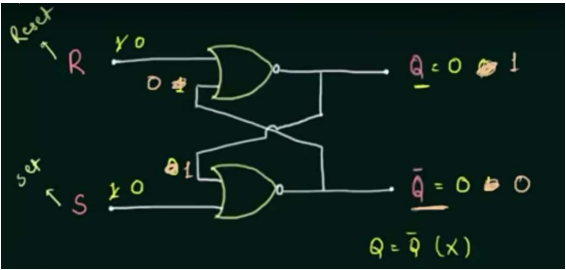
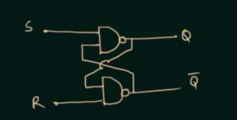


SR LATCH



Level sensitive

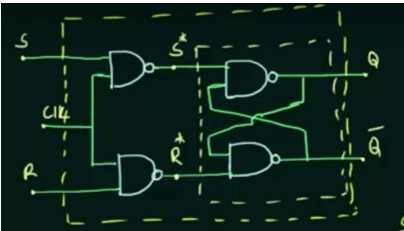


S	R	Q	Q̄
0	0	Not used	
0	1	1	0
1	0	0	1
1	1	Memory	

S	R	Q	Q̄
0	0	Memory (as before)	
0	1	0	1
1	0	1	0
1	1	Not used	

FLIP FLOP

Edge sensitive



clk	S	R	Q	Q̄
0	x	x	Memory	
1	0	0	Memory	
1	0	1	0	1
1	1	0	1	0
1	1	1	Not used	

Characteristic Table :-

clk = 1

Q _n	S	R	Q _{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	x
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	x

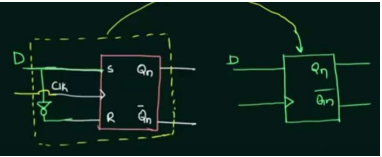
n.s. = p.s.

Excitation table:-

Q _n	Q _{n+1}	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

$$Q_{n+1} = S + Q_n R$$

D FLIP FLOP



clk	D	Q _{n+1}
0	x	Q _n
1	0	0
1	1	1

Only to store data we use this FF

Char. Table :-

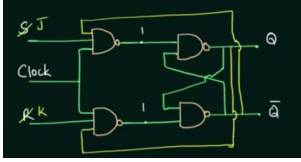
Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

$Q_{n+1} = D$

Excitation Table :-

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

JK FLIP FLOP



Start initially with Q=0

Truth Table :-

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	\bar{Q}_n (toggle)

Ch. Table :-

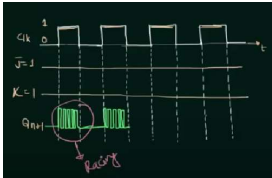
Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Excitation Table :-

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

$$J = Q_{n+1} \quad K = \bar{Q}_{n+1}$$

$$Q_{n+1} = \bar{Q}_n J + Q_n \bar{K}$$



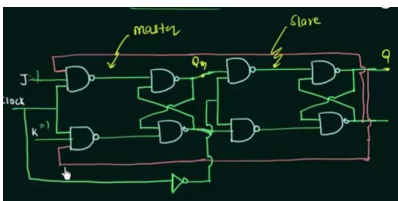
Unstable diff than toggle

Delay > T/2

Toggle is controlled But

Conditions to overcome Race:-
 i) $T/2 < \text{prop delay of FF}$
 ii) edge trig
 iii) Master - Slave

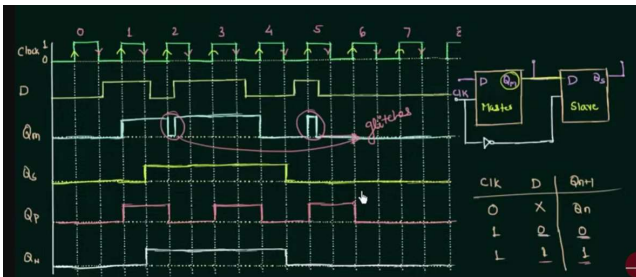
Master slave is same as neg edge trig



master

slave

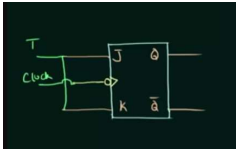
Glitches = sudden change in the output signal



Master slave work on level triggering
 Which has the same effect as neg edge triggering

There are no glitches in slave that why we use master slave FF
 Master slave output is same as the neg edge output

T Flip Flop - Toggle



T.T. for T FF -

clk	T	Q _{n+1}
0	X	Q _n (memory)
1	0	Q _n (memory)
1	1	\bar{Q}_n (toggling)

Ch Table:-

Q _n	T	Q _{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table:-

Q _n	Q _{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

T FF is neg edge triggered

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

PRESET AND RESET

>> They are the direct inputs or overriding inputs or asynchronous inputs.

