



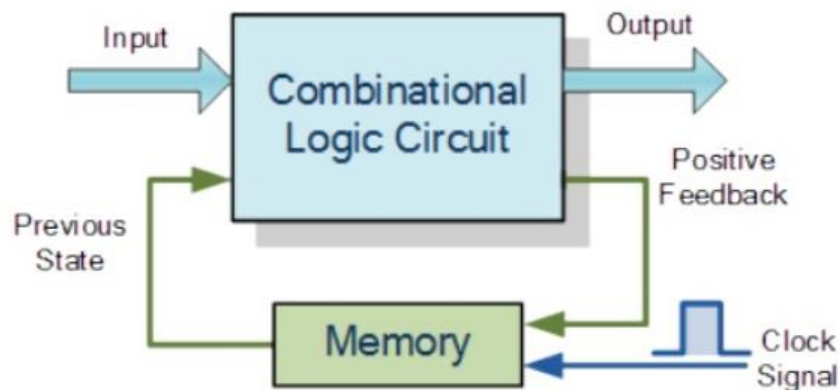
# DIGITAL LOGIC DESIGN

---

## UNIT-4

# Sequential Circuits

- Sequential circuit consists of feedback path and several memory elements



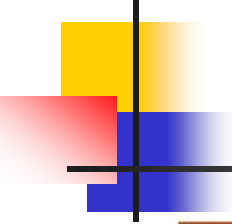
- Sequential circuit = Combinational Logic + Memory Elements

# Memory Elements in Sequential Circuits



---

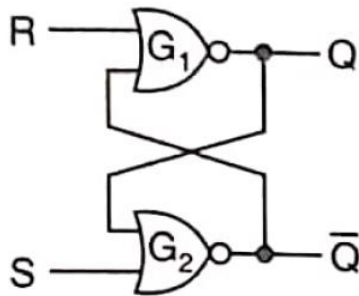
- Memory Element can either be a latch or flip-flop
- Latch output is dependent on input and does not require a clock while flip flop output depends on clock and input.
- One latch or one flip-flop can store 1 bit of information.



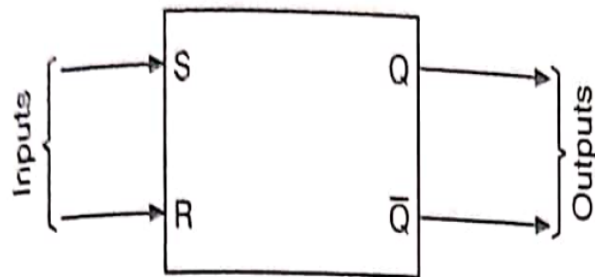
# Differences between Combinational Circuits and Sequential Circuits

Combinational	Sequential
Output of any instance of time depends only upon the input variables	Output is generated dependent upon the present input variables and also on the basis of past history of these inputs
Memory unit is not required. i.e. it doesn't allocate any memory to the elements.	Memory unit is required. . i.e. it allocates any memory to the elements.
Faster	Slower
Easy to design	Difficult
Parallel adder	Serial adder
Ex- Half and full adder Half and full subtractor MUX , DEMUX	Ex- Flip flops Shift registers Binary counters

# SR LATCH USING NOR GATE



Logic diagram



Logic symbol

S	R	$Q_n$	$Q_{n+1}$	State
0	0	0	0	No Change (NC)
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	x	Indeterminate (invalid)
1	1	1	x	

Truth table

# SR LATCH USING NOR GATE

Inputs		Previous state		Next state		
S	R	Q	Q'	Q <sub>+</sub>	Q <sub>+</sub> '	
0	0	0	1	0	1	No Change (NS=PS)
		1	0	1	0	
0	1	0	1	0	1	Reset (NS=0)
		1	0	0	1	
1	0	0	1	1	0	Set (NS=1)
		1	0	1	0	
1	1	0	1	1	1	Indeterminate (NS=1 OR 0)
		1	0	1	1	

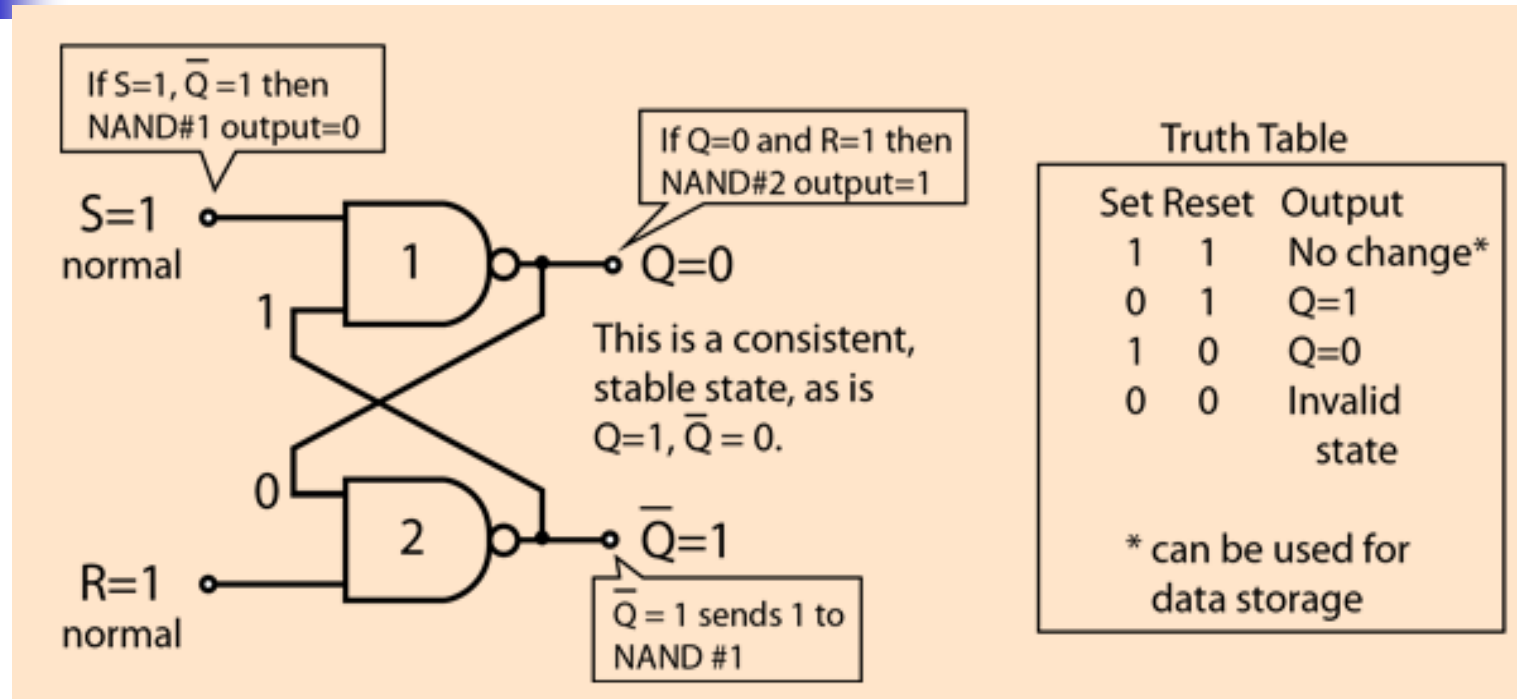


# SR LATCH USING NOR GATE

---

- SET=0, RESET=0: has no effect on the output state. Q and Q' will remain in whatever state they were prior to the occurrence of this input condition. (No change State)
- SET=0, RESET=1: this will always reset Q=0, where it will remain even after RESET returns to 0
- SET=1, RESET=0: this will always set Q=1, where it will remain even after SET returns to 0.
- SET=1, RESET=1; this condition tries to SET and RESET the latch at the same time, and it produces  $Q=Q'=0$ . here outputs are unpredictable. This input condition should not be used.
- Suppose inputs are applied in this way
  - S=0, R=0 then  $Q=0, Q'=1$  (No change)
  - S=0, R=1 then  $Q=0, Q'=1$ , (Reset state)
  - S=1, R=0 then  $Q=1, Q'=0$  (set state)
  - S=0, R=0 then  $Q=1, Q'=0$  (no change)

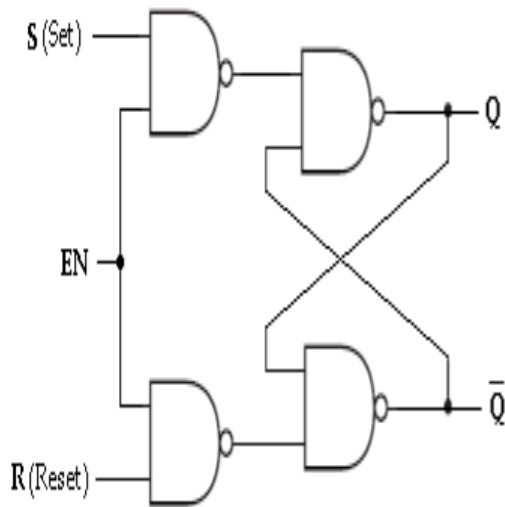
# SR LATCH USING NAND GATE



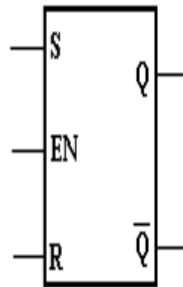
- Here States are reversed



# GATED SR LATCH USING NAND GATE



SR Latch with enable input using NAND gates



Logic Symbol

EN	S	R	$Q_n$	$Q_{n+1}$	State
1	0	0	0	0	No change (NC)
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	X	Indeterminate
1	1	1	1	X	
0	X	X	0	0	No change (NC)
0	X	X	1	1	

Functional Table

WHEN EN=1 SR LATCH OTHERWISE PREVIOUS STATE IS RETAINED



# OTHER LATCHES

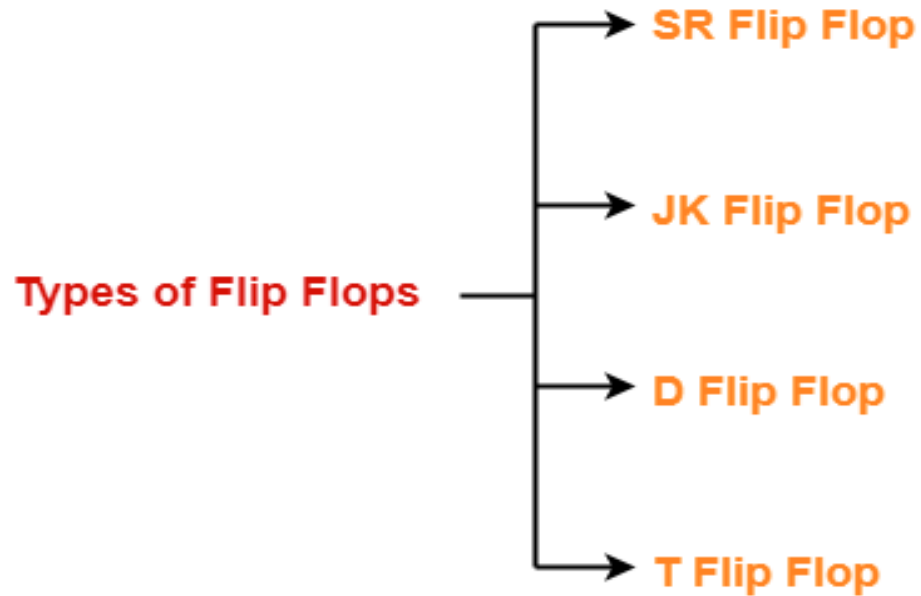
---

- D LATCH
- JK LATCH
- T LATCH
- Working of this will be same as working of flip-flops only difference is latch need not require clock



# FLIP-FLOP

---



# TRIGGERING OF FLIP-FLOP

Level  
Triggering



Positive level



Negative level

Edge  
Triggering



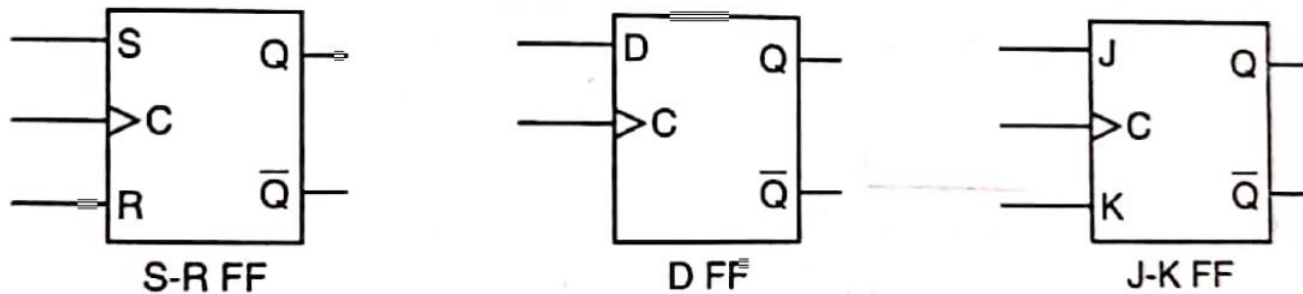
Positive edge



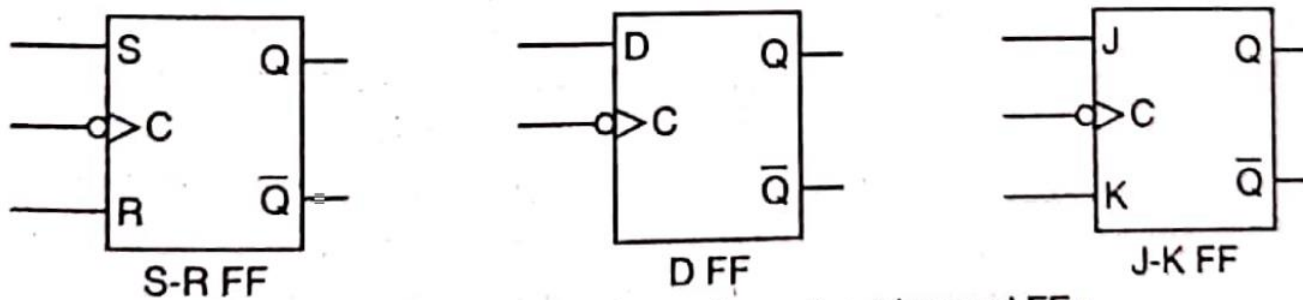
Negative edge

CLOCKS WILL NORMALLY BE EDGE TRIGGERING

# TRIGGERING OF FLIP-FLOP

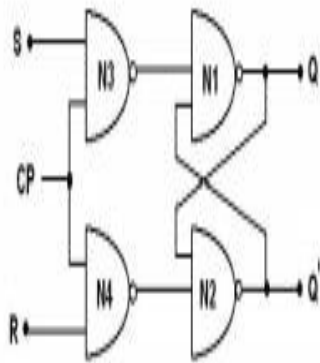


(a) Logic symbols of positive edge-triggered FFs

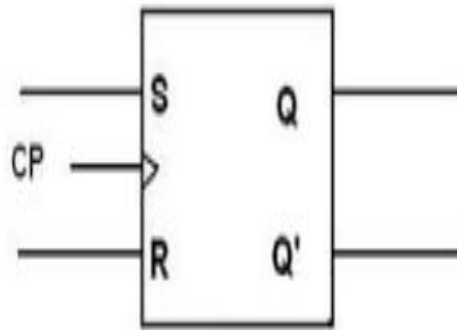


(b) Logic symbols of negative edge-triggered FFs

# SR FLIP-FLOP



a) Logic diagram



b) Block diagram

C	S	R	$Q_n$	$Q_{n+1}$	State
↑	0	0	0	0	No Change (NC)
↑	0	0	1	1	
↑	0	1	0	0	Reset
↑	0	1	1	0	
↑	1	0	0	1	Set
↑	1	0	1	1	
↑	1	1	0	x	Indeterminate
↑	1	1	1	x	
0	x	x	0	0	No Change (NC)
0	x	x	1	1	

c) Functional Table

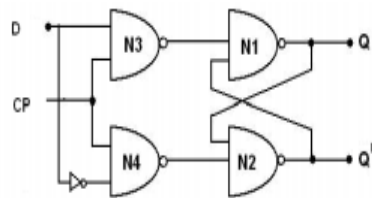


# SR FLIP-FLOP

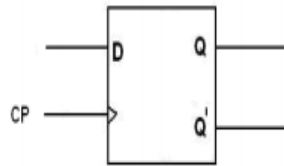
---

1. When  $CP=0$  the output of N3 and N4 are 1 regardless of the value of S and R. This is given as input to N1 and N2. This makes the previous value of Q and  $Q'$  unchanged.
2. When  $CP=1$  the information at S and R inputs are allowed to reach the latch and change of state in flip-flop takes place.
3.  $CP=1, S=1, R=0$  gives the SET state i.e.,  $Q=1, Q'=0$ .
4.  $CP=1, S=0, R=1$  gives the RESET state i.e.,  $Q=0, Q'=1$ .
5.  $CP=1, S=0, R=0$  does not affect the state of flip-flop.
6.  $CP=1, S=1, R=1$  is not allowed, because it is not able to determine the next state. This condition is said to be a race condition.

