Digital Electronic circuit

EC(EE)302

Sequential Logic Circuit

Budhaditya Biswas

Sequential Circuit → A digital circuit whose output not only depends upon the present state of inputs but also previous outputs. Sequential circuit has memory in it.

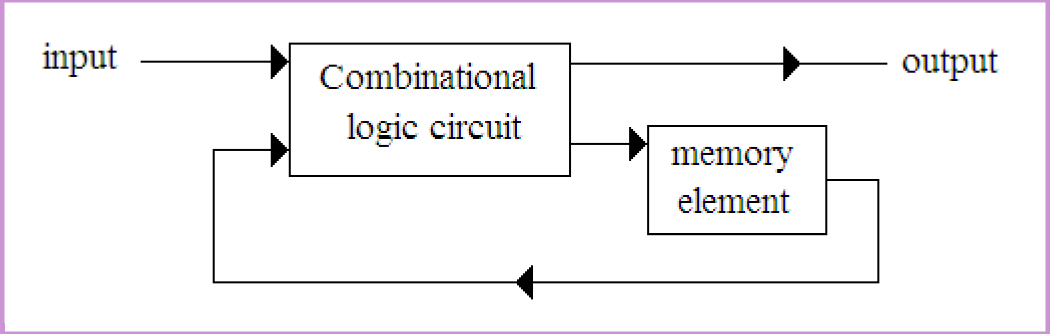


Figure: Block diagram of Sequential logic circuit

The memory elements used are **flip flops** and **latches** which are capable of storing binary information.

Flip flop & Latches

Flip-flops can be either simple (transparent or opaque) or clocked (synchronous or edge-triggered); the simple ones are commonly called latches. The word *latch* is mainly used for storage elements, while clocked devices are described as *flip-flops*. A latch is level-sensitive, whereas a flip-flop is edge-sensitive. That is, when a latch is enabled it becomes transparent, while a flip flop's output only changes on a single type (positive going or negative going) of clock edge

GATE → LATCHES

LATCHES → FLIP FLOP

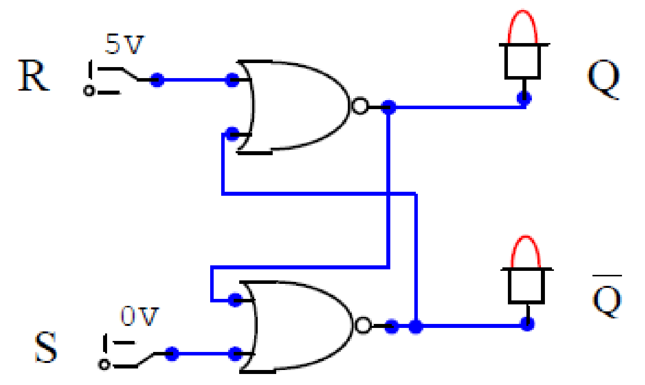
What are the difference between flip-flop and latches (find in google)

Flip flop → The basic memory element is known as *flip-flop. It has two stable* ***states, which*** are known as **1 state (set)** and **0 state (reset).** It can store 1 bit of binary information and maintains its output state until directed by an input signal to change its state. Sometimes it is also called *bistable multivibrator or 1 bit memory element. It is abbreviated as* ***FF.***

Flip-flops are classified as

- RS flip-flop
- D flip-flop
- JK flip-flop
- T flip-flop.

RS flip-flop → **R**eset **S**et flip-flop. It has two inputs (S & R) and two outputs (Q and $\bar{Q}$). $\bar{Q}$ is the complement of Q. It can be constructed using NAND gates or NOR gates.



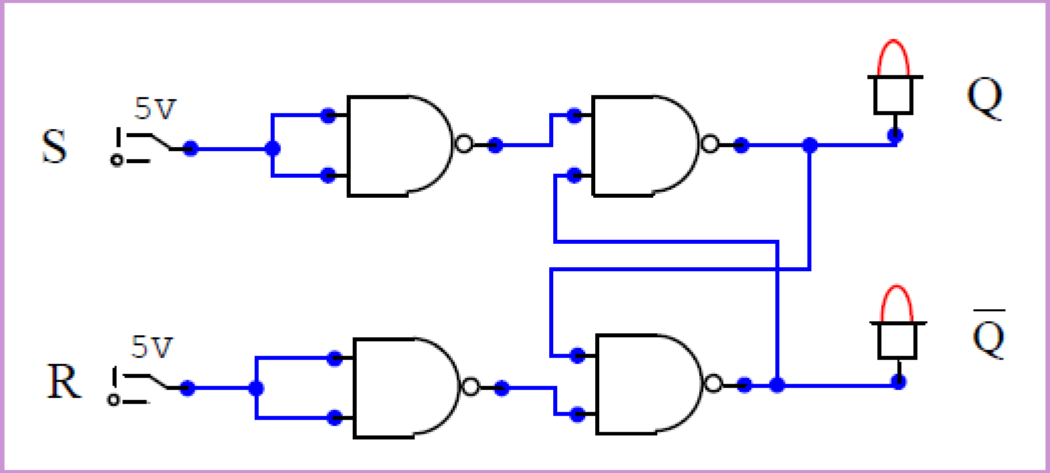
Basic Flip flop using NOR gates

The forth condition R=S=1 must be avoided by proper design.

Truth table for RS Flip flop

R	**S**	**Q**	$\overline{Q}$	**State**
0	0	NC	NC	No Change
0	1	1	0	SET
1	0	0	1	RESET
1	1	?	?	Indeterminate/race/forbidden

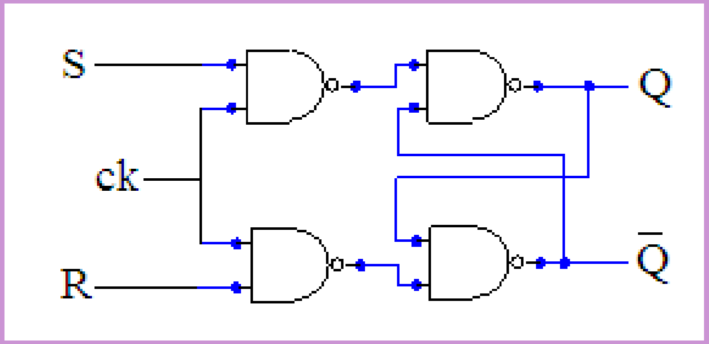
NAND gate implementation

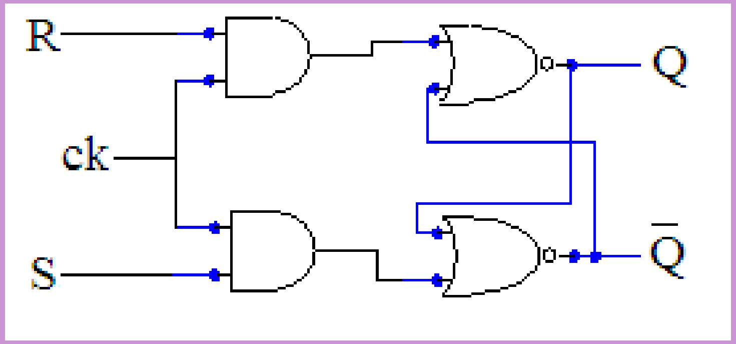


the flip flop is called ‘set’ when Q=1 and called ‘cleared’ when Q=0. The input S is called ‘set input’ because if S=1, Q=1 and R is called the clear or reset input because is R=1, Q=0.

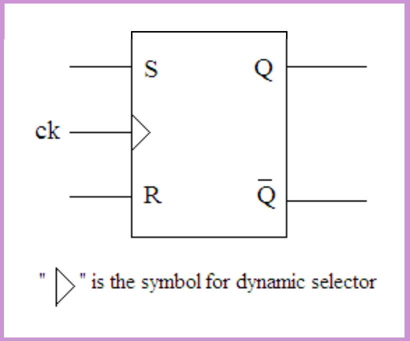
Clocked SR Flip flop

It is often required to set or reset the flip flop in synchronism with a train of pulses known as *clock pulse (abbreviated as* ***ck****). Such a circuit is shown in the figure below and it is referred* to as a ***Clocked SR Flip flop***.



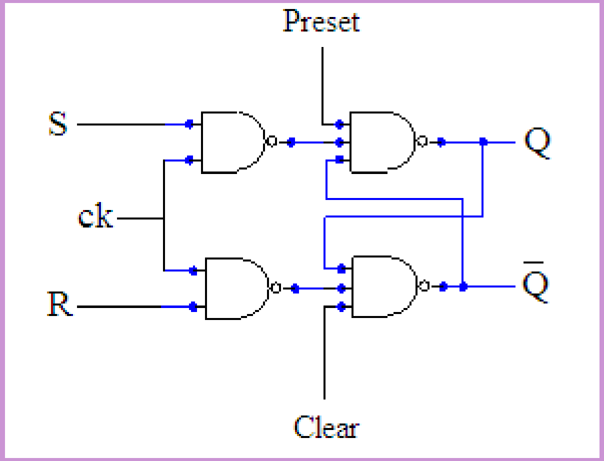


Symbol→

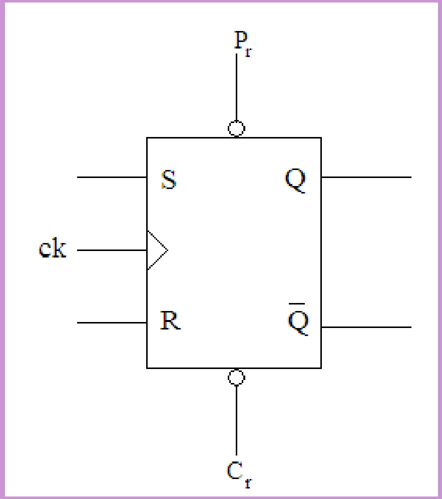


Preset and Clear →

In the previous flip flop, when the power is switched on the state of the circuit is uncertain. It may be set state (Q = 1) or reset state (Q = 0). In many applications it is desired to initially set or reset the flip flop, i.e. the initial state of the flip flop is to be assigned. This is accomplished by using directs inputs referred to as Preset (Pr) and clear (Cr) inputs. This input does not depend upon the clock and can be applied at any time instant. An SR flip flop with preset and clear inputs is shown in the figure.



An SR ff with preset and clear



Logic Symbol

Summary of operation of SR flip-flop

Inputs			Output	Operation Preformed
ck	Cr	Pr	Q	
1	1	1	Normal	Normal mode of operation
0	0	1	0	Clear
0	1	0	1	Preset

The condition Pr = Cr = 0 must not be used, since this leads to an uncertain state.