>> The synchronous inputs are S,R,J,K,D & T.

$\text{preset} = 0 \Rightarrow Q_n = 1$
 $\text{clear} = 0 \Rightarrow Q_n = 0$ because $\bar{Q}_n = 1$

Whatever be the value of clock and synch. inputs

Preset	Clear	Q _n
0	0	Not used
0	1	1
1	0	0
1	1	ff will perform normally

Asynchronous inputs = set reset
Synchronous inputs = s r j k d t

Introduction to State Diagram

State table:-

Q _n	Q _{n+1}	Q _{n+1}	Q _{n+1}
0	0	1	1
0	0	1	1
1	0	1	1
1	0	1	1

$S_0 = 00$
 $S_1 = 01$
 $S_2 = 10$
 $S_3 = 11$

for JK FF:-

Q _n	J	K	Q _{n+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

$S_0 = 00$
 $S_1 = 01$
 $S_2 = 10$
 $S_3 = 11$

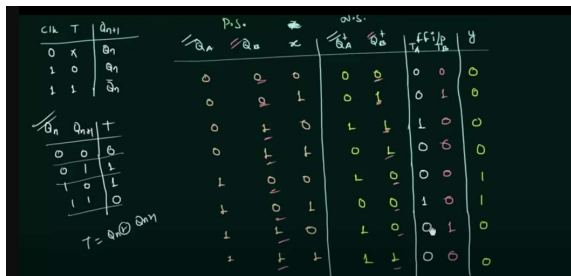
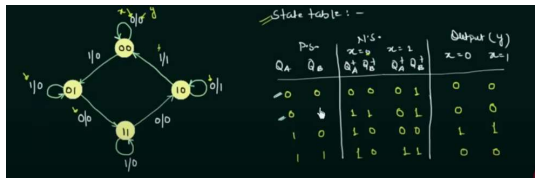
State eqⁿ:-

$$Q_{n+1} = \bar{Q}_n J + Q_n \bar{K}$$

Design Procedure for Clocked Sequential Circuits

- Step 1: A State diagram or timing diagram is given, which describes the behaviour of the circuit that is to be designed.
- Step 2: Obtain the state table.
- Step 3: The number of states can be reduced by state reduction method.
- Step 4: Do state assignment, (If required)
- Step 5: Determine the number of flip-flops required and assign letter symbols.
- Step 6: Decide the type of flip-flop to be used.
- Step 7: Derive the circuit excitation table from state table.
- Step 8: Obtain the expression for circuit output and flip flop input.
- Step 9: Implement the circuit.

Next stage as well as the output must be same for two states for them to be reduced

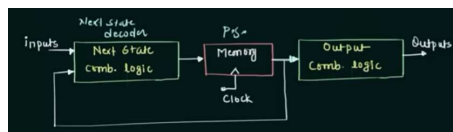


$$\begin{aligned}
 T_A &= \bar{Q}_A Q_B \bar{x} + Q_A \bar{Q}_B x \\
 T_B &= \bar{Q}_A Q_A x + Q_A Q_B \bar{x} \\
 y &= Q_A \bar{Q}_B x + Q_A Q_B \bar{x} \\
 &= Q_A \bar{Q}_B (x + \bar{x}) \\
 &= Q_A \bar{Q}_B \cdot 1 \\
 y &= Q_A \bar{Q}_B
 \end{aligned}$$

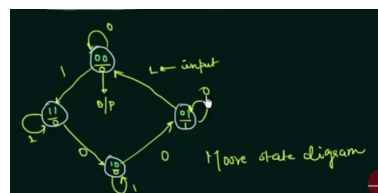
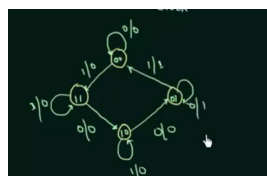
MEALY AND MOORE CIRCUITS

