Unit o6: Sequential Circuits

- **Difference between Asynchronous & Synchronous sequential circuits**
- **Asynchronous counters:**
  1. Up-Counter
  2. Down-Counter
  3. Mod-Counter
- **Working of Shift Registers:**
  1. Serial In Serial Out
  2. Serial In Parallel Out
  3. Parallel In Serial Out
  4. Parallel In Parallel Out
- **Application of Shift Register as a Ring Counter**

Difference between Synchronous & Asynchronous Seq. Circuits

SN	Synchronous Seq. Circuits	Asynchronous Sequential Circuits
1.	In synchronous circuits, memory elements are clocked Flip-flops	In asynchronous circuits, memory elements are either unclocked FFs or time delay elements
2.	The transition from one state to another takes place only by the application of specified clock signal, even if the inputs change	The transition from one state to another takes place immediately once the inputs change
3.	The maximum operating speed of clock depends on time delays involved	Because of the absence of clock, asynchronous circuits can operate faster than synchronous circuits
4.	Easier to design	More difficult to design
5.	These are predictable & reliable.	There are chances for asynchronous circuits to enter into a wrong state
6.	It consumes large power and dissipates large amount of heat.	Power consumption and heat dissipation are comparatively lower.

Asynchronous Counters

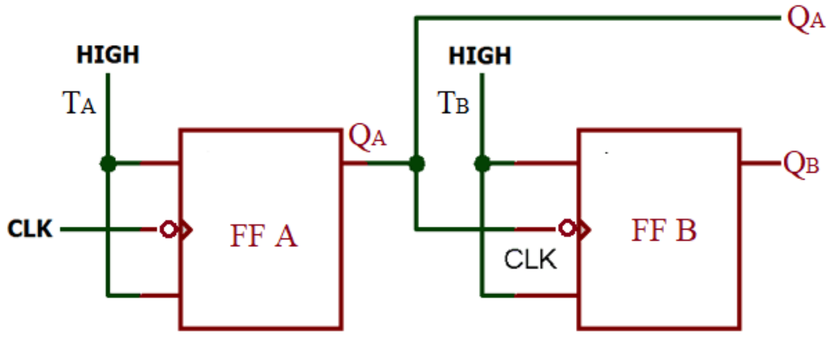
A digital counter is a set of Flip-Flops whose state changes in response to pulses applied at the input to the counter. FFs are interconnected such that their combined state at any time is the binary equivalent of the total number of pulses that have occurred up to that time. Thus, as its name implies, a counter is used to count pulses. Counters may be: Asynchronous [FFs are not triggered simultaneously] or Synchronous [FFs are triggered simultaneously].

Asynchronous counters are also called as Ripple Counters. These are simplest type of counters, the easiest to design and require least amount of hardware. Asynchronous counter uses T Flip-Flops to perform counting operations. The actual hardware used is usually JK FFs connected in Toggle mode [i.e., J and K are connected to logic 1]

- A counter may be an up-counter or down-counter
- An up-counter is a counter which counts in the upward direction, i.e., 0, 1, 2, 3 ..... N whereas, a down-counter is a counter which counts in the downward direction. i.e., N, N-1, N-2,.....1, 0.
- Each of the counts of the counter is called the state of the counter. The number of states through which counter passes before returning to starting state is called the modulus of the counter. Hence, the modulus of counter is equals to total number of counts
- For Ex: 2-bit counter has 4 states and hence it is called as mod-4 counter. Similarly, 3-bit counter is called as mod-8 counter and 4-bit counter is called mod-16 counter
- Note: in general, a n-bit counter will have 'n' flip-flops and 2^n states.

<u>2-Bit Up Counter</u>

The logic diagram of a 2-bit ripple up counter is shown in figure below. Two toggle (T) flip-flops are used here. External clock is applied to the clock input of flip-flop A and Q_A output is applied to the clock input of the next flip-flop i.e. FF-B. Both FFs are negatively edge triggered [i.e., it will change its state only at negative edge of the clock]



OPERATION ➔ Initially, let both FFs be in the reset state [i.e., Q_BQ_A= 00]

1] As soon as the first negative clock edge is applied, FF-A will toggle and Q_A will be equal to 1. Q_A is connected to clock input of FF-B. Since, Q_A has changed from 0 to 1, it is treated as the positive clock edge by FF-B. There is no change in Q_B because FF-B is a negative edge triggered FF. Thus, Q_BQ_A = 01 after the first clock pulse

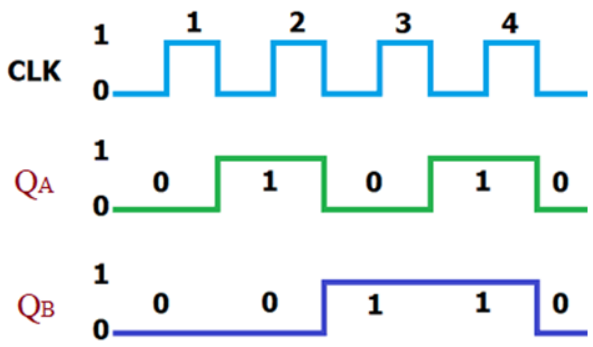
2] On the arrival of second negative clock edge, FF-A toggles again and Q_A=0. The change in Q_A acts as a negative clock edge for FF-B. So it will also toggle and Q_B will be 1. Thus, Q_BQ_A=10 after the second clock pulse

3] On arrival of third negative clock edge, FF-A toggles again and Q_A become 1 from 0. Since this is a positive going change, FF-B does not respond to it and remains inactive. So Q_B does not change and continues to be equal to 1. Q_BQ_A = 11 after third clock pulse

4] On arrival of 4th negative clock edge, FF-A toggles again and Q_A becomes 0 from 1. This negative change in Q_A acts as clock pulse for FF-B. Hence, it toggles to change Q_B

