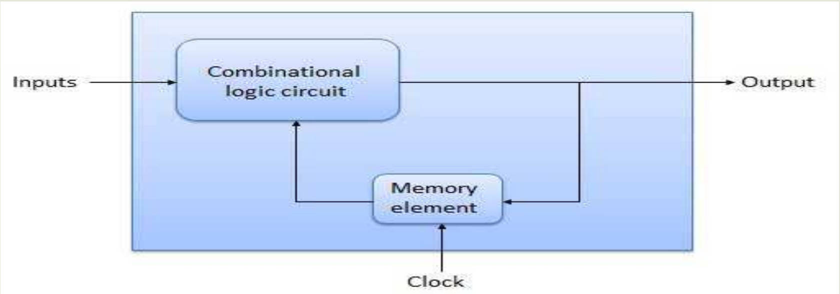
# Logical Organization of Computer CHAPTER 5

## Sequential Circuits

Developed By:
Vivek Vyas
Department of MCA
DDU

## ❑ Introduction

- The combinational circuit does not use any memory. Hence the previous state of input does not have any effect on the present state of the circuit. But sequential circuit has memory so output can vary based on input. This type of circuits uses previous input, output, clock and a memory element.



## Introduction

- A **Sequential circuit** combinational logic circuit that consists of inputs variable (X), logic gates (Computational circuit), and output variable (Z).

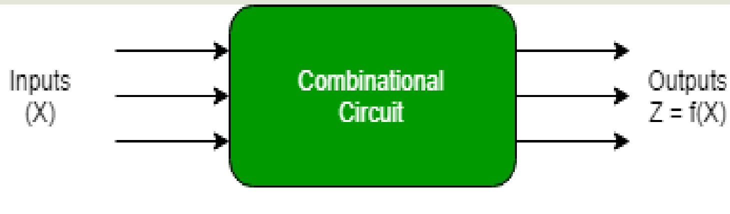


Figure: Combinational Circuits

## Introduction

- Combinational circuit produces an output based on input variable only, but **Sequential circuit** produces an output based on **current input and previous input variables**. That means sequential circuits include memory elements which are capable of storing binary information. That binary information defines the state of the sequential circuit at that time. A latch capable of storing one bit of information.

## Introduction

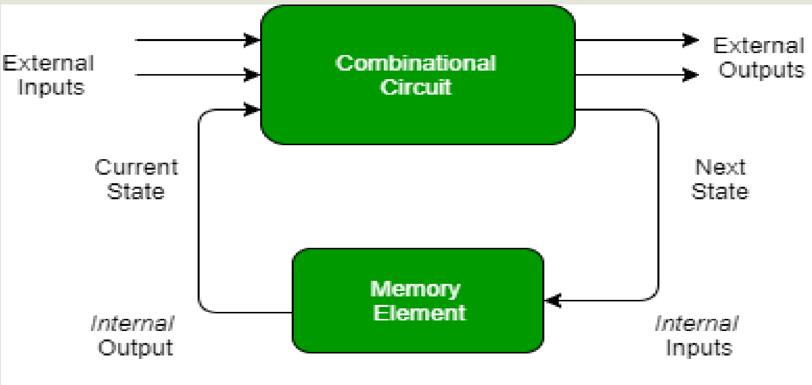


**Figure:** Sequential Circuit

## Introduction

- **Types of Sequential Circuits –** There are two types of sequential circuit :
- **Asynchronous sequential circuit –** These circuit **do not use a clock signal** but uses the pulses of the inputs. These circuits are **faster** than synchronous sequential circuits because there is clock pulse and change their state immediately when there is a change in the input signal.
- We use asynchronous sequential circuits when speed of operation is important and **independent** of internal clock pulse. But these circuits are more **difficult** to design and their output is **uncertain**.

## ❑ Introduction

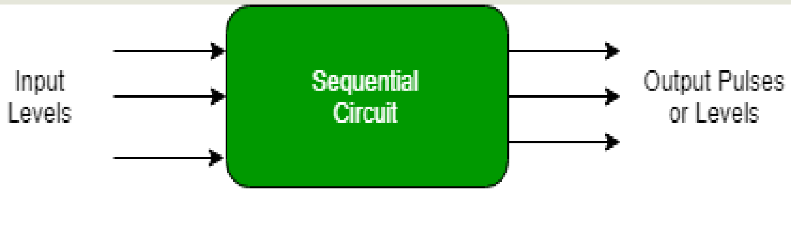


**Figure:** Asynchronous Sequential Circuit

## ❑ Introduction

- **Synchronous sequential circuit** – These circuit **uses clock signal** and level inputs (or pulsed). The output pulse is the same duration as the clock pulse for the clocked sequential circuits.
- Since they wait for the next clock pulse to arrive to perform the next operation, so these circuits are bit **slower** compared to asynchronous.
- We use sequential circuits to design Counters, Registers, RAM.