MOORE - Doesn't depend upon the input
Diff is the way the output is calc

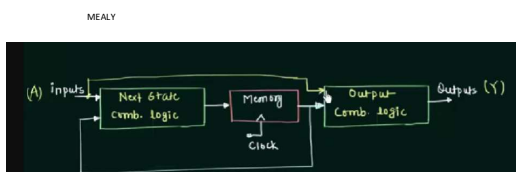
i) Moore Ckt/Moore State Mach :- O/p is the fun of P.S. only
ii) Mealy Ckt/Mealy State Machine :- O/p is the fun of P.S. as well as I/P
* only in the way the o/p is generated



mealy



Conversion to a state diagram which follows moore prop

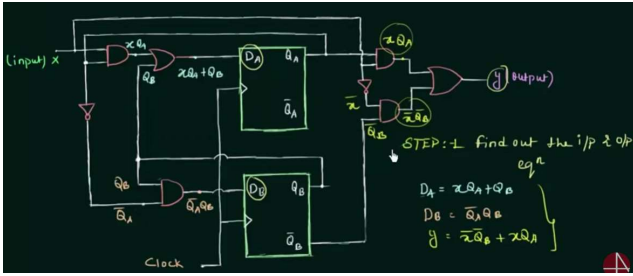


Mealy - no output at all

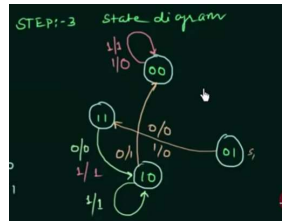
If same logic is applied

ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS

Write the equations for the inputs and outputs
Then write the state table
Then make the state diagram



Q_A	Q_B	X	Q_A^+	Q_B^+	Y
0	0	0	0	0	1
0	0	1	0	0	0
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	1	0	1

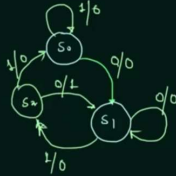


PATTERN DETECTOR

Step 1 state diagram (Moore machine)

$S_0 = \text{Reset (powerup)}$
 $S_1 = 0$
 $S_2 = 01$

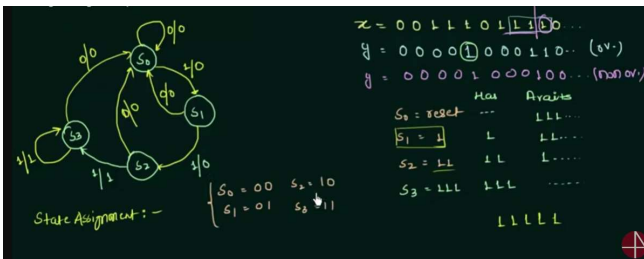
Step 2 state assignment:-



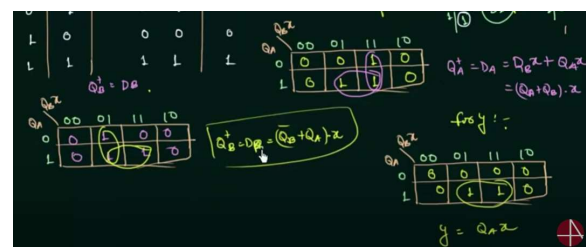
Considering DV

Step 2 state assignment:-

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



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



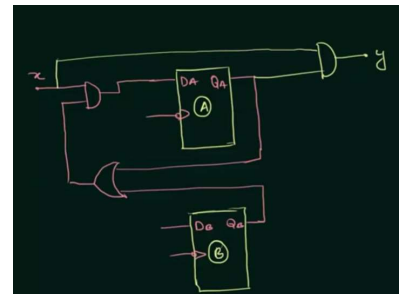
$$\begin{aligned} NLS &= D \\ D_A &= Q_A^+ \\ D_B &= Q_B^+ \end{aligned}$$

Synchronous Seq. Ckt.

1. These circuits are easy to design.
2. A clocked flip flop acts as memory element.
3. they are Slower.
4. The status of memory element is affected only at the active edge of clock, if input is changed.

Asynchronous Seq. Ckt.

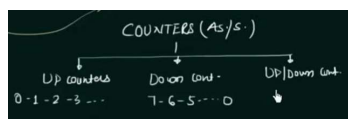
1. These circuits are difficult to design.
2. An unclocked flip flop or time delay element is used as memory element
3. Faster as clock is not present
4. The status of memory element will change any time as soon as input is changed





1. Flip flops are connected in such a way that the o/p of first flip flop drives the clock of next flip flop.
2. Flip flops are not clocked simultaneously.
3. Circuit is simple for more number of states.
4. Speed is slow as clock is propagated through number of stages

1. There is no connection between o/p of first flip flop and clock of next flip flop.
2. Flip flops are clocked simultaneously.
3. Circuit becomes complicated as number of states increases.
4. Speed is high as clock is given at a same time.



