STLD

Unit 2 — Sequential Logic Circuits

- Latches & Flip Flops

Latches & FFs are basic elements that are used to store information. One flip flop & latch can store one bit of data.

The main difference between Latches & FFs is that a latch checks input continuously & changes the output whenever there is change in the input. But, flip flop is a combination of latch and a clock that continuously checks input & clocks the output time adjusted by the clock signal.

In both latches & flip flops, output not only depends on the current inputs, but also depends on the previous inputs & outputs.

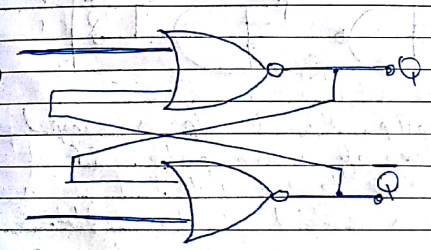
* Latch

- Latches are asynchronous — which means, the output of the latch depends on its input.
- Most computers today are Synchronous — means that output changes simultaneously to the rythm of a clock signal.
- 4 types of Latches are —

(i) SR Latch

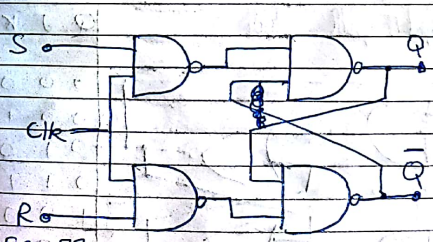
- Set/Reset latch (SRL) is an asynchronous device that relies on S & R inputs.
- Can be made using NOR gate.
- When S = 1, R = 1, then both outputs are 0. Which seems absurd. Thus, SR = 11 is a 'not allowed' state.

NOR Gate SR Flip Flop



S	R	Q	Q̄	
1	0	1	0	SET
0	0	1	0	
0	1	0	1	RESET
0	0	0	1	
1	1	0	0	UNDEFINED

NAND Gate SR FF



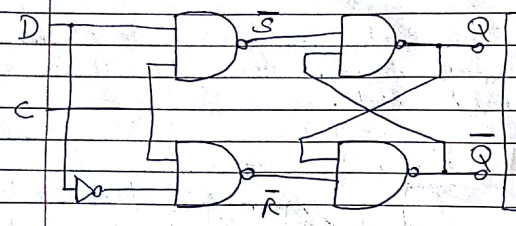
S	R	Q	Q̄	
0	0	No Change		
0	1	0	1	
1	0	1	0	
1	1	X	X	(Invalid)

SR FF

- Consists of 2 AND gates & a basic NOR FF.
- Output is 0 as long as clock pulse is 0 irrespective of S & R inputs.
- When clock pulse is 1, S & R I/Ps pass through the basic FF.
- When S = R = 1, output is 0 [Invalid State]

• D-Latch & D- Flip Flop

- D-Latch is the simple extension of SR latch.
- It removes the possibility of invalid input/state.
- It is also called transparent latch.

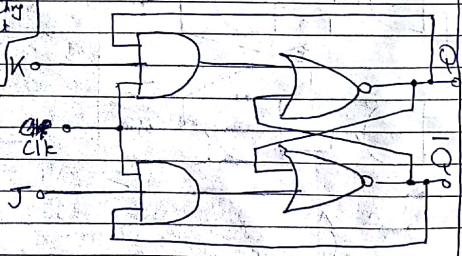


C	D	Next State of Q
0	X	No change
1	0	Q = 0 (Reset State)
1	1	Q = 1 (Set state)

- In D-flip flop, the D I/p is sampled during the existence of a clock pulse. If it is 1, FF is switched to set state. If it is 0, then the FF is switched to clear state.

• JK Flip Flop

Clk	J	K	O/P	Action
X	0	0	Q_n	No Change
↓	0	1	0	Reset
↓	1	0	1	Set
↓	1	1	$\bar{Q}_n$	Toggle



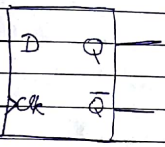
Q	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

- When input 1 is applied to both J & K, FF switches to complement state (If Q=1, it switches to Q=0).
- The output of FF Q is ANDed with I/Ps K & clk. Then FF will be cleared during clock pulse only if output Q was previously 1.

• T - Latch & T - Flip Flop

- T latch is formed when the inputs of the JK latch are shorted. When the input is high, then the output toggles.