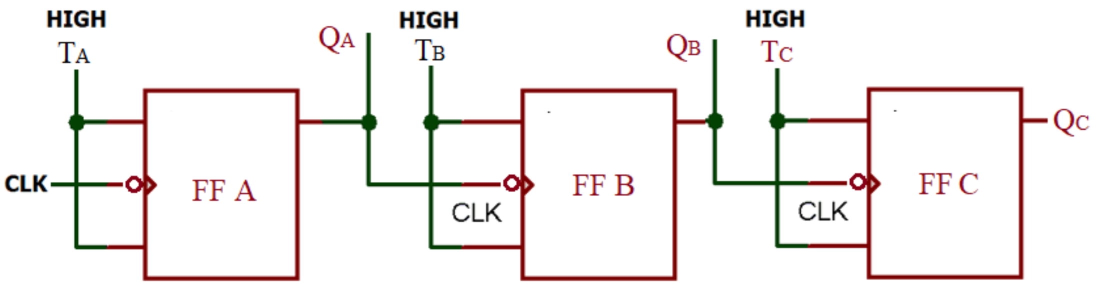
from 1 to 0. Q_BQ_A = 00 after the fourth clock pulse. i.e., counter will RESET itself after fourth clock pulse ! Truth table and waveforms are shown below:

Clock	Q_B	Q_A
Initially	0	0
1st	0	1
2nd	1	0
3rd	1	1
4th	0	0



3-Bit Up Counter

The logic diagram of a 3-bit ripple up counter is shown in figure below. Three toggle (T) flip-flops are used here. External clock is applied to the clock input of flip-flop A. Q_A output is applied to the clock input of the next flip-flop i.e. FF-B. Similarly, Q_B output is applied to the clock input of FF-C. All three FFs are negatively edge triggered [i.e., it will change its state only at negative edge of the clock]



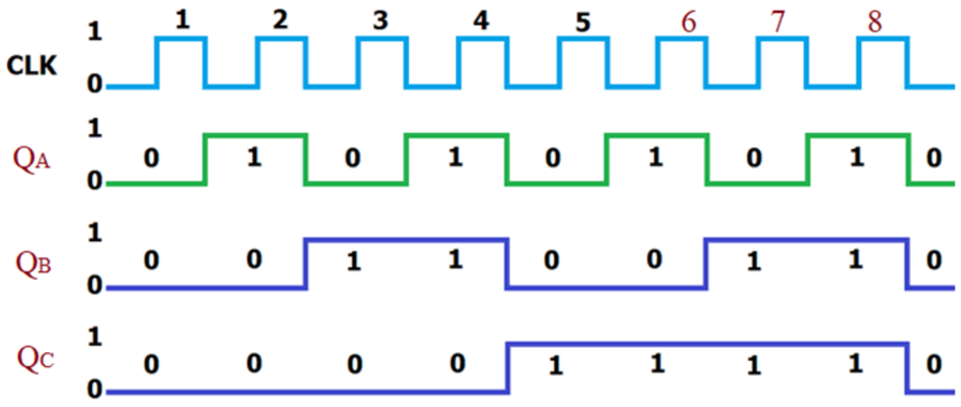
OPERATION ➔ Initially, let all FFs be in the reset state [i.e.,$Q_CQ_BQ_A$= 000]

1] As soon as the first negative clock edge is applied, FF-A will toggle and Q_A will be equal to 1. Q_A is connected to clock input of FF-B. Since, Q_A has changed from 0 to 1, it is treated as the positive clock edge by FF-B. There is no change in Q_B because FF-B is a negative edge triggered FF. Similarly, there is no change in Q_C. Thus, $Q_CQ_BQ_A$ = 001 after the first clock pulse

2] On the arrival of second negative clock edge, FF-A toggles again and Q_A=0. The change in Q_A acts as a negative clock edge for FF-B. So it will also toggle and Q_B will be 1. Since, Q_B has changed from 0 to 1, it is treated as the positive clock edge by FF-C. There is no change in Q_C because FF-C is a negative edge triggered FF. Thus, $Q_CQ_BQ_A$=010 after the second clock pulse

3] Similarly, count will follow the order $Q_CQ_BQ_A$=011, $Q_CQ_BQ_A$=100, $Q_CQ_BQ_A$=101, $Q_CQ_BQ_A$=110, $Q_CQ_BQ_A$=111 on the arrival of third, fourth, fifth, sixth and seventh negative clock edge respectively.

4] On arrival of 8th negative clock edge, FF-A toggles again and Q_A becomes 0 from 1. This negative change in Q_A acts as clock pulse for FF-B. Hence, it toggles to change Q_B from 1 to 0. Again this negative change in Q_B acts as clock pulse for FF-C. Hence, it toggles to change Q_C from 1 to 0 giving $Q_CQ_BQ_A$ = 000 after the 8th clock pulse (i.e., counter will RESET itself after 8th clock pulse ! Truth table and waveforms are shown below:



Truth table of 3-Bit Up Counter:

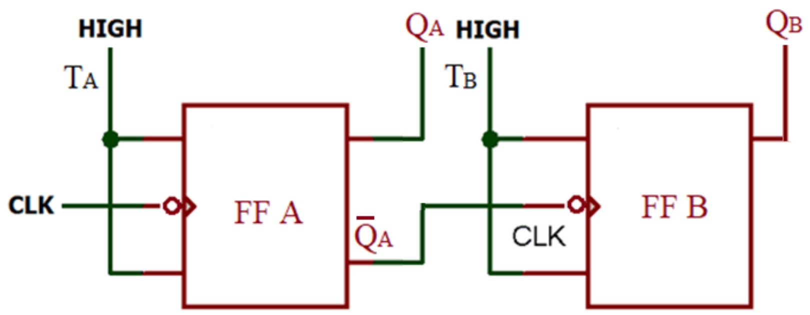
Clock	Q_C	Q_B	Q_A
Initially	0	0	0
1st	0	0	1
2nd	0	1	0
3rd	0	1	1
4th	1	0	0
5th	1	0	1
6th	1	1	0
7th	1	1	1
8th	0	0	0

2-Bit Down Counter

The logic diagram of a 2-bit ripple down counter is shown in figure below. Two toggle (T) flip-flops are used here. External clock is applied to the clock input of flip-flop A and $\overline{Q}_A$ output is applied to the clock input of the next flip-flop i.e. FF-B. Both FFs are negatively edge triggered [i.e., it will change its state only at negative edge of the clock]

OPERATION ➔ Initially, let both FFs be in the RESET state [i.e., Q_BQ_A= 00]