# D FLIP-FLOP



a) Logic diagram



b) Block diagram

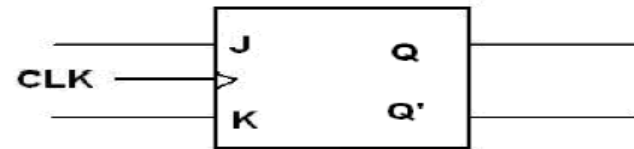
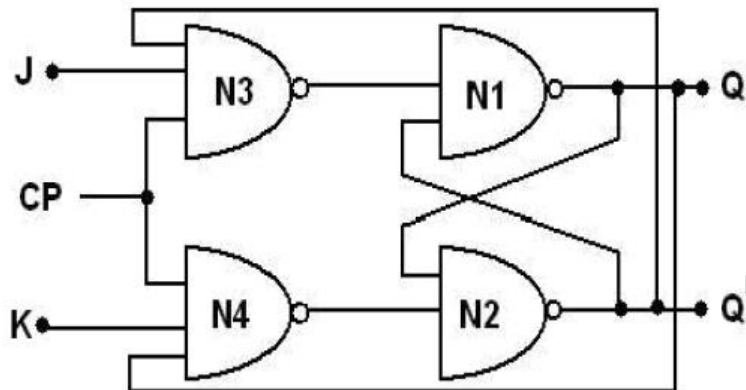
Truth table

CP	D	Q
0	x	Previous state
1	0	0
1	1	1

- The D flip-flop is the modified form of R-S flip-flop. S-R flip-flop is converted to D flip-flop by adding an inverter between S and R and only one input D is taken instead of S and R. So one input is D and complement of D is given as another input. The logic diagram and the block diagram of D flip-flop with clocked input
- When the clock is low both the NAND gates (N1 and N2) are disabled and Q retains its last value. When clock is high both the gates are enabled and the input value at D is transferred to its output Q. D flip-flop is also called —Data flip-flop. (Normally D-Flip-Flop is used in the design of Registers)



# EDGE TRIGGERED JK FLIP-FLOP

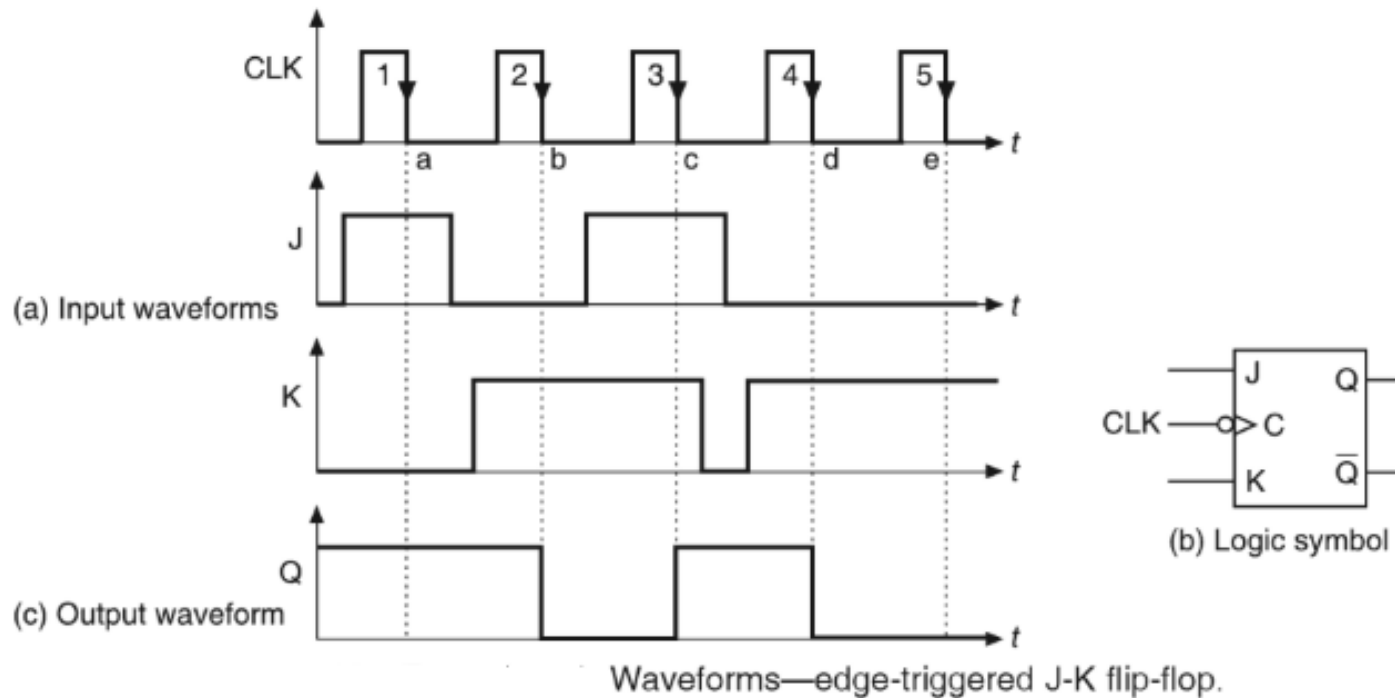


b) Block diagram

J	K	$Q_{(t+1)}$	Comments
0	0	$Q_t$	No change
0	1	0	Reset / clear
1	0	1	Set
1	1	$Q'_t$	Complement/ toggle.

c) Functional Table

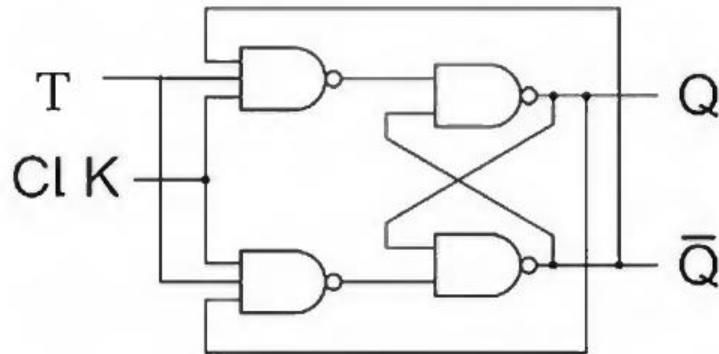
# EDGE TRIGGERED JK FLIP-FLOP



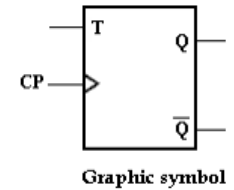
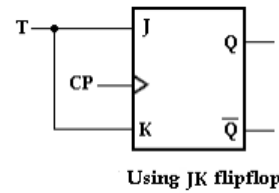
# EDGE TRIGGERED JK FLIP-FLOP

1. When  $J=0$ ,  $K=0$  then both N3 and N4 will produce high output and the previous value of Q and Q' retained as it is(No Change).
2. When  $J=0$ ,  $K=1$ , N3 will get an output as 1 and output of N4 depends on the value of Q. The final output is  $Q=0$ ,  $Q'=1$  i.e., reset state
3. When  $J=1$ ,  $K=0$  the output of N4 is 1 and N3 depends on the value of Q'. The final output is  $Q=1$  and  $Q'=0$  i.e., set state
4. When  $J=1$ ,  $K=1$ , If  $Q=1$ ,  $Q'=0$  then N4 passes '0' to N2 which produces  $Q'=1$ ,  $Q=0$  which is reset state. When  $J=1$ ,  $K=1$ , Q changes to the complement of the last state. The flip-flop is said to be in the toggle state. (This State is normally used in the design of Counters)

# EDGE TRIGGERED T FLIP-FLOP



If  $T=0$   $NS=PS$  and if  $T=1$   $NS=PS'$ .  
 $NS = \text{NEXT STATE}(Q_{N+1})$   
 $PS = \text{PREVIOUS STATE}(Q_N)$

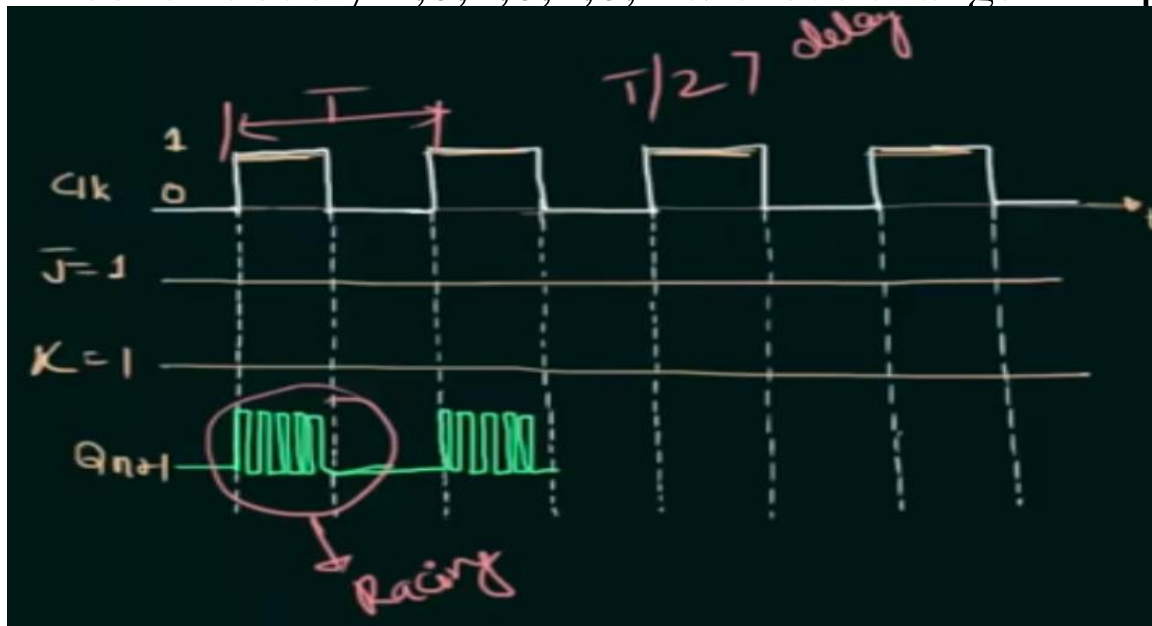


T	$Q_{n+1}$	State
0	$Q_n$	No Change
1	$Q_n'$	Toggle

Truth table for T Flip-Flop

# RACE AROUND CONDITION JK FLIP-FLOP

1. In level triggering, if  $J=1$ ,  $K=1$ , the output changes its state continuously 1,0,1,0,1,0,1 without change in input.



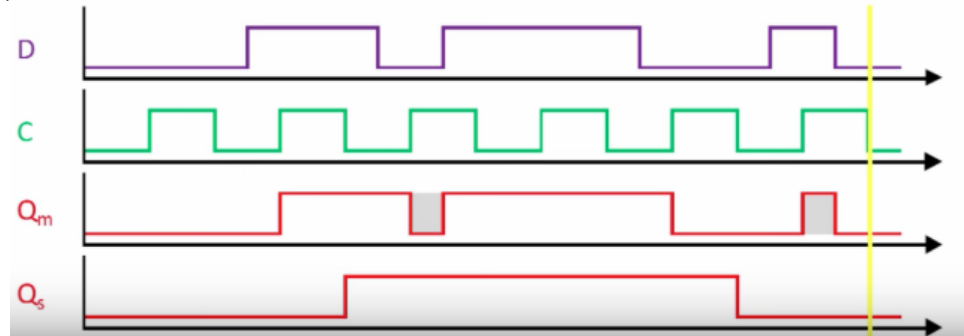
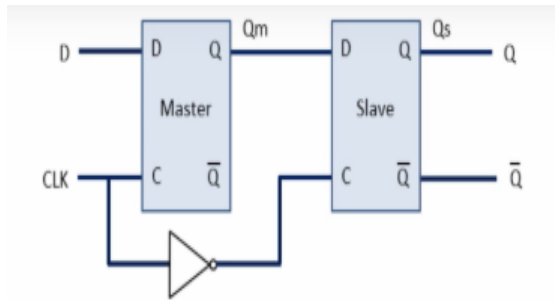
[https://www.youtube.com/watch?v=trPGhO7MPnw&itct=CAsQpDAYCSITCLDq0NCn5toCFVTZwQodmDYD3jIGcmVsbWZ1SPD0oZnyg\\_OXPg%3D%3D&app=desktop](https://www.youtube.com/watch?v=trPGhO7MPnw&itct=CAsQpDAYCSITCLDq0NCn5toCFVTZwQodmDYD3jIGcmVsbWZ1SPD0oZnyg_OXPg%3D%3D&app=desktop)

(See this video to understand about it)

To avoid this go for edge triggering

# MASTER SLAVE D FLIP-FLOP

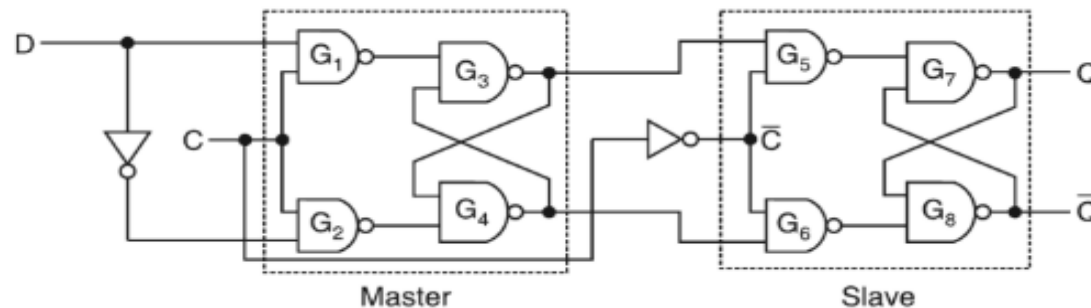
To avoid Race Around Condition we go for Master Slave Flip-Flops. We can use SR or JK or D or T Flip-Flops



- Master will be active at positive level of the clock  $Q_m = D$  and  $Q_s = PS$  (Previous State)
- Slave will be active at the negative level of the clock  $Q_m = PS$ , and  $Q_s = Q_m$
- If  $clk = 1$   $Q_m = D$  but output will still be the previous state ( $Q_s$ ). So here without change in input the output remains same. Hence Race Around Condition is avoided.

# MASTER SLAVE D FLIP-FLOP

To avoid Race Around Condition we go for Master Slave Flip- Flops. We can use SR or JK or D or T Flip-Flops

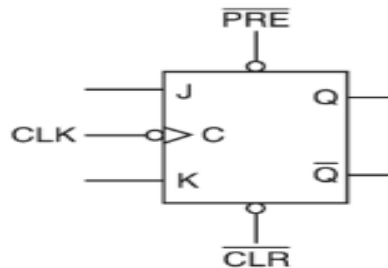


(a) Logic diagram

Inputs		Output	Comments
D	CLK	Q	
0		0	RESET
1		1	SET

(b) Truth table

# JK FLIP-FLOP WITH PRESET AND CLEAR

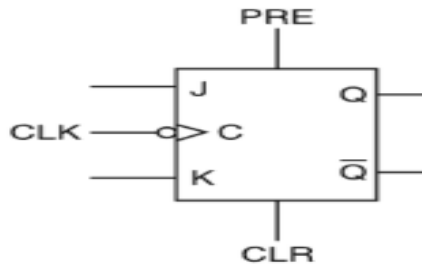


(a) Logic symbol

DC SET (PRE)	DC RESET (CLR)	FF response
0	0	Not used
1	0	$Q = 0$
0	1	$Q = 1$
1	1	Clocked operation

(b) Truth table

J-K flip-flop with active-LOW PRESET and CLEAR inputs.



(a) Logic symbol

DC SET (PRE)	DC RESET (CLR)	FF response
0	0	Clocked operation
0	1	$Q = 0$
1	0	$Q = 1$
1	1	Not used

(b) Truth table

J-K flip-flop with active-HIGH PRESET and CLEAR inputs.



# FLIP-FLOP EXCITATION TABLE

Used in conversion of Flip-Flops and design of Synchronous Counter

$Q_t$	$Q_{t+1}$	S	R	J	K	D	T
0	0	0	x	0	x	0	0
0	1	1	0	1	x	1	1
1	0	0	1	x	1	0	1
1	1	x	0	x	0	1	0

- Depending on output, input is derived
- See this video how excitation table is written for SR Flip-Flop. Same Procedure can be adopted for other flip-flops.
- <https://www.youtube.com/watch?v=uiKKRPZbuXA>



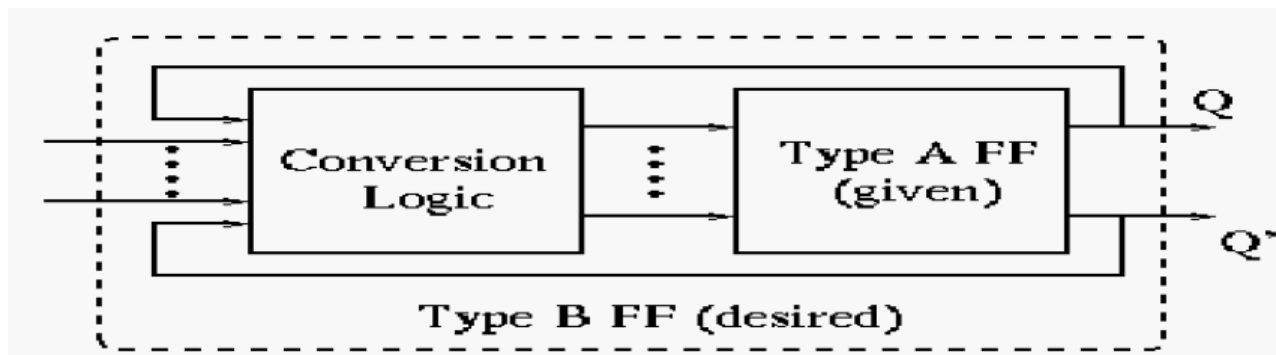
# SR FLIP-FLOP

Inputs		Previous state	Next state
S	R	Q	Q <sub>+</sub>
0	0	0	0
		1	1
0	1	0	0
		1	0
1	0	0	1
		1	1

# FLIP-FLOP CONVERSIONS

Procedure:

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.