Clk	D	Q	$\bar{Q}$	States
0	1	0	1	Start
1	1	1	0	Store 1
0	0	Q	$\bar{Q}$	No change
1	0	0	1	Store 0.



- T flip flop is the single input version of JK flip flop. Operation is as follows —
- If T = 0, then Present State = Next State = 0.
- If T = 1, Present State = 0 & Next state = 1.

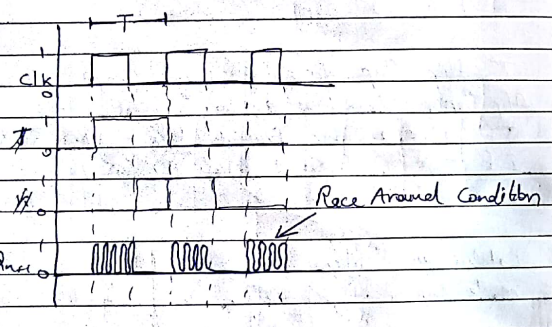
	Latches	Flip Flops
1.	Latches are building blocks made up of logic gates.	FFs are building blocks made by latches.
2.	Latches continuously checks the input & changes output correspondingly.	FFs continuously checks I/Ps & changes output accordingly as determined by clock.
3.	Latches can not be used as register.	FFs can be used as a register.
4.	It is based on enable function input.	It works on the basis of clock pulses.
5.	Level Triggered	Edge Triggered.

Page No.
Date

Race Around Condition

In JK flip flop,

Clk	J	K	Q_{n+1}
0	x	x	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	?



Solution –

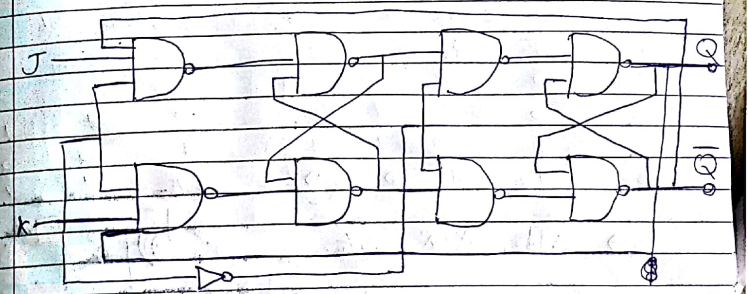
(i) Edge Triggering

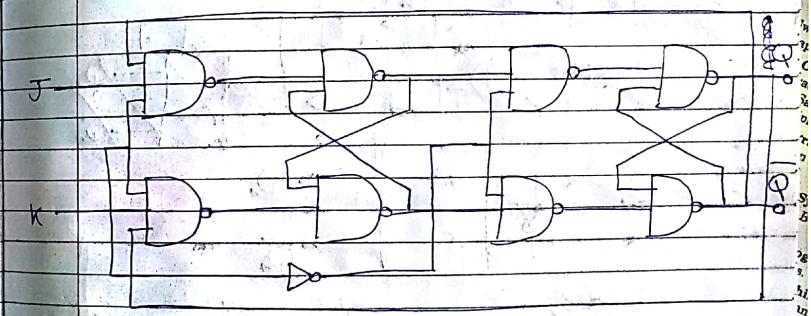
(ii) $T/2 < T'$
where T' is the time taken to generate output.

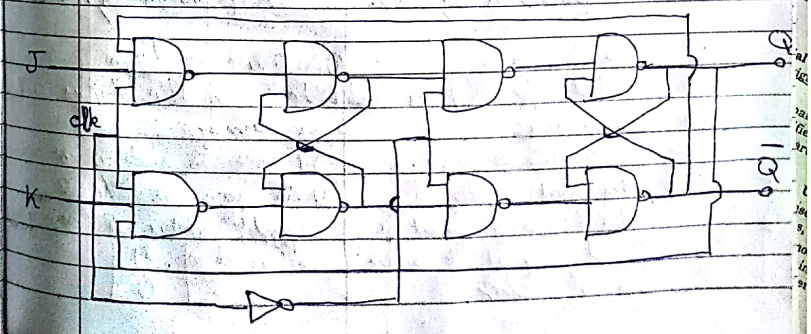
(iii) Use Master-Slave FF (MS).

