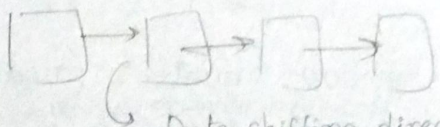
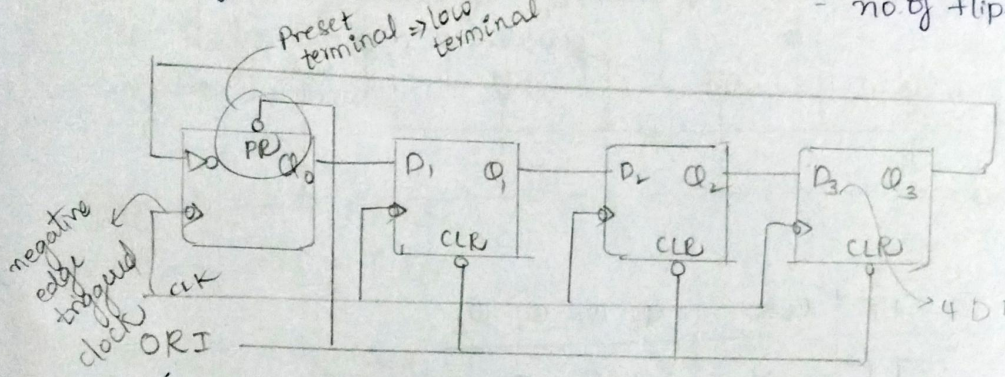
17/4/22



Ring Counter:

It is application of shift register.

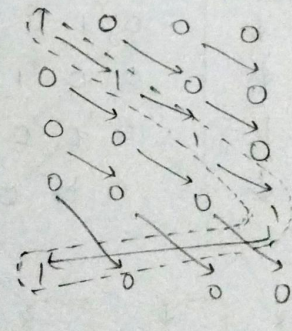
- Design Ring counter using shift count register
- n bit total states are 2^n → for normal counter
- Ring counter ⇒ No of seats states = no of bits
  = no of flipflops (F/F)

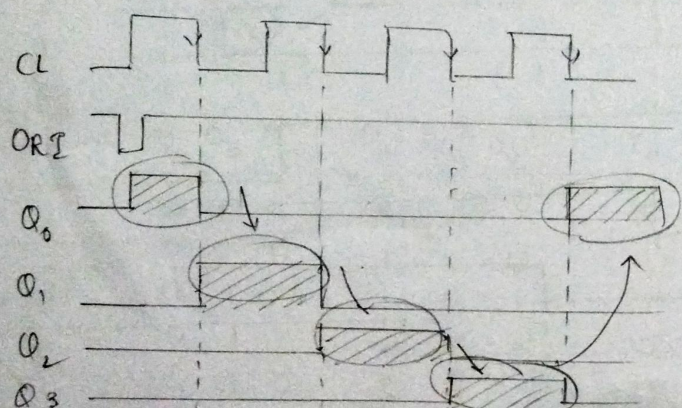


Over riding input
To initiate 1st state, we provide ORI

if PR=0, Q=1
if CLR=0 Q=0 → irrespective of other i/p's

ORI	CLRK	Q_0 Q_1 Q_2 Q_3
⊓	×	1 0 0 0
1	↓ (edge triggering)	0 1 0 0
1	↓	0 0 1 0
1	↓	0 0 0 1
1	↓	1 0 0 0



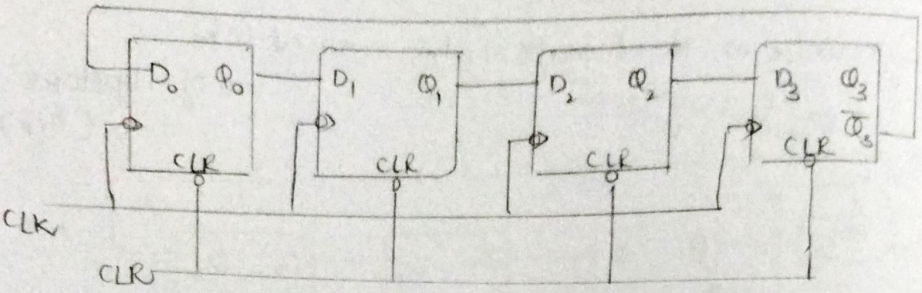


We don't need OR

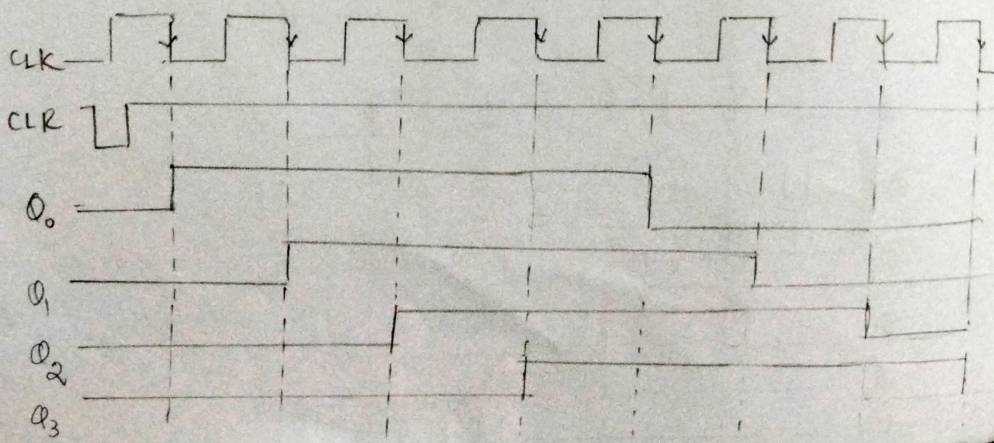
Johnson's Counter: (Twisted ring counter)

Switch Tail Ring counter

- It is application of shift register
- No. of states = 2 × No of
  = 2 × No. of bits



CLR	CLK	Q_0 Q_1 Q_2 Q_3
	×	0 0 0 $\bar{0}$
1	↓	1 0 0 0
1	↓	1 1 0 0
1	↓	1 1 1 0
1	↓	1 1 1 1
1	↓	0 1 1 1
1	↓	0 0 1 1
1	↓	0 0 0 1
1	↓	0 0 0 0



Sequence Counter:

Q. Make a given arbitrary sequence counter which counts from 7, 4, 1, 6, 2, 5

Steps to design:

1. Identify no. of F/F and Type of F/F
2. Write state Table of selected F/F excitation
3. Make a state Table
4. Solve Boolean Expression
5. Draw Circuit.