# JK FLIP-FLOP TO D-FLIP-FLOP CONVERSION

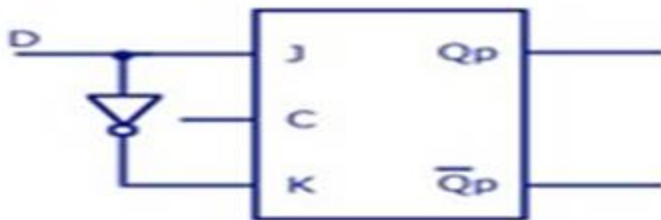
D Input	Outputs $Q_p$ $Q_{p+1}$		J-K Inputs J K	
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	1	X	0

D	0	1
	0	1
0	0	X
1	1	X

$J = D$

D	0	1
	0	1
0	X	1
1	X	0

$K = \overline{D}$



1. Available=JK, Required=D
2. Write characteristic table(functional table) of D flip-flop.
3. Write Excitation table of JK Flip-Flop
4. Write Boolean Expressions for JK Flip-flop
5. Draw the circuit.

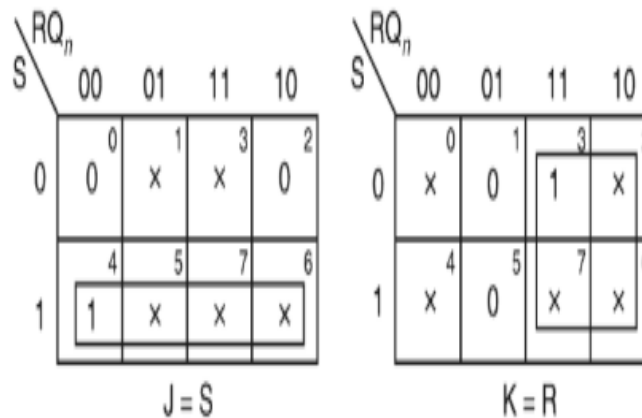
<https://www.youtube.com/watch?v=ApJ972OYyXQ>

(See this video if procedure is not clear)

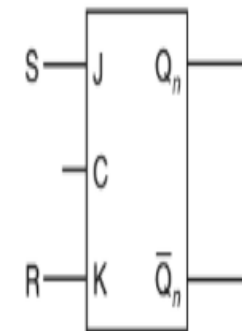
# JK FLIP-FLOP TO SR-FLIP-FLOP CONVERSION

External Inputs		Present State	Next State	Flip-flop Inputs	
S	R	$Q_n$	$Q_{n+1}$	J	K
0	0	0	0	0	x
0	0	1	1	x	0
0	1	0	0	0	x
0	1	1	0	x	1
1	0	0	1	1	x
1	0	1	1	x	0

(a) Conversion table



(b) K-maps for J and K



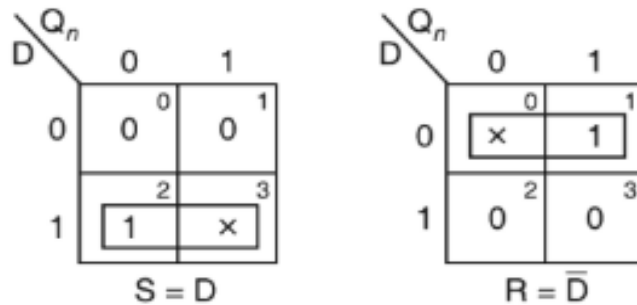
(c) Logic diagram

1. Available=JK, Required=SR
2. Write characteristic table(functional table) of SR flip-flop.
3. Write Excitation table of JK Flip-Flop
4. Write Boolean Expressions for JK Flip-flop
5. Draw the circuit.

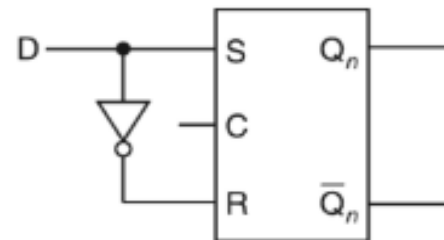
# SR FLIP-FLOP TO D-FLIP-FLOP CONVERSION

External Inputs	Present State	Next State	Flip-flop Inputs	
D	$Q_n$	$Q_{n+1}$	S	R
0	0	0	0	x
0	1	0	0	1
1	0	1	1	0
1	1	1	x	0

(a) Conversion table



(b) K-maps for S and R



(c) Logic diagram



# SEQUENTIAL CIRCUITS

---

---

## Synchronous sequential circuits

---

1. In synchronous circuits, memory elements are clocked FFs.
2. In synchronous circuits, the change in input signals can affect memory elements upon activation of clock signal.
3. The maximum operating speed of the clock depends on time delays involved.
4. Easier to design.

---

## Asynchronous sequential circuits

---

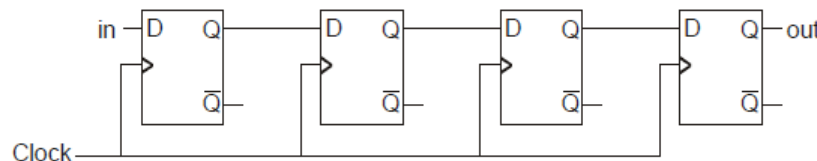
1. In asynchronous circuits, memory elements are either unclocked FFs or time delay elements.
  2. In asynchronous circuits, change in input signals can affect memory elements at any instant of time.
  3. Because of the absence of the clock, asynchronous circuits can operate faster than synchronous circuits.
  4. More difficult to design.
-

# SHIFT REGISTERS

Different types of Shift Registers are available

- Serial In - Serial Out Shift Register(SISO)
- Serial In- Parallel Out Shift Register(SIPO)
- Parallel In - Serial Out Shift Register(PISO)
- Parallel In - Parallel Out Shift Register (PIPO)
- Bidirectional Shift Registers for 2 types of Shift
- Universal Shift Register

## Serial In - Serial Out Shift Register(SISO)

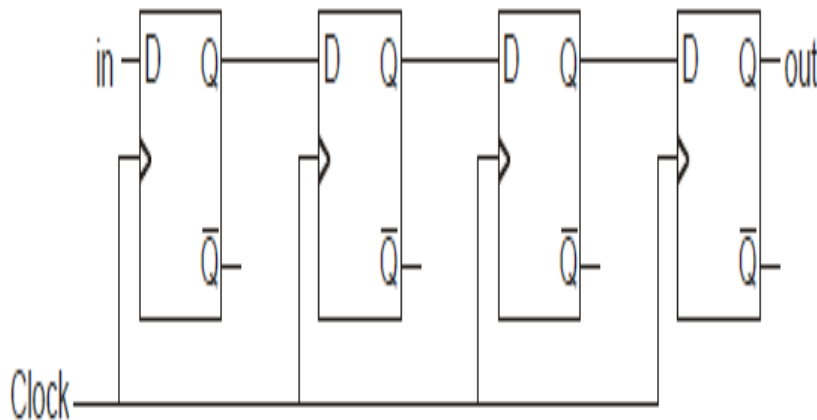


- Serial means single input and serial output means only one output
- The input data is applied sequentially to the D input of the first flip-flop on the left (FF0). During each clock pulse, one bit is transmitted from left to right.
- Assume a data input to be 1. the data input reaches FF3 after 4 clock pulses.



# SHIFT REGISTERS

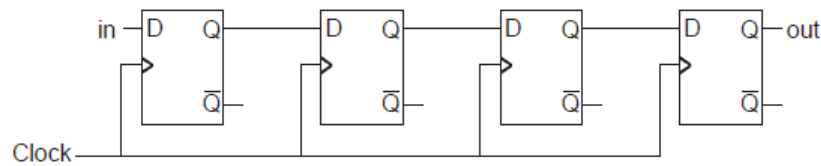
Serial In - Serial Out Shift Register(SISO)



No of Cloc k	Input	Q <sub>0</sub>	Q <sub>1</sub>	Q <sub>2</sub>	out
0	X	0	0	0	0
1	1	1	0	0	0
2	0	0	1	0	0
3	1	1	0	1	0
4	1	1	1	0	1

# SHIFT REGISTERS

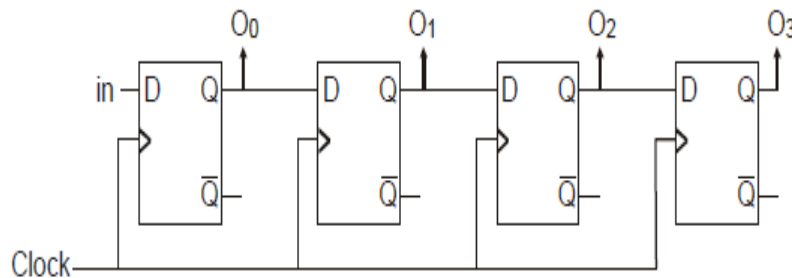
## Serial In - Serial Out Shift Register(SISO)



No of Clock	Input	$D_B$	$D_C$	$D_D$	$Q_0$	$Q_1$	$Q_2$	$Q_3$
0	X	$PS(Q_0)$	$PS(Q_1)$	$PS(Q_2)$	0	0	0	0
1	1	0	0	0	1	0	0	0
2	0	1	0	0	0	1	0	0
3	1	0	1	0	1	0	1	0
4	1	1	0	1	1	1	0	1

# SERIAL IN- PARALLEL OUT SHIFT REGISTERS

- Serial means single input and parallel output means output is taken in parallel
- At each clock,  $Q_0 = \text{in}$ ,  $Q_1 = \text{PS of } Q_0$ ,  $Q_2 = \text{PS of } Q_1$ ,  $Q_3 = \text{PS of } Q_2$
- After 4 clocks  $Q_3 = \text{D}$ .



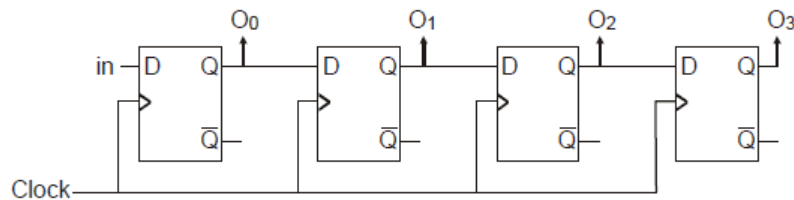
Logic Diagram

No of Cloc k	Input	$Q_0$	$Q_1$	$Q_2$	$Q_3$
0	X	0	0	0	0
1	1	1	0	0	0
2	0	0	1	0	0
3	1	1	0	1	0
4	1	1	1	0	1

Functional Table

Similarly other type of Shift Registers

# SERIAL IN- PARALLEL OUT SHIFT REGISTERS



Logic Diagram

No of Clock	Input	$D_B$	$D_C$	$D_D$	$Q_0$	$Q_1$	$Q_2$	$Q_3$
0	X	PS( $Q_0$ )	PS( $Q_1$ )	PS( $Q_2$ )	0	0	0	0
1	1	0	0	0	1	0	0	0
2	0	1	0	0	0	1	0	0
3	1	0	1	0	1	0	1	0
4	1	1	0	1	1	1	0	1

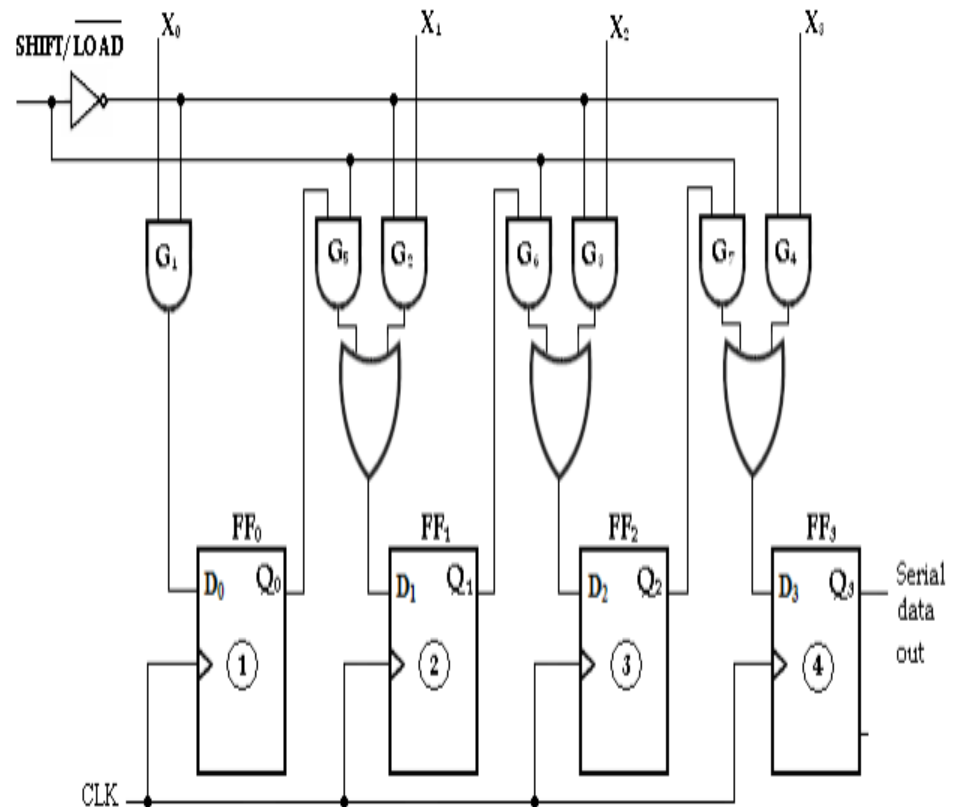
# PARALLEL IN SERIAL OUT SHIFT REGISTER

In this type, the bits are entered in parallel i.e., simultaneously into their respective stages on parallel lines.

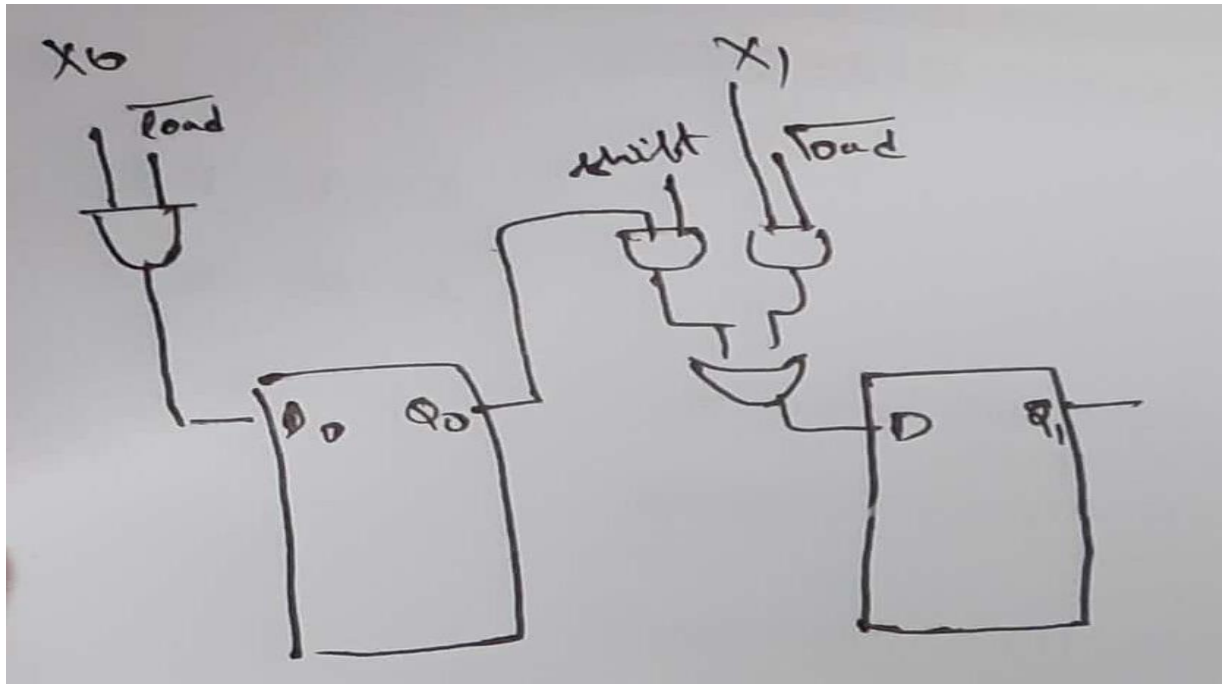
There are four data input lines,  $X_0$ ,  $X_1$ ,  $X_2$  and  $X_3$  for entering data in parallel into the register.