Logic-1

CLK

J_A Q_A K_A (A)

J_B Q_B K_B (B)

J_C Q_C K_C (C)

Q_A Q_B Q_C

LSB

Counter outputs

MSB

State	0	1	2	3
Initially	0	0	0	0
3^{st} (1)	0	0	1	1
2^{nd} (1)	0	1	0	2
3^{rd} (1)	0	1	1	3
4^{th} (1)	1	0	0	4
3^{th} (1)	1	0	1	5
6^{th} (1)	1	1	0	6
7^{th} (1)	1	1	1	7
1^{st} (1)	0	0	0	0

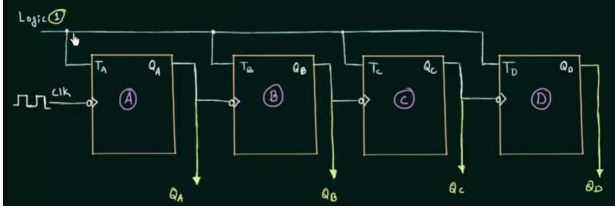
8 States

$2^n = 2^3 = 8$

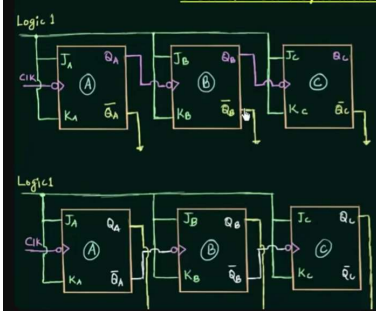
Maximum count:

$\sqrt{2^n - 1} = \sqrt{2^3 - 1} = \sqrt{7} = 2$

4 Bit Asynchronous Up Counter



3 BIT DOWN COUNTER

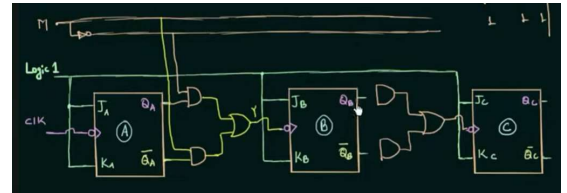


UPDOWN COUNTER

>> We have designed the up counters and down counters separately.
 >> But in practice both these modes are combined.
 >> A mode control input (M) is used to select either up or down mode.
 >> A combinational circuit is required between each pair of flip flops.

M	Q	\bar{Q}	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

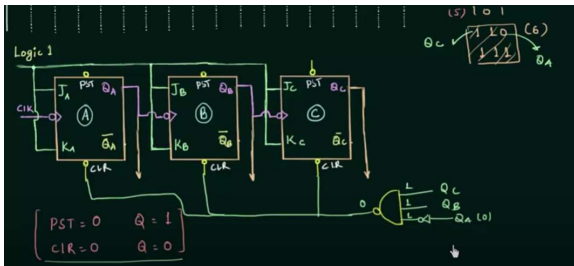
$Y = \bar{Q}Q + M$
 $Y = \bar{Q}Q + M$



MODULUS COUNTER

>> 2-bit ripple counter is called MOD-4 or modulus 4 counter.
 >> 3-bit ripple counter is called as MOD-8 counter.

Modulus of counter = number of states



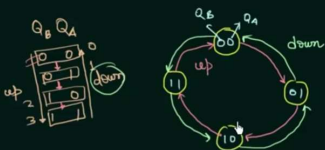
State Diagram of a Counter

2-bit up counter

Maximum Count :- $2^n - 1$

$$\begin{aligned}
 M.C. &= 2^2 - 1 \\
 &= 4 - 1 \\
 &= 3 \text{ (11)}
 \end{aligned}$$

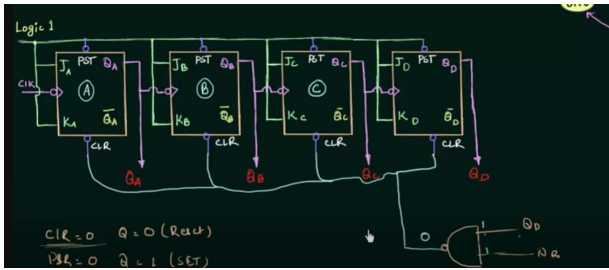
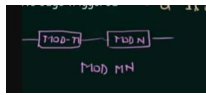
5 states



Decade (BCD) Ripple Counter

IMPORTANT POINT'S:

- 1) Negative edge triggered $\rightarrow Q_1$ clock \rightarrow UP counter
- 2) Positive edge triggered $\rightarrow \bar{Q}_1$ is clock \rightarrow UP counter
- 3) Negative edge triggered $\rightarrow Q_1$ clock \rightarrow Down counter
- 4) Positive edge triggered $\rightarrow \bar{Q}_1$ is clock \rightarrow Down counter



Mod 10 counter / BCD counter

How to Design Synchronous Counters

Step 1: Decide the number of flip flops.