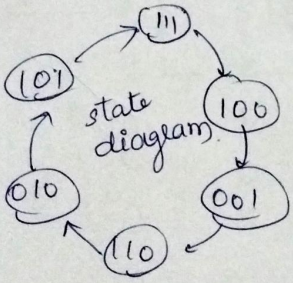
Step1: No. of F/F = 3

F/F = T F/F

Step2:

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

Step3 [7, 4, 1, 6, 2, 5]



Step4:

Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	T_A	T_B	T_C
0	0	0	x	x	x	x	x	x
0	0	1	1	1	0	1	1	1
0	1	0	1	0	1	1	1	1
0	1	1	x	x	x	x	x	x
1	0	0	0	0	1	1	0	1

0	1	1	1	0	1 0
1	1 0	0	1 0	1	0 0
1	1 1	1	0 0	0	1 1

Step5:

T_A

Q_C \ $Q_A Q_B$	00	01	11	10
0	X	1	1	1
1	1	X	0	0

$T_A = \bar{Q}_C + \bar{Q}_A$

T_B

	00	01	11	10
0	X	1	0	0
1	1	X	1	0

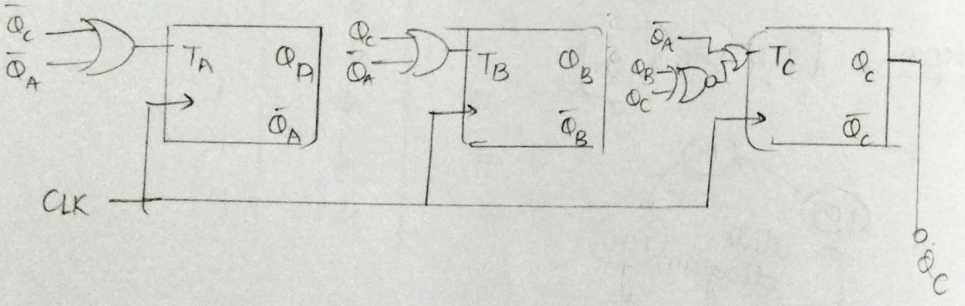
$T_B = Q_C + \bar{Q}_A$

T_C

$\bar{Q}_C$ / Q_C \ $\bar{Q}_A\bar{Q}_B$	$\bar{Q}_A\bar{Q}_B$	$\bar{Q}_A Q_B$	$Q_A Q_B$	$Q_A \bar{Q}_B$
$\bar{Q}_C$	X	1	0	1
Q_C	1	X	1	0

$T_C = \bar{Q}_A + \bar{Q}_B\bar{Q}_C + Q_B Q_C$

Step6:



Arbitrary sequence counter

- Make a counter for given sequence 0, 1, 3, 5, 6.

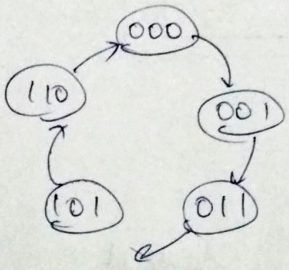
Step 1: Total bits = Total F/F = 3

F/F = T F/F

Step 2: Excitation table

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

Step 3:



Q_A	Q_B	Q_C	Q_A^+	Q_B^+	Q_C^+	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	x	x	x	x	x	x
0	1	1	1	0	1	1	1	0
1	0	0	x	x	x	x	x	x
1	0	1	1	1	0	0	1	1
1	1	0	0	0	0	1	1	0
1	1	1	x	x	x	x	x	x

Step 4:

T_A

0	x / 0	1	x
0	1	x	0

$T_A = Q_B$

T_B

0	X	1	X
1	1	X	1

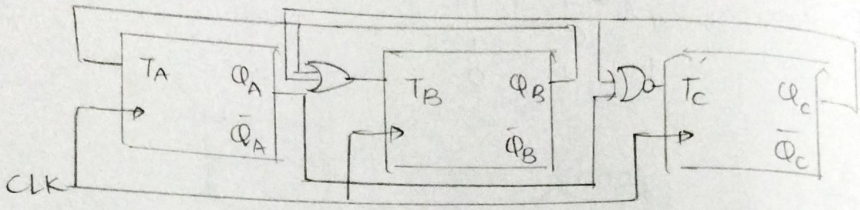
$T_B = Q_C + Q_B$

T_C

1	X	0	X
0	0	X	1

$T_C = Q_A Q_C + \bar{Q}_A \bar{Q}_C$
$= Q_A \odot Q_C$

Steps:



Sequence Generator

5 digit

- Generate sequence of 10110...
- No. of F/F can be identify by

Step 1:

$L \leq 2^n - 1$

L → length of sequence; n → no. of F/F (minimum)

$n = 3$

Step 2:

shifted by 1 bit → B; shifted again by 1 bit → C; feedback resistant → F

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

10110...

$5 \leq 2^n - 1$
$5 + 1 \leq 2^n$
$6 \leq 2^n$
$n = 3$

101 state is repeated so we have to add 1 more flipflop.

After adding 1 more flipflop check whether states are repeated or not

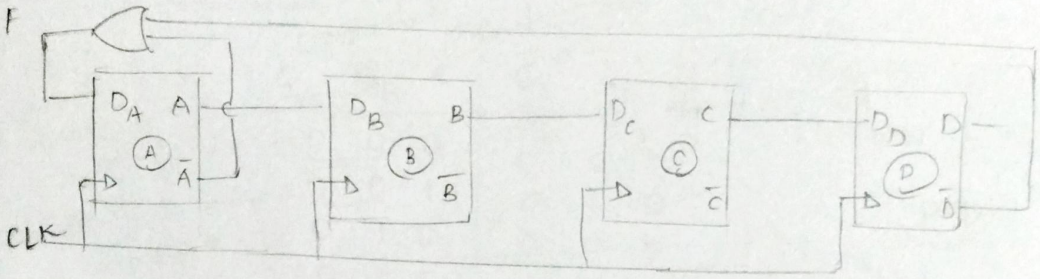
Step 3:-

F, CD \ AB	00	01	11	10
00	x	x	x	x
01	x	1	0	x
11	x	x	x	0
10	x	1	x	1

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

Step 4:

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



Note:- If any state is repeated then we have to add one more F/F.

19/4/22

Q. Generate Sequence of 110010...

No. F/F can be identify by

Step 1:

$L \leq 2^n - 1$

$6 \leq 2^n - 1$

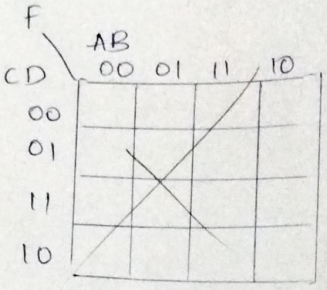
$7 \leq 2^n$

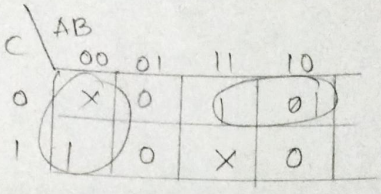
$\boxed{n = 3}$

Step 2:

A	B	C	F
1	0	1	0
1	01	0	1
0	01	1	0
0	0	1	1
1	0	0	1
0	1	0	0

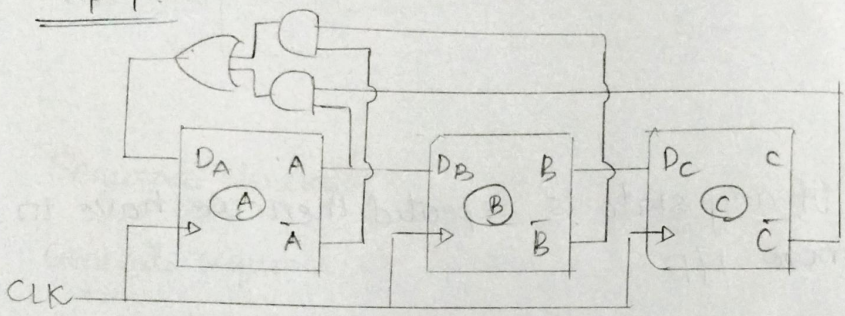
Step 3:





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

Step 4:



Sequence Detector

0. Design circuit to detect 110 from given input data stream.

If input data is

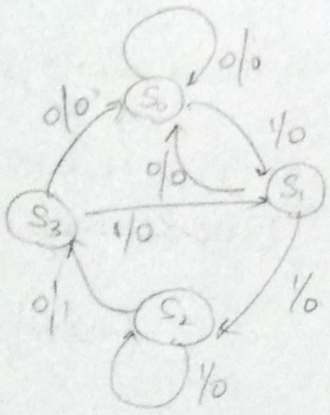
$x = 0111011010110\ldots$

$y = 0000100100001$

110 ↘

$S_0 = 0$ (Reset)
$S_1 = 1$
$S_2 = 11$
$S_3 = 1101$

$S_0 = 0$
$S_1 = 11$
$S_2 = 110.$



$S_0 = 0$

$00 \rightarrow 0$ (go to S_0)

$01 \rightarrow 0$ (go to S_1)

$S_1 = 1$

$10 \rightarrow 0$ (go to S_0)

$11 \rightarrow 0$ (go to S_2)

$S_2 = 11$

$110 \rightarrow 1$ (go to S_3)

$111 \rightarrow 0$ (go to S_2)

$S_3 = 110$

$1100 \rightarrow 0$ (go to S_0)

$1101 \rightarrow 0$ (go to S_1)

State Table :

current state Q_A	current state Q_B	i/p x	next state Q_A^+ (D_A)	next state Q_B^+ (D_B)	o/p y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	0

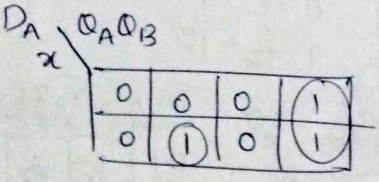
S_0 with i/p 0 = S_0 with o/p 0 S_0

S_0 with i/p 1 = S_1 with o/p 0

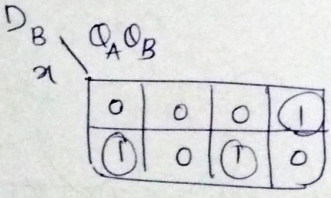
	Q_A	Q_B
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

4 states so, make circuit based on 2 F/F's

K-Map



$D_A = Q_A\bar{Q}_B + x\bar{Q}_AQ_B$



$D_B = x\bar{Q}_A\bar{Q}_B + xQ_AQ_B + \bar{x}Q_A\bar{Q}_B$

Q. Make a circuit which detect 111 from given i/p data

$x = 101111011111$

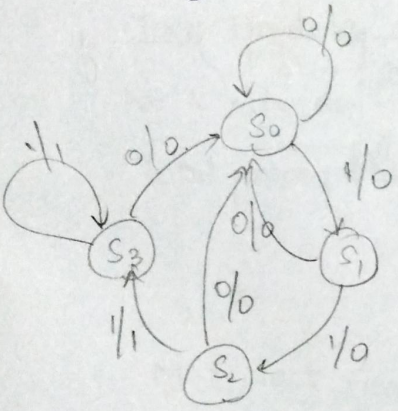
$y = 000011000111$

$S_0 = 0$ [reset]

$S_1 = 1$

$S_2 = 11$

$S_3 = 111$



$S_0 = 0$

$00 \rightarrow 0$ (go to S_0)

$01 \rightarrow 0$ (go to S_1)

$S_1 = 1$

$10 \rightarrow 0$ (go to S_0)

$11 \rightarrow 0$ (go to S_2)

$S_2 = 11$

$110 \rightarrow 0$ (go to S_0)

$111 \rightarrow 01$ (go to S_3)

$S_3 = 111$

$1110 \rightarrow 0$ (go to S_0)

$1111 \rightarrow 1$ (go to S_3)

	Q_A	Q_B
S_0	0	0
S_1	0	1
S_2	1	0
S_3	1	1

Q_A	Q_B	x	Q_A^+	Q_B^+	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	1	1	1

Q_A^+

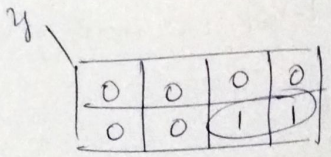
x \ Q_AQ_B	$\bar{Q}_A\bar{Q}_B$	$\bar{Q}_AQ_B$	$Q_A\bar{Q}_B$	Q_AQ_B
$\bar{x}$	0	0	0	0
x	0	1	1	1

$Q_A^+ = Q_Bx + Q_A\bar{x}$

Q_B^+

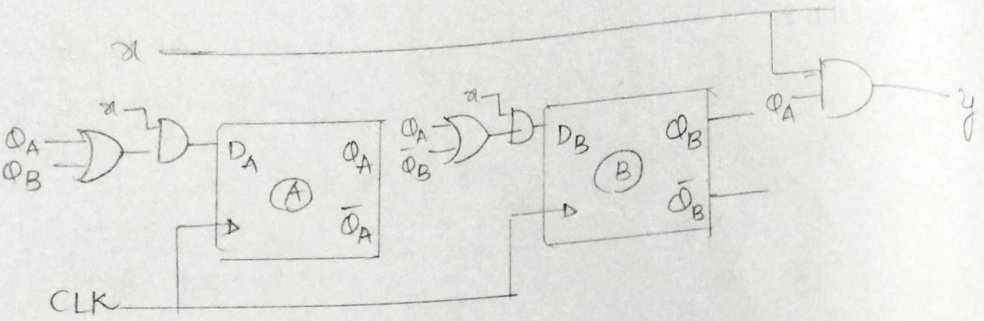
x \ Q_AQ_B	$\bar{Q}_A\bar{Q}_B$	$\bar{Q}_AQ_B$	$Q_A\bar{Q}_B$	Q_AQ_B
$\bar{x}$	0	0	0	0
x	1	0	1	1

$Q_B^+ = Q_A\bar{x} + \bar{Q}_Bx$



$y = Q_A x$

Circuit diagram



21/4/22

Example of counter

Q. If a counter is having 10 F/F, it is initially zero. What count will it hold after 2060 pulses

10 F/F → 10 bits counter

2^{10} = 1024 pulse

Why it is 10 bit counter

- After 1024 pulse holds zero count

- 10 F/F counter means, it is 10 bits counter
- It count total 2^{10} pulses with one complete round = 1024.
- Another 1024, 2048 again it will be zero count
- 2060 - 2048 = 12 pulses.