## ❑ Introduction

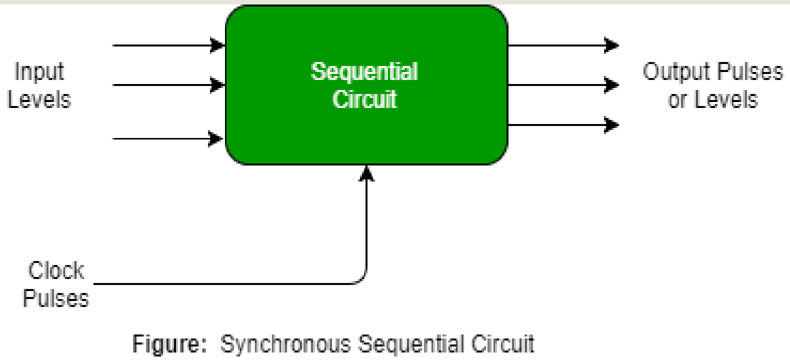
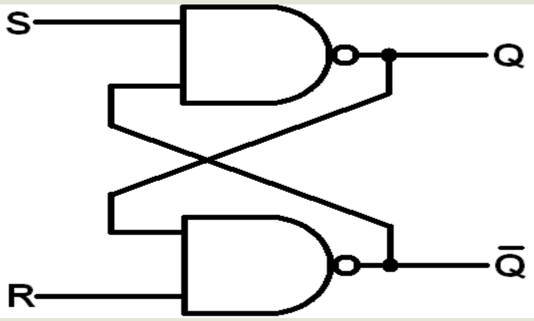
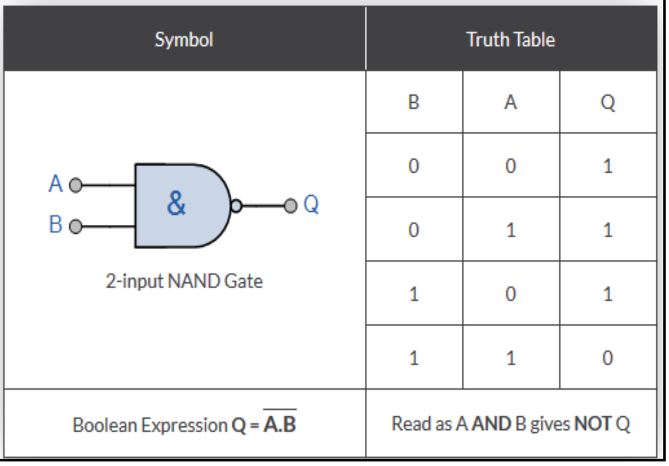


**Figure:** Synchronous Sequential Circuit

## 1. Latch

- The basic storage element is called latch. As the name suggest it latches 0 or 1.
- SR NAND Latch

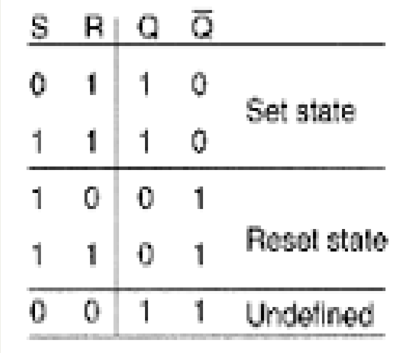



A
B
&
Q
2-input NAND Gate

| Symbol | Truth Table | | |
|---|---|---|---|
| | B | A | Q |
| | 0 | 0 | 1 |
| | 0 | 1 | 1 |
| | 1 | 0 | 1 |
| | 1 | 1 | 0 |
| Boolean Expression **Q = $\overline{A.B}$** | Read as A **AND** B gives **NOT** Q | | |

# 1. Latch

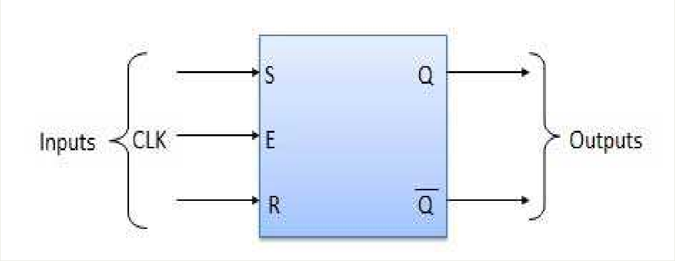
- SR NAND Latch Truth Table



| S | R | Q | $\bar{Q}$ | |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | Set state |
| 1 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 1 | Reset state |
| 1 | 1 | 0 | 1 | |
| 0 | 0 | 1 | 1 | Undefined |