Page No.
Date

* Master Slave JK Flip flop







- D flip flop to T-flip flop

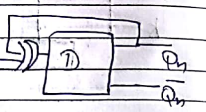
Characteristic Table (T) (QP)

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

Excitation Table (D)

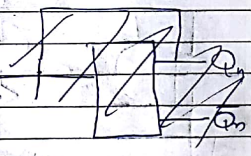
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

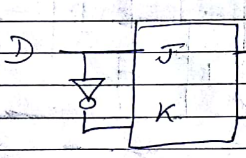
→ Add XOR gate



- JK ff to D ff

Use Not gate.



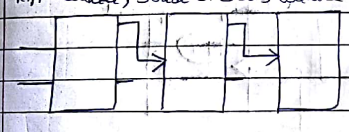
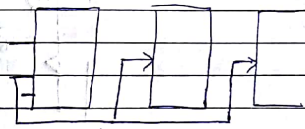


-	Synchronous Seq. Ckt.	Asynchronous Seq. Ckt.
①	Easy to design	① Difficult to design
②	Clocked flip flop is used as memory	② Unclocked flip flop is used as memory
③	Slower due to clock.	③ Faster as clock is not present
④	When input is changed, status of memory element is affected only at the active edge of clock.	④ As soon as I/p is changed, status of memory element will change.

* COUNTERS

- Counters are sequential circuits capable of counting clock pulses.
- Counting can be ascending or descending.
- n bit counter will have 'n' flipflops & 2^n states.
- These are used to perform the timing function as in digital watches, to create time delays, to produce non sequential binary counts, etc.

Counter

	Asynchronous Counter	Synchronous Counter
Two Types	Ripple Counter, Serial or Series Counter	
→		
→	The output of first FF drives the clock for second.	The ~~FF~~ clock of first FF drives the clock for others.
→	FFs are not clocked simultaneously	FFs are clocked simultaneously
→	Design & implementation is easy.	Very complex.
→	Mostly T-FF used.	Mostly D-FF used.
→	Low speed / Slower	High Speed / Faster.

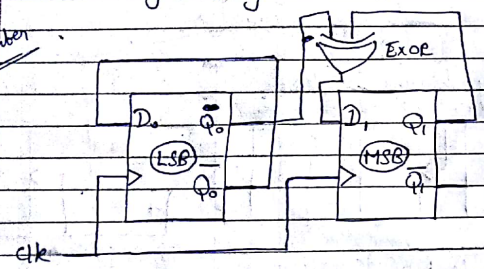
Page No.
Date

* Counters that count in ascending order are Up Counters
* " " " " descending are called Down Counters

• 2-bit Synchronous Counter

Steps -
1. Decide the no. of FFs. (Each FF stores 1 bit)
2. Write excitation table of FF.
3. Circuit excitation table
4. Using K-map, find I/P equation.
5. Draw logic dig.

2-bit Counter



Present State		Next State		Input	
Q_{1P}	Q_{0P}	Q_{1N}	Q_{0N}	D_1	D_0
0	0	0	1	0	1
0	1	1	0	1	0
1	0	1	1	1	1
1	1	0	0	0	0

Present State K Map for D_1

Q_{1P} \ Q_{0P}	0	1
0	0	1
1	1	0

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

Present State K Map for D_0

Q_{1P} \ Q_{0P}	0	1
0	1	1
1	0	0

By K-maps,

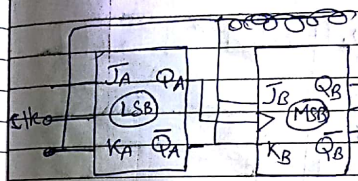
$D_1 = Q_{0P}\overline{Q_{1P}} + \overline{Q_{0P}}Q_{1P} = Q_{0P} \oplus Q_{1P}$

~~$D_0 = Q_{0P}\overline{Q_{1P}} + \overline{Q_{0P}}Q_{1P}$~~

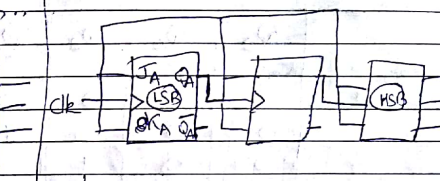
* Asynchronous Counter (Ripple Counter)