J-K Flip Flop →The uncertainty in the state of an S-R Flip flop when S=R=1 can be
eliminated by converting it into a J-K flipflop. The data inputs are J and K which are ANDed with $\overline{Q}$ and Q respectively, to obtain S and R inputs; i.e.

$$S = J.\overline{Q}$$

$$R = K.Q$$

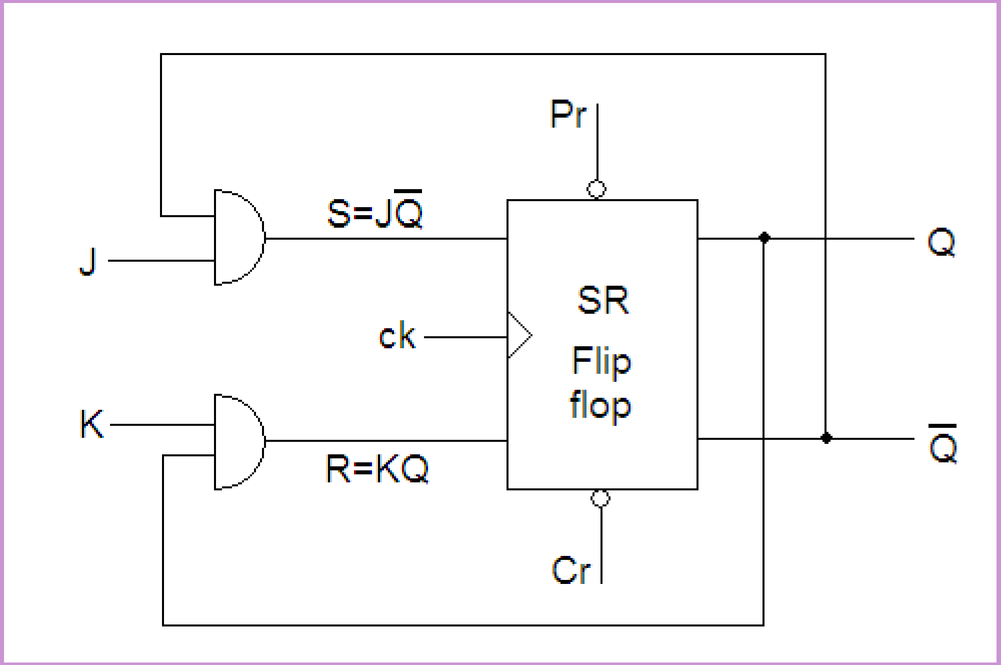


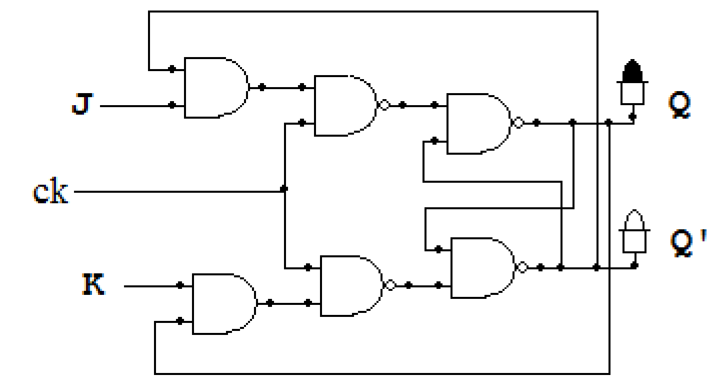
Figure: An SR flipflop converted into JK flipflop.

Truth table of this circuit →

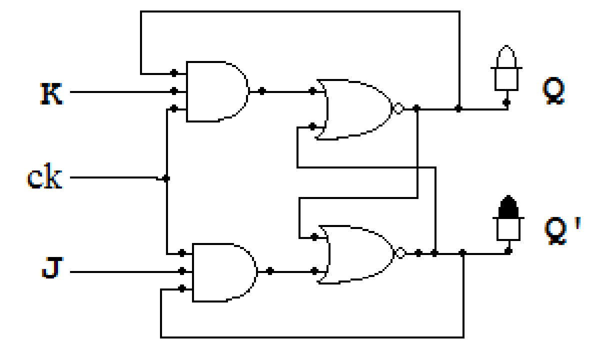
Data Inputs		Outputs		Input to SR FF		output	
J_n	K_n	Q_n	$\overline{Q}_n$	S_n	R_n	Q_{n+1}	
0	0	0	1	0	0	0	Q_n
0	0	1	0	0	0	1	
0	1	0	1	0	0	0	0
0	1	1	0	0	1	0	
1	0	0	1	1	0	1	1
1	0	1	0	0	0	1	
1	1	0	1	1	0	1	$\overline{Q}_n$
1	1	1	0	0	1	0	

Truth table of JK flipflop

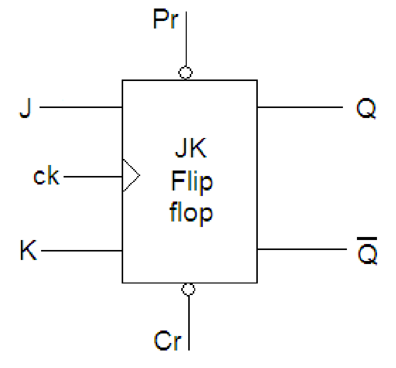
Inputs		**Outputs**
J_n	K_n	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q}_n$



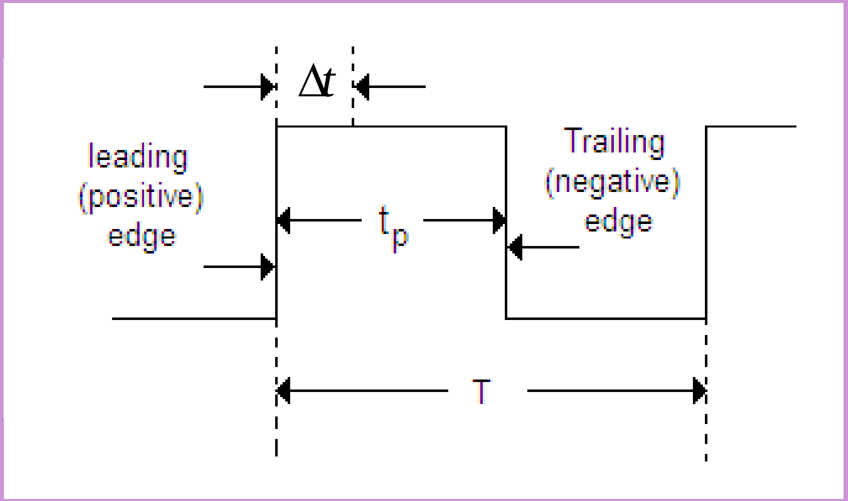
JK flip flop using NAND



JK flipflop using NOR



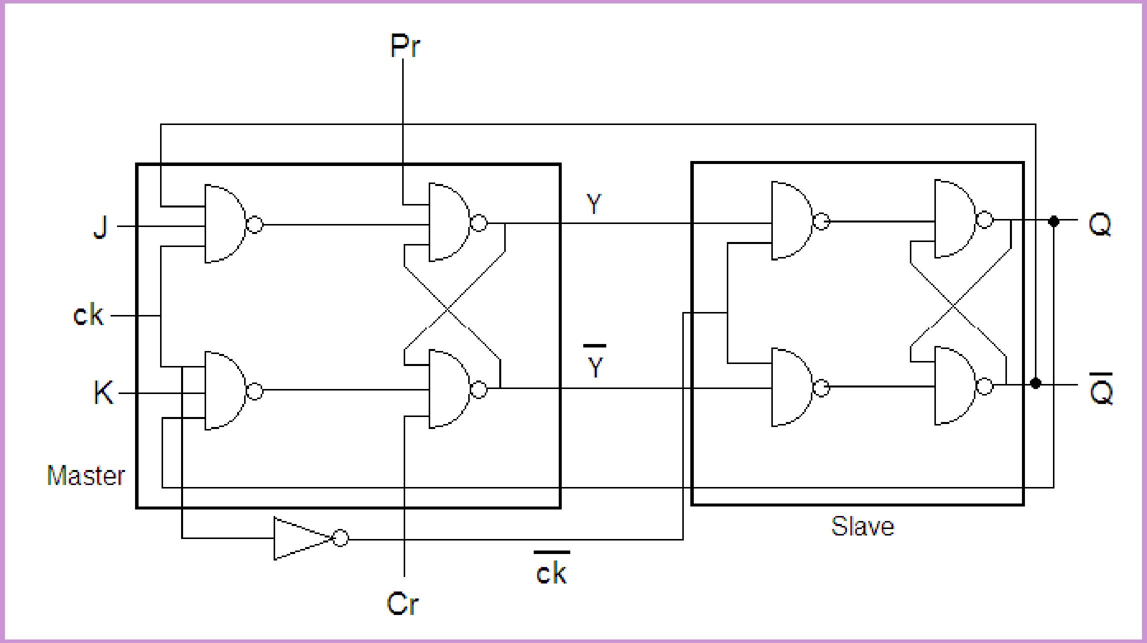
The Race Around Condition →



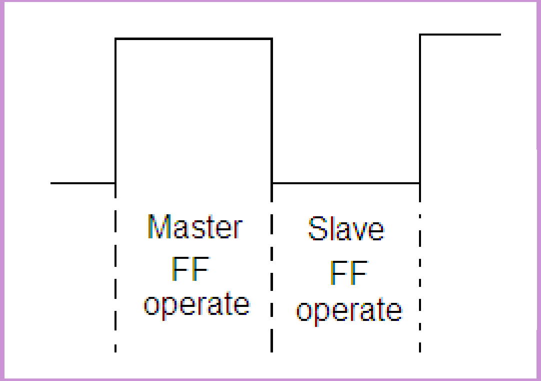
Let Δt be the propagation delay through two NAND gates in series of a JK flip flop. A condition come when J=K=1 and Q=0 and a pulse as shown in the figure is applied at the clock input. So after a time interval Δt the output will change to Q=1. Now we have J=K=1 and Q=1 and after another time interval of Δt the output will change back to Q=0. Hence we conclude that for the duration tp of the clock pulse, the output will oscillate back and forth between 0 and 1. At the end of the clock pulse, the value of Q is uncertain. This situation is referred to as the **race around condition.**

The Race around condition can be avoided if $t_p < \Delta t < T$. However, it may be difficult to satisfy this condition because of very small propagation delays of ICs. A more practical method for overcoming this difficulty is the use of **Master –Slave (MS) Configuration.**

The Master – Slave JK Flip flop →



The Master Slave FF is constructed from two separate ffs. One circuit serves as a master and the other as a slave and the overall circuit is referred to as a Master Slave Flip flop. In the logic diagram of a clocked JK flip flop Y and *Y* are the outputs of the Master flip flop. When ck = 0, $\overline{ck}$=1 *and so Slave flip flop is enable and* output Q and $\overline{Q}$ *are exactly equal to Y and* $\overline{Y}$ respectively. The master flip flop is disable because ck =0. Now when ck =1, Master flip flop is enable and the Slave flip flop is disable. The intermediate outputs Y and *Y are produced according to the truth table of the JK flip* flop.



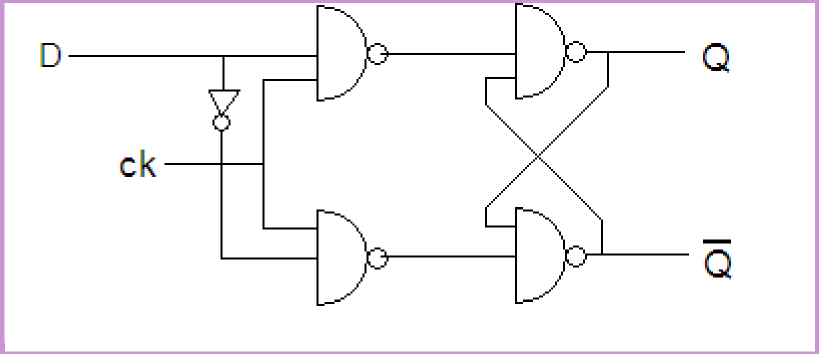
The Race around condition is not arrived in MS flip flop because the main cause for arriving race around condition is that the feedback inputs (Q and $\overline{Q}$*) are changed while the ck* remains 1. But in Master Slave flip flop there is no chance to change the outputs Q and $\overline{Q}$ because when ck=1 slave disable and Q and $\overline{Q}$ *are the outputs of slave flip flop. So these* cannot be changed.