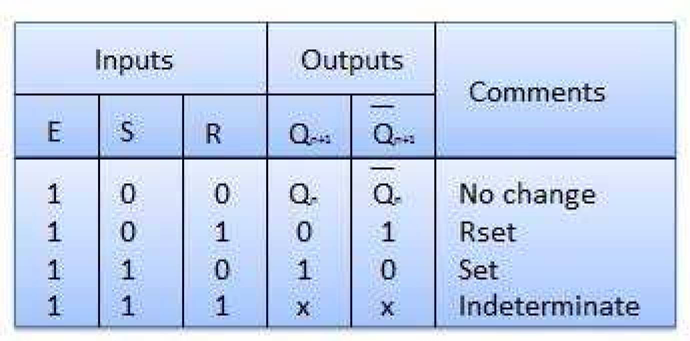
# 2. S-R Flip Flop

- It is basically S-R latch using NAND gates with an additional **enable** input. It is also called as level triggered SR-FF. For this, circuit in output will take place if and only if the enable input (E) is made active. In short this circuit will operate as an S-R latch if E = 1 but there is no change in the output if E = 0.



## 2. S-R Flip Flop

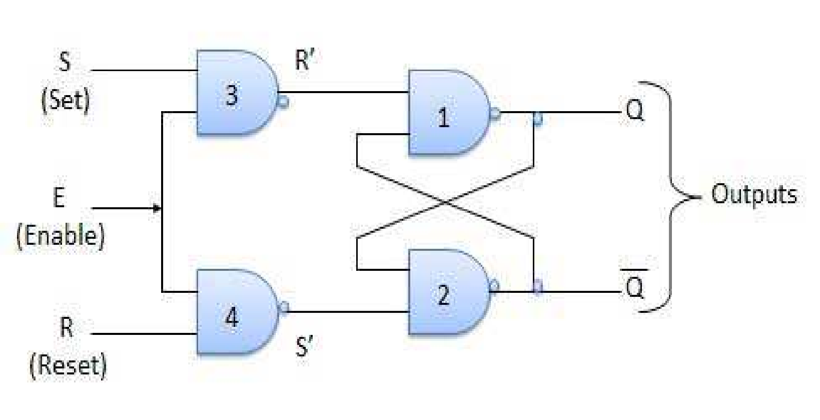
- Truth Table

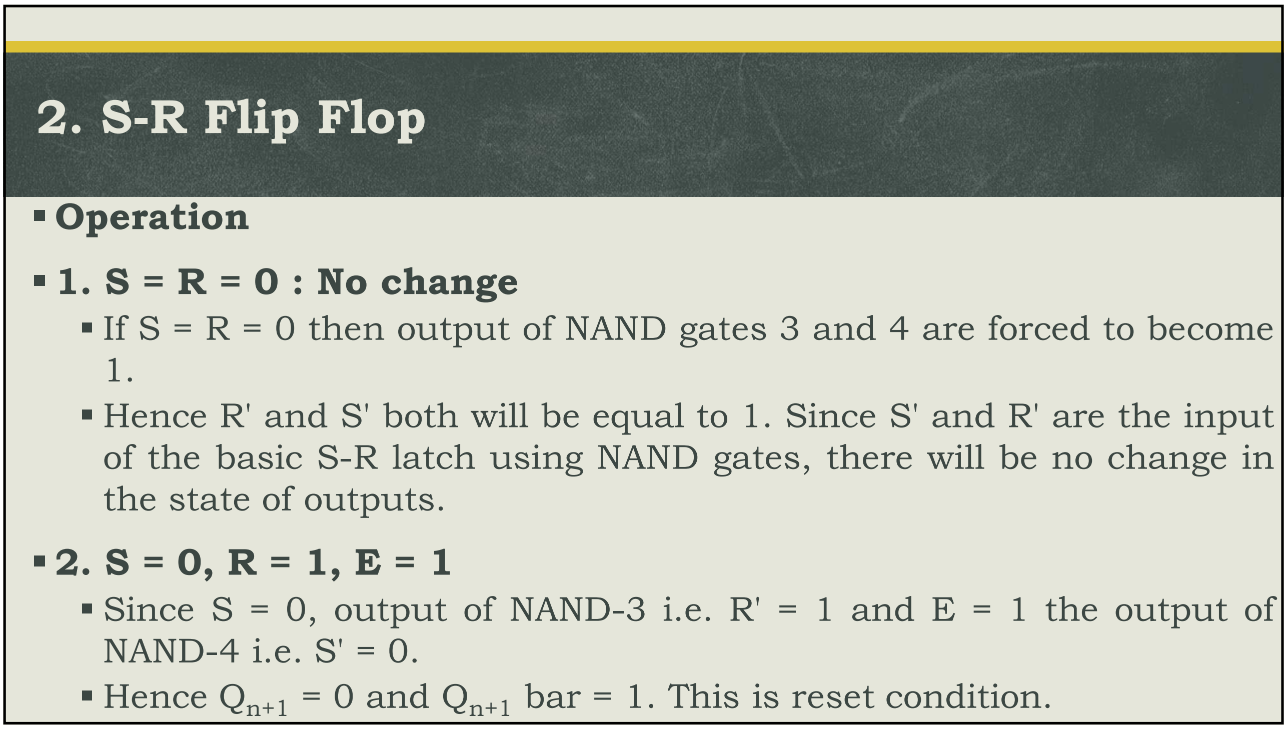


| Inputs | | | Outputs | | Comments |
|---|---|---|---|---|---|
| E | S | R | $Q_{n+1}$ | $\overline{Q}_{n+1}$ | |
| 1<br>1<br>1<br>1 | 0<br>0<br>1<br>1 | 0<br>1<br>0<br>1 | $Q_n$<br>0<br>1<br>x | $\overline{Q}_n$<br>1<br>0<br>x | No change<br>Rset<br>Set<br>Indeterminate |