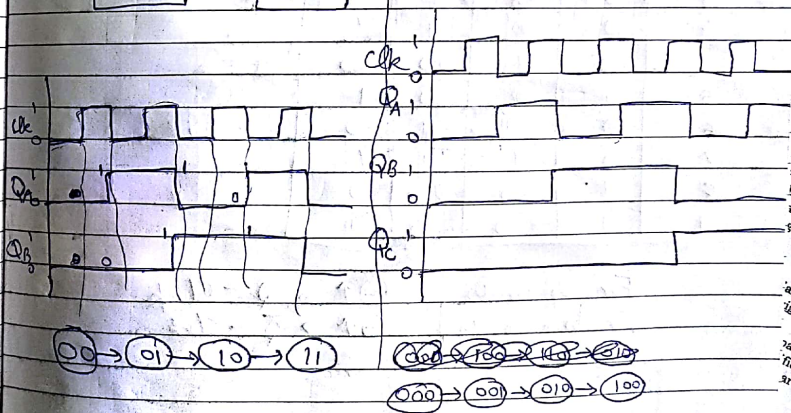
[Using Negative Edge Triggered FF]

2-bit

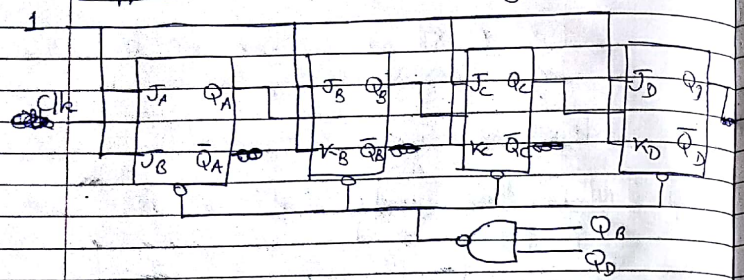


3-bit





Ripple : BCD / Decade Counter using JK FF



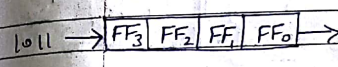
	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

For Up Counter, $\ominus\uparrow \rightarrow Q$ – Clock
For down counter, $\ominus\uparrow \rightarrow \bar{Q}$ – Clock

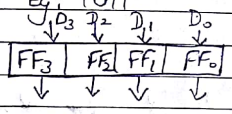
* Register & Data Format & Classification

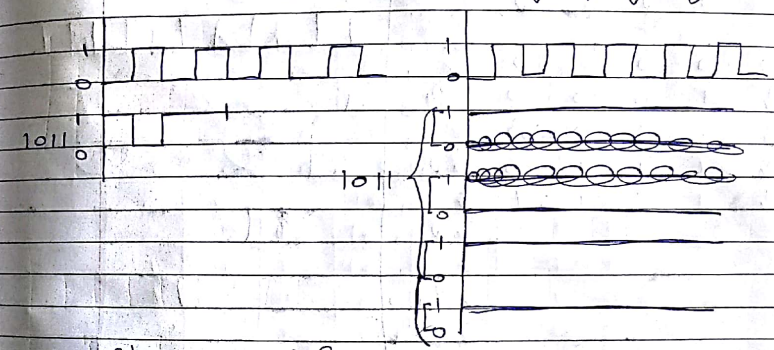
→ Flip flop is 1-bit memory cell.
→ Group of flip flops is known as REGISTER
→ 'n' bit register consists of 'n' no. of FFs.