When ck=1 the first flip flop producing the outputs Y and $\overline{Y}$ *and when ck*=0 *the slave* flip flop operate and it just transfer the inputs Y and $\overline{Y}$ *to its output Q and* $\overline{Q}$. *So this flip flop* only follow only follows the previous flip flop. That's why the first flip flop is called the **master** flip flop and the second one is called the **slave** flip flop.

This flip flop is a negative edge triggering flip flop. Because it gives the output Q and $\overline{Q}$ *when ck goes from 1 to 0.*

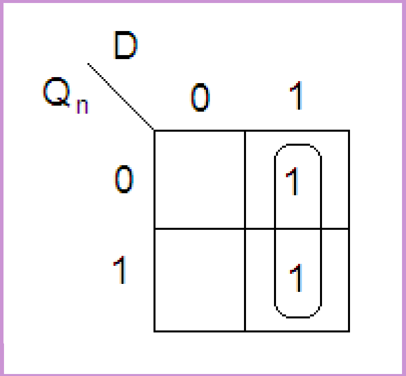
D flip flop →

The D flip flop is a modification of clocked RS flip flop. The D input goes directly to the S input and its compliment is applied to the R input. This arrangement reduces the input number from two to one and also eliminate the indeterminate state. The circuit and characteristics table are shown in the figure.

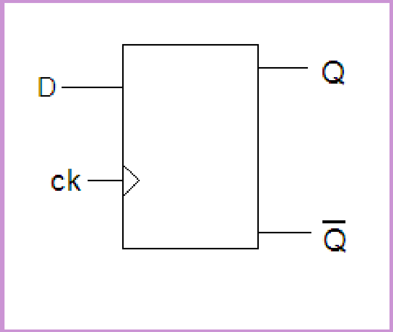


Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

Characteristics Table



$Q_{n+1} = D$
Characteristics Equation



Logic Symbol

so we see that when ck = 1 the output of D flip flop is follow the D input. It just transfer data. That's why it receives the designation D (DATA) flip flop.

T flip flop →

In a JK flip flop, if J=K; the resulting flip flop is referred to as a T type flip flop. In has only one input, referred to as T input. The truth table for T flip flop is shown in the figure =. It is clear that if T = 1 it acts as a toggle switch. For every clock pulse the output Q changes it state.

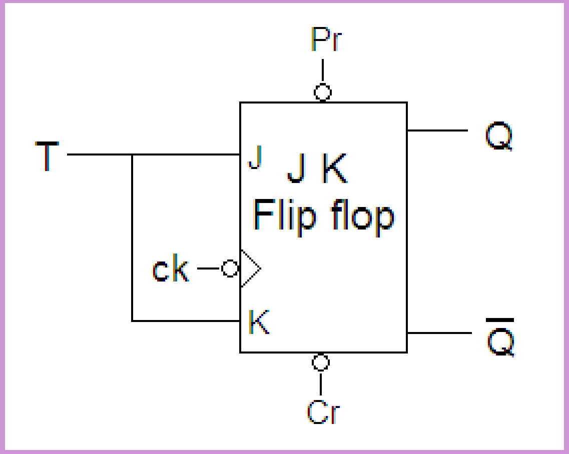


Figure: A JK flip flop converted into T flip flop

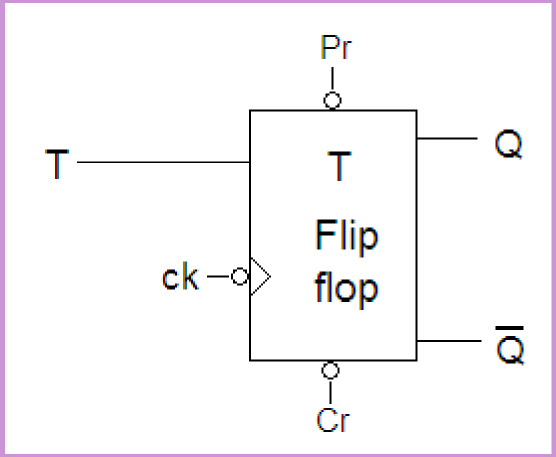


Figure: Logic symbol of T flip flop

Truth table of T flip flop →

T_n	Q_{n+1}
0	Q_n
1	$\bar{Q}_n$

Excitation table of Flip flop →

A truth table of a flip flop is also referred to as the characteristics table and specifies the operational characteristics of the flip flop.

In the design of sequential circuits, we usually come across situations in which the present state and next state of the circuit are specified and we have to find the input conditions that must satisfy to cause the desired transition of the state. By the present state and next state we mean the state of the circuit prior to and after the clock pulse respectively. For example in a SR flip flop $Q_n = 0$ and we want the output will not change when the next clock pulse arrive. Now from the truth table of the SR flip flop we see that

If $S_n = R_n = 0$ output $Q_{n+1} = Q_n$
and if $S_n = 0$; $R_n = 1$ output $Q_{n+1} = 0 = Q_n$ (in this case only)

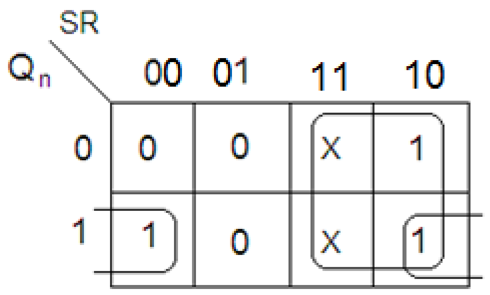
so we conclude from the above condition that the S_n input must be 0, whereas the R_n input may be zero may be one (don't care condition). Similarly input condition can be found for all possible situations. A tabulation of these conditions is known as the **Excitation Table**.

Excitation table of flip flop →

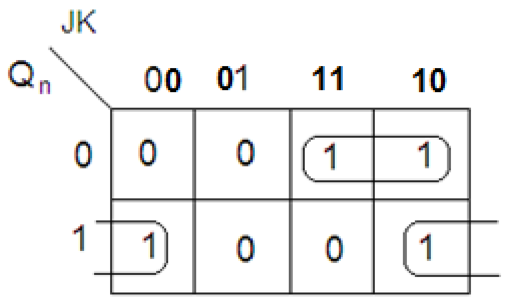
Present State	Next State	SR Flip flop		JK Flip flop		T flip flop	D flip flop
$\mathbf{Q_n}$	$\mathbf{Q_{n+1}}$	$\mathbf{S_n}$	$\mathbf{R_n}$	$\mathbf{J_n}$	$\mathbf{K_n}$	$\mathbf{T_n}$	$\mathbf{D_n}$
0	0	0	X	0	X	0	0
0	1	1	0	1	X	1	1
1	0	0	1	X	1	1	0
1	1	X	0	X	0	0	1

Characteristics equations of flip flops →

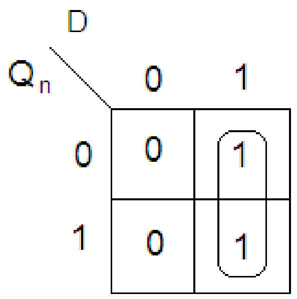
The characteristics equations of flip flops are useful in analyzing circuits made of them. Here, next state Q_{n+1} is expressed as a function of present state Q_n and input to flip flop. Karnaugh Map can be used to get the optimized expression and truth table of each flip flop is mapped into it.



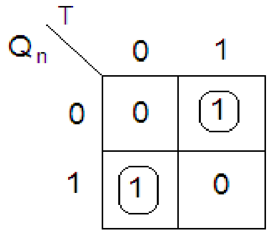
(a) $Q_{n+1} = S + \overline{R}Q_n$



(c) $Q_{n+1} = J\overline{Q}_n + \overline{K}Q_n$

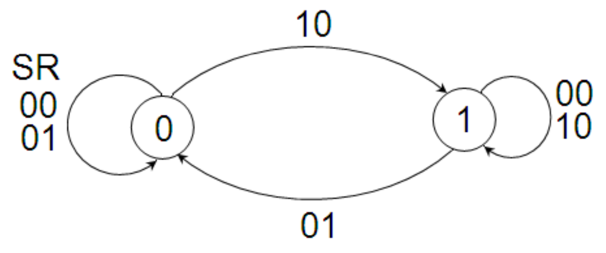


(b) $Q_{n+1} = D$

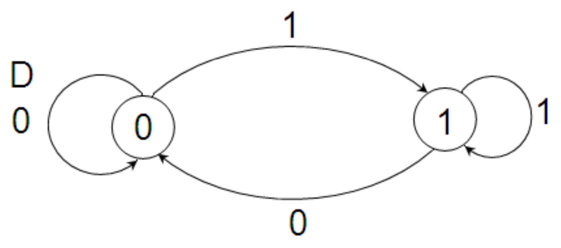


(d) $Q_{n+1} = T\overline{Q}_n + \overline{T}Q_n$

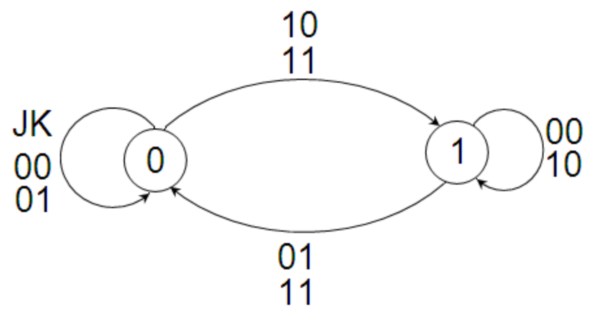
State Transition Diagram of Flip flops →



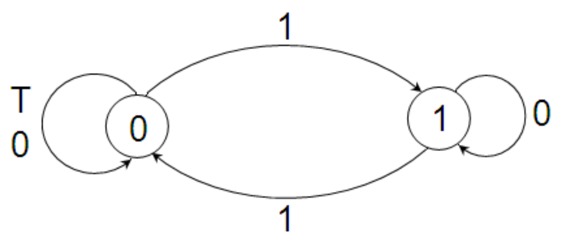
(a) SR flip flop



(b) D flip flop



(c) JK flip flop



(d) T flip flop

Conversion of flip flops →

Example: Convert an D type flip flop to an JK flip flop.

Step 1 → look into JK flip flop truth table and specifically note Q_n → Q_{n+1} transitions for a given combination of inputs JK and Present state Q_n.

Step 2 → from the Q_n → Q_{n+1} transition write the input combination of the desire flip flop with the help of the excitation table of that flip flop.

Step 3 → from the state synthesis table (step 2) with the help of K Map write the design equation of the desire flip flop.