## 2. S-R Flip Flop

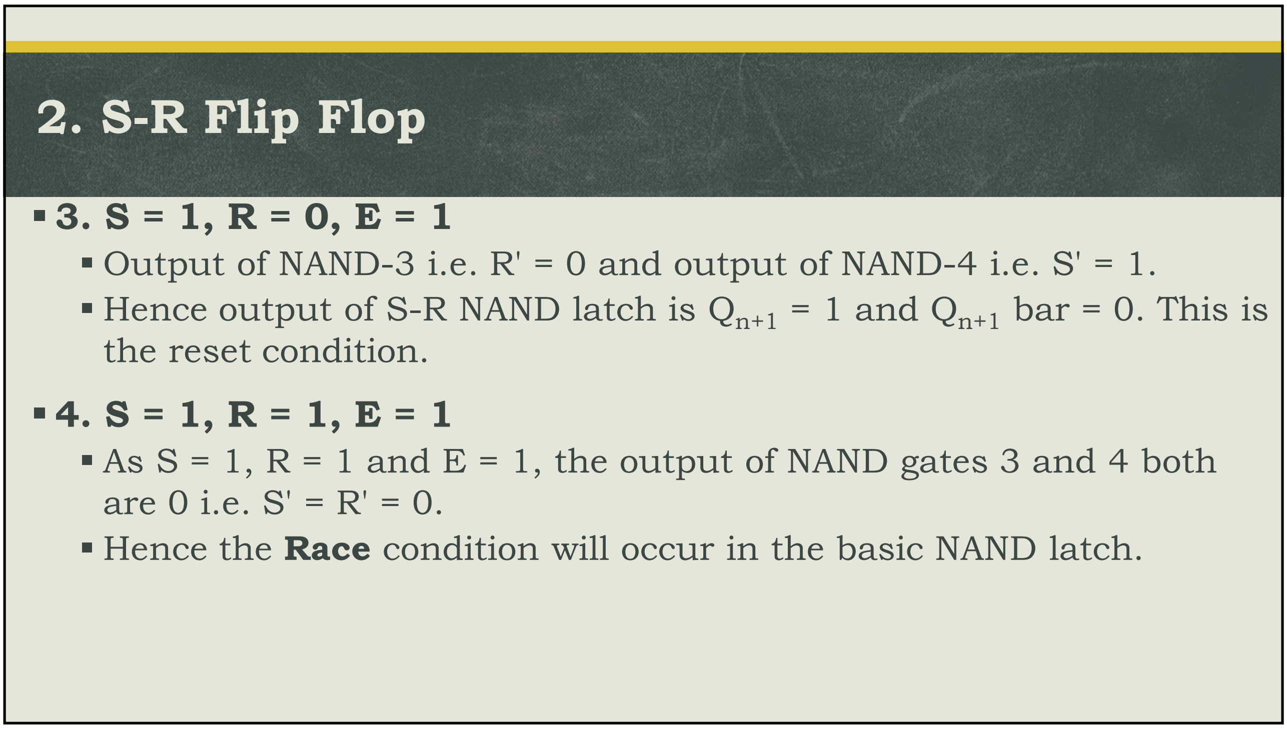
- **Circuit Diagram**





## 2. S-R Flip Flop

- **Operation**
- **1. S = R = 0 : No change**
  - If S = R = 0 then output of NAND gates 3 and 4 are forced to become 1.
  - Hence R' and S' both will be equal to 1. Since S' and R' are the input of the basic S-R latch using NAND gates, there will be no change in the state of outputs.
- **2. S = 0, R = 1, E = 1**
  - Since S = 0, output of NAND-3 i.e. R' = 1 and E = 1 the output of NAND-4 i.e. S' = 0.
  - Hence $Q_{n+1}$ = 0 and $Q_{n+1}$ bar = 1. This is reset condition.



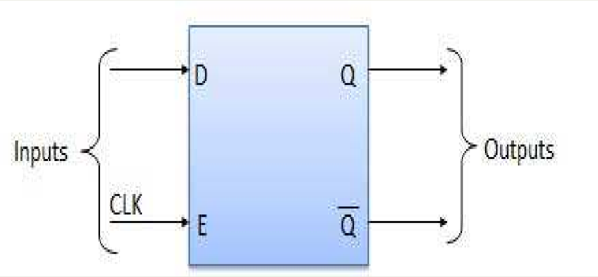
## 2. S-R Flip Flop

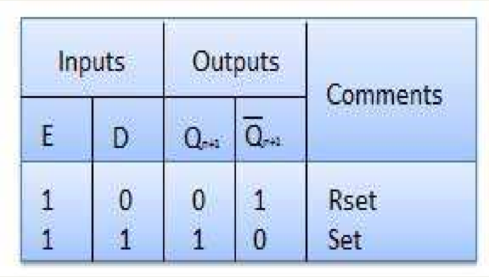
- **3. S = 1, R = 0, E = 1**
  - Output of NAND-3 i.e. R' = 0 and output of NAND-4 i.e. S' = 1.