Serial Data Input
Eg. 1011



Parallel Data Input
Eg. 1011





* Classification of Register

(i) Depending on Input & Output

a SISO →

b SIPO →

c PISO →

d PIPO

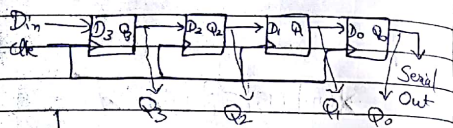
(ii) Depending on Application

(a) Shift Register

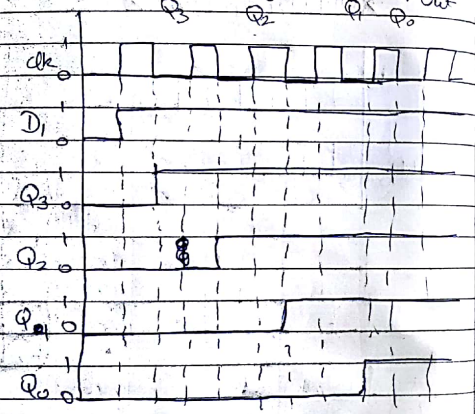
(b) Storage Register

* Shift Register

(i) In SISO



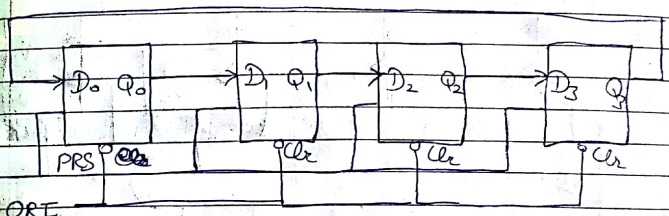
Clk	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
↓	1	0	0	0
↓	1	1	0	0
↓	1	1	1	0
↓	1	1	1	1



* Ring Counter (Special type of Shift Register)

→ Last FF's output is connected to input of first FF.

→ No. of state = No. of FFs used.



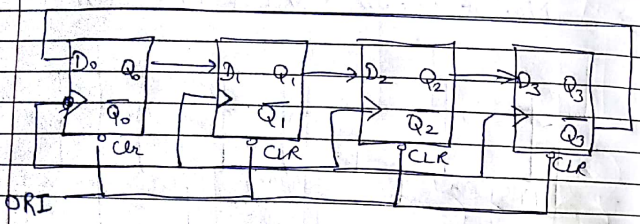
ORI (Overwriting I/P)	Clk	Q_0	Q_1	Q_2	Q_3
⊓ (↓)	X	1	0	0	0
1	↓	0	1	0	0
1	↓	0	0	1	0
1	↓	0	0	0	1
1	↓	1	0	0	0

When [(Preset) PR = 0, Q = 1
CLR = 0, Q = 0]

Page No.
Date

* Johnson's Counter

→ Last FF's complemented o/p is I/p for 1st FF.
→ No. of states = 2 × No. of FF.



ORI	CLK	Q_0	Q_1	Q_2	Q_3
⊓(↓)	X	0	0	0	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