State synthesis table for D to JK flip flop conversion →

Truth Table of desire Flip flop

J_n	K_n	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

State synthesis table for D to JK flip flop conversion →

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

Excitation Table of given Flip flop

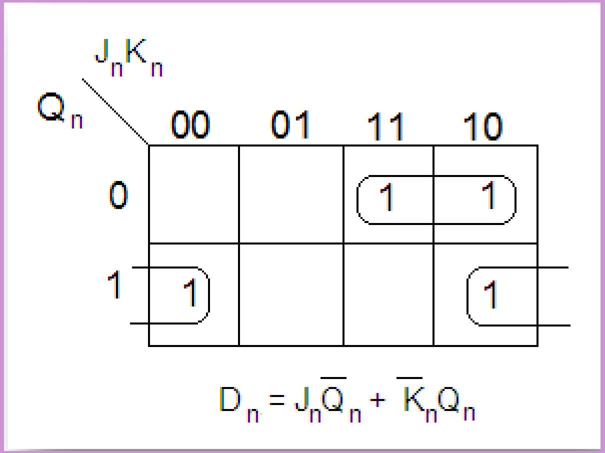
State synthesis table for D to JK flip flop conversion →

Truth Table of desire Flip flop

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

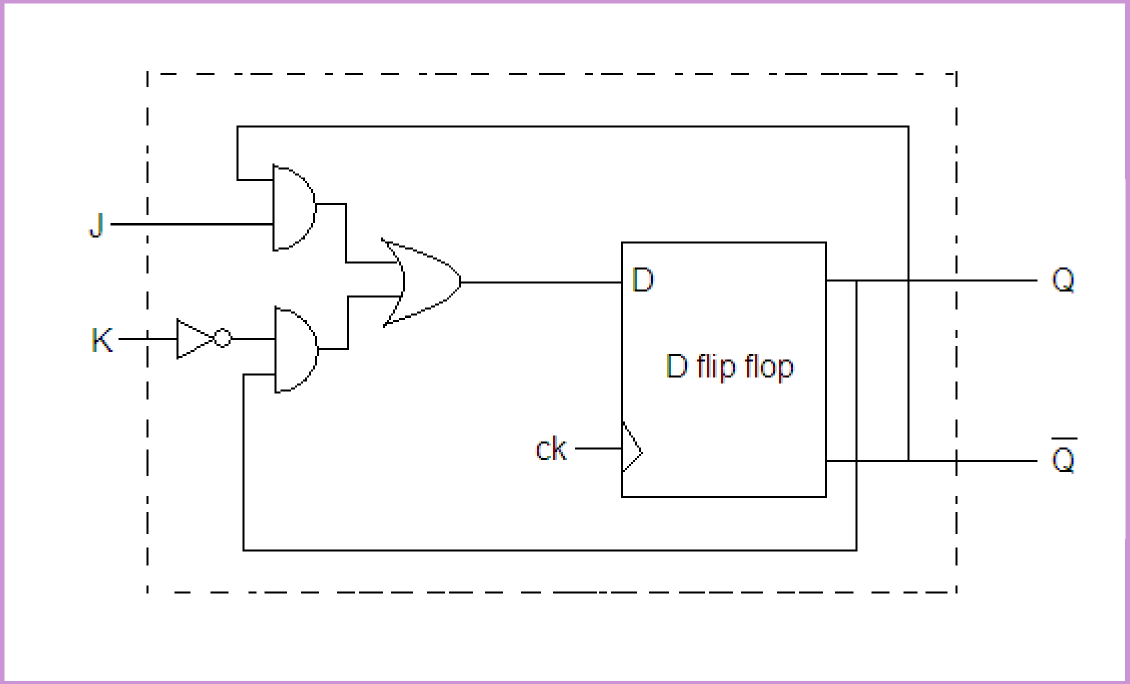
Excitation Table of given Flip flop

K Map simplification for input D



$D_n = J_n\overline{Q}_n + \overline{K}_nQ_n$

Figure: JK flip flop from D flip flop



Register →

A register is a group/array of binary storage cells (flip flop) suitable for holding the binary information. A group of flip flops constitute a register. Since each flip flop is a binary cell capable of storing one bit of binary information, an n-bit of register has a group of n flip flops and is capable of storing any binary information containing n bit.

The data can be entered in serial (one bit at a time) or in parallel form (all the bits simultaneously) and can be retrieved in the serial or parallel form.

Register are classified depending upon the way in which data are entered are retrieved. There are four possible modes of operation:

- SISO (**S**erial **I**n **S**erial **O**ut)
- SIPO (**S**erial **I**n **P**arallel **O**ut)
- PISO (**P**arallel **I**n **S**erial **O**ut)
- PIPO (**P**arallel **I**n **P**arallel **O**ut)

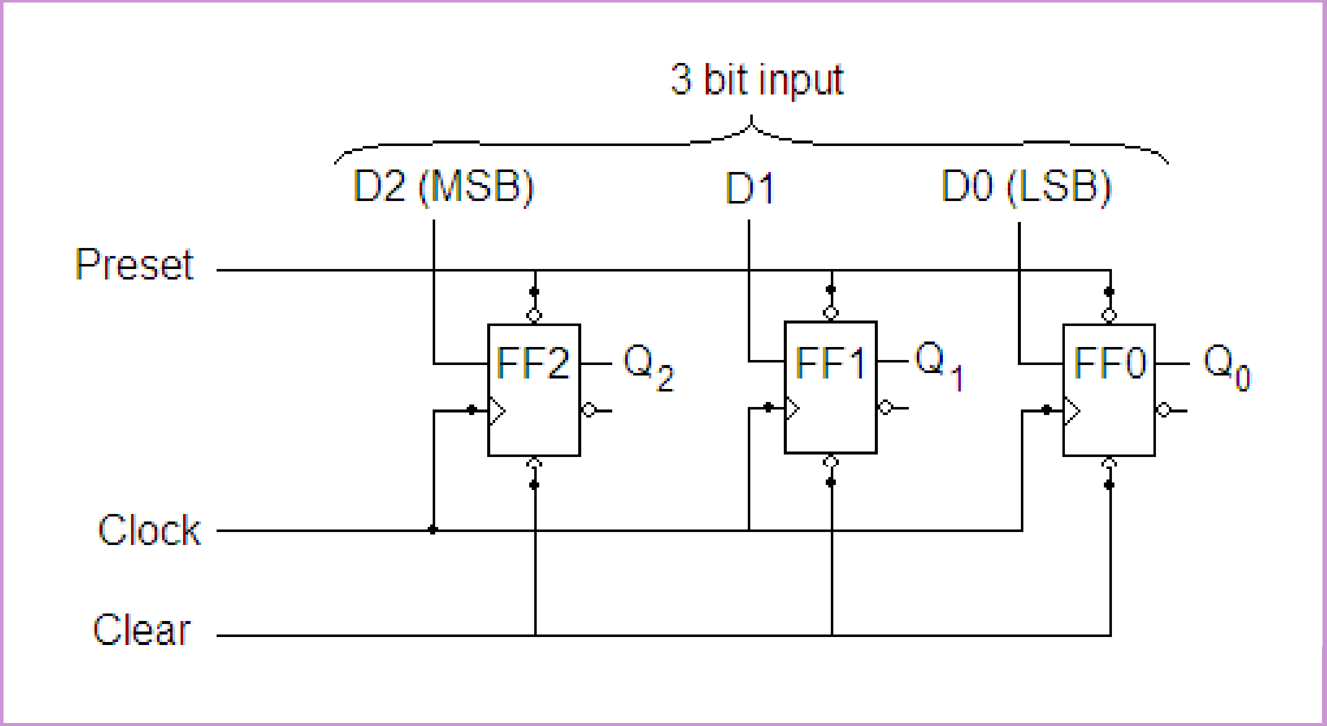


Figure: A 3-bit Register using D type Flip flop.

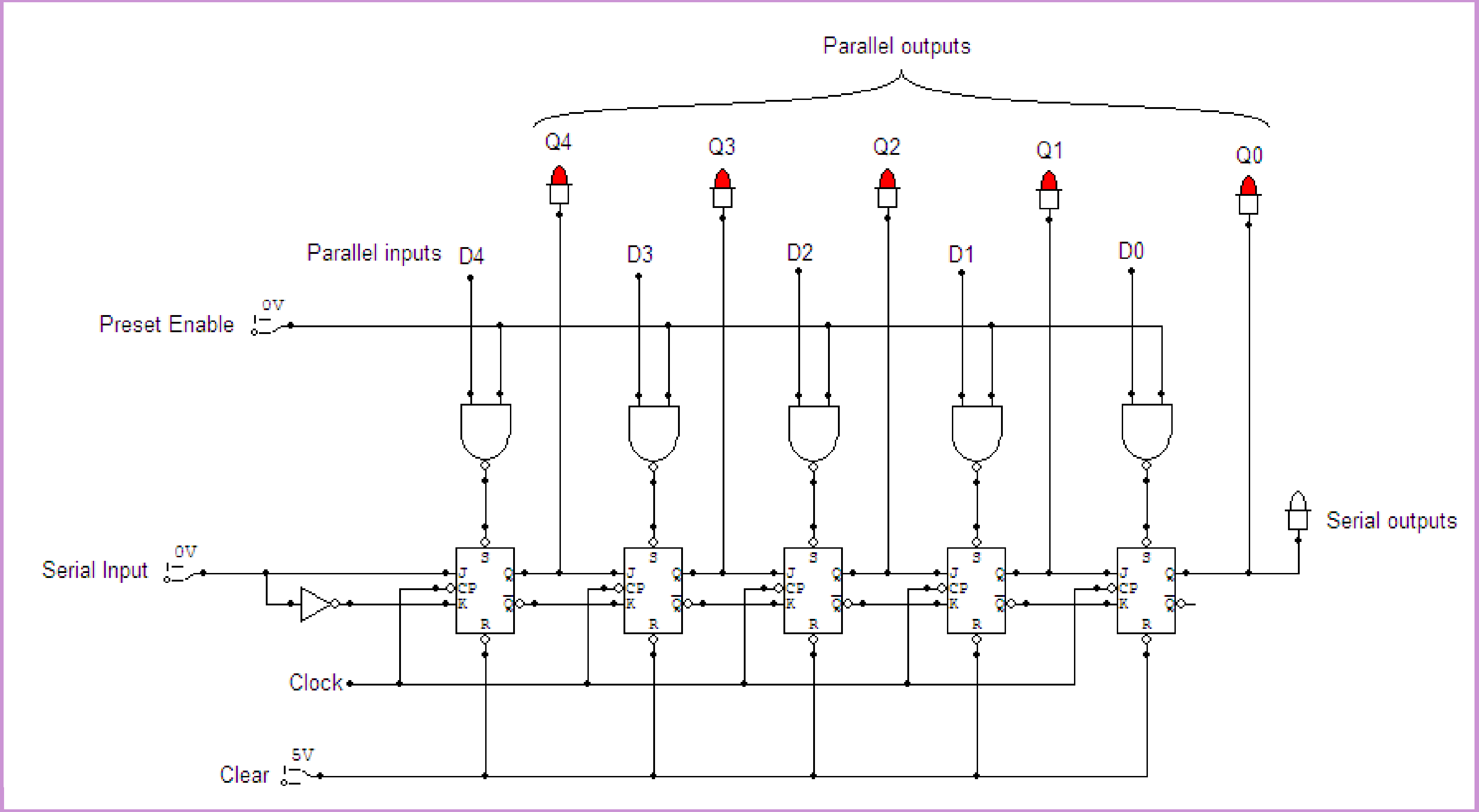


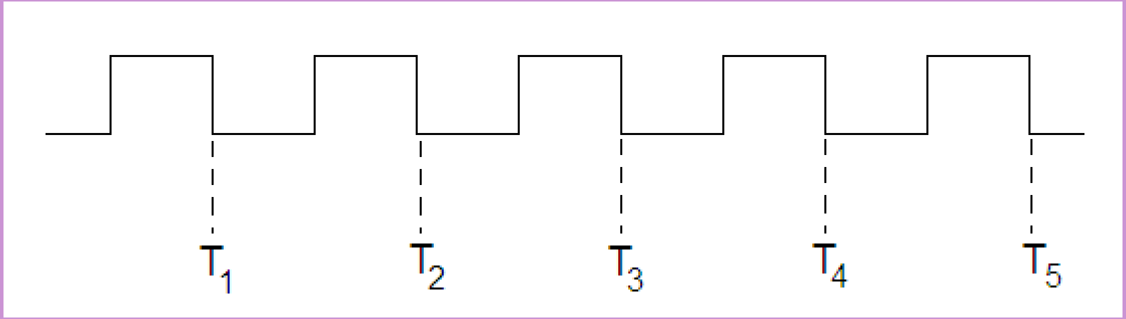
Figure: A 5 bit shift register using 5 Master Slave SR (or JK flip flop)

Registers in which data are entered or/and taken out in serial form are referred to as a **shift register**, since bits are shifted in the flip flops with the occurrence of the clock pulses either in the Right direction (**Right Shift Register**) or in the Left direction (**Left Shift Register**). In the **Bi-directional shift registers** data can be shifted from left to right as well as in the reverse direction.