  - Hence output of S-R NAND latch is $Q_{n+1}$ = 1 and $Q_{n+1}$ bar = 0. This is the reset condition.
- **4. S = 1, R = 1, E = 1**
  - As S = 1, R = 1 and E = 1, the output of NAND gates 3 and 4 both are 0 i.e. S' = R' = 0.
  - Hence the **Race** condition will occur in the basic NAND latch.

## 3. D Flip Flop

- Data Flip Flop or D Flip Flop is the simple gated S-R latch with a NAND inverter connected between S and R inputs.
- Hence S = R = 0 or S = R = 1, these input condition will never appear. This problem is avoid by SR = 00 and SR = 1 conditions.

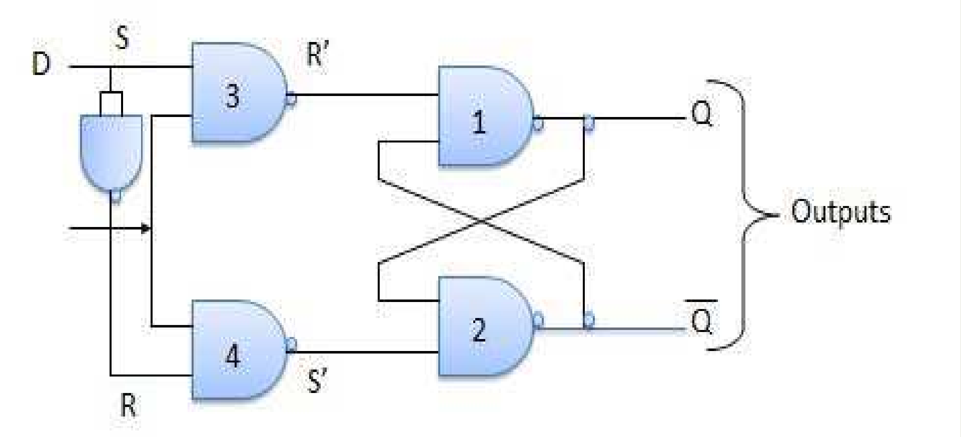
## 3. D Flip Flop





| Inputs | | Outputs | | Comments |
|---|---|---|---|---|
| E | D | $Q_{n+1}$ | $\overline{Q}_{n+1}$ | |
| 1 | 0 | 0 | 1 | Rset |
| 1 | 1 | 1 | 0 | Set |