- So after 2060 pulses = 2060 - 2048
= 12
it will hold 12 count

10 bits to represent 12

↳ 0000001100 ← status of counter after 2060 pulses.

2) A 4 bits mod 16 counter is made by JK F/F. If propagation delay of each F/F is 50nsec. The maximum clock frequency can be used.

For 4 bits, we need 4 JK F/F's

- Total propagation delay = $n \times t_{pd}$
= 4×50 nsec
= 200 nsec

- Max clock frequency = $\frac{1}{p.d} = \frac{1}{200\text{nsec}}$ = 5 MHz (Mega Hertz)

3) A certain JK F/F has propagation delay of t_{Pd} = 12 nsec. The largest ripple counter that can operate at 10MHz is

Total propagation delay = $\frac{1}{f} = \frac{1}{10\text{MHz}}$ = 100nsec

$n = \frac{\text{Total propagation delay}}{t_{pd}}$

$= \frac{100}{12} \simeq 8$

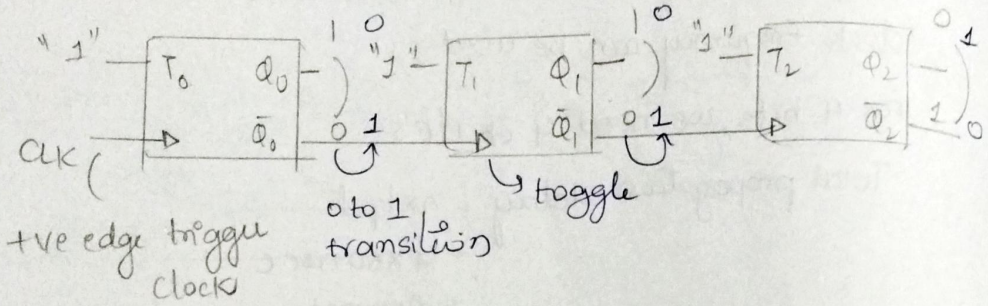
Total 8 no. of F/F which can be connected to ripple counter with frequency of 10MHz

⇒ 8 bits

With 8 bits max mod of counter = (mod) = 2^n
$= 2^8$
$= 256$

mod 256 counter

4) A given figure shows ripple counter with positive edge trigger clock. If present state of counter is given by $Q_2Q_1Q_0 = 011$, the next state will be ?



- Here we have +ve edge trigger clock.
- When transition happens from 0 to 1, there will be change in state
  - $T = 1$ → toggle our output.

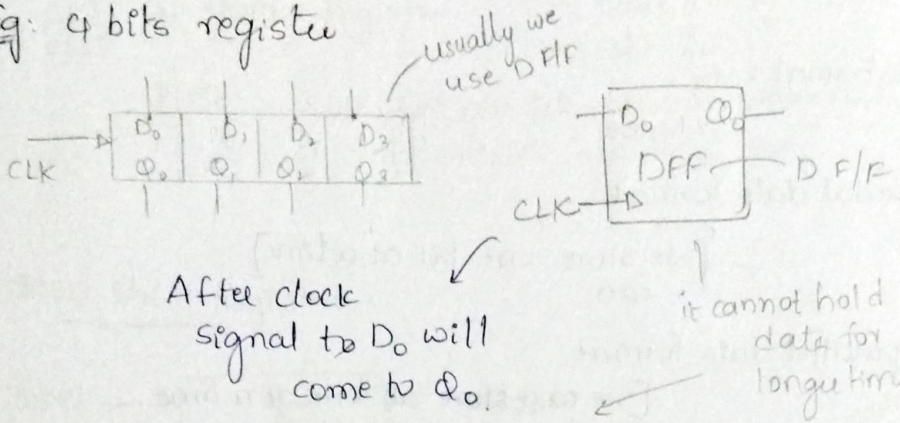
$$Q_2Q_1Q_0 = 100$$

Register :

- Flip is used to store 1 bit data
- But if you want to increase the capacity of storage, you can't F/F's, you can use register
- Group of F/F's, to increase the storage capacity is referred as Register.

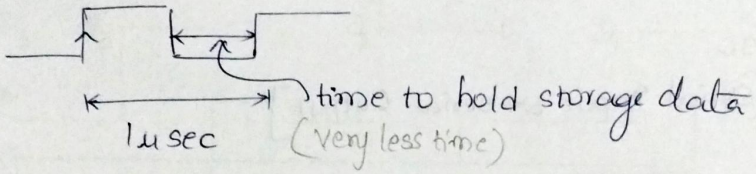
- So n bits register has n no. of F/F, this register can store n bits word by using single register

Eg: 4 bits register



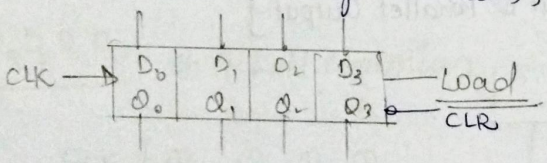
For example, $f = 1MHz$

$$T = \frac{1}{f} = \frac{1}{1MHz} = 1\mu sec.$$



Additional feature we are adding

- To hold data for longer duration, Load terminal is used.



Two types of Load.

① Synchronous load

- In synchronous load, register is operational when load & clock high.

② Asynchronous load

- In asynchronous load, register is operational when load is high.
- so when load is low, it will hold data for longer duration.

Note: Clear is active low terminal, it will make "0" output with all terminals

Data Format: → 2 types

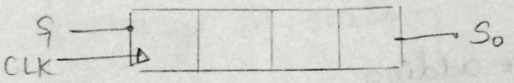
① Serial data format
[We can store one bit at a time]

② parallel data format
[we can store all bits at a time

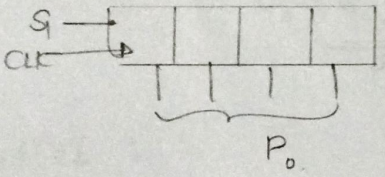
Classification of Register

-We can classify Registers based on input and output

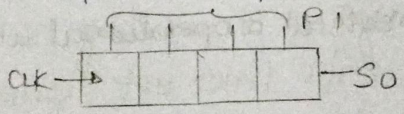
① SISO
[Serial Input & serial Output].