1] As soon as the first negative clock edge is applied, FF-A will toggle and Q_A will be equal to 1 and $\overline{Q}_A$ will be equal to 0. $\overline{Q}_A$ is connected to clock input of FF-B. Since, $\overline{Q}_A$ has changed from 1 to 0, it is treated as the negative clock edge by FF-B. It will toggle and Q_B will be equal to 1. Thus, Q_BQ_A = 11 after the first clock pulse

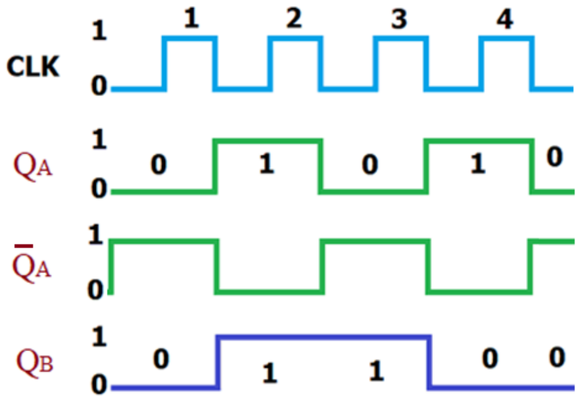


2] On the arrival of second negative clock edge, FF-A toggles again and Q_A will change from 1 to 0. Similarly, $\overline{Q}_A$ will change from 0 to 1. This change in $\overline{Q}_A$ acts as a positive clock edge for FF-B. So, it will not change its state and Q_B continues to be equal to 1. Thus, Q_BQ_A=10 after the second clock pulse

3] On arrival of third negative clock edge, FF-A toggles again and Q_A will change from 0 to 1. Similarly, $\overline{Q}_A$ will change from 1 to 0. It is treated as the negative clock edge by FF-B. It will toggle and Q_B will be equal to 0. Thus, Q_BQ_A = 01 after the third clock pulse

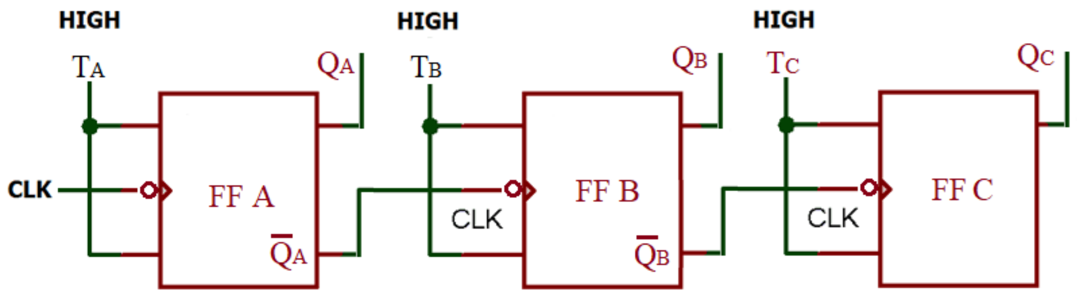
4] On arrival of 4th negative clock edge, FF-A toggles again and Q_A will change from 1 to 0. Similarly, $\overline{Q}_A$ will change from 0 to 1. This change in $\overline{Q}_A$ acts as a positive clock edge for FF-B. So it will not change its state and Q_B continues to be equal to 0. Thus, Q_BQ_A=00 after the fourth clock pulse. i.e., counter will RESET itself after 4th clock pulse ! Truth table & waveforms are shown below:

Clock	Q_B	Q_A
Initially	0	0
1st	1	1
2nd	1	0
3rd	0	1
4th	0	0



3-Bit Down Counter

The logic diagram of a 3-bit ripple down counter is shown in figure below. Three toggle (T) flip-flops are used here. External clock is applied to the clock input of flip-flop A. $\overline{Q}_A$ output is applied to the clock input of the next flip-flop i.e. FF-B. Similarly, $\overline{Q}_B$ output is applied to the clock input of FF-C. All three FFs are negatively edge triggered [i.e., it will change its state only at negative edge of the clock]

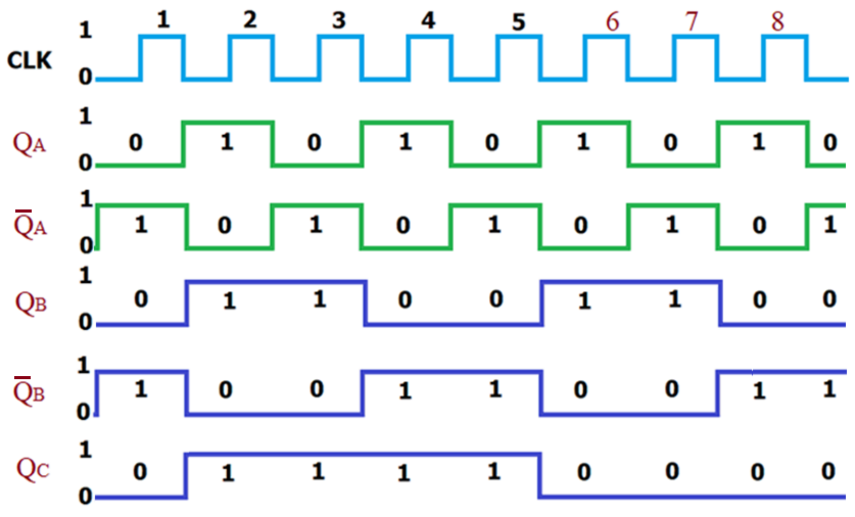


OPERATION ➔ Initially, let all FFs be in the reset state [i.e.,$Q_CQ_BQ_A$= 000]

1] As soon as the first negative clock edge is applied, FF-A will toggle and Q_A will be equal to 1 and $\overline{Q}_A$ will be equal to 0. $\overline{Q}_A$ is connected to clock input of FF-B. Since, $\overline{Q}_A$ has changed from 1 to 0, it is treated as the negative clock edge by FF-B. It will toggle and Q_B will be equal to 1 and $\overline{Q}_B$ will be equal to 0. Also, $\overline{Q}_B$ has changed from 1 to 0, it is treated as the negative clock edge by FF-C. It will toggle and Q_C will be equal to 1. Thus, $Q_CQ_BQ_A$ = 111 after the first clock pulse

2] On the arrival of second negative clock edge, FF-A toggles again and Q_A will change from 1 to 0. Similarly, $\overline{Q}_A$ will change from 0 to 1. This change in $\overline{Q}_A$ acts as a positive clock edge for FF-B. So, it will not change its state as FF-B is negatively edge triggered flip-flop and Q_B continues to be equal to 1. Since, Q_B has not changed there will not be any change in Q_C either. Thus, $Q_CQ_BQ_A$=110 after the second clock pulse

3] Similarly, count will follow the order $Q_CQ_BQ_A$=101, $Q_CQ_BQ_A$=100, $Q_CQ_BQ_A$=011, $Q_CQ_BQ_A$=010, $Q_CQ_BQ_A$=001 on the arrival of third, fourth, fifth, sixth and seventh negative clock edge respectively. After the 8th clock pulse, counter will RESET itself and $Q_CQ_BQ_A$=000.