## 3. D Flip Flop



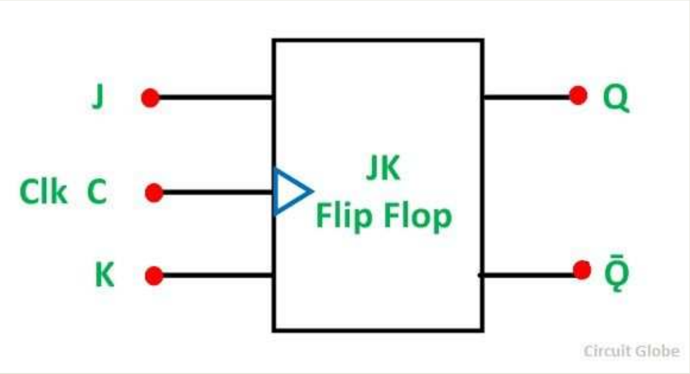
## 3. D Flip Flop

- Operation
- **1. E = 0:** Latch is disabled. Hence no change in output.
- **2 E = 1 and D = 0:** If E = 1 and D = 0 then S = 0 and R = 1. Hence irrespective of the present state, the next state is Qn+1 = 0 and Qn+1 bar = 1. This is the reset condition.
- **3 E = 1 and D = 1:** If E = 1 and D = 1, then S = 1 and R = 0. This will set the latch and Qn+1 = 1 and Qn+1 bar = 0 irrespective of the present state.

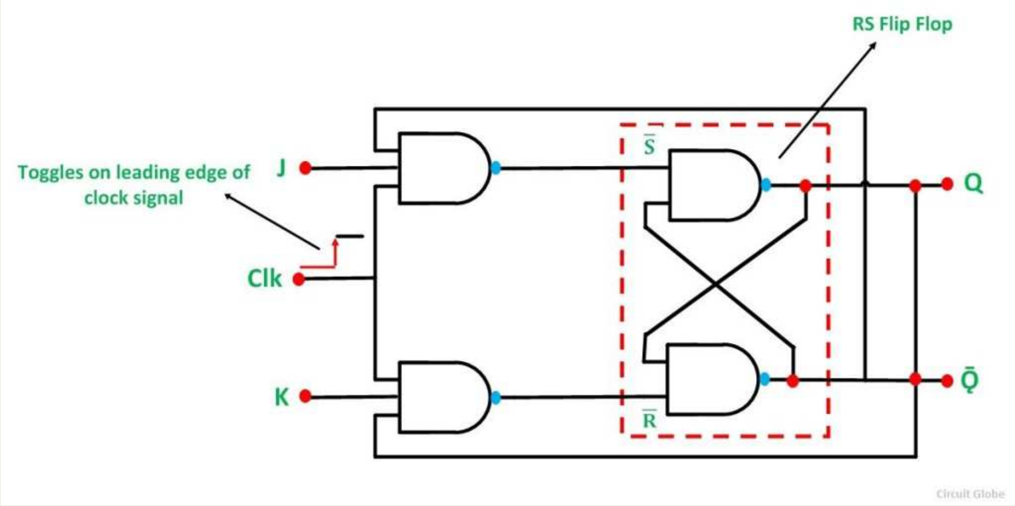
## 4. J K Flip Flop

- The **JK Flip Flop** is the most widely used flip flop. It is considered to be a universal flip-flop circuit.
- The sequential operation of the JK Flip Flop is same as for the SR flip-flop with the same **SET** and **RESET** input.
- The difference is that the JK Flip Flop does not have the invalid input states of the SR Latch (when S and R are both 1).
- The JK Flip Flop name has been kept on the inventor name of the circuit known as **Jack Kilby.** The basic **symbol** of the JK Flip Flop is shown below.