Que: Design 2-bit synchronous up counter

Step 2: Excitation table of FF.

Step 3: State diagram and circuit excitation table.

Step 4: Obtain simplified equations using K' map.

Step 5: Draw the logic diagram.

How to Design Synchronous Counters

Step 1: Decide the number of flip flops.

Que: Design 2-bit synchronous up counter

Step 2: Excitation table of FF.

Step 3: State diagram and circuit excitation table.

Step 4: Obtain simplified equations using K' map.

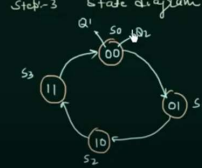
Step 5: Draw the logic diagram.

Step-2: Excitation table of JK flipflop

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

JK flip flop

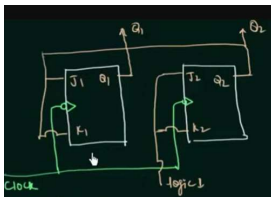
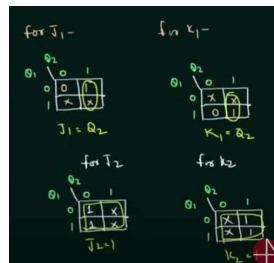
Step-3 State diagram

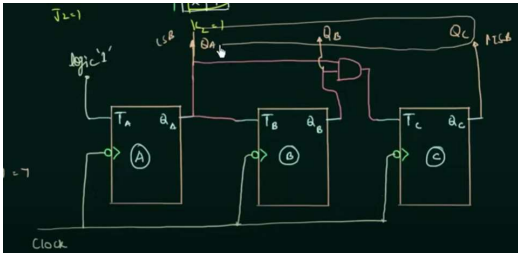


6:00 / 12:56

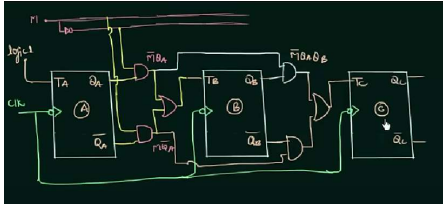
CKT excitation table:-

Q_1, Q_2	Q_1^+, Q_2^+	J_1, K_1	J_2, K_2
0 0	0 1	0 X	1 X
0 1	1 0	1 X	X 1
1 0	1 1	X 0	1 X
1 1	0 0	X 1	X 1

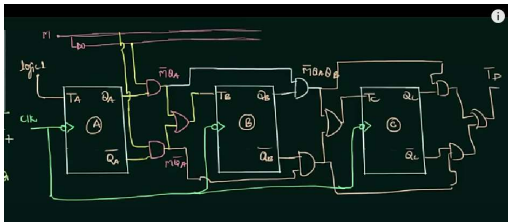




3 bit sync counter



3 bit updown sync counter



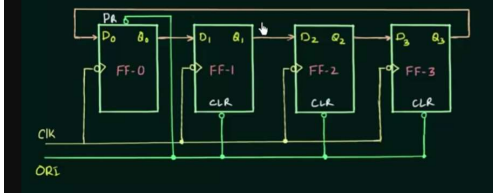
4 bit updown counter

RING COUNTER

Difference from register = output of the last FF is connected to the input of the first FF

Ring Counter

- >> Ring counter is a typical application of shift register
- >> The only change is the output of last ff is connected to the input of first ff.



$$\begin{aligned} \overline{PR} = 0 & \quad Q = 1 \\ \overline{CLR} = 0 & \quad Q = 0 \end{aligned}$$

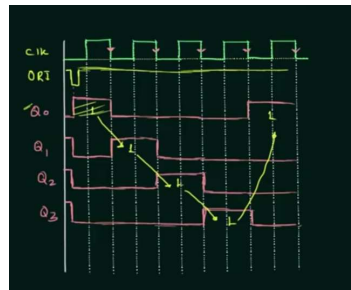
ORI	CLK	Q ₃	Q ₂	Q ₁	Q ₀
1	X	1	0	0	0
1	↓	0	1	0	0
1	↓	0	0	1	0
1	↓	1	0	0	0