A register is referred to as a **Universal register** if it can be operated in all the four possible modes and also as a bidirectional register. [IC 74178, 74179, 74194 (4 bit) 74198 (8 bit) are the Universal registers].

Assuming a 5 bit data 10011.

- Serial Input = first clear the flip flop. The Preset enable input is to be held at '0' so that 'preset' for every flip flop is '1'. The digital word entered corresponding to LSB in the serial input.



At	**T_1**	**$Q_4 = 1$**	**$Q_3 = 0$**	**$Q_2 = 0$**	**$Q_1 = 0$**	**$Q_0 = 0$**
At	**T_2**	**$Q_4 = 1$**	**$Q_3 = 1$**	**$Q_2 = 0$**	**$Q_1 = 0$**	**$Q_0 = 0$**
At	**T_3**	**$Q_4 = 0$**	**$Q_3 = 1$**	**$Q_2 = 1$**	**$Q_1 = 0$**	**$Q_0 = 0$**
At	**T_4**	**$Q_4 = 0$**	**$Q_3 = 0$**	**$Q_2 = 1$**	**$Q_1 = 1$**	**$Q_0 = 0$**
At	**T_5**	**$Q_4 = 1$**	**$Q_3 = 0$**	**$Q_2 = 0$**	**$Q_1 = 1$**	**$Q_0 = 1$**

The number of clock pulses required for storing the n bit data are n. The process of entering the data is also referred to as **writing into the register**.

The data stored into the register can be retrieved (also referred to as **reading**) in two ways, **serial out** and **parallel out**. The data in serial form is obtained at Q_0 when clock pulses are applied. The number of clock pulse required is same as the number of bits (five in this case). In the parallel form the outputs are available at Q_4 Q_3 Q_2 Q_1 Q_0 and no clock pulse is required for this.

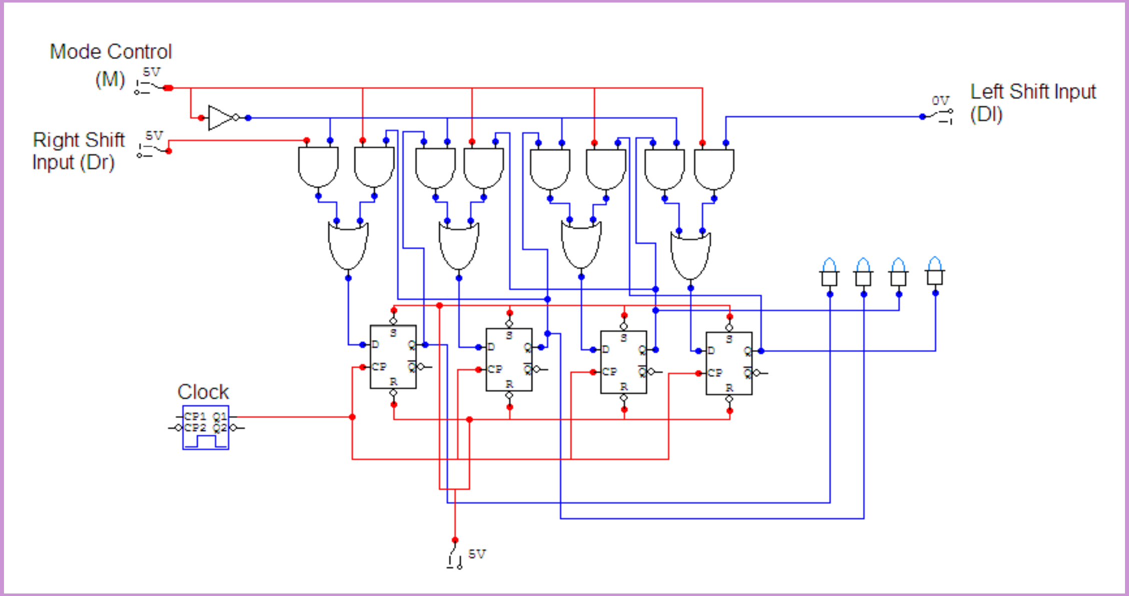
Parallel In →

Data can be entered in the parallel form in two ways, **Asynchronous loading** and **Synchronous loading**. Data can be entered in parallel form making use of the preset input. After clearing the flip flops, if the data input lines are connected to the parallel inputs (D_4 D_3 D_2 D_1 D_0) and a 1 is applied at the preset enable input. The data are written into the register. This is referred to as **asynchronous loading** since no clock pulse is required for this purpose.

The data can be entered in parallel form by using D types flip flops. The data inputs are connected to the D inputs of the flip flops, when the clock pulse are applied this data are stored in the flip flops. It is referred to as **synchronous loading.**

Bi-Directional Register →

A shift register which can be able to shifted its data towards life or right according to its requirements. Right shift is required when Binary numbers are divided by 2. If the number is odd then it causes an error of 0.5. Left shift is required when binary number is multiplied by 2. A four bit binary shift register is shown in the figure.



If M = 1, the input D_R shifted to the right on the occurrence of the clock pulse

If M = 0, the input D_L shifted to the left on the occurrence of the clock pulse.

Counter →
A sequential logic circuit consisting of flip flops and are able to count pulses arrived at it clock input. There are two types of counters, name by **Synchronous counter** and **Asynchronous counter**.

Synchronous Counters → The counter in which a common clock input is connected to all the flip flops and all the flip flops are clocked simultaneously. Example **Ring counter, Johnson Counter, Twisted Ring Counter**.

Asynchronous Counter → The counter in which the first flop is clocked by the external clock pulse and each successive flip flops are clocked from the previous flip flop outputs (may be Q or $\overline{Q}$). Therefore the asynchronous counter are not clocked simultaneously. It is also called **Ripple Counter**. Such as **Modulus N Counter**.

Definition of Counters → A sequential circuit that goes through a prescribed sequence of states upon the application of input pulses is called a counter. The input pulses, called count pulses. A 'n' nit binary counter consists of n flip flops and can count 2^n number of binary states.

Modulus of the counter → The number of the states that a counter can count is called 'The modulus' or MOD of that counter. Example MOD 5 counter counts 5 binary states. If it is desire to have a modulus m counter the number of flip flops required is determined by the equation, as the maximum value of N which satisfies the equation $m \leq 2^n$.

Design a 3 bit binary counter → Number of steps are $2^3 = 8$ i.e. from 000 to 111. So the number of flip flop required is 3.

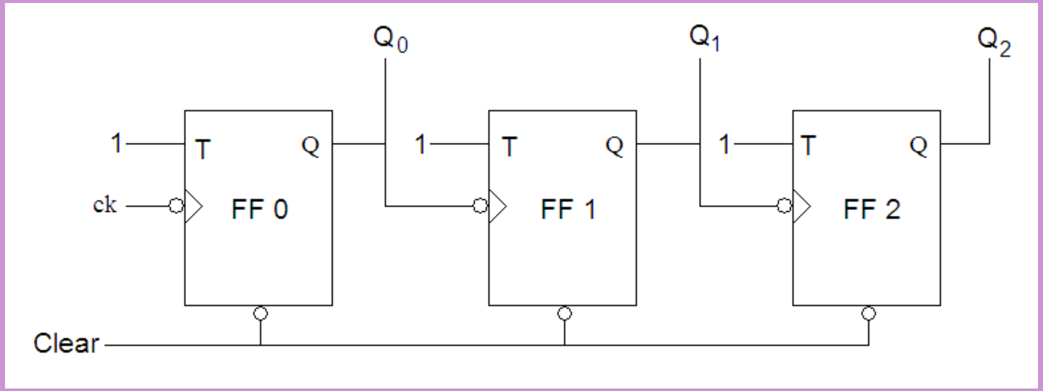
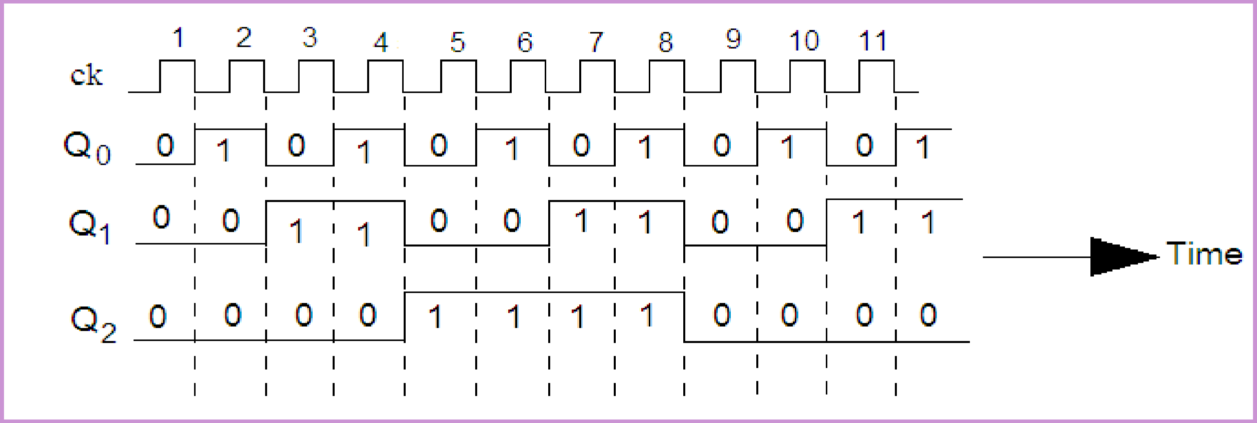


Figure: 3 bit binary Up counter.