SHIFT/LOAD input is the control input, which allows four bits of data to load in parallel into the register.

When SHIFT/LOAD is LOW, gates  $G_1$ ,  $G_2$ ,  $G_3$  and  $G_4$  are enabled, allowing each data bit to be applied to the D input of its respective Flip-Flop. When a clock pulse is applied, the Flip-Flops with  $D = 1$  will set and those with  $D = 0$  will reset, thereby storing all four bits simultaneously.



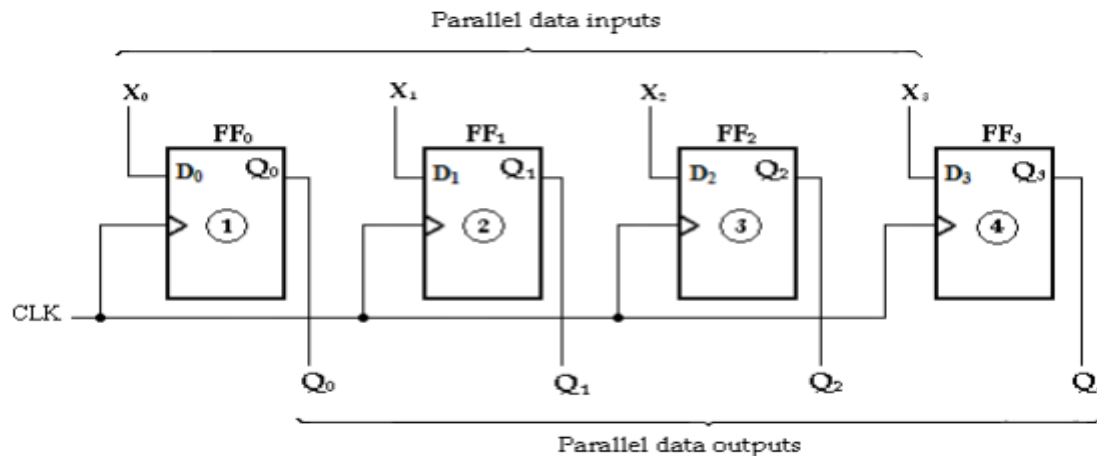
# PARALLEL IN SERIAL OUT SHIFT REGISTER



# PARALLEL IN SERIAL OUT SHIFT REGISTER

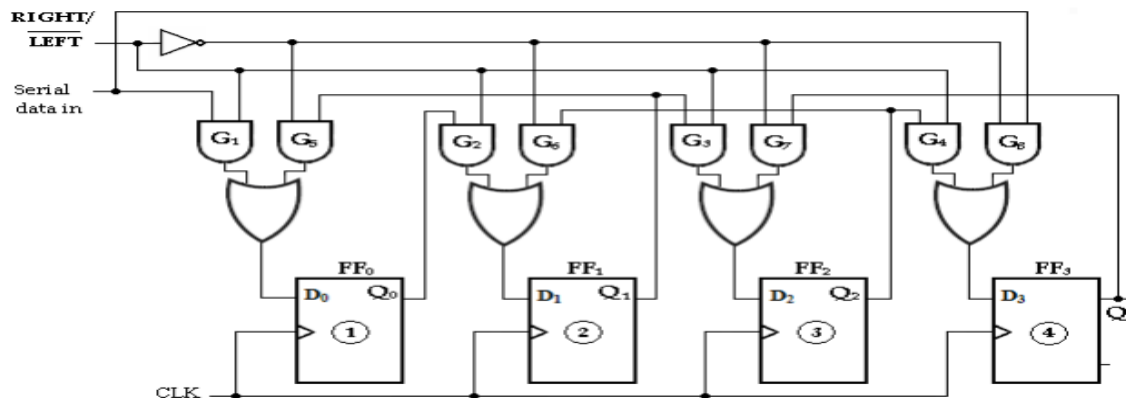
When SHIFT/LOAD is HIGH, gates G1, G2, G3 and G4 are disabled and gates G5, G6 and G7 are enabled, allowing the data bits to shift right from one stage to the next. The OR gates allow either the normal shifting operation or the parallel data entry operation, depending on which AND gates are enabled by the level on the SHIFT/LOAD input.

## Parallel-In Parallel-Out Shift Register:



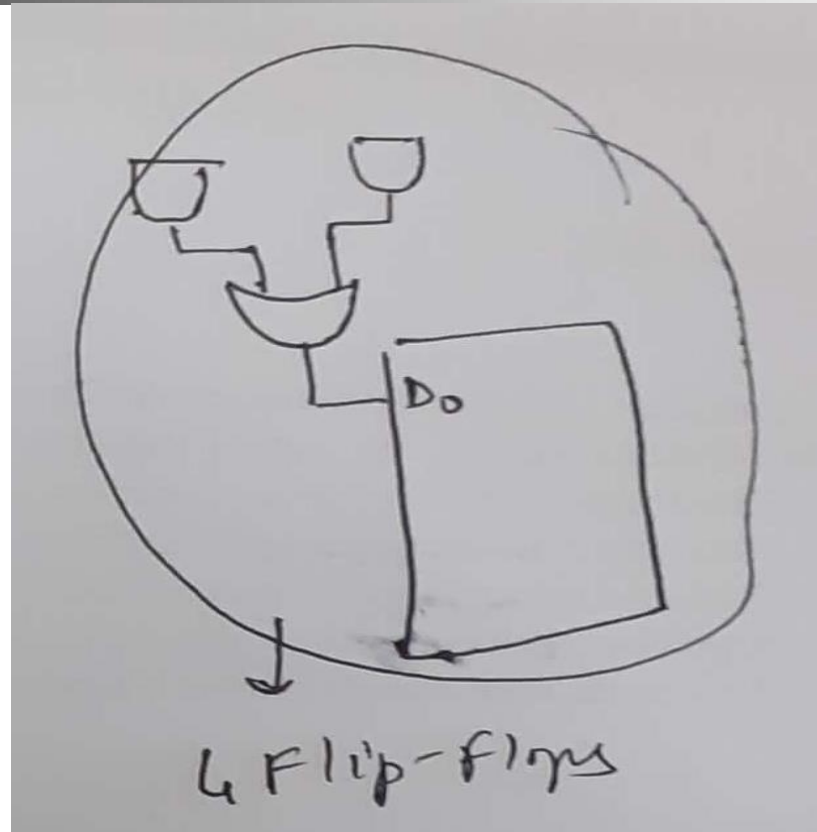
# BI-DIRECTIONAL SHIFT REGISTER

- Here data can be shifted either left or right.
- A HIGH on the RIGHT/LEFT control input allows data bits inside the register to be shifted to the right, and a LOW enables data bits inside the register to be shifted to the left. When the RIGHT/LEFT control input is HIGH, gates G1, G2, G3 and G4 are enabled, and the state of the Q output of each Flip-Flop is passed through to the D input of the following Flip-Flop. When a clock pulse occurs, the data bits are shifted one place to the right.
- When the RIGHT/LEFT control input is LOW, gates G5, G6, G7 and G8 are enabled, and the Q output of each Flip-Flop is passed through to the D input of the preceding Flip-Flop. When a clock pulse occurs, the data bits are then shifted one place to the left.

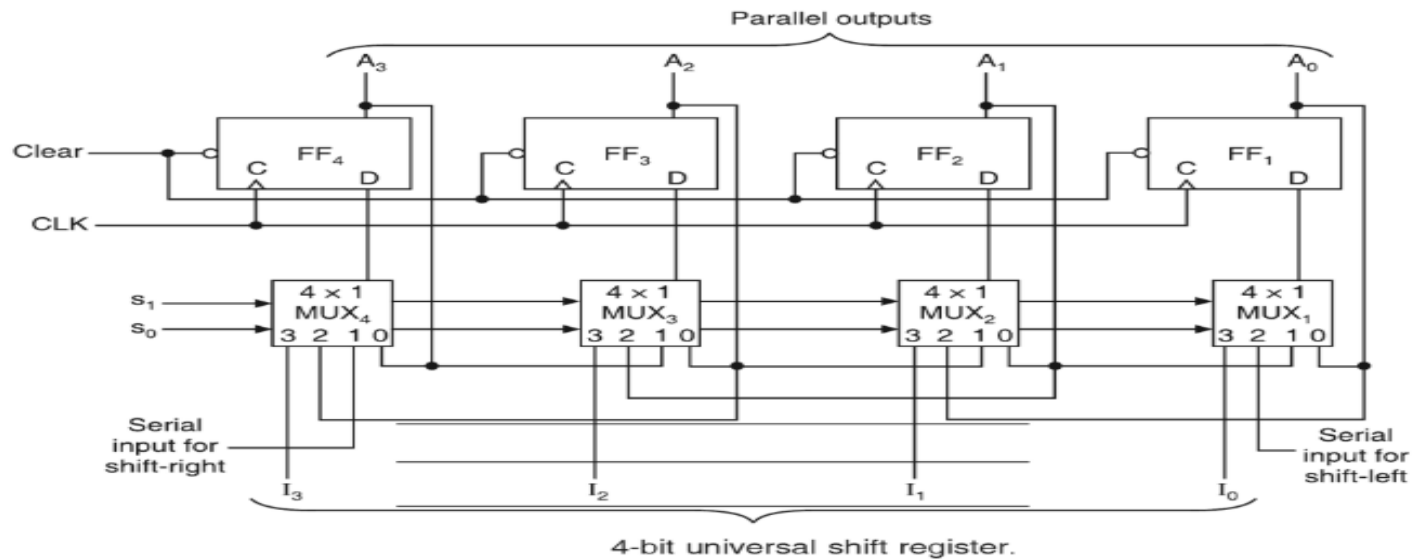




# BI-DIRECTIONAL SHIFT REGISTER



# UNIVERSAL SHIFT REGISTER



Function table for the register

Mode control		Register operation
$S_1$	$S_0$	
0	0	No change
0	1	Shift right
1	0	Shift left
1	1	Parallel load

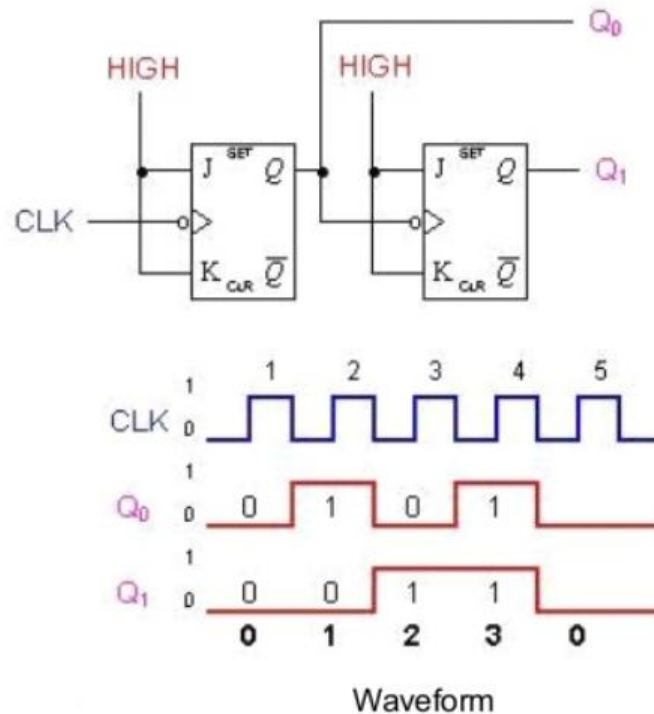


# COUNTER

---

- A *counter* is a register that goes through a predetermined sequence of states upon the application of clock pulses.
  - Asynchronous counters
  - Synchronous counters
- **Asynchronous Counters** (or *Ripple counters*)
  - the clock signal (CLK) is only used to clock the first FF.
  - Each FF (except the first FF) is clocked by the preceding FF.
- **Synchronous Counters**,
  - the clock signal (CLK) is applied to all FF, which means that all FF shares the same clock signal,
  - thus the output will change at the same time.

# 2 BIT ASYNCHRONOUS COUNTER OR MOD-4 UP COUNTER

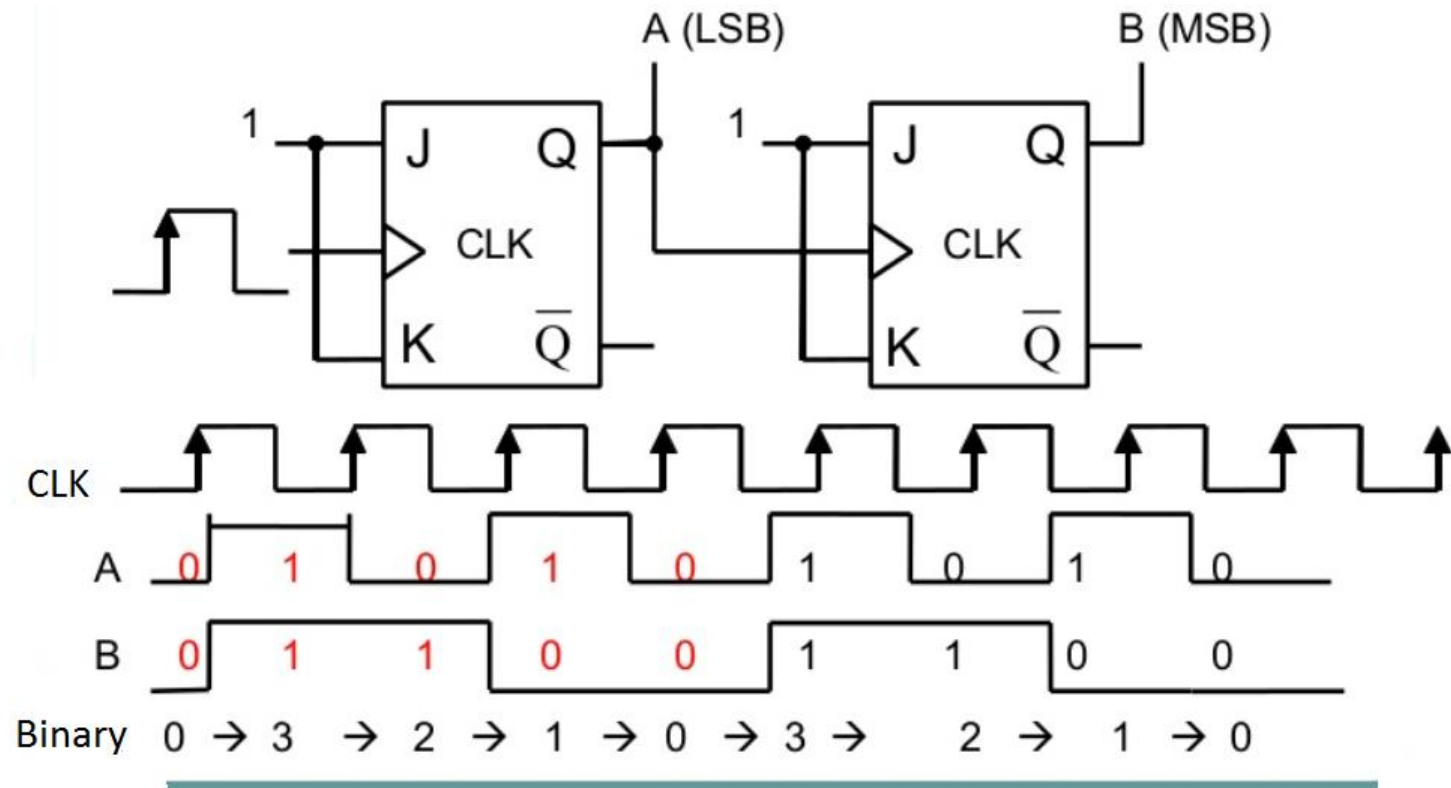


**A two-bit asynchronous counter is shown on the left. The external clock is connected to the clock input of the first flip-flop (FF0) only. So, FF0 changes state at the falling edge of each clock pulse, but FF1 changes only when triggered by the falling edge of the Q output of FF0.**

If working not clear view this video

<https://www.youtube.com/watch?v=iaIu5SYmWVM&list=PLuYnCh-Sh1Xd5cLa-CfK883tPmJwrjSwF>

# ASYNCHRONOUS MOD-4 DOWN COUNTER





# DESIGN PROCEDURE

# ASYNCHRONOUS COUNTER

---

- Each Flip-Flop will give 2 states 0 and 1. So N flip-flop will give  $2^n$  states
- First Determine no of Flip-Flop Required. If it is a 2 bit 2 flip-flop are required.