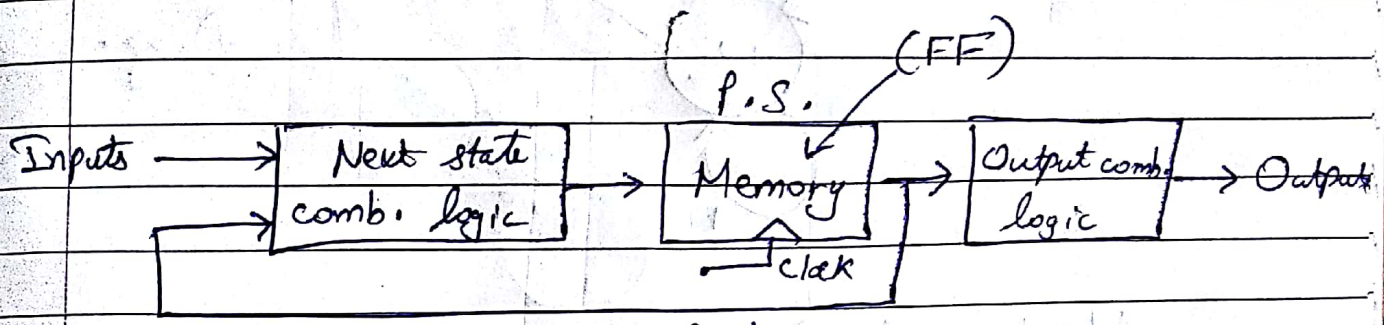
Page No.
Date

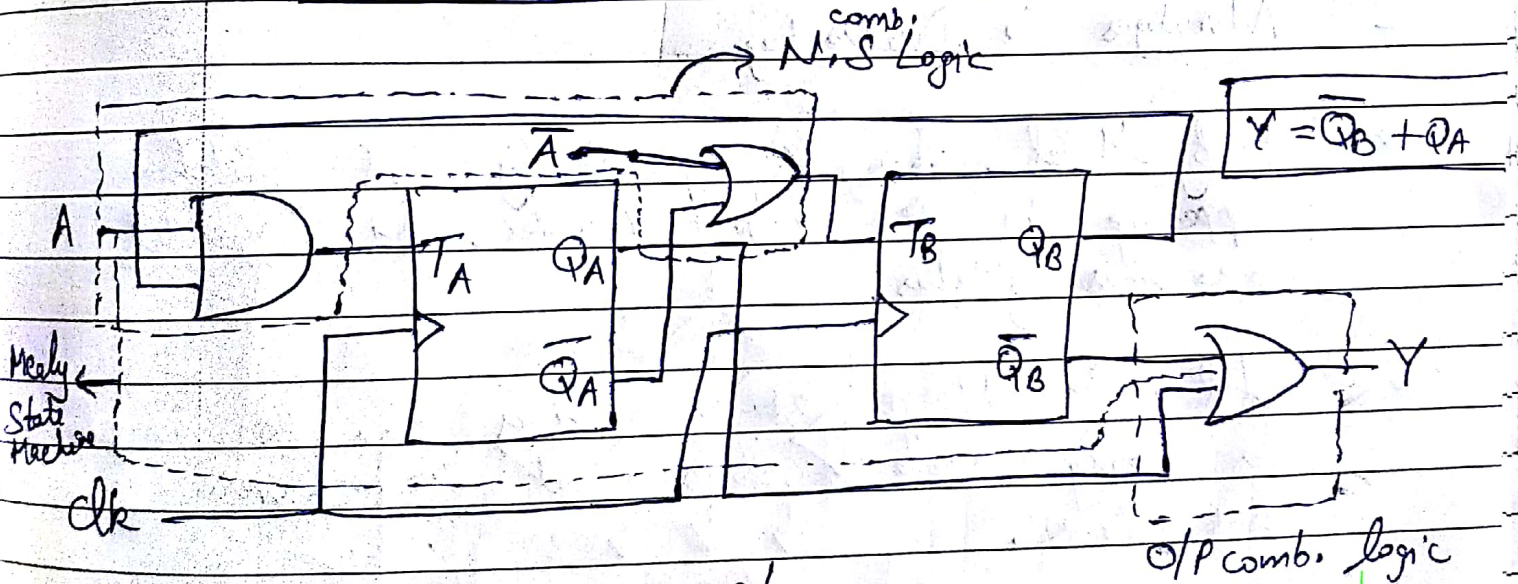
*	Combinational Circuits	Sequential Circuits
1.	Output variables are dependent only on the present I/P variables.	O/P depends on present I/P as well as previous o/p.
2.	Faster	Slower
3.	Easy to design	Hard to design
4.	E.g. Adder	Eg. Counter
5.	No memory unit.	It uses memory unit.