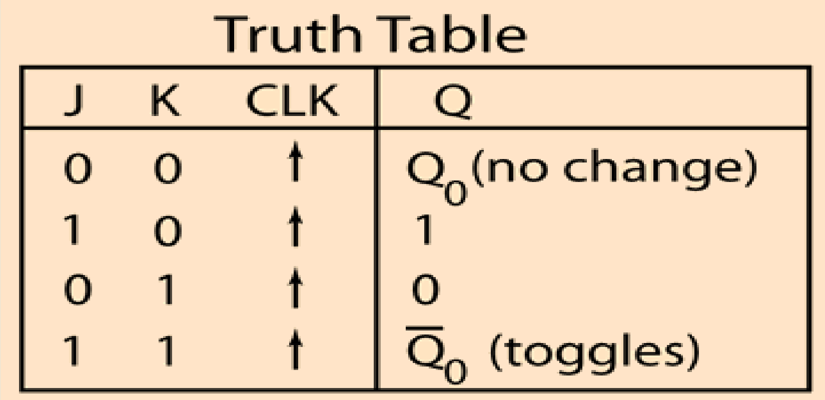
## 4. J K Flip Flop



# 4. J K Flip Flop



# 4. J K Flip Flop



Truth Table

| J | K | CLK | Q |
|---|---|---|---|
| 0 | 0 | ↑ | $Q_0$(no change) |
| 1 | 0 | ↑ | 1 |
| 0 | 1 | ↑ | 0 |
| 1 | 1 | ↑ | $\overline{Q}_0$ (toggles) |

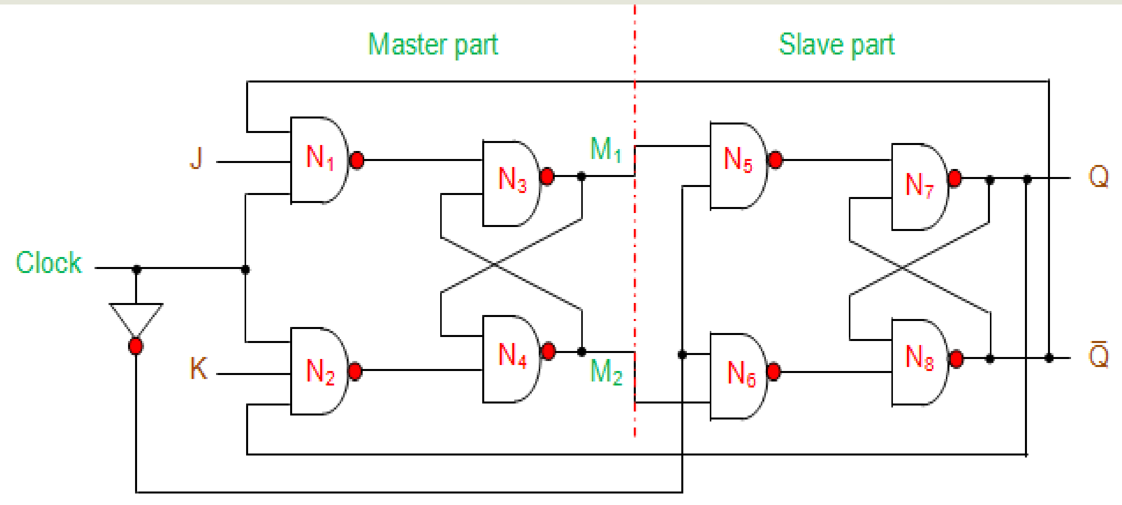
## 5. Master Slave J K Flip Flop

- **Race Around Condition In JK Flip-flop**
- For J-K flip-flop, if J=K=1, and if clk=1 for a long period of time, then Q output will toggle as long as CLK is high, which makes the output of the flip-flop unstable or uncertain. This problem is called race around condition in J-K flip-flop.
- This problem (Race Around Condition) can be avoided by ensuring that the clock input is at logic "1" only for a very short time. This introduced the concept of **Master Slave JK** flip flop.

## 5. Master Slave J K Flip Flop

- The Master-Slave Flip-Flop is basically a combination of two JK flip-flops connected together in a series configuration.
- Out of these, one acts as the **"master"** and the other as a **"slave"**. The output from the master flip flop is connected to the two inputs of the slave flip flop whose output is fed back to inputs of the master flip flop.
- In addition to these two flip-flops, the circuit also includes an **inverter**. The inverter is connected to clock pulse in such a way that the inverted clock pulse is given to the slave flip-flop.
- In other words if CP=0 for a master flip-flop, then CP=1 for a slave flip-flop and if CP=1 for master flip flop then it becomes 0 for slave flip flop.