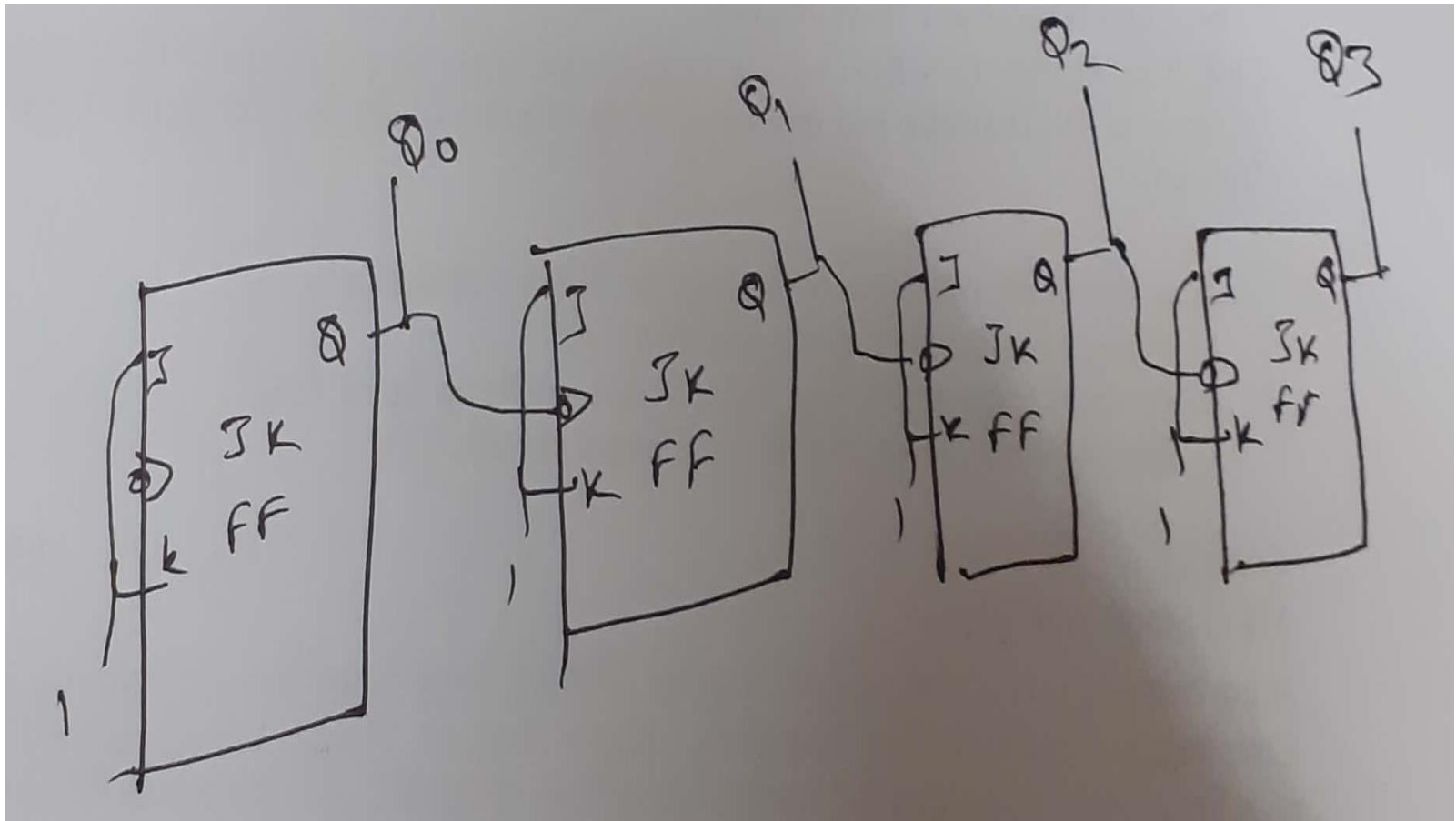
MOD-n Counter :(If n is a multiple of 2 )then

- if Mod-16 counter is written then it has 16 states starting from 0000 to 1111 in binary. or to get 16 states, n has to be 4.

MOD-n Counter :

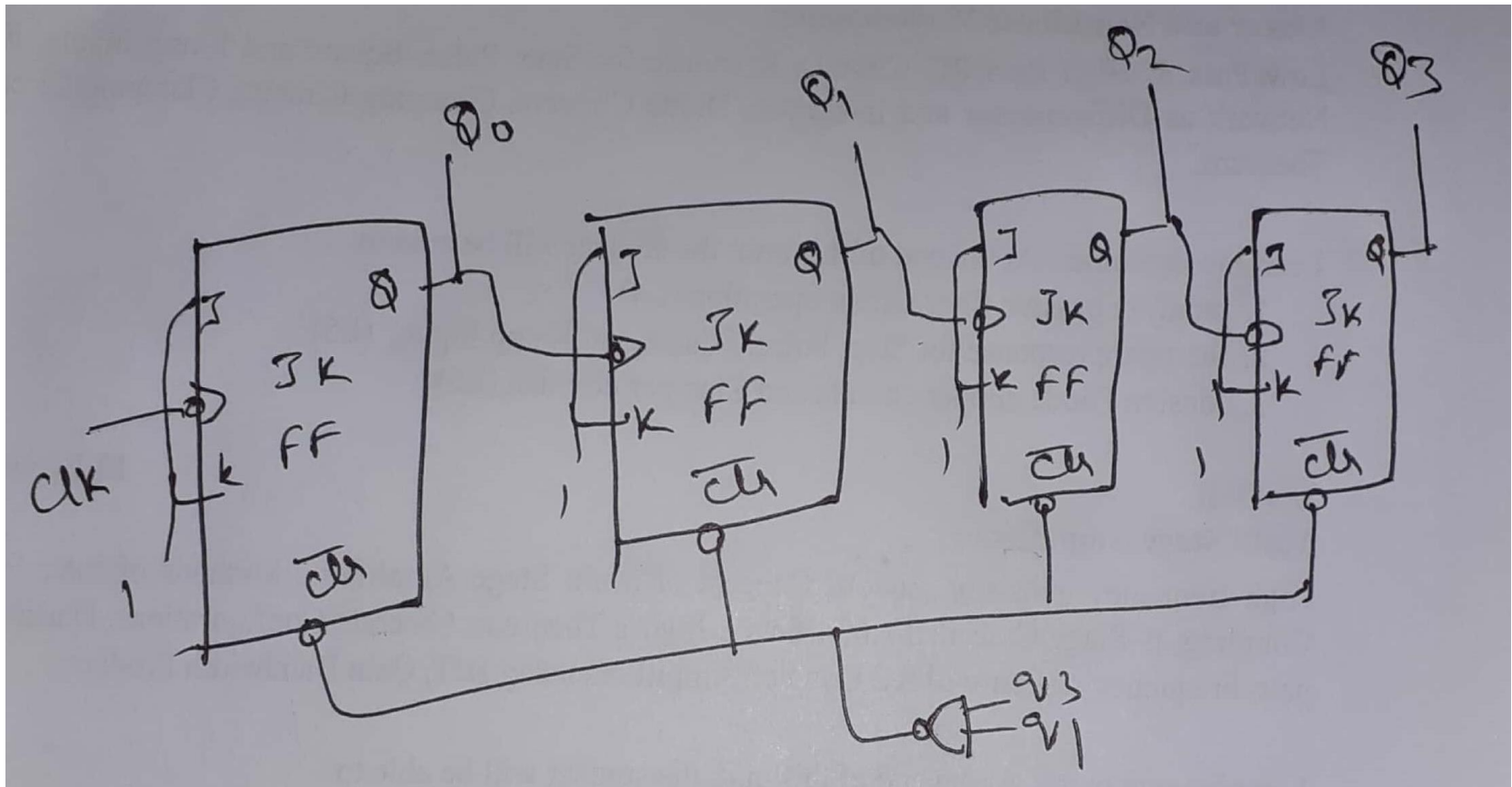
- If n is not a multiple of 2
- First identify no of states, then write the state in binary format. Then the maximum state is considered if it is a r bit then r flip flop are required. But r flip-flop will give  $2^r$  States. In order to get maximum state use AND gate for active clear or NAND gate for low clear
- Ex:MOD-10 counter means it has 10 states starting from 0000 to 1001 in binary since the last state is a 4 bit number 4 flip flop are required. But 4 flip-flops will give output as 16 states. So we design the circuit in such a way at 1010 the flip-flop has to be reset to zero by connecting NAND gate with inputs as Q3,Q1 for low clear.

# MOD-10 ASYNCHRONOUS COUNTER



# MOD-10(1010)

## ASYNCHRONOUS COUNTER





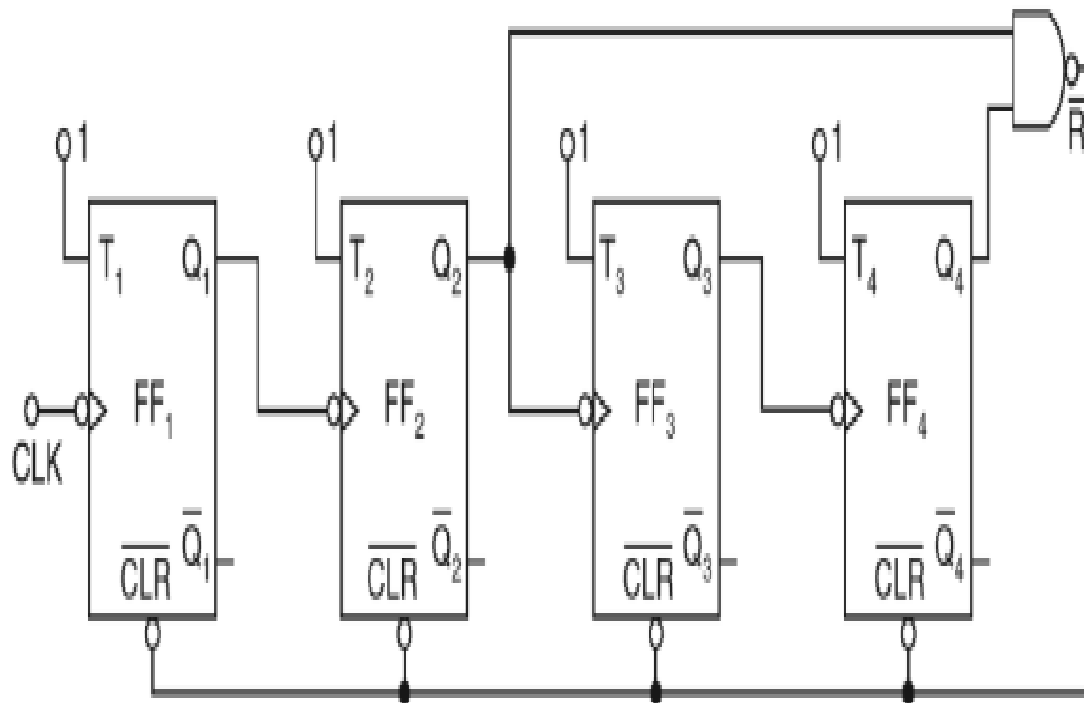
# MOD-10 ASYNCHRONOUS COUNTER

Handwritten truth table for a MOD-10 asynchronous counter:

$q_3$	$q_2$	$q_1$	$q_0$	$\overline{Clr}$
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Don't care state: 1000 ( $q_3 q_2 q_1 q_0$ )

# MOD-10 ASYNCHRONOUS COUNTER



(c) Logic diagram

After pulses	Count			
	$Q_4$	$Q_3$	$Q_2$	$Q_1$
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
10	0	0	0	0

(a) Count table

# MOD-10 ASYNCHRONOUS COUNTER

Working:

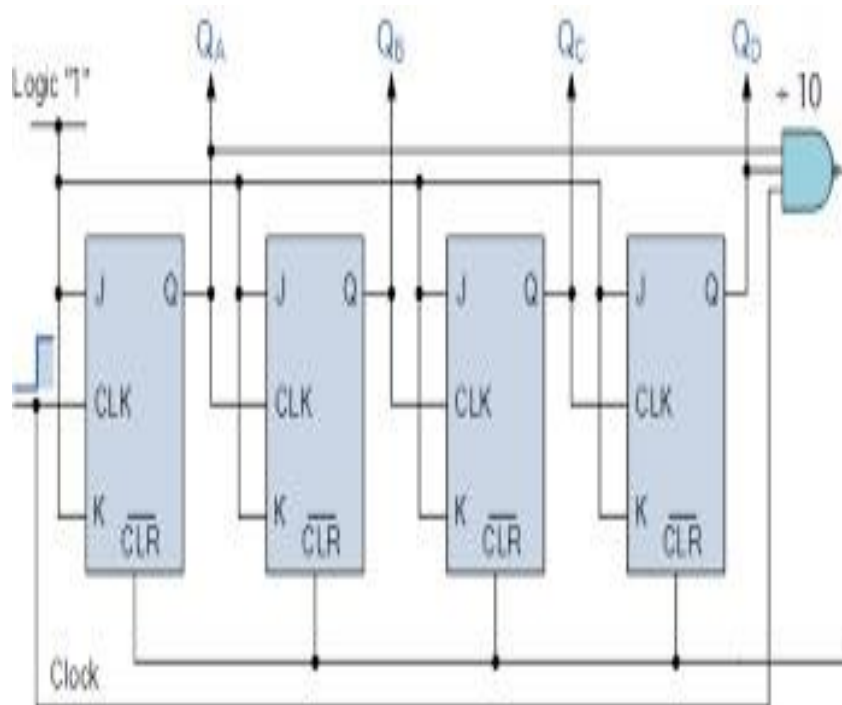
- At first clock output will be 0001 then  $\text{clr}=1$  as  $Q_D=0, Q_B=0$  then the output will change to 0010 at next clock then  $\text{clr}=1$  as  $Q_D=0, Q_B=1$ . This process will continue until output reaches 1001
- When output is 1001  $\text{clr}=1$  as  $Q_D=1, Q_B=0$  so at next clock output reaches 1010 but immediately  $\text{clr}=0$  as  $Q_D=1, Q_B=1$  then all the flip-flop will reset to 0000 and then once again count continues

If any doubts view this video

[https://www.youtube.com/watch?v=fKVZpupyP\\_o](https://www.youtube.com/watch?v=fKVZpupyP_o)

Disadvantages: It will take more time for operation as clock is not common for all flip-flops but it is easy to design