* no of states = no of bits used

ORI = overriding inputs

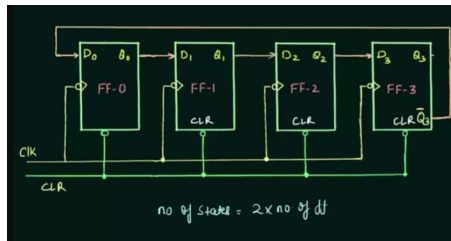
Number of states = n

ORI	CLK	Q ₃	Q ₂	Q ₁	Q ₀
1	X	1	0	0	0
1	↓	0	1	0	0
1	↓	0	0	1	0
1	↓	1	0	0	0



Johnson Counter

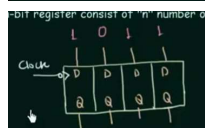
Better than the ring counter due to more number of states
Number of states = 2^n



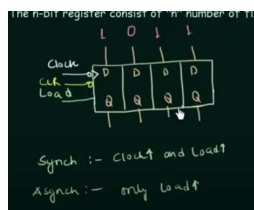
CLR	CLK	Q ₀	Q ₁	Q ₂	Q ₃	
1	X	0	0	0	0	①
1	↓	1	0	0	0	②
1	↓	1	1	0	0	③
1	↓	1	1	1	0	④
1	↓	1	1	1	1	⑤
1	↓	0	1	1	1	⑥
1	↓	0	0	1	1	⑦
1	↓	0	0	0	1	⑧
1	↓	0	0	0	0	⑨

REGISTER

» Flip Flop is 1-bit memory cell.
» To increase the storage capacity, we have to use group of flip-flop. This group of ff is known as REGISTER.
» The n-bit register consist of "n" number of flip-flops and is capable of storing "n-bit" word.



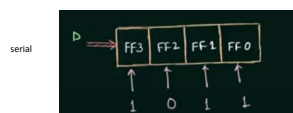
Clock is applied to all directly
This can be a problem as the data stored will get changed as clock pulse changes



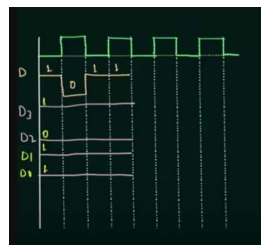
Hence load are applied

Load = Independent control

» Data can be entered in serial or in parallel form.
one bit at a time
all bits at a time



parallel



Serial = temporal code
Parallel = special code

Classification of Reg :-
i) Depending on i/p & o/p
a) SISO c) PISO
b) SIPO d) PIPO
ii) Depending on application
a) Shift Reg.
b) Storage Reg.

Shifts the data

Stores the data

PIPO

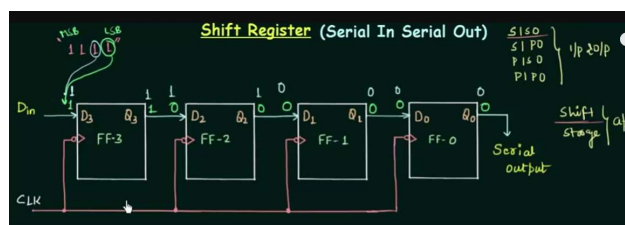
Serial IN and Serial OUT

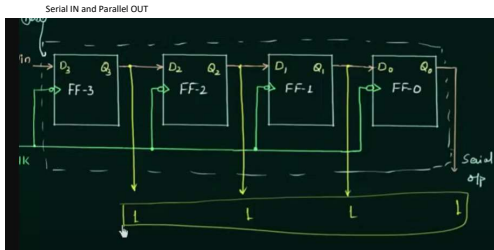
On input = 1111
The clock stores on rising edge

Beneficial for long distance transmission
Only one input line is required

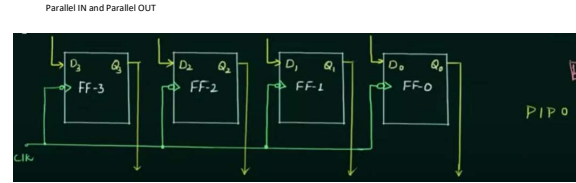
Number of clock pulses required to store data = 4
But disadv is that we need 4 pulses to output the data as well