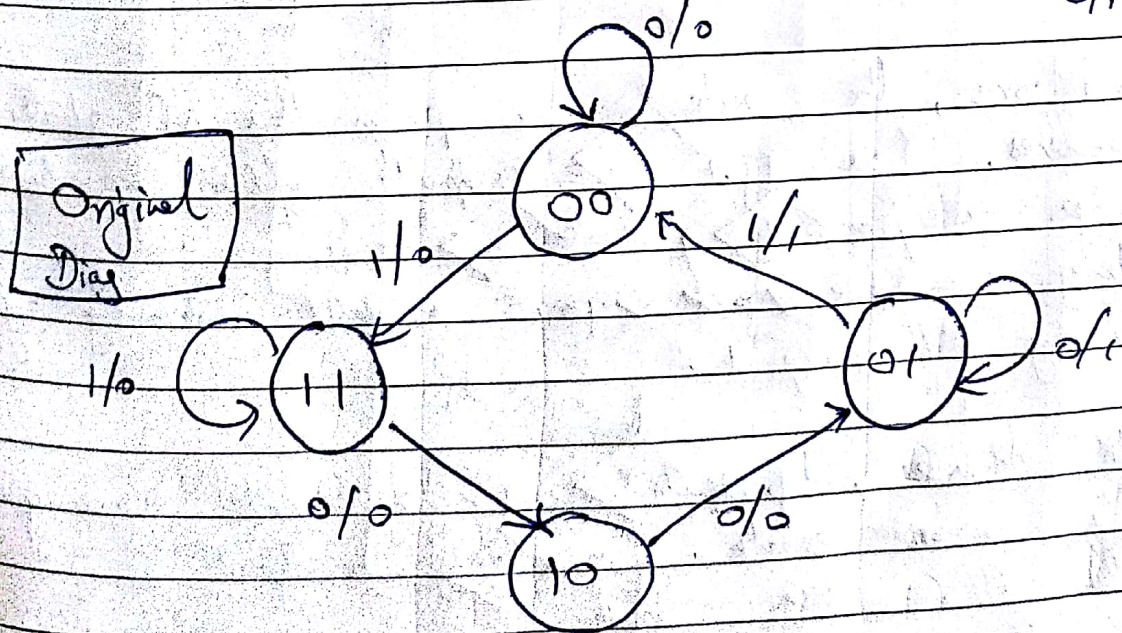
Page No.
Date

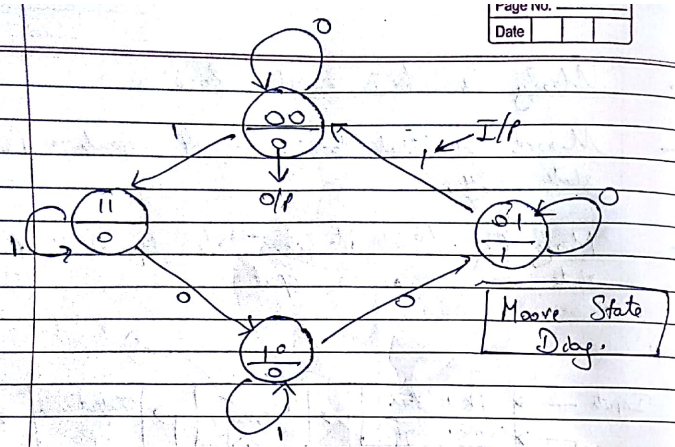
Mealy & Moore State Machines

- Moore Ckt./State Machine – O/P depends on Present state only.
- Mealy Ckt./State Machine – O/P depends on present state as well as input.









- Advantages & Disadvantages

* Mealy Modal has a possibility of glitches appearing in the output variables. Moore modal overcomes glitches.

* All the Moore state machines can be implemented as mealy state machines but the converse is not true.

* In Moore, the outputs are properties of states themselves. You get the output after machine reaches a particular state.
Mealy machine gives you output instantly after recieving the input.

* In Moore, outputs are held until you go to some other state. In Mealy, output is not held after the clock cycle.

Algorithmic State Machines

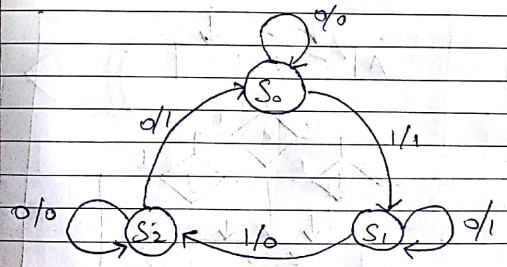
* ASM chart are advanced versions of state diagrams.