Truth table of 3-Bit Down Counter:

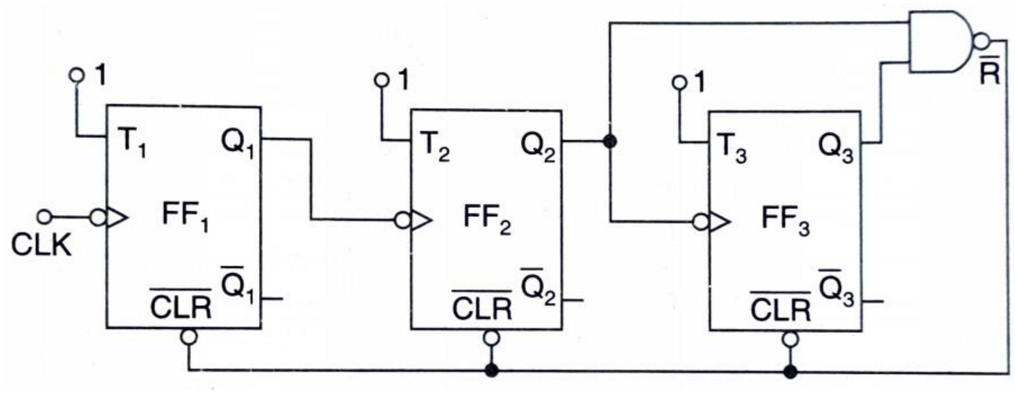
Clock	Q_C	Q_B	Q_A
Initially	0	0	0
1st	1	1	1
2nd	1	1	0
3rd	1	0	1
4th	1	0	0
5th	0	1	1
6th	0	1	0
7th	0	0	1
8th	0	0	0

MOD Counters:

N-bit asynchronous counter can have $(2^N - 1)$ possible counting states. For eg. Mod-8 for a 3-bit counter (0-7), Mod-16 for a 4-bit counter (0-15). But it is also possible to use the basic asynchronous counter configuration to construct special counters with counting states less than their maximum output number. Such counters are called as modulo or MOD counters. By using one or more external logic gates few output states can be skipped. Thus, in the case of MOD counters, it does not count to all their possible states, but instead it counts to specific count value and then return to zero. Counters can also have additional inputs called CLEAR and PRESET which makes it possible to clear the count to zero (all Q = 0) or to preset the counter to some initial value

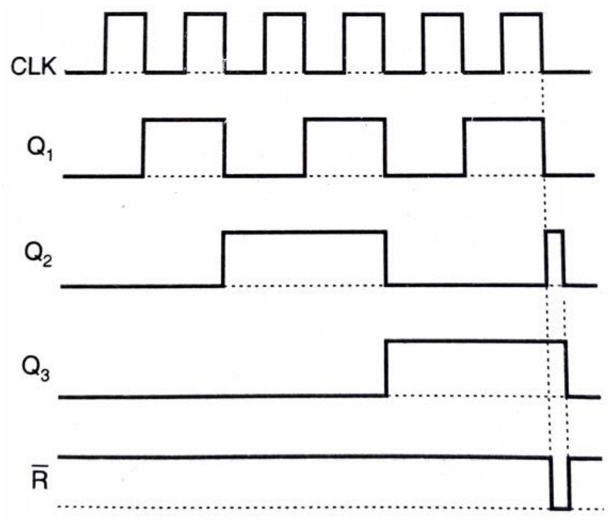
1. Mod-6 Asynchronous Counter:

A mod-6 counter has six stable states 000, 001, 010, 011, 100 and 101. It requires three Flip-flops. Three FFs have eight possible states out of which only six are utilized and remaining states 110 and 111 are invalid. When the sixth clock pulse is applied, the counter temporarily goes to 110 state, but immediately resets to 000 because of the feedback provided. Figure below shows logic diagram of mod-6 counter:



OPERATION ➔

The working of mod-6 counter is exactly same as the 3-bit Up Counter except it will count only SIX counts. Q_1, Q_2 and Q_3 are outputs of flip-flops 1, 2, 3 respectively. A NAND Gate is used in order to skip remaining states. Its inputs are Q_3 and Q_2 whereas $\overline{R}$ is the output. At the arrival of sixth clock pulse, both inputs to NAND gate becomes 1 and hence, output will be 0. This output is connected to CLEAR terminal of all FFs which will reset the count to 000.

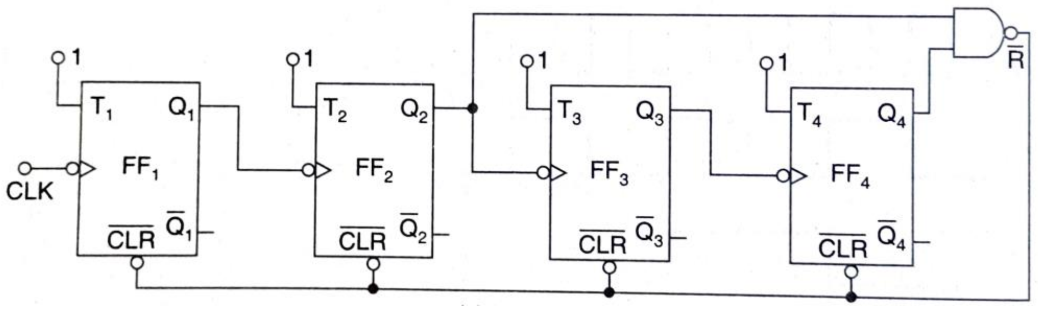


Truth table of Mod-6 Counter:

Clock	Q_3	Q_2	Q_1	$\overline{R}$
Initially	0	0	0	1
1st	0	0	1	1
2nd	0	1	0	1
3rd	0	1	1	1
4rth	1	0	0	1
5th	1	0	1	1
6th	1 $\downarrow$	1 $\downarrow$	0 $\downarrow$	0
	0	0	0	

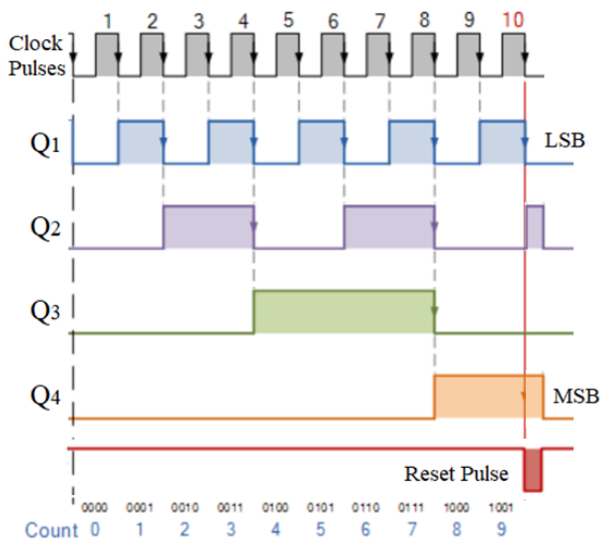
<u>2. Mod-10 Asynchronous Counter:</u>

A mod-10 counter is also called as Decade Counter. It has ten stable states 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001. It requires four T Flip-flops. Four FFs have 16 possible states out of which only ten states are utilized and remaining six states (1010, 1011, 1100, 1101, 1110, 1111) are invalid. When the tenth clock pulse is applied, the counter temporarily goes to 1010 state, but immediately resets to 0000 because of the feedback provided. Figure below shows logic diagram of mod-10 counter:



OPERATION ➔

The working of mod-10 counter is exactly same as the 4-bit Up Counter except it will count only TEN counts. Q_1, Q_2, Q_3 and Q_4 are outputs of flip-flops 1, 2, 3, 4 respectively. A NAND Gate is used in order to skip remaining states. Its inputs are Q_4 and Q_2 whereas $\overline{R}$ is the output. At the arrival of 10th clock pulse, both inputs to NAND gate becomes 1 and hence, output will be 0. This output is connected to CLEAR terminal of all FFs which will reset count to 0000



Truth table of Mod-10 Counter:

Clock	Q_4	Q_3	Q_2	Q_1	$\overline{R}$
Initially	0	0	0	0	1
1st	0	0	0	1	1
2nd	0	0	1	0	1
3rd	0	0	1	1	1
4rth	0	1	0	0	1
5th	0	1	0	1	1
6th	0	1	1	0	1
7th	0	1	1	1	1
8th	1	0	0	0	1
9th	1	0	0	1	1
10th	1 $\downarrow$	0 $\downarrow$	1 $\downarrow$	0 $\downarrow$	0
	0	0	0	0	

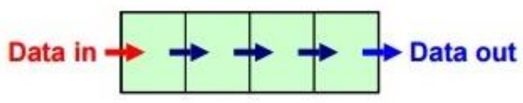
Shift Registers:

Flip-flops can be used to store single bit of binary data (1 or 0). However, in order to store multiple bits of data, we need multiple flip-flops. 'N' flip-flops are to be connected in an order to store N-bits of data. A Register is a device which is used to store such information. It is a group of flip-flops connected in series used to store multiple bits of data. The information stored within registers can be transferred with the help of shift registers. Shift Register is a group of flip-flops used to store multiple bits of data. The bits stored in such registers can be made to move within the registers and in/out of the registers by applying clock pulses. An N-bit shift register can be formed by connecting 'N' flip-flops where each flip-flop stores a single bit of data.

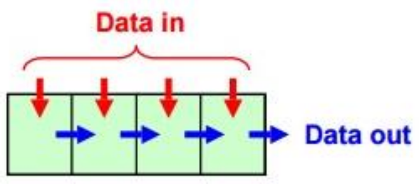
A shift register basically consists of several "D Flip-flops", one for each data bit (either a logic 0 or 1) connected together in a series arrangement so that the output from one D flip-flop becomes the input of the next flip-flop and so on. Data bits may be fed in or out of a shift register serially (that is one after the other from either the left or the right direction) or all together at the same time in a parallel configuration. Shift Registers are used for data storage or for the movement of data and are therefore commonly used inside calculators or computers to store data such as two binary numbers before they are added together, or to convert the data from either a serial to parallel or parallel to serial format. The D flip-flops that make up a shift register are all driven by a common clock signal making them synchronous device.

Generally, shift registers operate in one of four different modes with the basic movement of data through a shift register being:

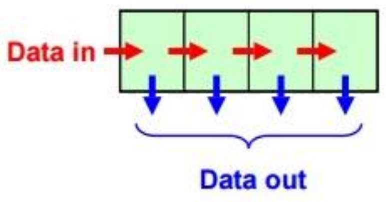
1. Serial-in to Serial-out (SISO): Data is shifted serially IN and OUT of the register, one bit at a time in either left/right direction under clock control.
2. Serial-in to Parallel-out (SIPO): Register is loaded with serial data, one bit at a time, with stored data being available at output in parallel form.
3. Parallel-in to Serial-out (PISO): The parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
4. Parallel-in to Parallel-out (PIPO): The parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.