Output waveforms (Timing Diagram)→

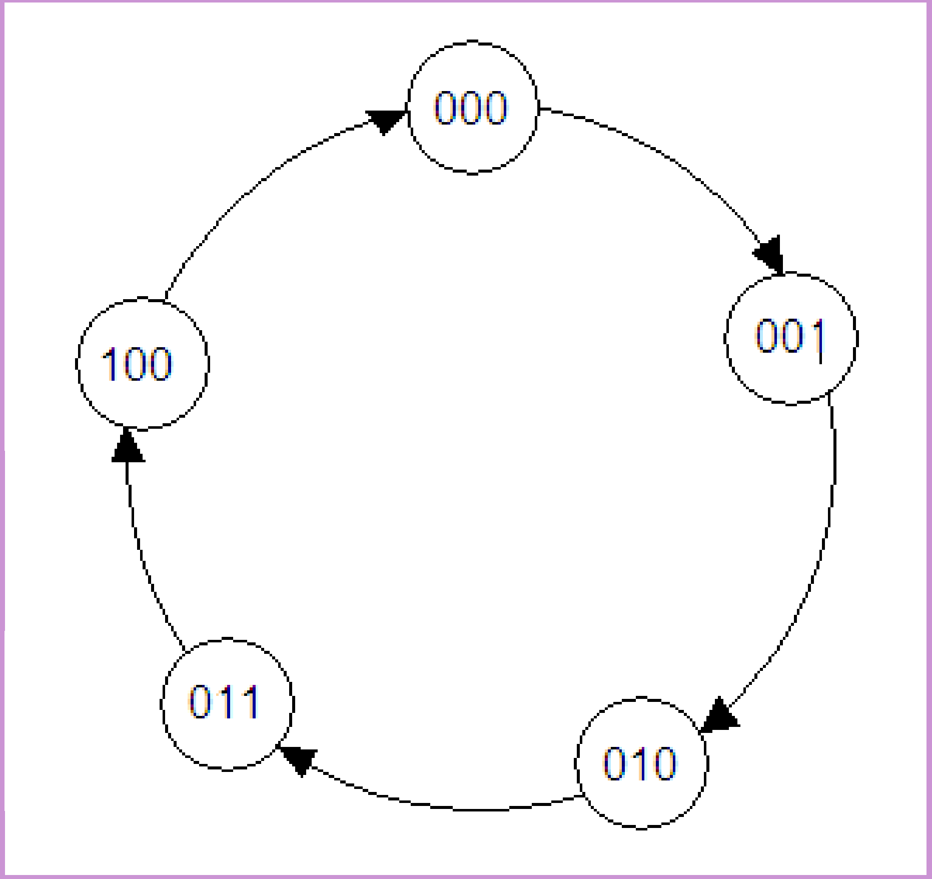


Design MOD 5 Synchronous counter using JK flip flop → Here 5 indicate total number of the states the counter can counts. Let the stated are 0, 1, 2, 3 and 4. Where 0 is the initial state. Number of flip flop required

$$2^n \geq 5$$

so n = 3. Let the flip flops are 0, 1 and 2.

State Diagram →


$$\begin{tikzcd}
 & 000 \arrow[dr] & \\
100 \arrow[ur] & & 001 \arrow[d] \\
011 \arrow[u] & & 010 \arrow[ll]
\end{tikzcd}$$

State table →

Present State			Next State			Flip flop inputs					
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
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	0	0	0	X	1	0	X	0	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	X	X	X	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X	X	X	X

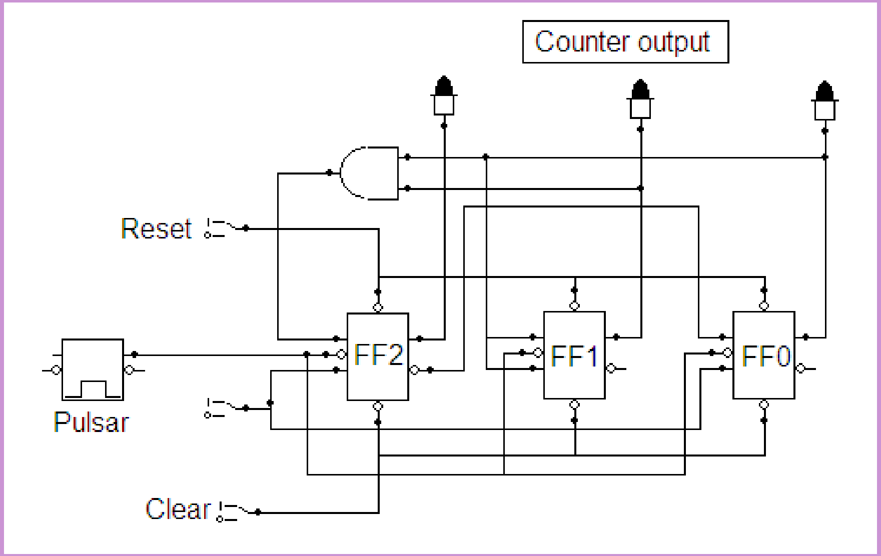
The *skipped* states are 101, 110 and 111. Enter 'X' for these states.

Using K map find the expression for all the flip flop input (Consider the present state to enter the cell).

$$J_2 = Q_0 Q_1 \qquad J_1 = Q_0 \qquad J_0 = \overline{Q_2}$$

$$K_2 = 1 \qquad K_1 = Q_0 \qquad K_0 = 1$$

From the above expressions constructs the logic circuit.



Lock out conditions → The regular binary counter counts the adjacent sequence in the form of ascending or descending order. For example in a 3 bit binary counter (up) the initial state is 000 and final state is 111. After counting 111 it returns its initial state i.e. 000. If the final state is prescribed then after counting this state the counter returns its initial state. If there are some unused states (for example in MOD 5 counter 101, 110 and 111 are the unused states), by chance, the counter happens to find itself in any one of them, its next state would not be known. It may just be possible that the counter might go from one unused state to another and never comeback it its used states. Of course such a situation makes the counter useless for its intended purpose. A counter whose unused states have this feature is said to suffer from **lock out**.

To ensure lock out does not occurs, we design the counter assuming the next state to be the initial state or the next used state, from each of the unused state.

Design a MOD Counter using reset function →

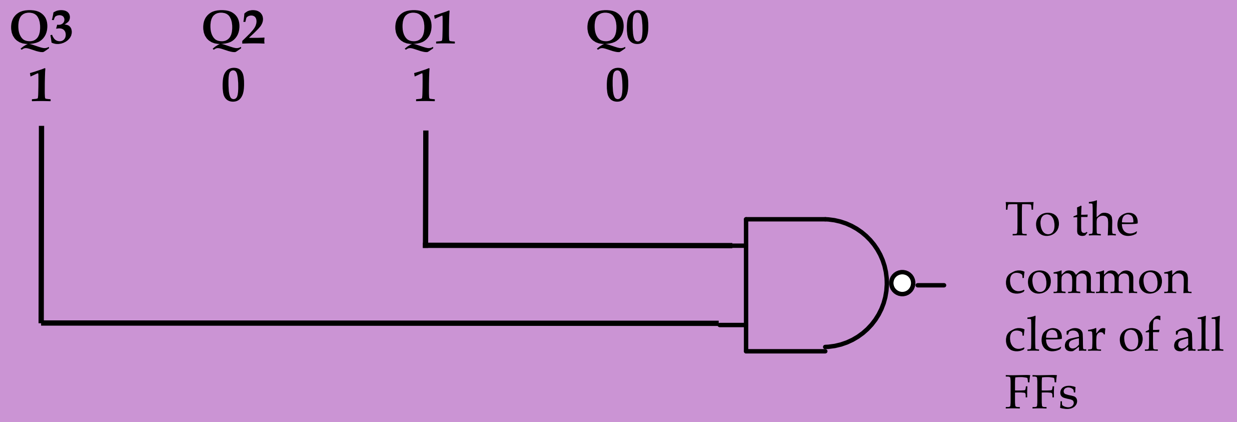
The most commonly used method for obtaining MOD counters with ***skipping states*** by using clear method. For example, in a MOD N counter when the N+1 sates come, in the same moment all the flip flops will get the reset pulse. So all the flip flops are cleared and the state will be the initial state. In this type of design of counter the initial state must be 0000.

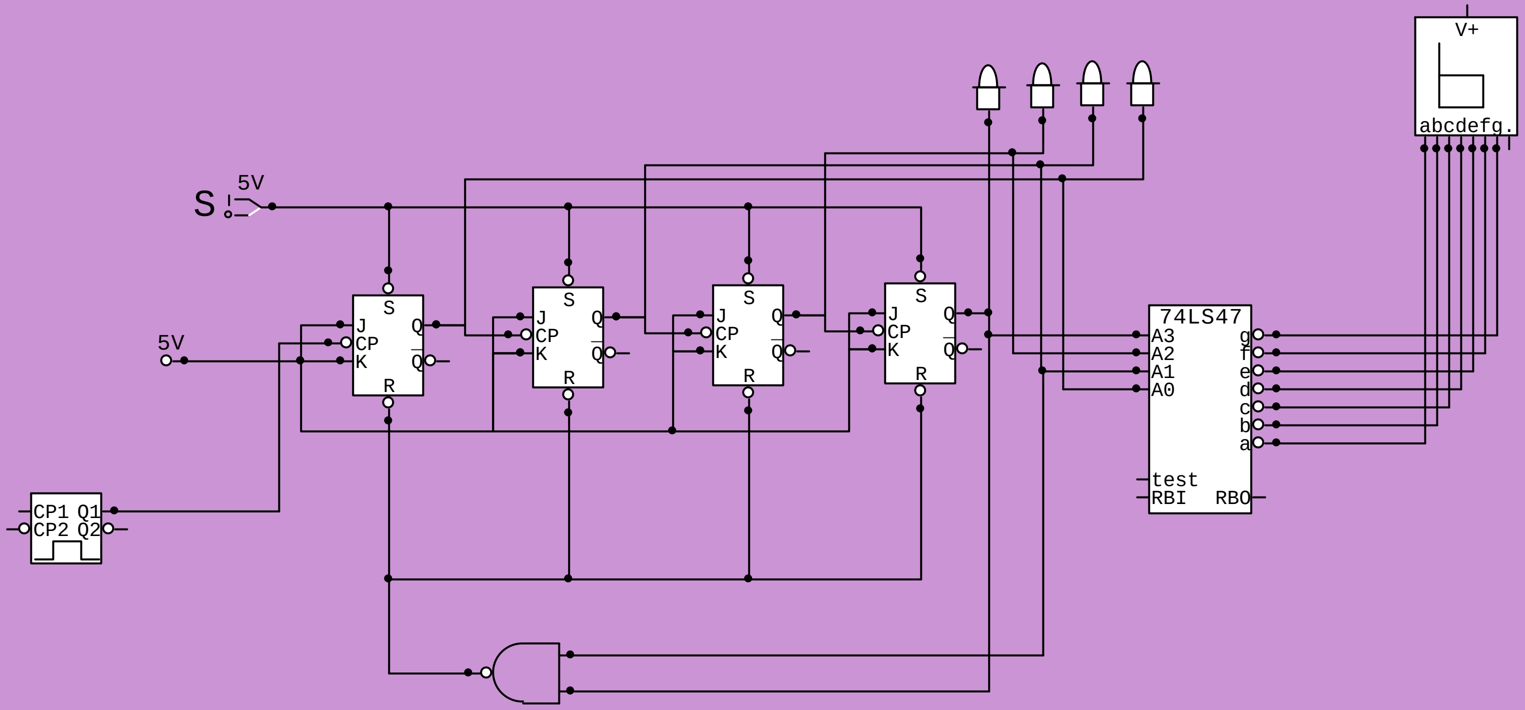
$$\left.\begin{array}{l}\textbf{Design a Decade counter using reset function} \rightarrow \\ \\ \textbf{Design a Asynchronous Decade counter} \rightarrow\end{array}\right\} \textbf{Same Solution}$$

How many flip flops required? Maximum Modulus (No. of counts)? Skipping states?

Decade counter or MOD 10 counter have ten counting states, starting from 0000 to 1001. But the maximum modulus for 4 bit counter is 16, i.e. 0000 to 1111. So 1010, 1011, 1100, 1101, 1110 and 1111 are skipping states in this design.