# MOD-9 ASYNCHRONOUS COUNTER



Logic Diagram

0000, 0001, 0010, ..., 1000, 0000

1001  
Q<sub>3</sub> Q<sub>2</sub> Q<sub>1</sub> Q<sub>0</sub>

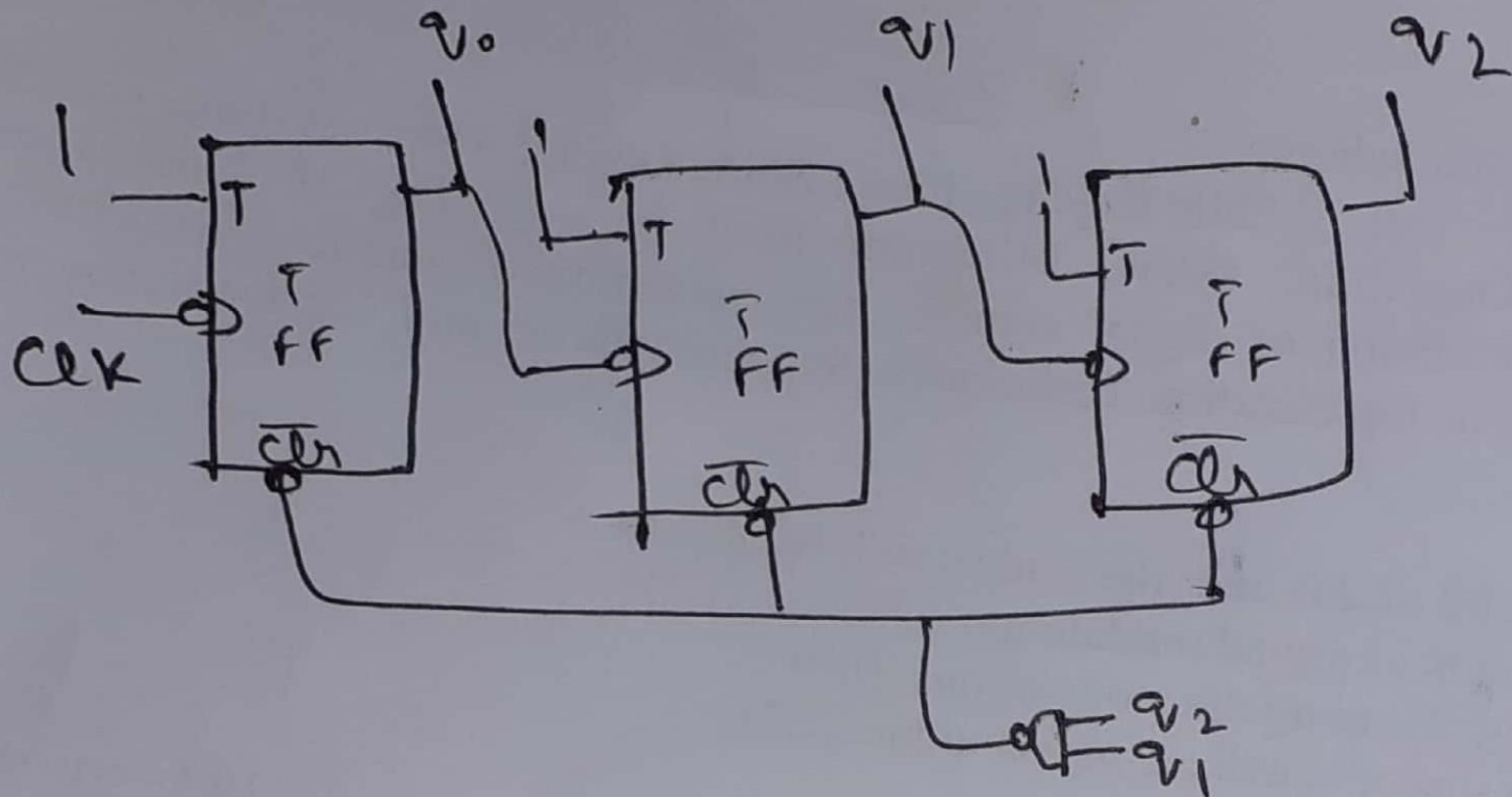
Functional Table here

$Q_D=A, Q_C=B, Q_B=C, Q_A=D$

Mod-6

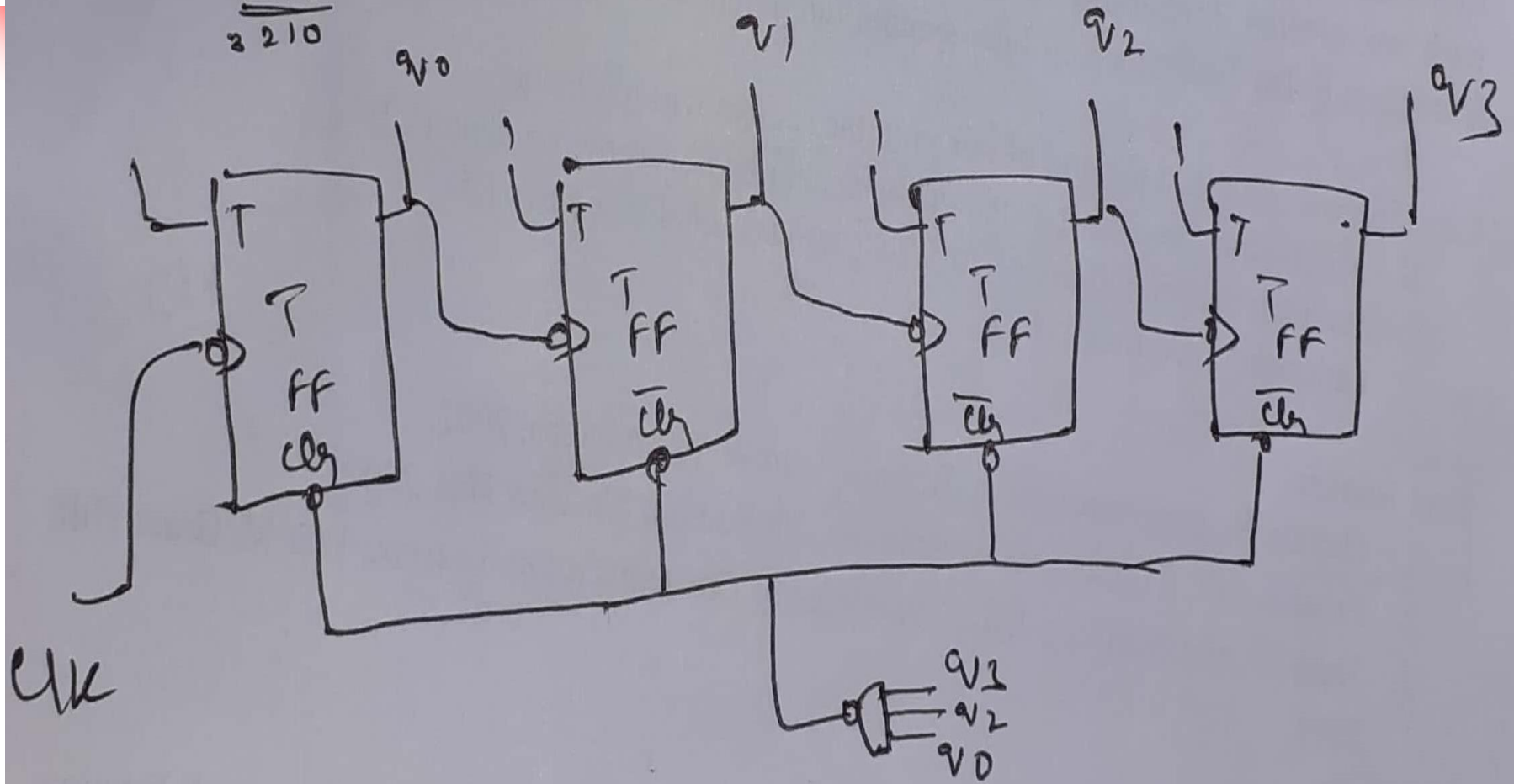
110

000  $\rightarrow$  101



Mod - 13: (0000 - 1100)

1101  
2 2 1 0



# SYNCHRONOUS COUNTER DESIGN PROCEDURE



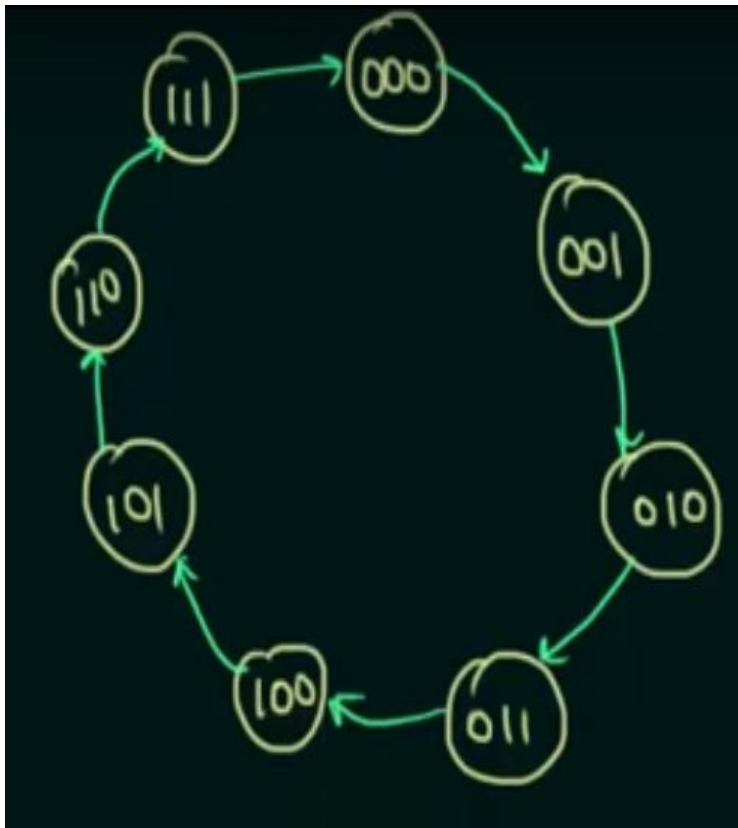
---

Advantage: Faster in operation and more components are required

Design Steps:

1. The given problem is determined with a state diagram.
2. From the state diagram, obtain the state table.
3. Assign binary values to each state (Binary Assignment) if the state table contains letter symbols.
4. Determine the number of Flip-Flops.
5. Choose the type of Flip-Flop (SR, JK, D, T) to be used.
6. From the state table, circuit excitation and output tables.
7. Using K-map or any other simplification method, derive the circuit output functions and the Flip-Flop input functions.
8. Draw the logic diagram.

# MOD-8 SYNCHRONOUS COUNTER USING T FF



State Diagram

P.S-			N.S			T <sub>C</sub> T <sub>B</sub> T <sub>A</sub>		
Q <sub>C</sub>	Q <sub>B</sub>	Q <sub>A</sub>	Q <sub>C</sub> <sup>+</sup>	Q <sub>B</sub> <sup>+</sup>	Q <sub>A</sub> <sup>+</sup>	T <sub>C</sub>	T <sub>B</sub>	T <sub>A</sub>
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

State table along with excitation table



# MOD-8 SYNCHRONOUS COUNTER USING T FF

for  $T_C$

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

$T_C = Q_B Q_A$

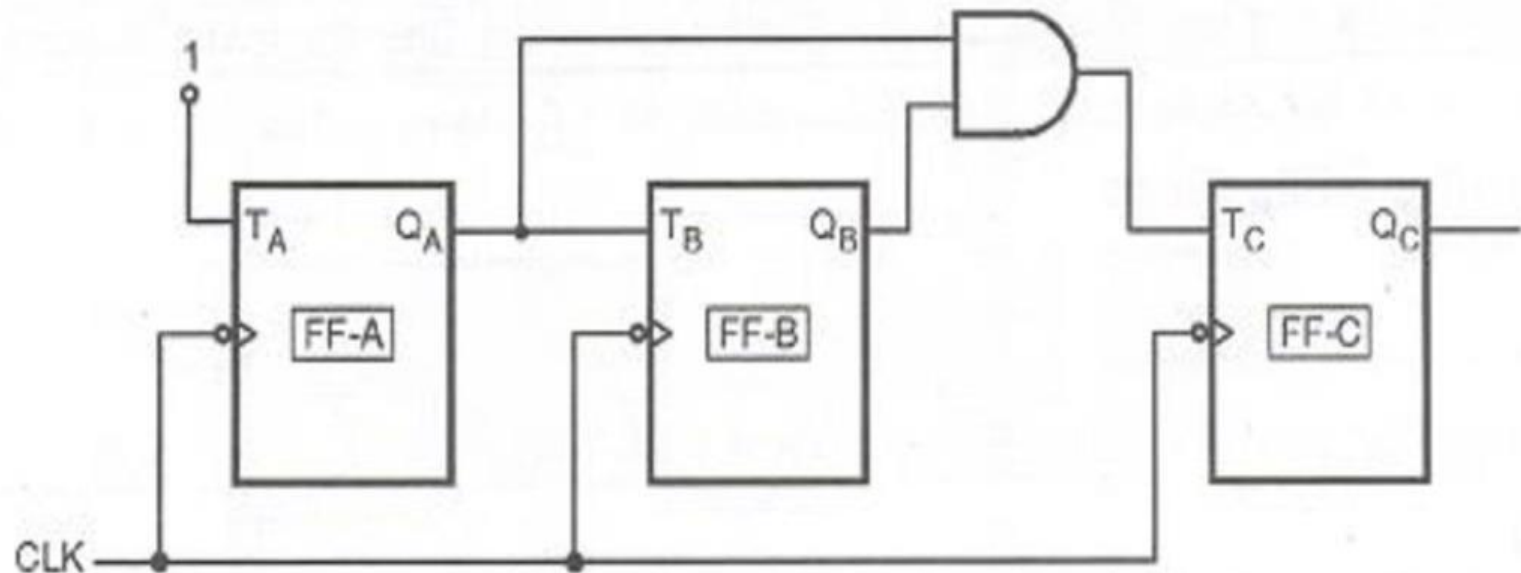
for  $T_B$

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

$T_B = Q_A$

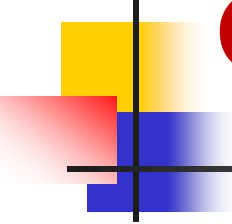
$T_A = 1$  AS ALL ARE 1 OR CAN VERIFY THROUGH K-MAPS

# MOD-8 SYNCHRONOUS COUNTER USING T FF



3 bit synchronous counter using T-flip flops

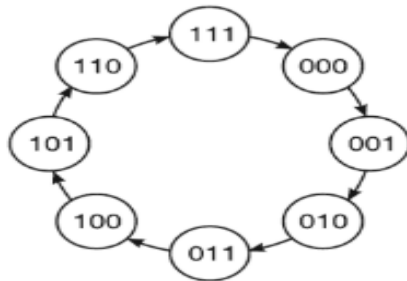
# MOD-8 SYNCHRONOUS COUNTER USING T FF

- 
- draw the State Diagram with 8 states starting from 000 to 111
  - As last number is 111 3 flip-flops are required
  - From the State Diagram write the State Table if PS=000 then NS=001
  - If PS=001 then NS=010 like this write for every state and if PS=111 then NS=000
  - After that write excitation table using PS,NS
  - Then obtain Boolean Expressions for  $T_a$ ,  $T_b$ ,  $T_c$
  - Then draw the logic diagram

See this video if you have doubts

<https://www.youtube.com/watch?v=6e8oV2blkGs>

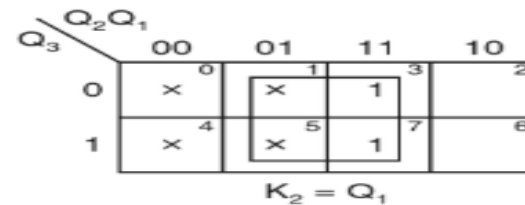
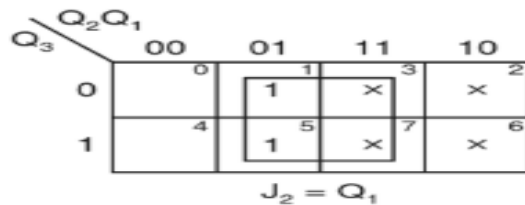
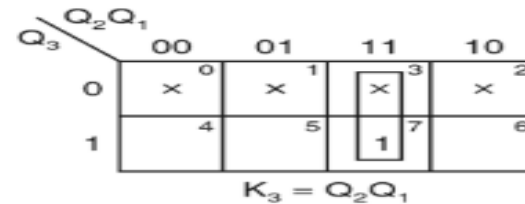
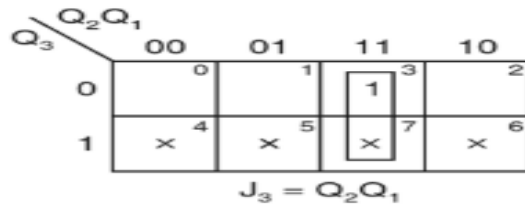
# MOD-8 SYNCHRONOUS COUNTER USING JK FF



(a) State diagram

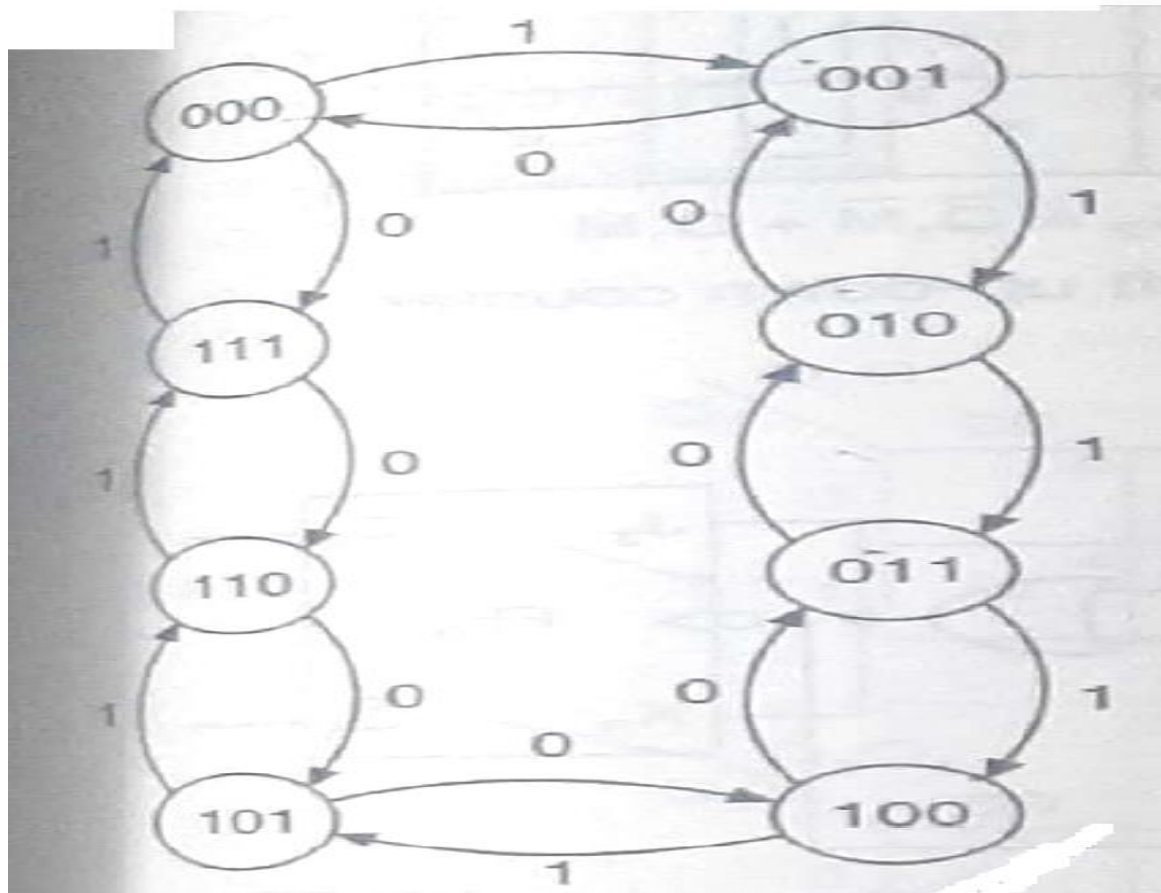
PS			NS			Required excitations					
$Q_3$	$Q_2$	$Q_1$	$Q_3$	$Q_2$	$Q_1$	$J_3$	$K_3$	$J_2$	$K_2$	$J_1$	$K_1$
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	0	1	x	0	0	x	1	x
1	0	1	1	1	0	x	0	1	x	x	1
1	1	0	1	1	1	x	0	x	0	1	x
1	1	1	0	0	0	x	1	x	1	x	1

(b) Excitation table



Karnaugh maps for a 3-bit up-counter.