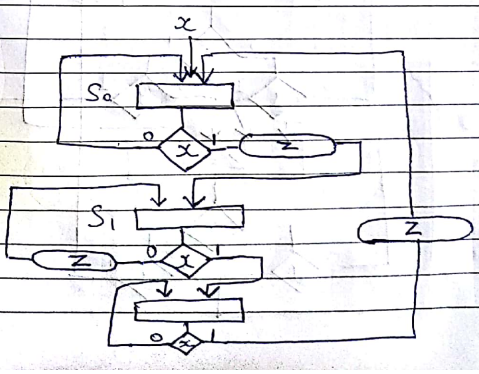
State Block

Decision box

Condition box (Mealy box)
* only used when o/p is high

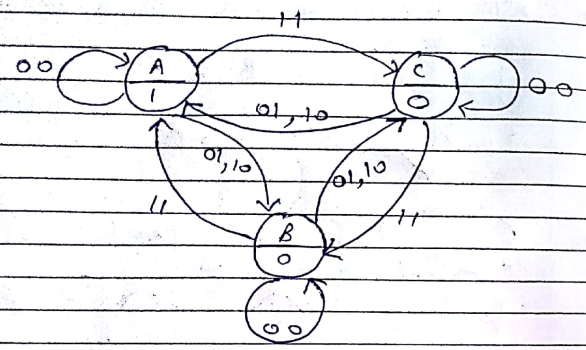
* Using Mealy State Machine

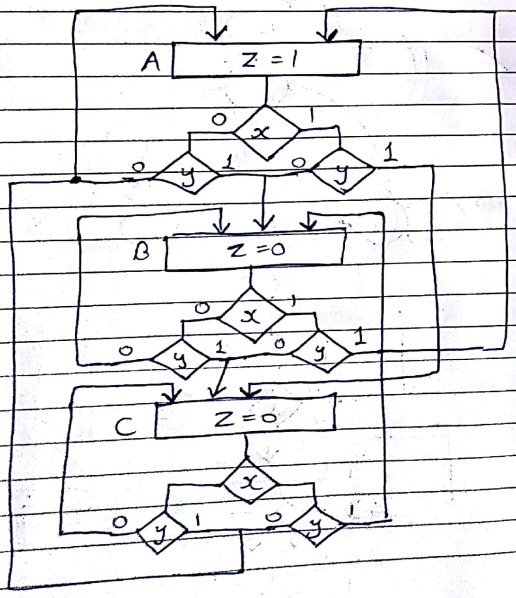




Page No.
Date

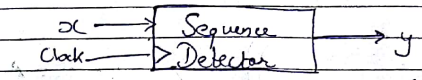
* Using Moore State Machine



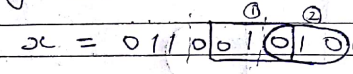


Page No.
Date

* Pattern or Sequence Detector



Eg. Scanning 010 from –



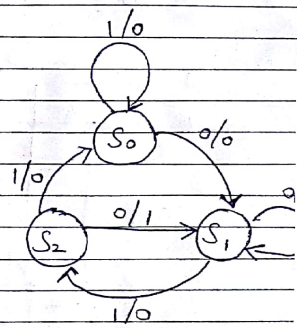
$\Rightarrow y = 0\,0\,0\,0\,0\,0\,1\,0\,1\,0$

Step 1 State Diagram

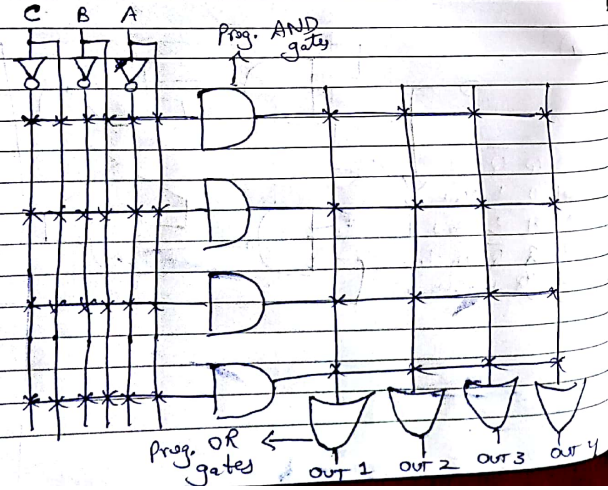
S_0 = Reset (power up)
S_1 = 0
S_2 = 01

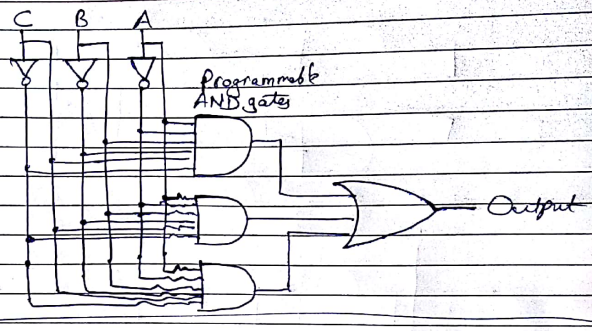
Step 2. State assignment

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



	PLA	PAL
(i)	Both AND & OR arrays are programmable	OR array is fixed & AND array is programmable.
(ii)	More costly.	Less Costly.
(iii)	Much more complex than PAL & PROM.	Less complex / Simpler.
(iv)	AND array can be programmed to get desired minterms.	And array can be programmed to get desired minterms.
(v)	Any boolean form in SOP can be implemented using PLA.	Any boolean form in SOP can be implemented using PAL.





Characteristics of CMOS

(i) Extremely large fan-out capability.
(ii) Lowest power dissipation of all gates.
(iii) Very high noise immunity.
(iv) Requires single power source.
(v) Lower propagation delay than NMOS.
(vi) Temperature stability is excellent.