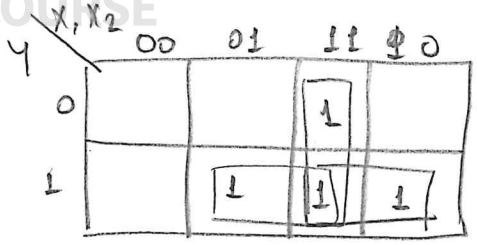
PS	NS				Output			
	00	01	10	11	00	01	10	11
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Transition/output table

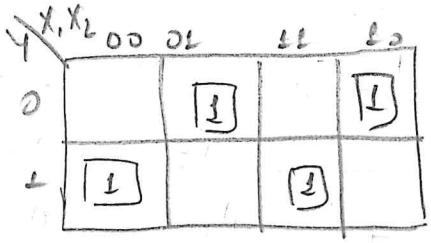
Select memory element and construct excitation/output functions

- suppose we use D-FF ($Q_n = Q = D$)





← state

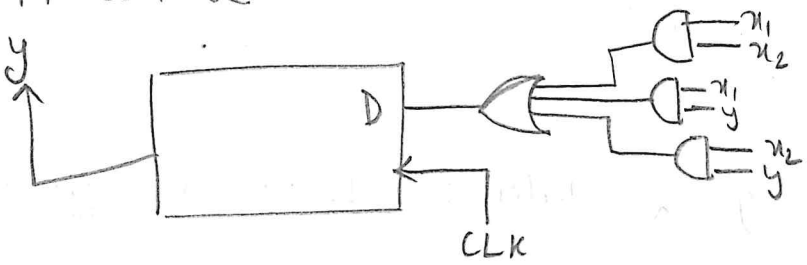


← o/p sum.

$Y = x_1x_2 + x_1y + x_2y$

$Z = x_1 \oplus x_2 \oplus y$

∴ D-FF will be



Note

Sum can be considered as state and carry can be considered as o/p