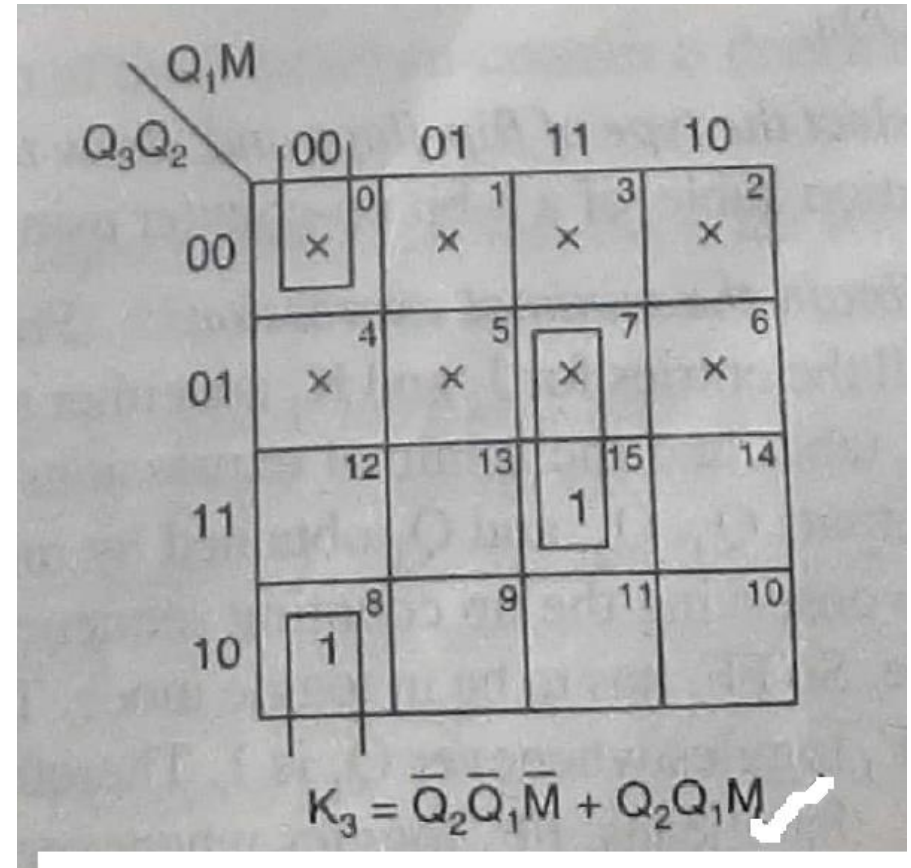
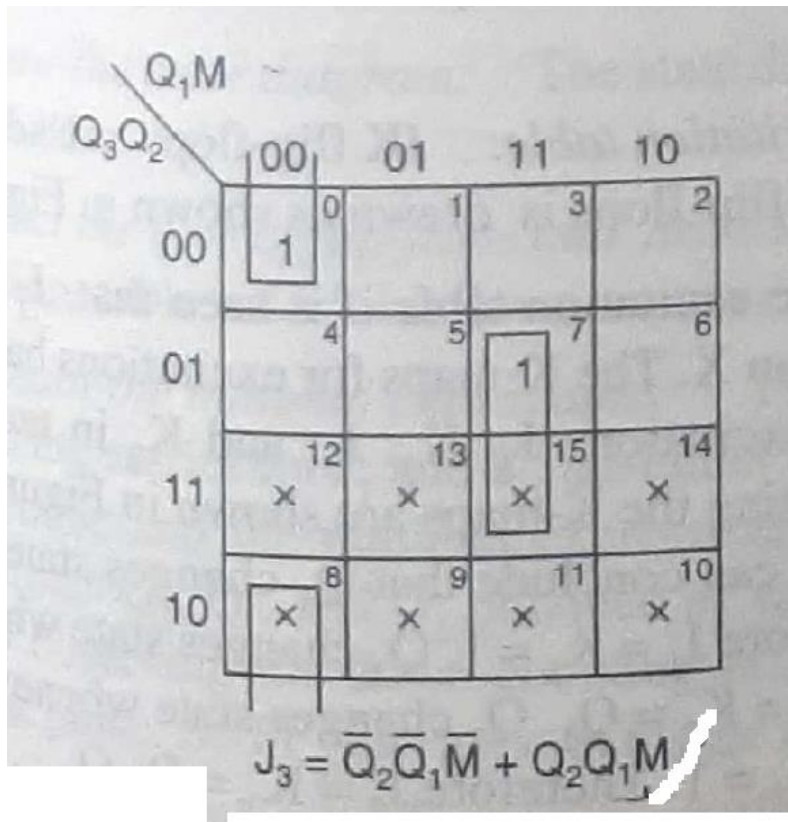
# 3 BIT UP/DOWN COUNTER USING JK FF



# 3 BIT UP/DOWN COUNTER USING JK FF

PS			Mode	NS			Required excitations					
Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>		Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	J <sub>3</sub>	K <sub>3</sub>	J <sub>2</sub>	K <sub>2</sub>	J <sub>1</sub>	K <sub>1</sub>
0	0	0	0	1	1	1	1	x	1	x	1	x
0	0	0	1	0	0	1	0	x	0	x	1	x
0	0	1	0	0	0	0	0	x	0	x	x	1
0	0	1	1	0	1	0	0	x	1	x	x	1
0	1	0	0	0	0	1	0	x	x	1	1	x
0	1	0	1	0	1	1	0	x	x	0	1	x
0	1	1	0	0	1	0	0	x	x	0	x	1
0	1	1	1	1	0	0	1	x	x	1	x	1
1	0	0	0	0	1	1	x	1	1	x	1	x
1	0	0	1	1	0	1	x	0	0	x	1	x
1	0	1	0	1	0	0	x	0	0	x	x	1
1	0	1	1	1	1	0	x	0	1	x	x	1
1	1	0	0	1	0	1	x	0	x	1	1	x
1	1	0	1	1	1	1	x	0	x	0	1	x
1	1	1	0	1	1	0	x	0	x	0	x	1
1	1	1	1	0	0	0	x	1	x	1	x	1

# 3 BIT UP/DOWN COUNTER USING JK FF





# 3 BIT UP/DOWN COUNTER USING JK FF

Truth Table for J<sub>2</sub> (Q<sub>1</sub> M):

Q <sub>3</sub> Q <sub>2</sub> \ Q <sub>1</sub> M	00	01	11	10
00	1	1	1	1
01	x	x	x	x
11	x	x	x	x
10	1	1	1	1

Equation for J<sub>2</sub>:

$$J_2 = \bar{Q}_1 \bar{M} + Q_1 M$$

Truth Table for K<sub>2</sub> (Q<sub>1</sub> M):

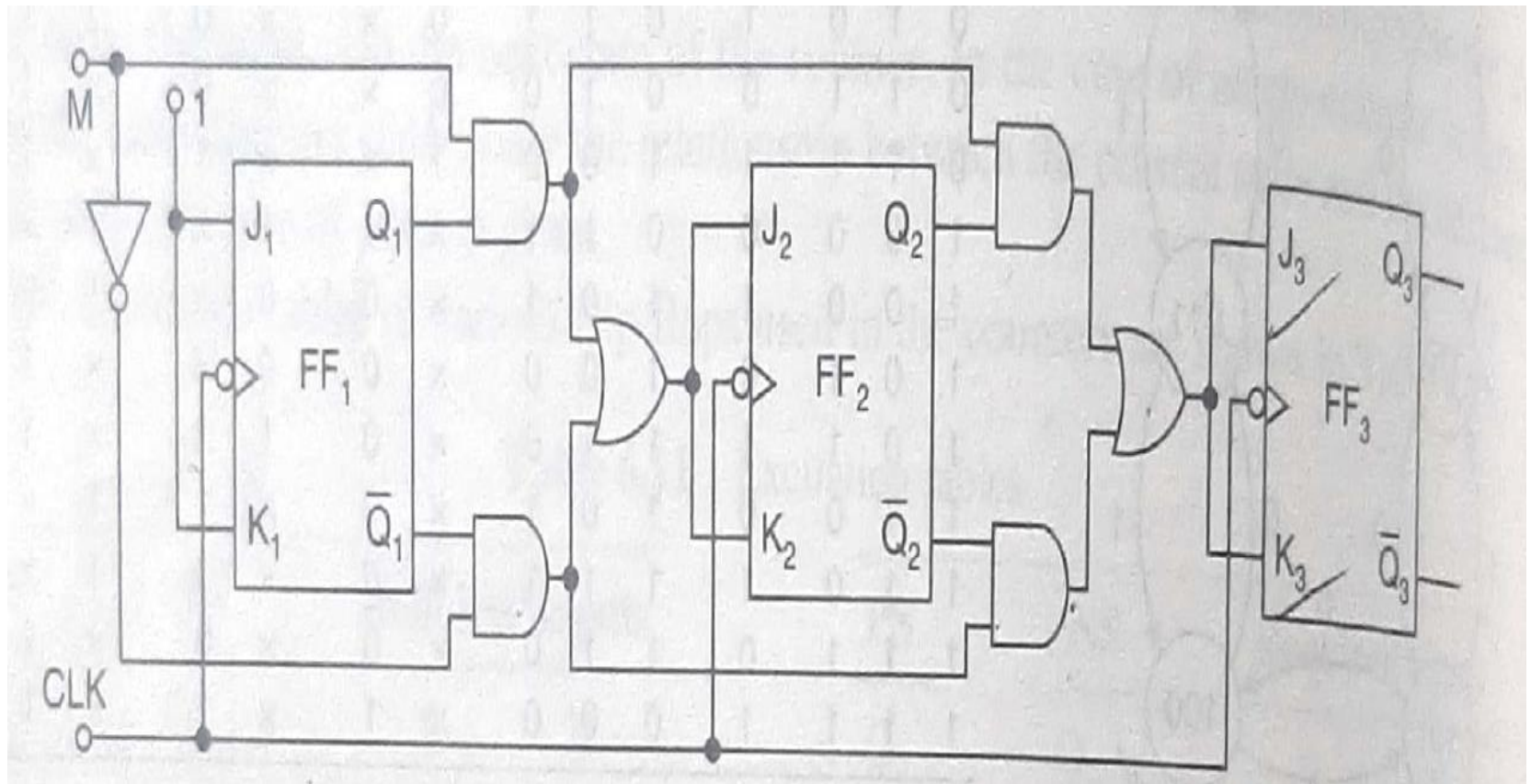
Q <sub>3</sub> Q <sub>2</sub> \ Q <sub>1</sub> M	00	01	11	10
00	x	x	x	x
01	1	1	1	1
11	1	1	1	1
10	x	x	x	x

Equation for K<sub>2</sub>:

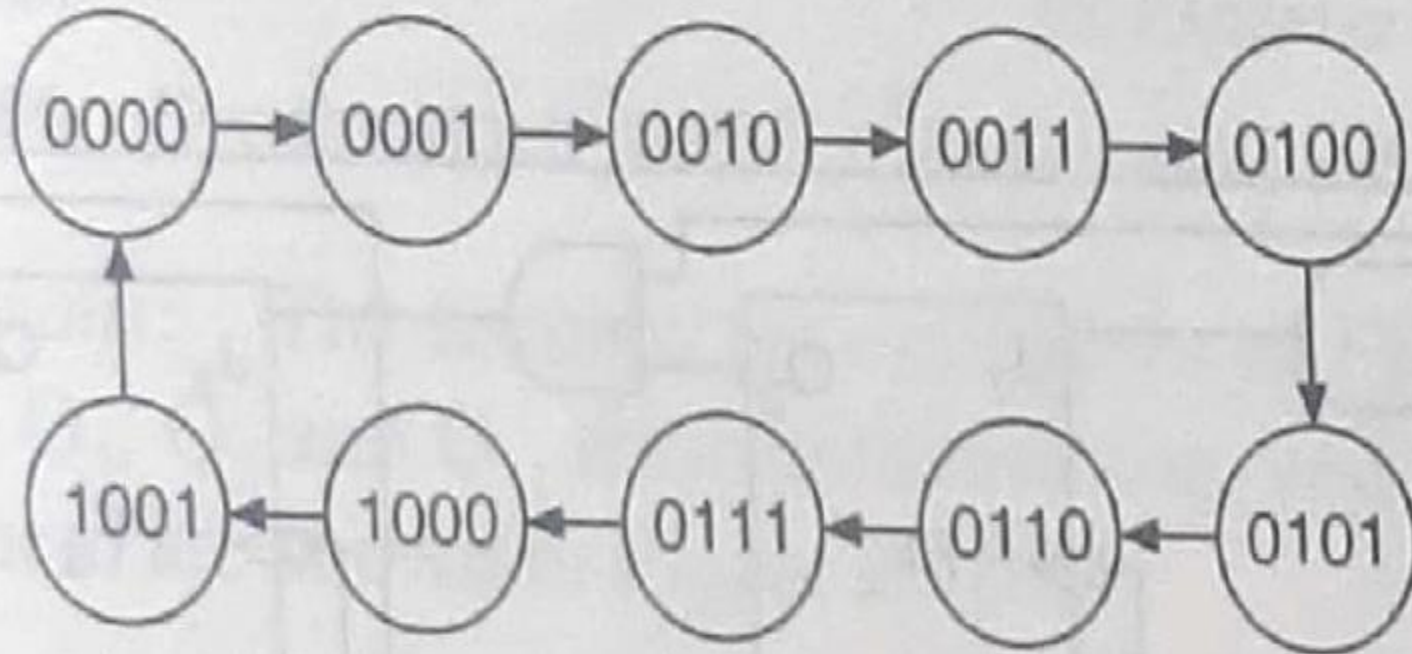
$$K_2 = \bar{Q}_1 \bar{M} + Q_1 M$$



# 3 BIT UP/DOWN COUNTER USING JK FF



# MOD-10 SYNCHRONOUS COUNTER

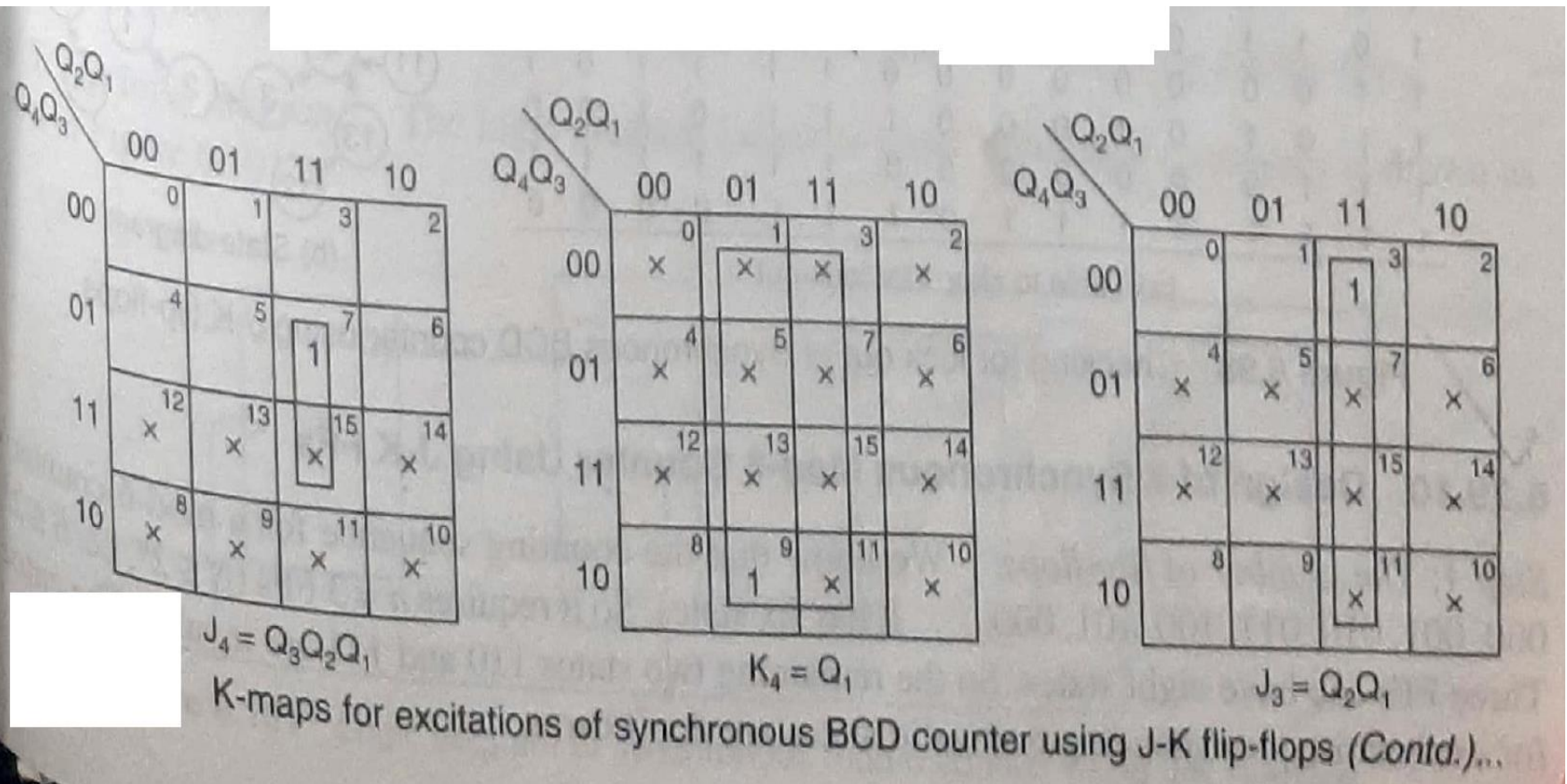


(a) State diagram

# MOD-10 SYNCHRONOUS COUNTER

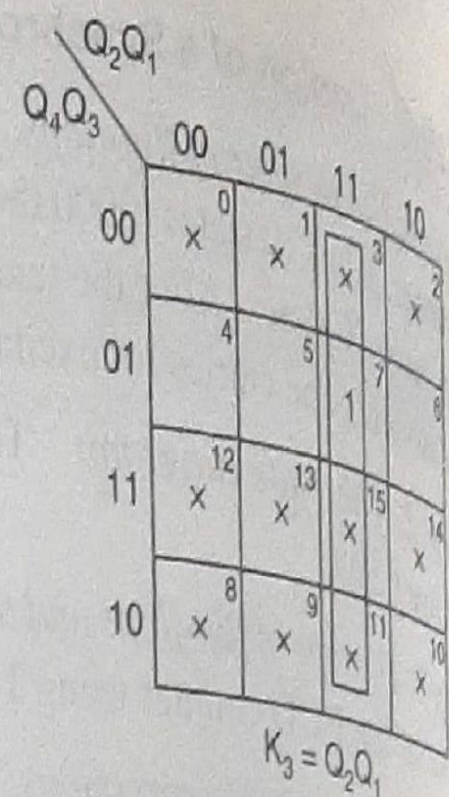
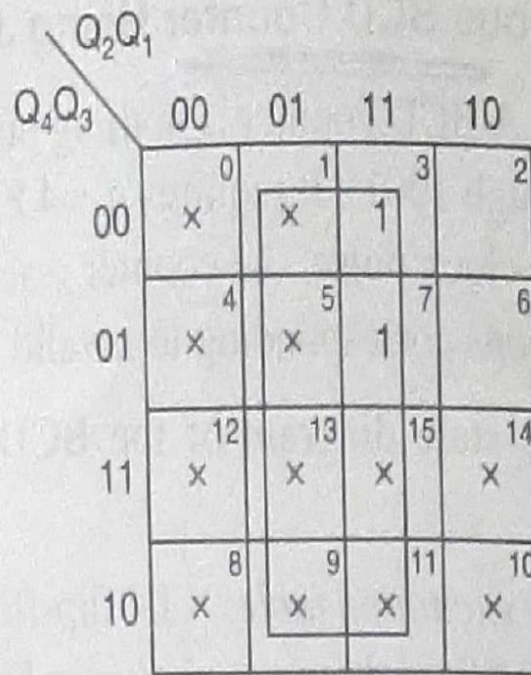
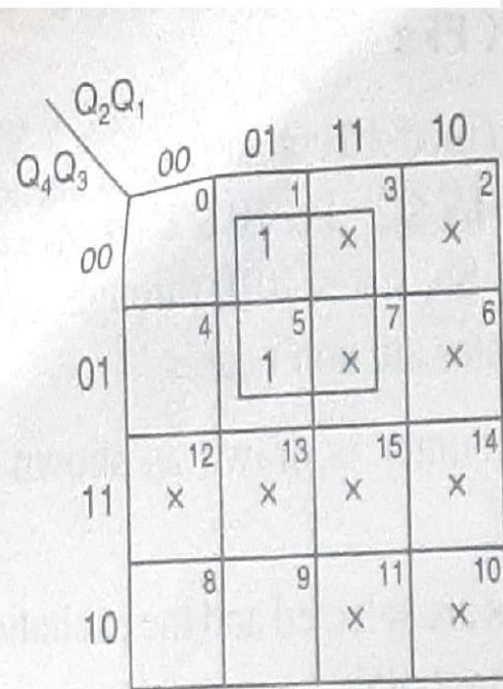
PS				NS				Required excitations							
Q <sub>4</sub>	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	Q <sub>4</sub>	Q <sub>3</sub>	Q <sub>2</sub>	Q <sub>1</sub>	J <sub>4</sub>	K <sub>4</sub>	J <sub>3</sub>	K <sub>3</sub>	J <sub>2</sub>	K <sub>2</sub>	J <sub>1</sub>	K <sub>1</sub>
0	0	0	0	0	0	0	1	0	x	0	x	0	x	1	x
0	0	0	1	0	0	1	0	0	x	0	x	1	x	x	1
0	0	1	0	0	0	1	1	0	x	0	x	x	0	1	x
0	0	1	1	0	1	0	0	0	x	1	x	x	1	x	1
0	1	0	0	0	1	0	1	0	x	x	0	0	x	1	x
0	1	0	1	0	1	1	0	0	x	x	0	1	x	x	1
0	1	1	0	0	1	1	1	0	x	x	0	x	0	1	x
0	1	1	1	1	0	0	0	1	x	x	1	x	1	x	1
1	0	0	0	1	0	0	1	x	0	0	x	0	x	1	x
1	0	0	1	0	0	0	0	x	1	0	x	0	x	x	1

# MOD-10 SYNCHRONOUS COUNTER



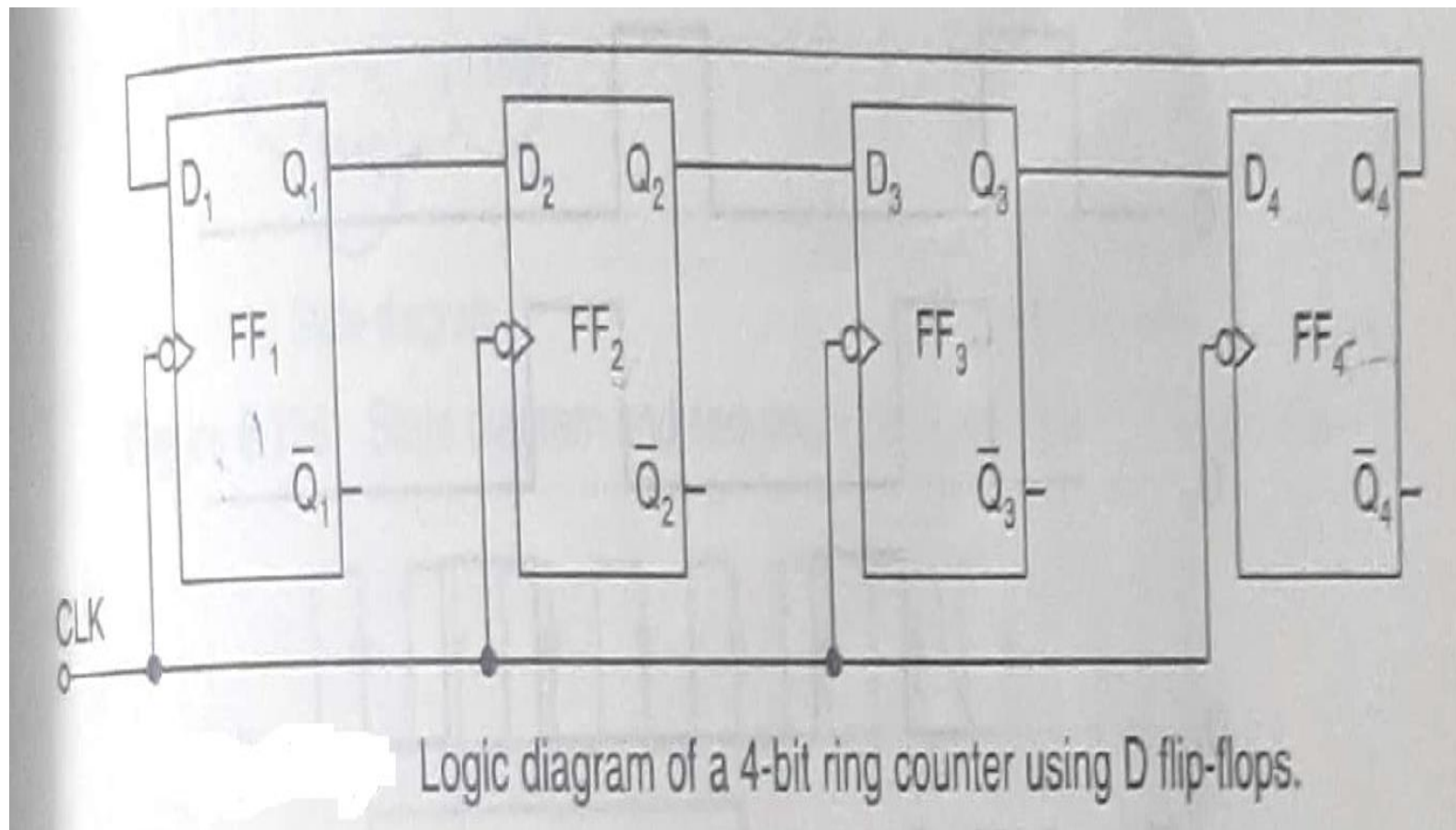


# MOD-10 SYNCHRONOUS COUNTER

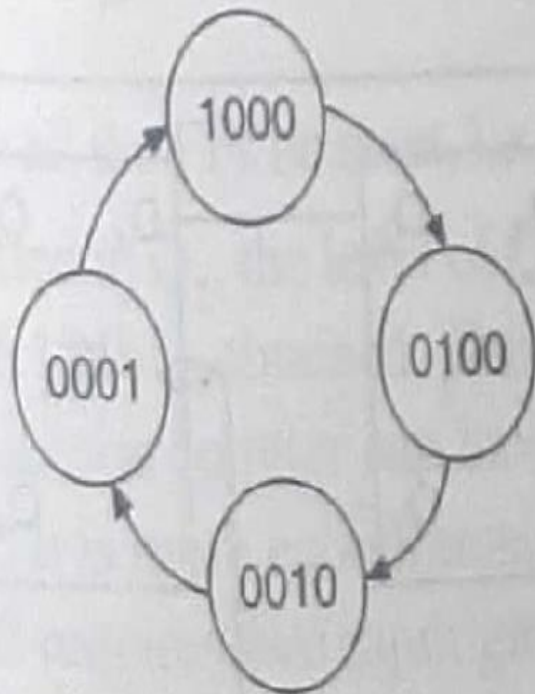


K-maps for excitations of synchronous BCD counter using J-K flip-flops.

# RING COUNTER



# RING COUNTER

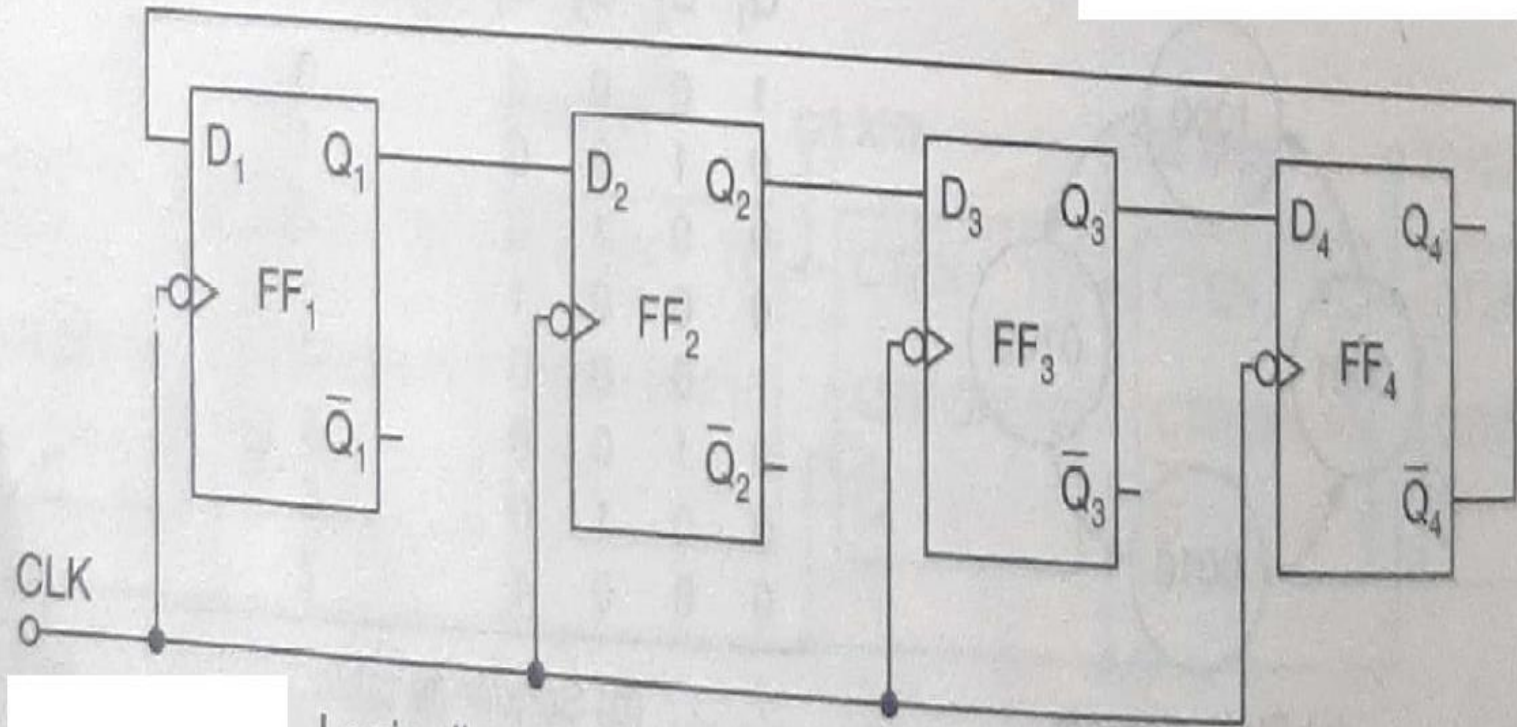


(a) State diagram

$Q_1$	$Q_2$	$Q_3$	$Q_4$	After clock pulse
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7

(b) Sequence table

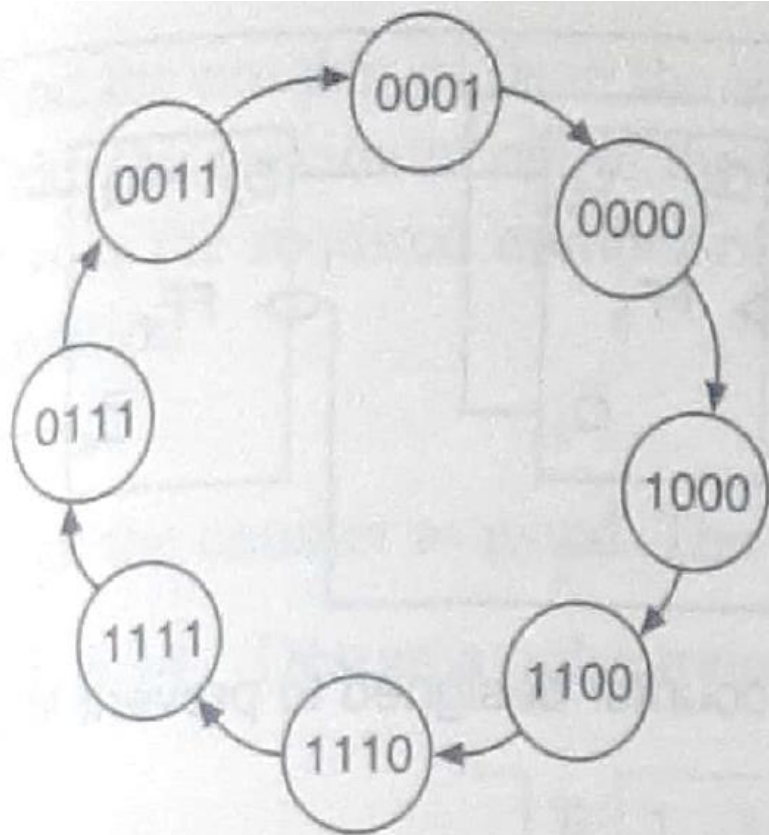
# JOHNSON COUNTER (TWISTED RING COUNTER)



Logic diagram of a 4-bit twisted ring counter using D flip-flops.



# JOHNSON COUNTER (TWISTED RING COUNTER)



(a) State diagram

$Q_1$	$Q_2$	$Q_3$	$Q_4$	After clock pulse
0	0	0	0	0
1	0	0	0	1
1	1	0	0	2
1	1	1	0	3
1	1	1	1	4
0	1	1	1	5
0	0	1	1	6
0	0	0	1	7
0	0	0	0	8
1	0	0	0	9

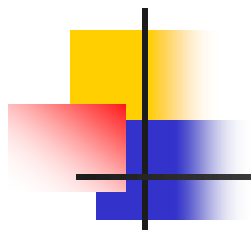
(b) Sequence table



# References

---

- Kumar, A. Anand. *Switching Theory and Logic Design*. PHI, 2014.
- Videos from NESCO Academy



---

THANK  
YOU