CLK	Q ₃	Q ₂	Q ₁	Q ₀
Initially	0	0	0	0
↓	1	0	0	0
↓	1	1	0	0
↓	1	1	1	0
↓	1	1	1	1





4 clock pulses to store the data
1 clock pulse to output the data



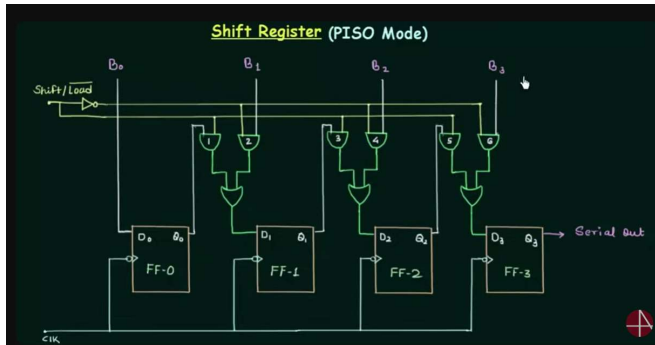
Input the number before clocks falling edge and then make clock = 0
This stores the data
Output is given parallelly

1 clock pulse to store the data

Storage register / buffer register

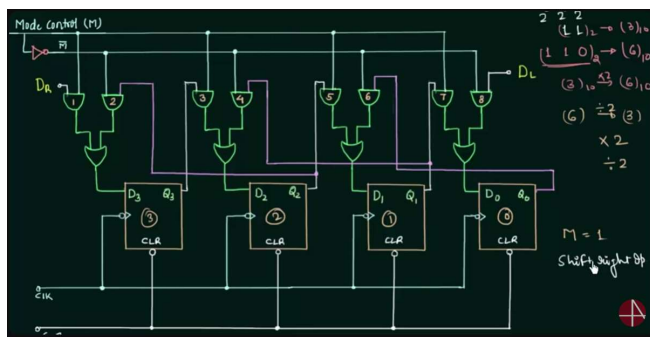


Buffer
Input in = output out



- 1) Load Mode → For input. Works when load = 0
- 2) Shift Mode → Shifts the data towards the right. Works when load = 1

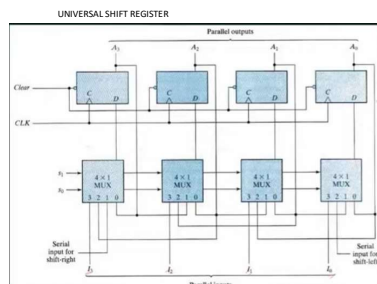
BIDIRECTIONAL SHIFT REGISTER



$M = 1$
Shift right by
 $M = 0$
Shift left

bidirectional ser.
+
parallel reading
= U.S.R.

All SISO SIPO PISO PISO are possible in this, hence universal



Mode control	Reg. Op.
$S_1 S_0$	
0 0	No change
0 1	Shift right
1 0	Shift left
1 1	Parallel load

Save data in flip flops

FSM = finite state machine

Flip flop are edge sensitive while latches are level sensitive

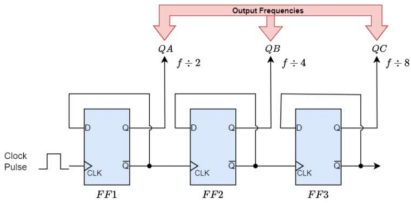
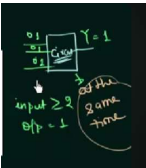
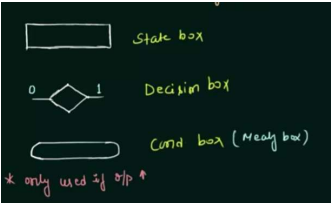


Fig-1: Freq Division Counter

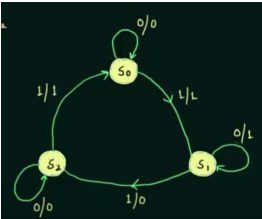


Coincidence logic

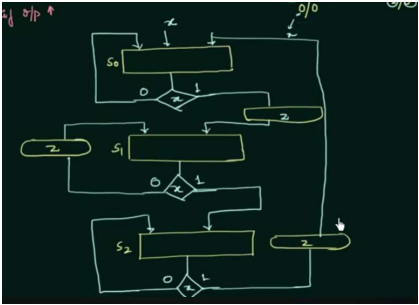
ASM CHART = Algorithmic State Machine



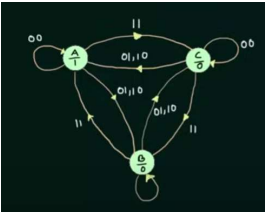
Condition box used only for mealy state machines