After counting 1001 it again returns to its initial state 0000. In this design when 1010 state (i.e. next to the final state) will come all the flip flops getting a clear pulse simultaneously. So all the flip flop output becomes 0 i.e. the initial state '0000' of the counting.

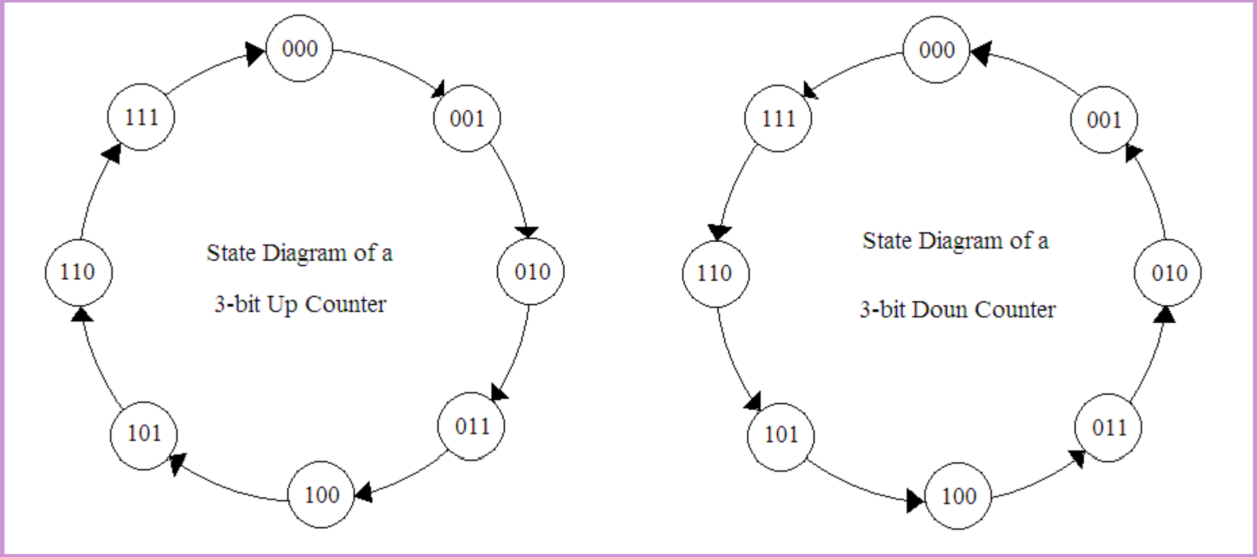




Decade Asynchronous counter using clear function

Up Down Synchronous Counter →

A counter with an ascending sequence is called *Up Counter* and the counter with a descending sequence is called *Down Counter*.



Implementation of a 3-bit UP/DOWN counter is done using T-flip flops.

State table of UP/DOWN counter

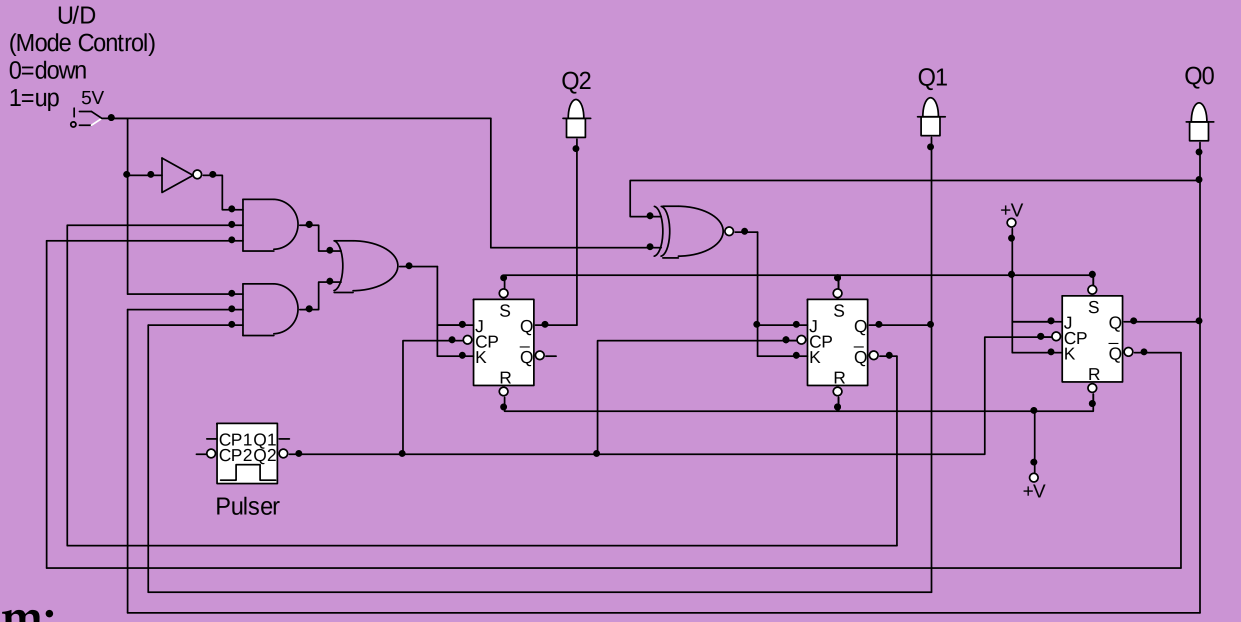
	Mode control	Present State			Next State			Flip-flop Inputs		
	U/D	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	T_2	T_1	T_0
Down	0	0	0	0	1	1	1	1	1	1
	0	0	0	1	0	0	0	0	0	1
	0	0	1	0	0	0	1	0	1	1
	0	0	1	1	0	1	0	0	0	1
	0	1	0	0	0	1	1	1	1	1
	0	1	0	1	1	0	0	0	0	1
	0	1	1	0	1	0	1	0	1	1
	0	1	1	1	1	1	0	0	0	1
up	1	0	0	0	0	0	1	0	0	1
	1	0	0	1	0	1	0	0	1	1
	1	0	1	0	0	1	1	0	0	1
	1	0	1	1	1	0	0	1	1	1
	1	1	0	0	1	0	1	0	0	1
	1	1	0	1	1	1	0	0	1	1
	1	1	1	0	1	1	1	0	0	1
	1	1	1	1	0	0	0	1	1	1

From the state table using K map simplification the expressions for T_2, T_1 and T_0 are

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

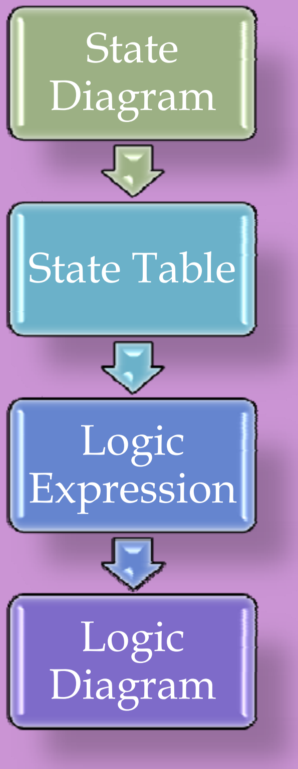
$$T_1 = \overline{U_D}\,\overline{Q_0} + U_D Q_0 = U_D \text{ XNOR } Q_0$$

$$T_0 = 1$$

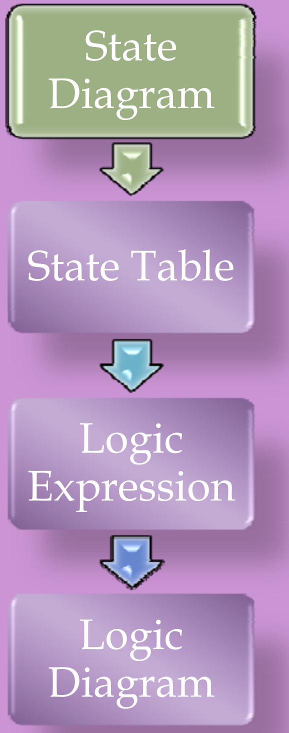


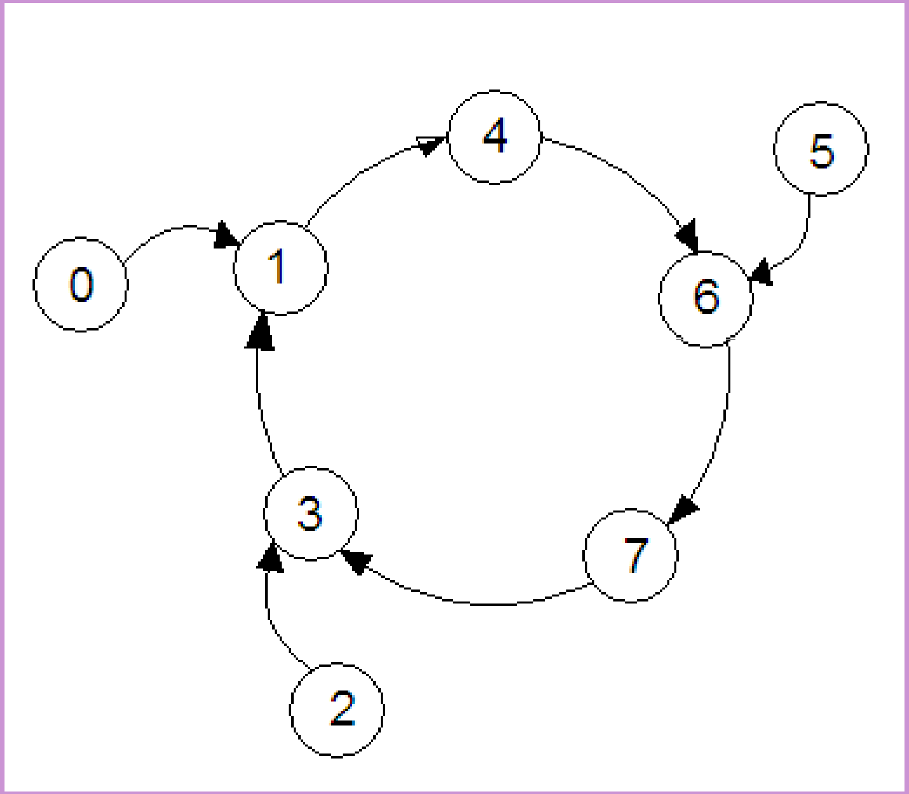
Logic Diagram:

- Design a synchronous counter for count the sequences 4 → 6 → 7 → 3 → 1 → 4…avoid lockout condition and use JK flip flops.

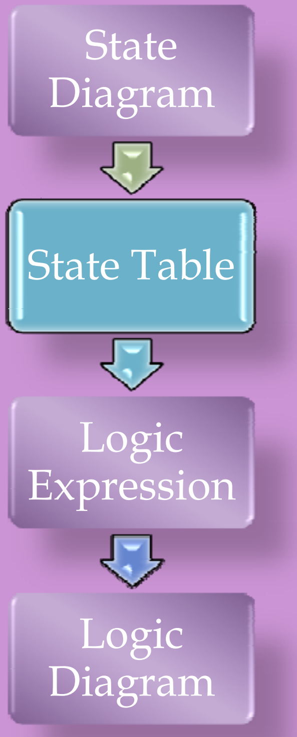


- Design a synchronous counter for count the sequences 4 → 6 → 7 → 3 → 1 → 4…avoid lockout condition and use JK flip flops.