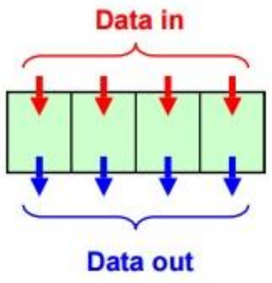
SISO - Serial IN Serial OUT Shift Register



PISO-Parallel IN Serial OUT shift register



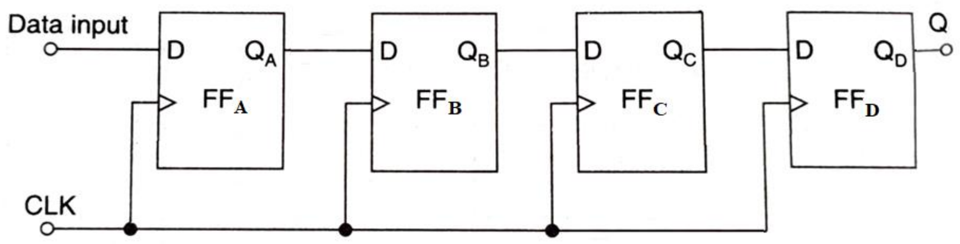
SIPO-Serial IN Parallel OUT shift register



PIPO-Parallel-IN Parallel-OUT shift register

1. Serial-in to Serial-out (SISO) Shift Register:

The shift register, which allows serial input (one bit after the other through a single data line) and produces a serial output is known as Serial-In Serial-Out shift register. The logic circuit given below shows a 4-bit serial-in serial-out shift register. The circuit consists of four D flip-flops which are connected in a serial manner. All these flip-flops are synchronous with each other since the same clock signal is applied to each flip flop.

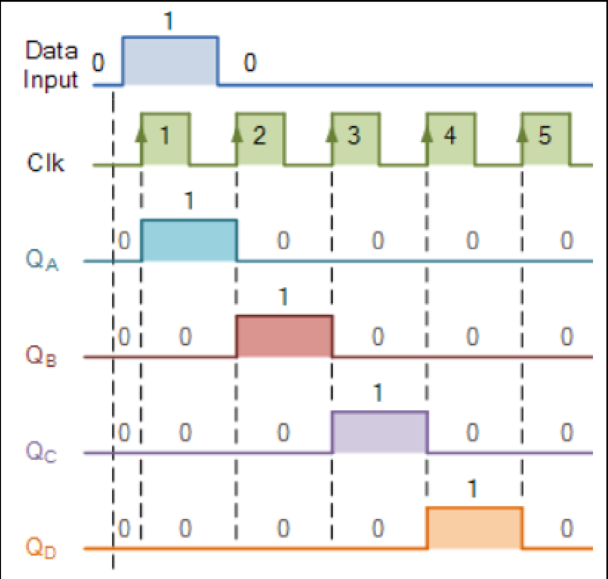


WORKING OPERATION:

Serial data is applied at the D input of the first FF. The output of FF-A (Q_A) is connected to the D input of FF-B, The output of FF-B (Q_B) is connected to the D input of FF-C and The output of FF-C (Q_C) is connected to the D input of FF-D.

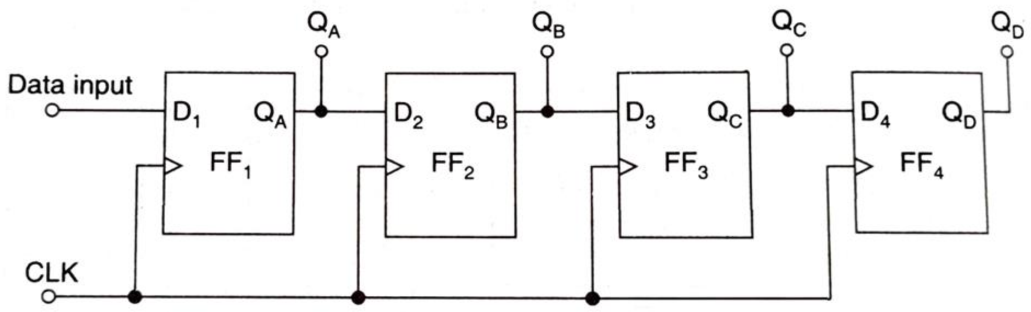
The data is outputted from the output terminal of last FF (i.e., Q_D). When serial data is transferred into a register, each new bit is clocked into the first Flip-flop at the positive edge of each clock pulse. The bit that was previously stored by FF-A is transferred to FF-B. The bit that was stored by FF-B is transferred to FF-C and so on. The bit that was stored by last flip-flop (FF-D) is shifted out. The truth table and following waveforms show the propagation of the logic "1" through the register from left to right as follows:

Clock Pulse	QA	QB	QC	QD
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1
5	0	0	0	0



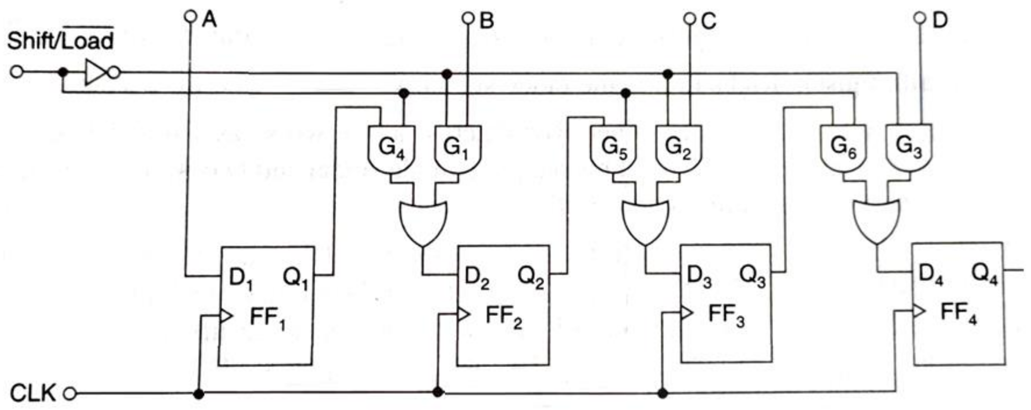
<u>2. Serial-in to Parallel-out (SIPO) Shift Register:</u>

Figure shows the logic diagram of 4-bit serial-in parallel-out shift register. In this type of register, the data bits are entered into the register serially, but the data stored in the register is shifted out in parallel form. Once the data bits are stored, each bit appears on its respective output line and all bits are available simultaneously. SIPO register can be used as SISO if the output is taken from the Q output of the last FF.



3. Parallel-in to Serial-out (PISO) Shift Register:

For a parallel-in serial-out shift register, data bits are entered simultaneously into their respective stages on parallel lines but data bits are transferred out of the register serially. Figure illustrates 4-bit parallel-in serial-out shift register using 4 D Flip-flops. A, B, C & D are data lines through which data is entered into register in parallel form.