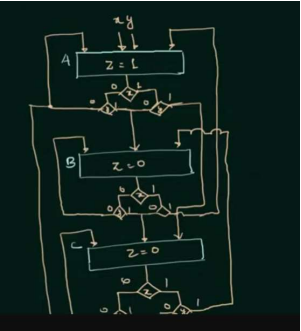
Mealy state machine



Z means that the output is 1
If output is 0 then don't do anything



Moore State Machine



PLA = Programmable Logic Array

PLD = Programmable Logic Device

Programmable Logic Array (PLA)

PLD

It is a type of fixed architecture logic device with programmable AND gates followed by programmable OR gates

A	B	C	Y ₁	Y ₂
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

$$Y_1 = A\bar{B}\bar{C} + A\bar{B}C + ABC$$

$$Y_1 = A\bar{B} + AC$$

$$Y_2 = B\bar{C} + AC$$

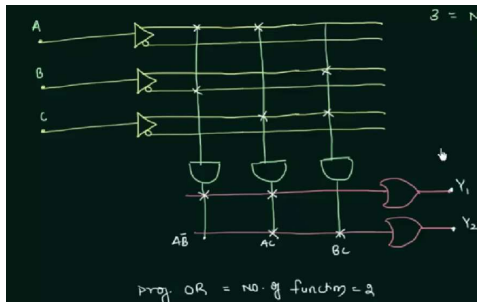
Step:-2

Step:-1 No. of i/p buffers

= number of variables



3 = No. of prog. AND gate = No. of connections (input to output)



PAL = Programmable Array Logic

>> It is most commonly used type of PLD.
>> Has programmable AND array and fixed OR array.

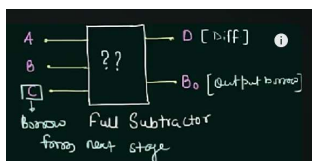
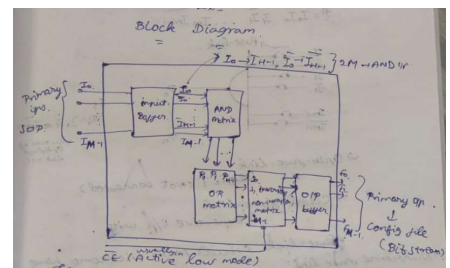
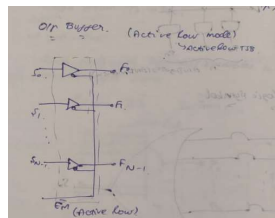
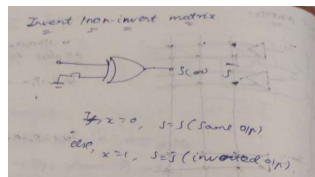
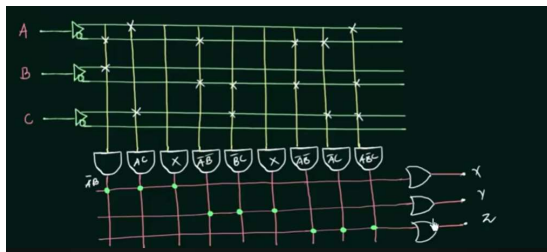
$$X(ABC) = \sum(2,3,5,7) = \bar{A}B + AC$$

$$Y(ABC) = \sum(0,1,5) = \bar{A}\bar{B} + \bar{B}C$$

$$Z(ABC) = \sum(0,2,3,5) = \bar{A}B + \bar{A}C + A\bar{B}C$$

Number of AND gates per function = maximum number of minterms
Number of OR gates = number of outputs

When we don't need an AND gate, simply fuse it



Full subtractor: $A - B - B_0$

STEPS:-

1. Identify available and required flip flop.
2. Make characteristic table for required flip flop.
3. Make excitation table for available flip flop.
4. Write boolean expression for available ff.
5. Draw the circuit.

Steps for conversion of FFs

Ex:- JK to D Flip flop conversion.

Av. ff = JK
req. ff = D

Qn	D	Qn+1	J	K
0	0	0	0	x
0	1	1	1	x
1	0	0	x	1
1	1	1	x	0

for J -		for K -	
Qn	D	Qn	D
0	1	x	x
x	x	1	0