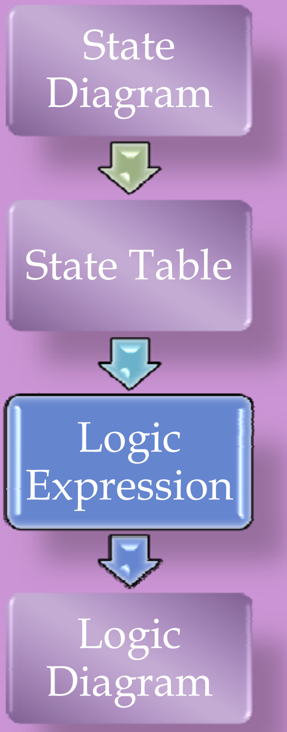


• **Design a synchronous counter for count the sequences 4 → 6 → 7 → 3 → 1 → 4…avoid lockout condition and use JK flip flops.**



Present State			Next State			Flip flop inputs					
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	1	0	0	1	X	0	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	1	0	X	X	1	X	0
1	0	0	1	1	0	X	0	1	X	0	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	1	1	X	1	X	0	X	0

• **Design a synchronous counter for count the sequences 4 → 6 → 7 → 3 → 1 → 4…avoid lockout condition and use JK flip flops.**



Using K map we simplify the expressions for the flip flop inputs

$$J_2 = \overline{Q_1} Q_0 \qquad J_1 = Q_2 \qquad J_0 = \overline{Q_2} + Q_1$$

$$K_2 = Q_1 Q_0 \qquad K_1 = \overline{Q_2} Q_0 \qquad K_0 = \overline{Q_1}$$

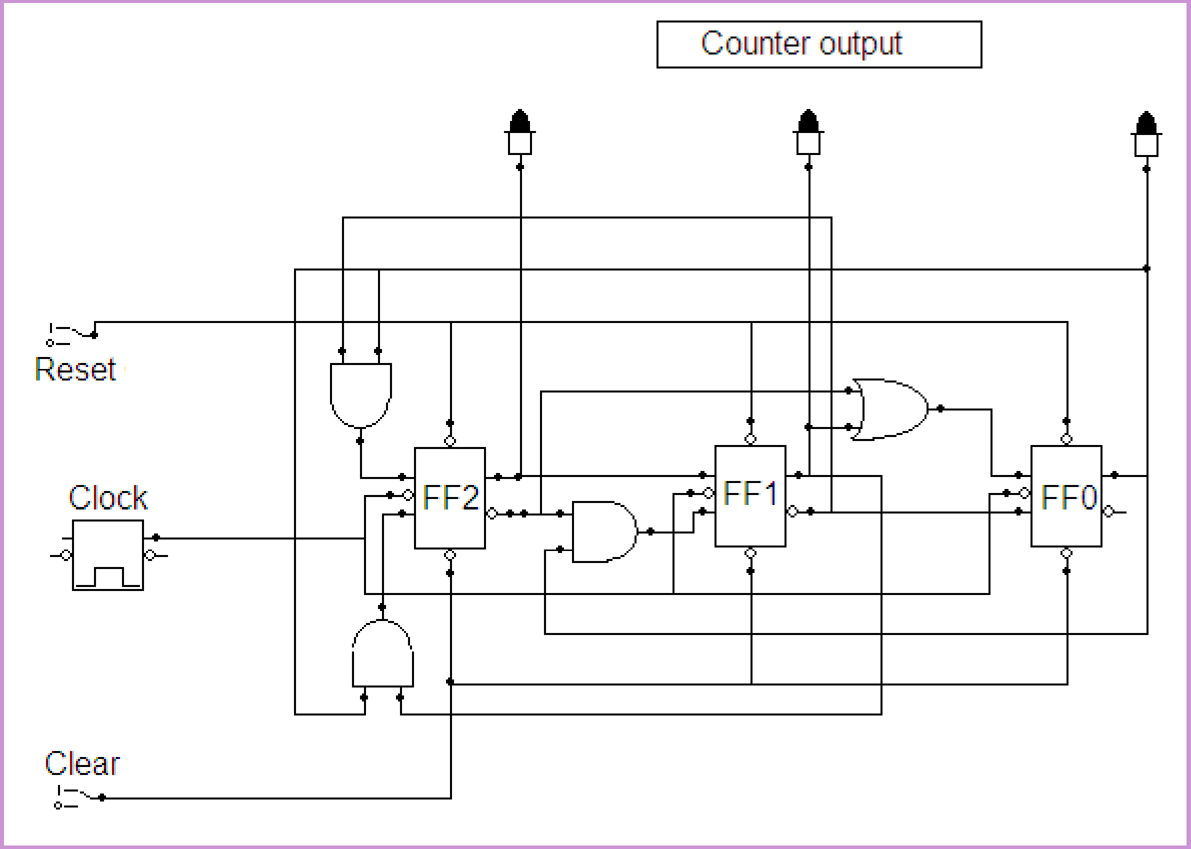
• **Design a synchronous counter for count the sequences 4 → 6 → 7 → 3 → 1 → 4…avoid lockout condition and use JK flip flops.**

State Diagram

State Table

Logic Expression

Logic Diagram



Ring counter

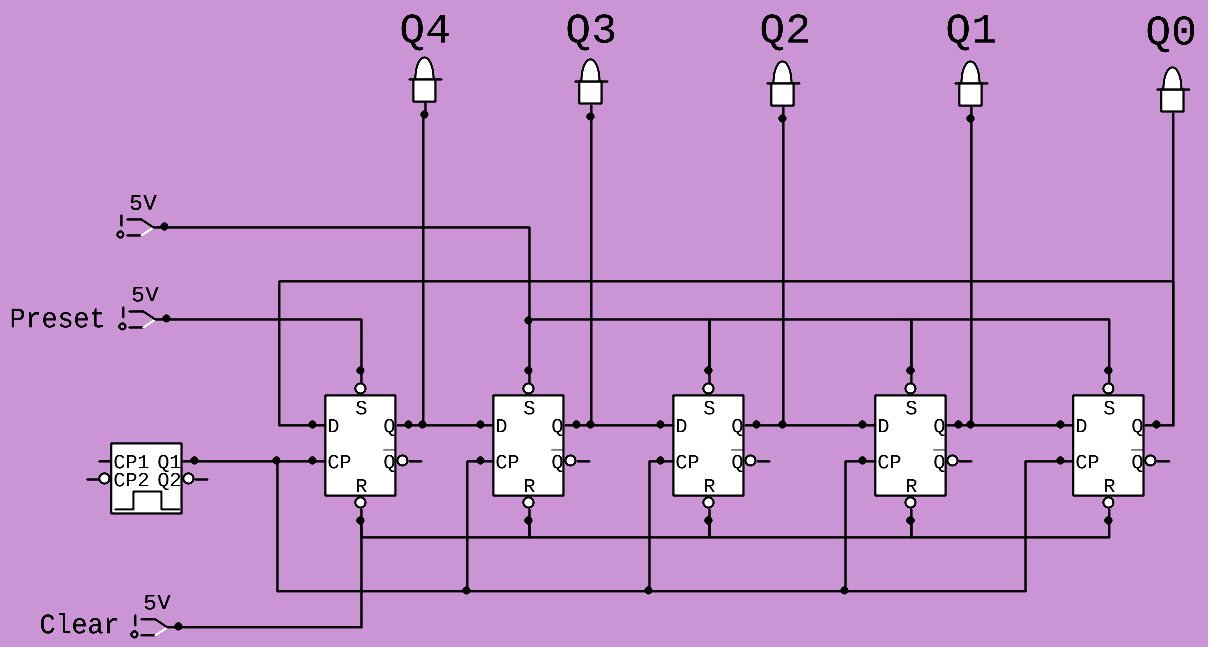
A **ring counter** is a type of counter composed of circular shift register. The output of the last shift register is fed to the input of the first register.

There are two types of ring counters:
A ***straight ring counter*** or ***Overbeck counter*** connects the output of the last shift register to the first shift register input and circulates a single one (or zero) bit around the ring. For example, in a 4-register one-shot counter, with initial register values of 1000, the repeating pattern is: 1000, 0100, 0010, 0001, 1000... . Note that one of the registers must be pre-loaded with a 1 (or 0) in order to operate properly.
A ***twisted ring counter***, also **called *Johnson counter***, connects the complement of the output of the last shift register to the input of the first register and circulates a stream of ones followed by zeros around the ring. For example, in a 4-register counter, with initial register values of 0000, the repeating pattern is: 0000, 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000... .

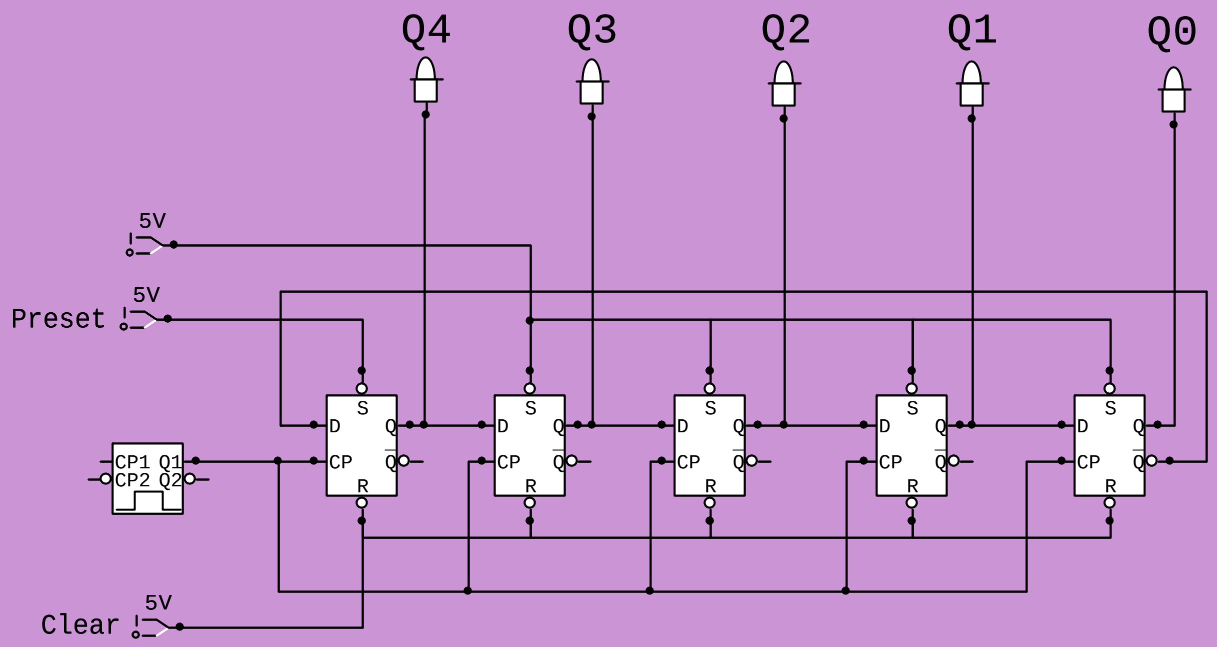
Four-bit ring counter sequences

Straight ring/*Overbeck counter*				
State	Q0	Q1	Q2	Q3
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
0	1	0	0	0

Twisted ring/*Johnson counter*				
State	Q0	Q1	Q2	Q3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
0	0	0	0	0



Ring Counter



Twister Ring / Johnson Counter

Thank you for listening