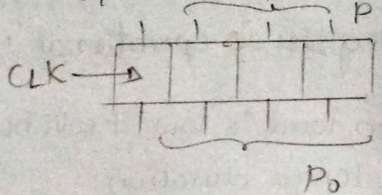
② SIPO [Serial Input & Parallel Output]



③ PISO [parallel Input & Serial Output]



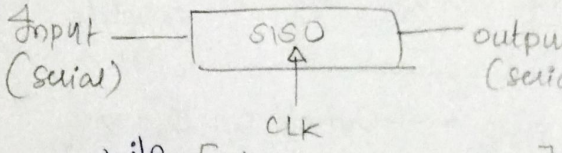
④ PIPO [parallel Input & parallel output]

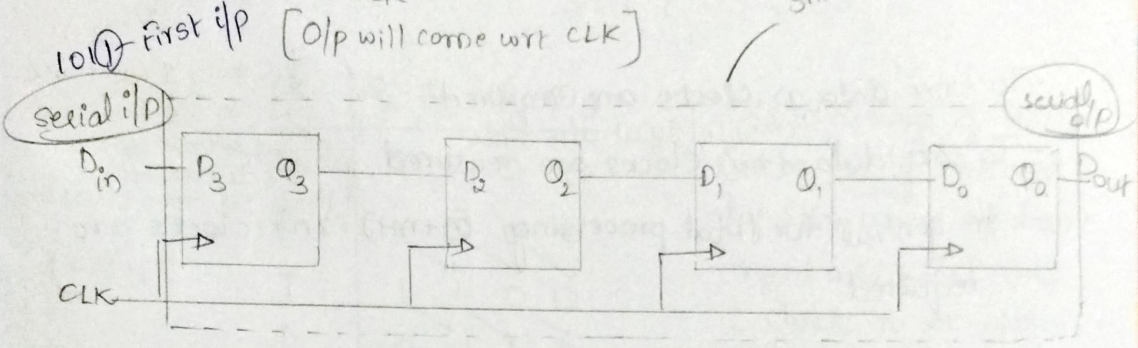


- Classifications can be done based on Applications

SISO
SIPO
PISO
① Shift Register — We shift data one by one and we take o/p
② Storage Register
PIPO — we store complete data at a time and we take complete o/p at a time

SISO Shift Register





Note:
$D_3 D_2 D_1 D_0$ ⟶ shift Right operation

$D_3 D_2 D_1 D_0$ ⟵ shift Left connection

CLK	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
↑	1	0	0	0
↑	1	1	0	0
↑	0	1	1	0
↑	1	0	1	1

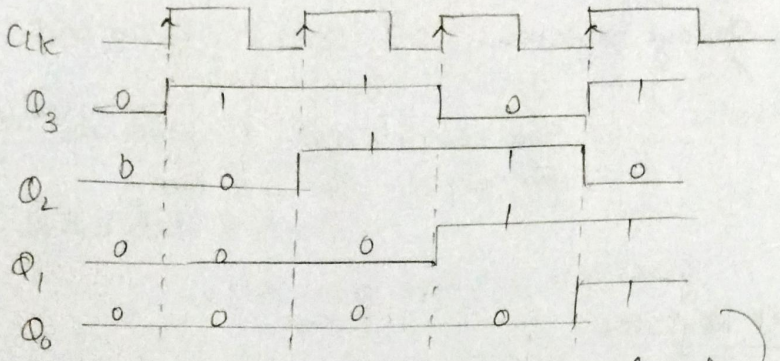
1011 first i/p
second i/p
third i/p
last i/p

4 bit shift register

After 4 clocks we store 4 bit data.
n n

- After n Clocks we store n bit data in shift Register.

But to get complete serial o/p, another n-1 clocks required

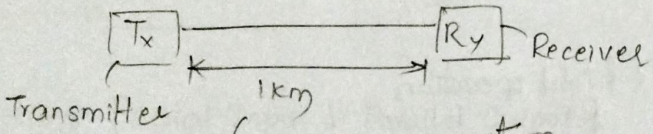


Our o/p is Q_0 - only.

1011

- To store data n clocks are required.
- To take data completely at o/p side (n-1) clocks are required.
- In general, for total processing (n+n-1) = 2n-1 clocks are required.

(like only 1 will come first in o/p side, but if we want o/p at o/p side completely then (n-1) more clocks are required)



(connection

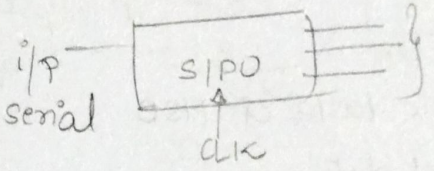
- for long distance we use this
- In serial connection, we need only one wire connection
- If parallel, we need 4 parallel wires, In this case cost of metal will increase for long distance

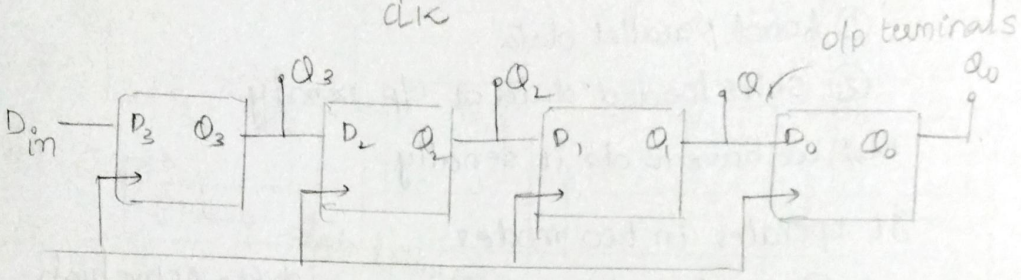
∴ For long Distance connection we use SISO.

26/4/22

SIPO - Used to convert serial data into parallel data

Serial to Parallel Converter

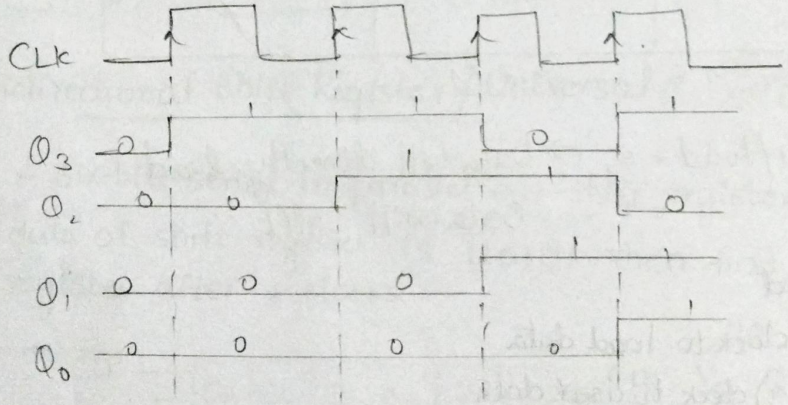




1011 → Shift Right data

CLK	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
↑	1	0	0	0
↑	1	1	0	0
↑	0	1	1	0
↑	1	0	1	1

- Disadvantage of SISO can be easily overcome by SIPO, where we didn't need additional n-1 clocks to get o/p.



o/p :- $Q_3 Q_2 Q_1 Q_0$

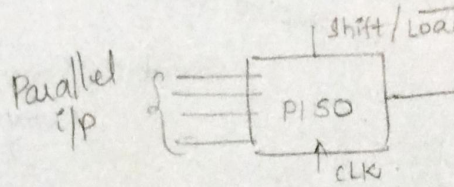
- We need, n clock for storing data
  0 clock for use of data
  n clock for total processing

In SISO $\frac{n}{n-1 \text{ use of data}} \Longleftrightarrow \frac{n}{0}$

$2n-1 \qquad\qquad n$

SIPO PISO [Storage Register / Buffer Register]

other names of PIPO (not PISO)