WORKING OPERATION:

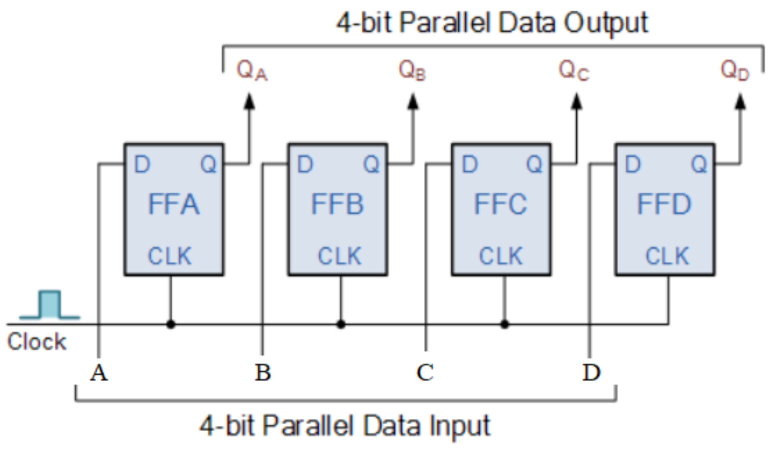
The signal Shift/$\overline{\text{Load}}$ allows

1. Data to be entered in parallel form into the register and
2. Data to be shifted out serially from terminal $\overline{Q}_4$

When Shift/$\overline{\text{Load}}$ line is LOW, gates G_4, G_5 and G_6 are disabled, but gates G_1, G_2, G_3 are enabled allowing data input to appear at D inputs of respective FFs. When Shift/$\overline{\text{Load}}$ line is HIGH, gates G_1, G_2, G_3 are disabled, but gates G_4, G_5 and G_6 are enabled allowing the data bits to shift right from one stage to next. When a clock pulse is applied, these data bits are shifted to the Q output terminals of the FFs and therefore, data is inputted in one step. The OR gate allows either the normal shifting operation or the parallel data entry depending on which AND gates are enabled by the level on Shift/$\overline{\text{Load}}$ input.

4. Parallel-in to Parallel-out (PIPO) Shift Register:

In 4-bit parallel-in parallel-out shift register, the data bits are entered into the register in parallel form and also data is taken out of register in parallel form. Immediately following simultaneous entry of all data bits, bits appear on the parallel output. Data is applied to the D input terminals of all the FFs. When a clock pulse is applied the D inputs are shifted into Q outputs of the FFs. The register now stores the data. The stored data is available instantaneously for shifting out in parallel form. Figure shows the logic diagram of 4-bit parallel-in parallel-out shift register.



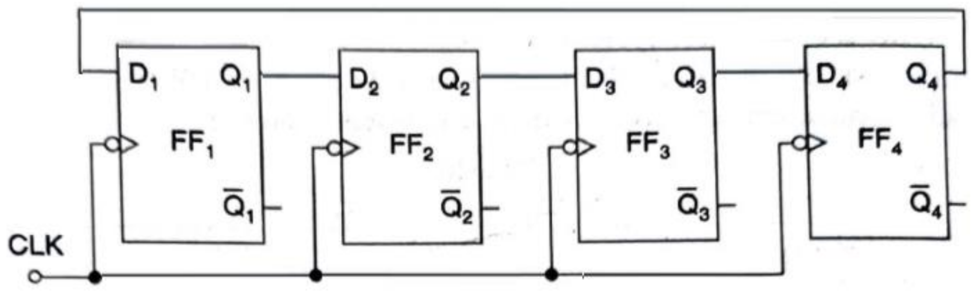
Shift Register Counters:

One of the applications of shift register is that they can be arranged to form several types of counters. Shift registers counters are obtained from serial-in serial-out shift register by providing feedback from output of last flip-flop to the input of first FF. These devices are called as counters because they exhibit a specified sequence of states. The most widely used shift register counters are: Ring Counter and Twisted Ring Counter.

Shift register counters are also known as Synchronous Counters. These are different from Asynchronous or Ripple counters. A common clock triggers all flip-flops simultaneously rather than one at a time in succession as in a Ripple counter.

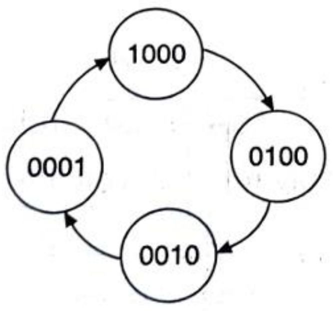
A. Ring Counter:

A Ring Counter is a circular shift register with only one flip-flop being set at any particular time, all others are cleared. This is the simplest shift register counter. The basic ring counter using D FFs is shown in the figure.

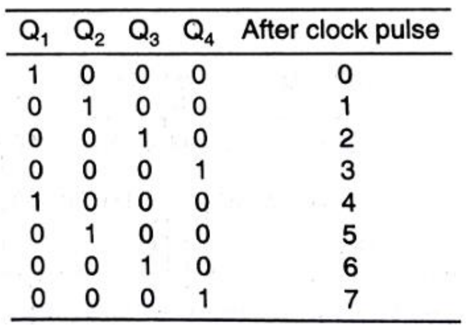

D_1 Q_1 FF_1 $\bar{Q}_1$
D_2 Q_2 FF_2 $\bar{Q}_2$
D_3 Q_3 FF_3 $\bar{Q}_3$
D_4 Q_4 FF_4 $\bar{Q}_4$
CLK

The FFs are arranged in a normal shift register i.e., the output Q of each stage is connected to the D input of the next stage, but the Q output of last FF is connected back to the D input of the first FF such that array of FF is arranged in a Ring and therefore, the named as Ring Counter.