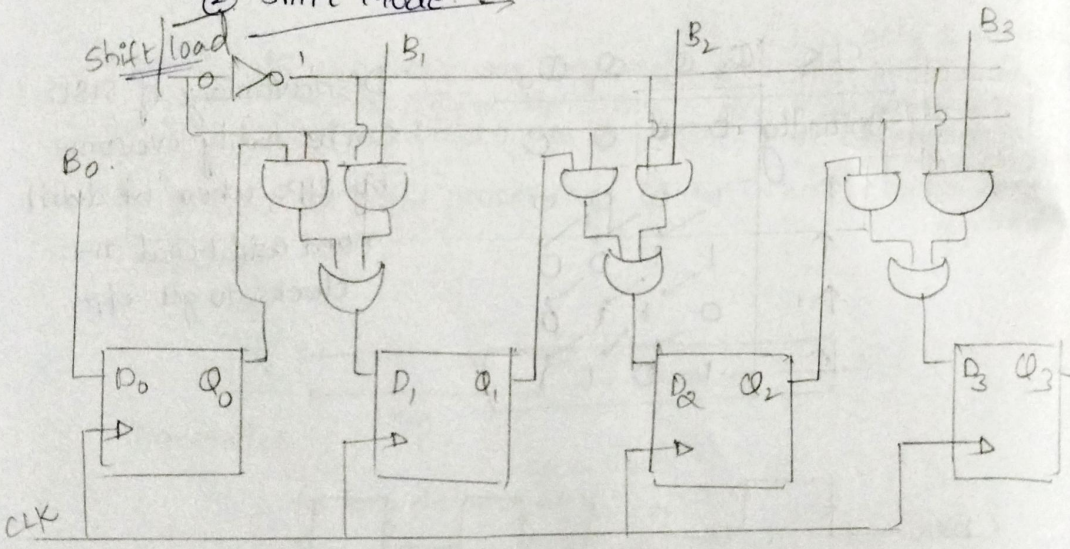


- There are two basic tasks of PISO
  1. Load parallel data
  2. Shift loaded data at o/p serially

But we have to do it serially

- It operates in two modes
  1. Load Mode = 0 ⇒ parallel data loading in D F/F's
  2. Shift Mode = 1 → terminal

shift = Active high
Load = Active low

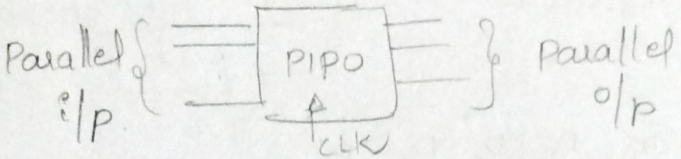


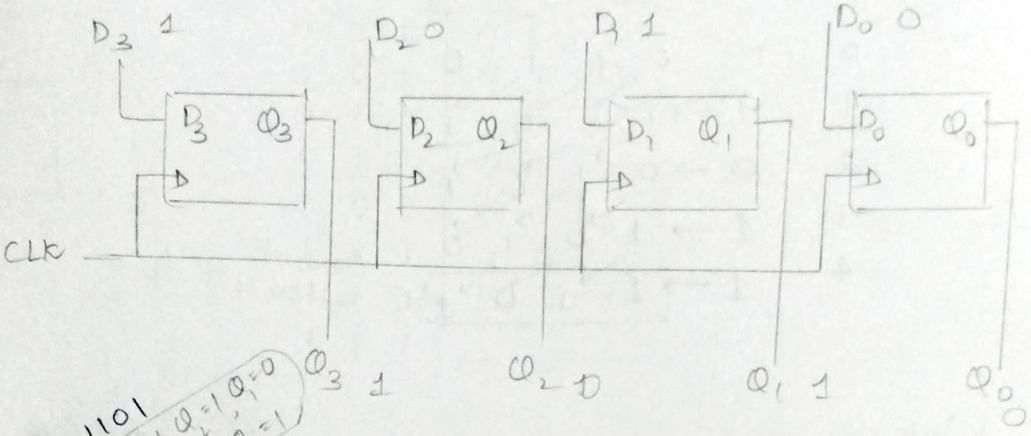
if shift/load = 0, ⇒ Data will directly load.
= 1 ⇒ Data will shift.

We need
- 1 clock to load data
- (n-1) clock to used data

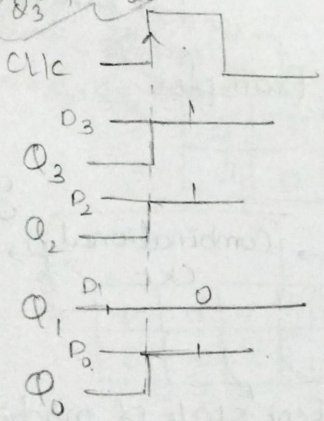
n clock is used for total processing.

PIPO





if 1101 $Q_3=1, Q_2=1, Q_1=0, Q_0=1$



we have to give i/p before clock (+ve edged)

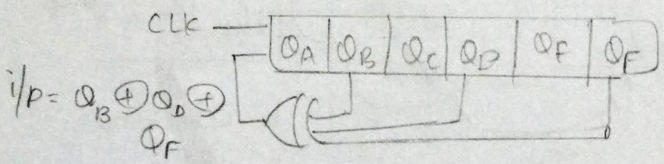
(Buffer register

- we need 1 clock to store data
- 0 clock to need use the data
- 1 clock is used for total processing.

Bidirectional Shift Register & Universal ⇒ Not taught

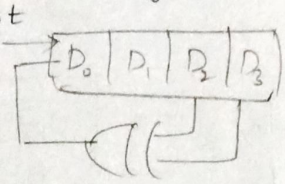
Examples based on shift register

Q1- A Six-bits serial in parallel out shift register, If initial data of shift register is 110101. Then find o/p of shift register after 3 clocks.



CLK	i/p	Q_A	Q_B	Q_C	Q_D	Q_E	Q_F
0	–	1	1	0	1	0	1
1	1 →	1	1	1	0	1	0
2	1 →	1	1	1	1	0	1
3	1 →	1	1	1	1	1	0

Q2. The initial content of A 4-bit serial in parallel shift register is 0110. After 4 clock pulses. Content of shift register is



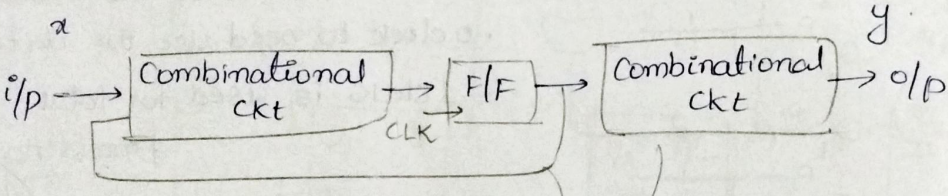
CLK	I_n	D_0	D_1	D_2	D_3
0	-	0	1	1	0
1	1 →	1	0	1	1
2	0 →	0	1	0	1
3	1 →	1	0	1	0
4	1 →	1	1	0	1

Content after 4th clock

28/4/22

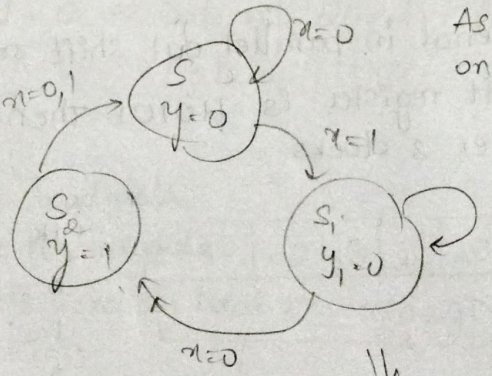
Moore state Machine and its Examples

we don't have don't care condition in Moore either dash/0/1



- Output y only depends on present state of machine
- Next state of Moore machine depends on present state and i/p x.

$S_0 = 00$
$S_1 = 01$
$S_2 = 10$



As y only depends on present state of m/c, so, o/p is written along with the state

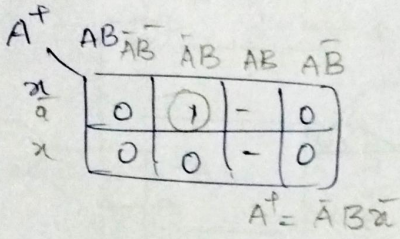
Example Question
This diagram will be mentioned in the Question itself

- In Moore state machine (diagram) along with state o/p should be mentioned as o/p depends on only present state.

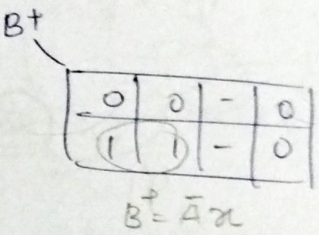
	A	B
S_0	0	0
S_1	0	1
S_2	1	0

A	B	x	A^+	B^+	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	-	-	-
1	1	1	-	-	-

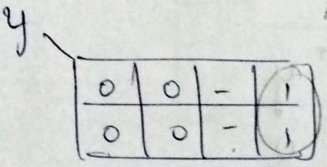
k-Map



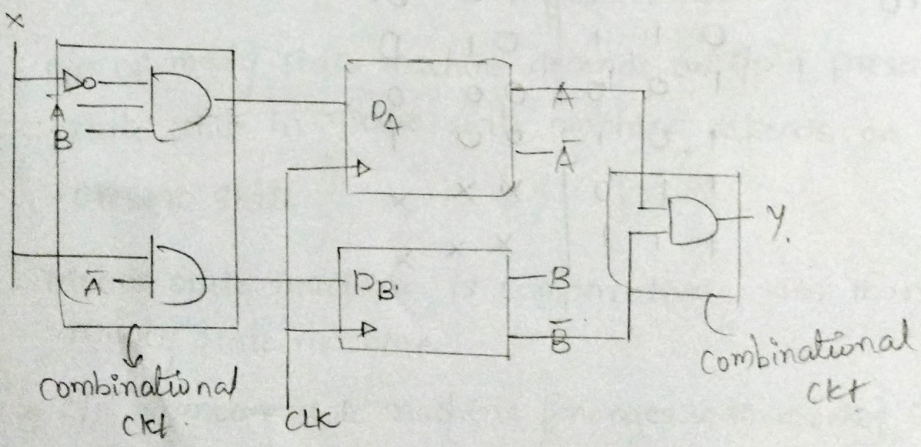
$A^+ = \bar{A}Bx$



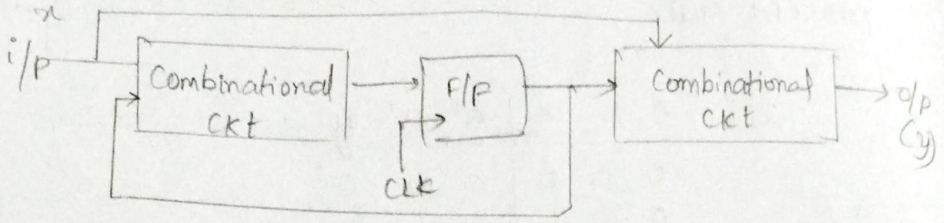
$B^+ = \bar{A}x$



$y = A\bar{B}$

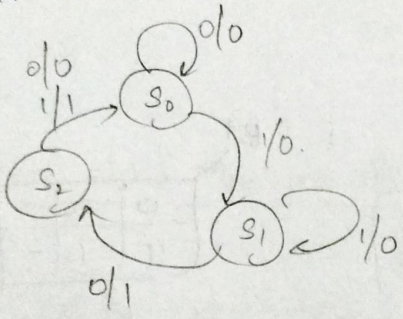


Mealy state Machine and its Examples



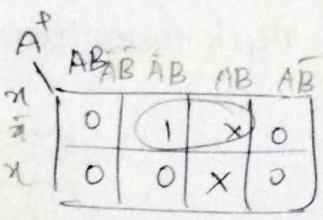
- In Mealy state machine o/p depends on present state and input
- In this next state depends on present states and i/p.

Ex:

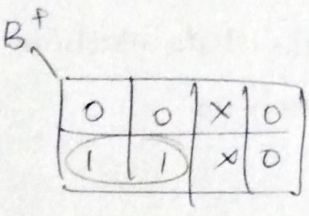


	A	B	x	A^+	B^+	y
S_0 0 0	0	0	0	0	0	0
S_1 0 1	0	0	1	0	1	0
S_2 1 0	0	1	0	1	0	ø1
	0	1	1	0	1	D
	1	0	0	0	0	0
	1	0	1	0	0	1
	1	1	0	X	X	X
	1	1	1	X	X	X

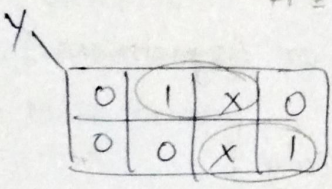
K-Map



$A^+ = Bx̄$

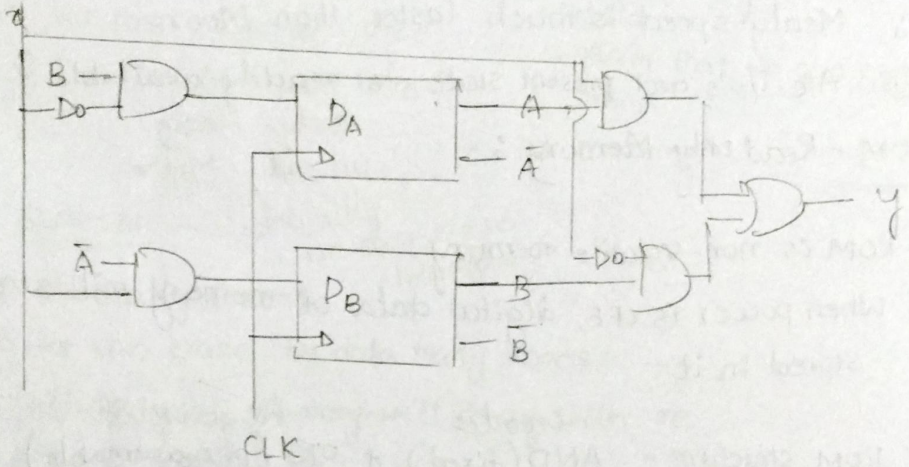


$B^+ = Āy$



$Y = Bx̄ + Ay$

Circuit Diagram



Comparison of Moore and mealy state machines

- o/p of mealy state machine depends on i/p & present state while in moore state machine depends on only present state o/p

- Moore state machine is comparatively safer than mealy state machine.