Initially, the first FF is preset to 1. So, the initial states is 1000 (i.e., Q_1=1, Q_2=0, Q_3=0 and Q_4=0). After each clock pulse, the contents of register are shifted to the right by one bit and Q_4 is shifted back to Q_1. The sequence repeats after four clock pulses. The number of distinct states in ring counter (i.e., mod of ring counter = number of FFs used in the counter)



State diagram

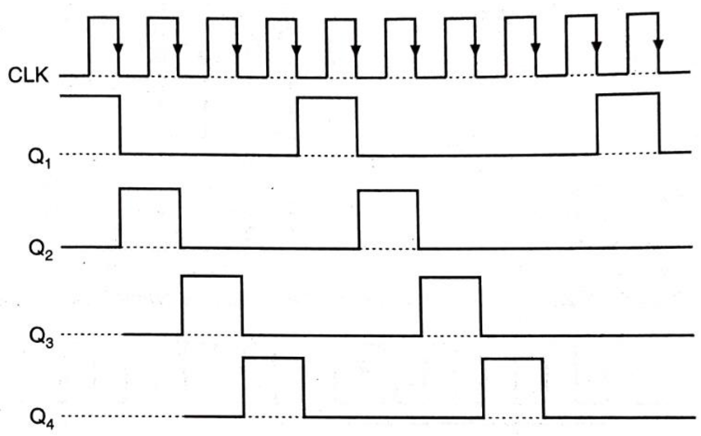


Q_1	Q_2	Q_3	Q_4	After clock pulse
1	0	0	0	0
0	1	0	0	1
0	0	1	0	2
0	0	0	1	3
1	0	0	0	4
0	1	0	0	5
0	0	1	0	6
0	0	0	1	7

Sequence table

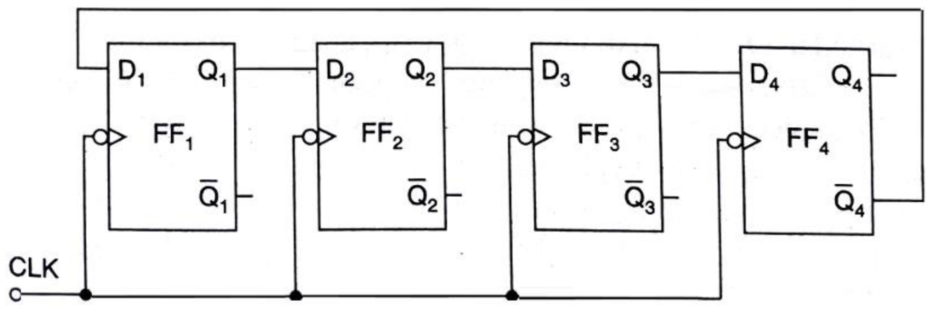
Figure shows timing diagram of 4-bit Ring Counter. N-bit Ring Counter can count only N states whereas, N-bit Ripple Counter can count 2^N states.

So, Ring counter is uneconomical compared to Ripple counter. But, it has advantage of requiring no decoder since, we can count simply by noting which FF is set. Also since, it is entirely a synchronous operation and requires no gates external to FFs, it has further advantage of being very fast.

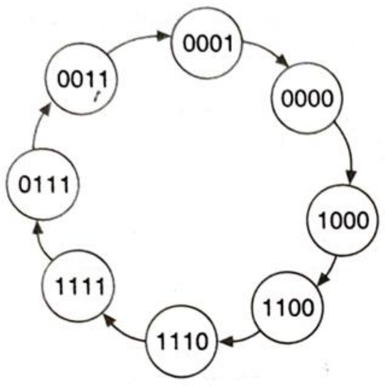


B. Twisted Ring Counter (Johnson Counter):

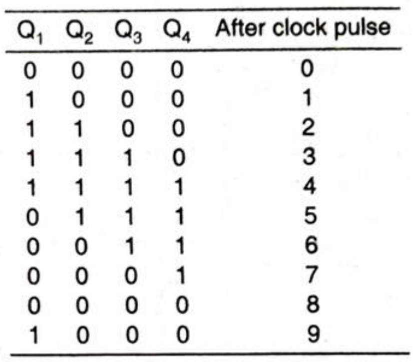
This counter is obtained from a serial-in serial-out shift register by providing feedback from the inverted output of last FF to the D input of first FF. The output Q of each stage is connected to the D input of the next stage, but the $\overline{Q}$ output of last FF is connected back to the D input of the first FF therefore, the named as Twisted Ring Counter. Following figure shows logic diagram for 4-bit Johnson Counter.



Initially, all FFs are RESET to 0. Thus, state of counter will be 0000. After each clock pulse, Q_1 is shifted to Q_2, Q_2 is shifted to Q_3, Q_3 to Q_4 and $\overline{Q}_4$ to Q_1 and sequence given below is obtained. This sequence is repeated after every eight clock pulses. 'N' FF Johnson counter can have '2n' unique states and can count up to '2n' pulses.



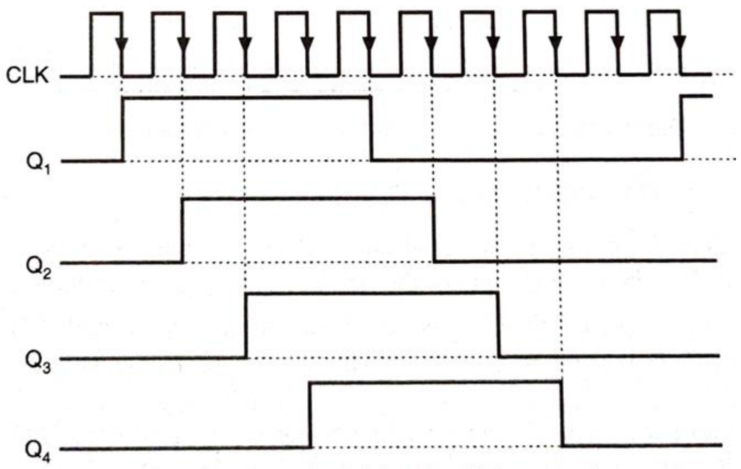
State diagram



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

Sequence table

Twisted Ring counter is more economical than Ring Counter but less economical than Ripple counter. Figure shows timing diagram of 4-bit Twisted Ring Counter.



Timing diagram of a 4-bit twisted ring counter

Important Questions:

1. Design and explain 4-bit asynchronous/ripple up counter with truth table and timing diagram (waveforms).
2. Design and explain 3-bit asynchronous/ripple down counter with truth table.
3. Design and explain MOD-10 decade up counter with truth table.
4. Describe operation of SISO shift register with proper diagram & truth table
5. Describe operation of SIPO shift register with proper diagram and truth table
6. Explain the operation of 4-bit Ring Counter using D flip-flops.
7. Describe operation of 4-bit Twisted Ring Counter