↳ o/p of moore state machine [changes with clock] o/p will change wrt i/p & CLK while with mealy state machine

- Mealy state machine is much faster than moore state machine

Synchronous - Moore
Asynchronous - Mealy

Q. Does we need synchronous or Asynchrous ?

How to choose btw moore & Mealy :

(i) Does we need synchronous or Asynchronous machine ?
↓ ↓
Moore Mealy

speed ?

(ii) Mealy - speed is much faster than Moore.

(iii) Are i/p's and present state is readily available ?
ready - Mealy
unready - Moore

ROM - Read only Memory :-

- ROM is non-volatile memory
- When power is OFF, digital data of memory will stay stored in it

ROM structure = AND (fixed) + OR (programmable)
Decoder + Programmable OR gates

- Size of ROM = $2^n \times y$

n = no. of inputs

y = no. of outputs

Ex: If i/p is 8 and Y (o/p) is 4.

Size = $2^8 \times 4$

$= 2^{10} = 1024$ bits

$= \left(\frac{1024}{8}\right)$ bytes

$= 128$ bytes

PLA (Programmable Logic Array)

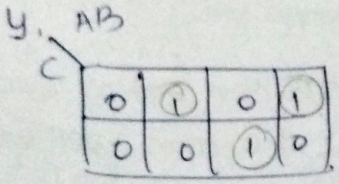
- It is fixed architecture logic device which have programmable AND and programmable OR gates.

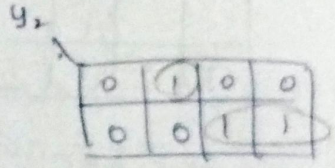
fixed AND
programmable OR

A	B	C	y_1	y_2
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Steps to program PLA :-

Step 1: Find Boolean function from the given table





$y_1 = \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

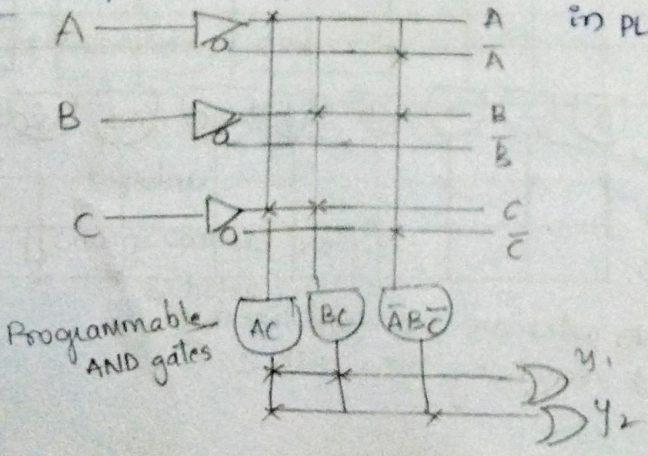
$= (A \oplus B)\bar{C} + ABC$

$y_2 = \bar{A}B\bar{C} + AC$

Step-2 Identify no.of input buffers.

We have three input variables, so we need 3 buffers.

Step-3: Implementation of boolean function in PLA



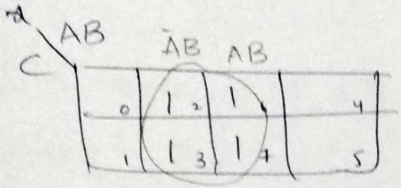
PAL (Programmable Array Logic)

- It is no most commonly used PLAD (Programmable logic devices)
- It has programmable AND gate and fixed OR gates
- Structure is simple than PLA
- It has issues regarding flexibility (more connections due to fixed OR gates, but it has fixed symbol structures. Here we need to program only programmable AND, but there we have to program both)

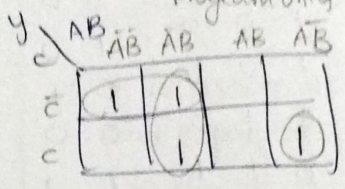
$x = \Sigma m(2,3,6,7)$

$y = \Sigma m(0,2,3,5)$

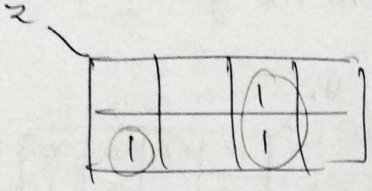
$z = \Sigma m(1,6,7)$



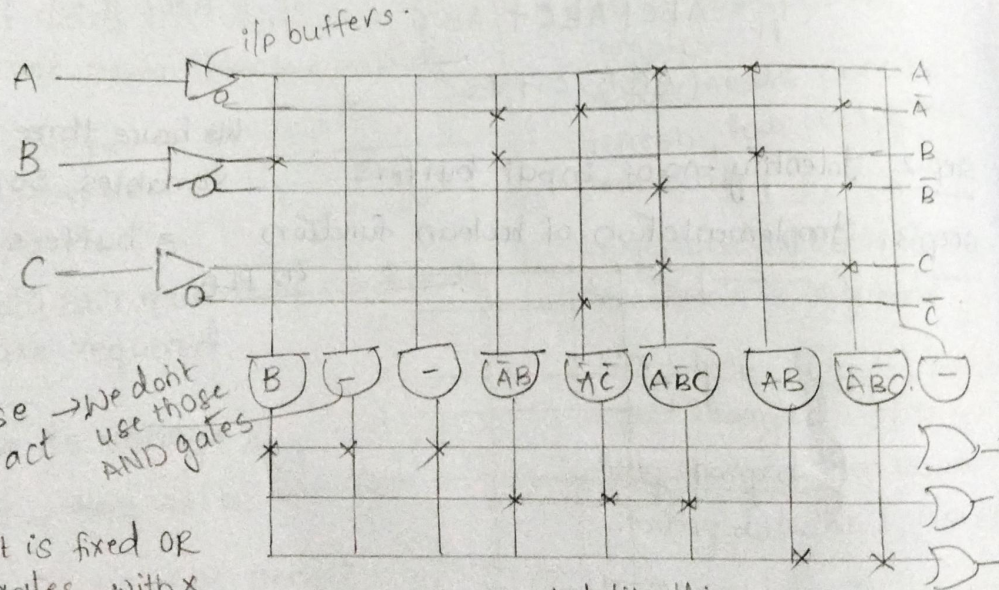
$x = B$



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



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



fuse → We don't use those AND gates intact

- As it is fixed OR gates, with x 3 programmable AND gates are connected like this with y next 3 and remaining three with z.