# 5. Master Slave J K Flip Flop



# 5. Master Slave J K Flip Flop

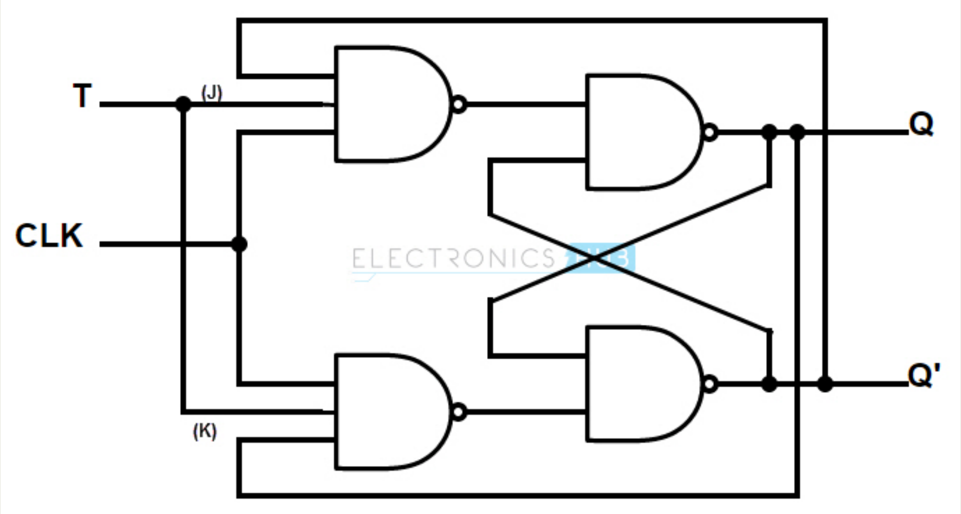
## Truth Table

| Trigger | Inputs | | Output | | | | | | Inference |
|---|---|---|---|---|---|---|---|---|---|
| | | | Present State | | Intermediate | | Next State | | |
| CLK | J | K | Q | $\bar{Q}$ | $M_1$ | $M_2$ | Q | $\bar{Q}$ | |
| ↑ | 0 | 0 | 0 | 1 | 0 | 1 | Latched | | No Change |
| ↓ | | | 0 | 1 | Latched | | 0 | 1 | |
| ↑ | | | 1 | 0 | 1 | 0 | Latched | | |
| ↓ | | | 1 | 0 | Latched | | 1 | 0 | |
| ↑ | 0 | 1 | 0 | 1 | 0 | 1 | Latched | | Reset |
| ↓ | | | 0 | 1 | Latched | | 0 | 1 | |
| ↑ | | | 1 | 0 | 0 | 1 | Latched | | |
| ↓ | | | 1 | 0 | Latched | | 0 | 1 | |
| ↑ | 1 | 0 | 0 | 1 | 1 | 0 | Latched | | Set |
| ↓ | | | 0 | 1 | Latched | | 1 | 0 | |
| ↑ | | | 1 | 0 | 1 | 0 | Latched | | |
| ↓ | | | 1 | 0 | Latched | | 1 | 0 | |
| ↑ | 1 | 1 | 0 | 1 | 1 | 0 | Latched | | Toggles |
| ↓ | | | 0 | 1 | Latched | | 1 | 0 | |
| ↑ | | | 1 | 0 | 0 | 1 | Latched | | |
| ↓ | | | 1 | 0 | Latched | | 0 | 1 | |

## 6. T Flip Flop

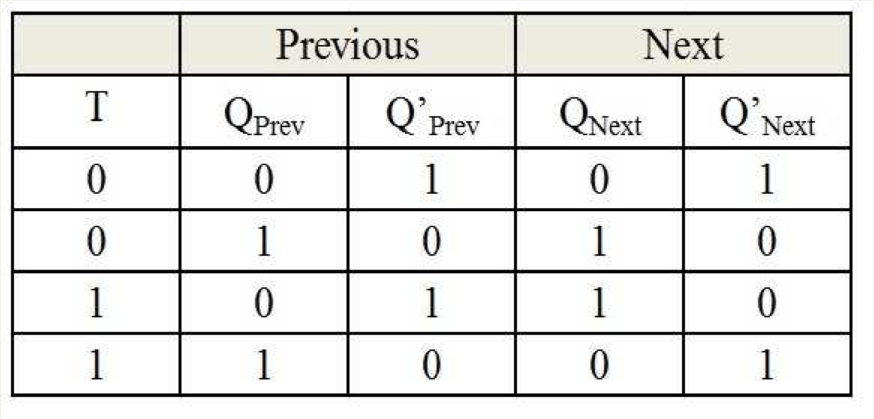
- The name T flip-flop is termed from the nature of toggling operation. The major applications of T flip-flop are counters and control circuits.
- **T flip flop is modified form of JK flip-flop** making it to operate in toggling region.
- Whenever the **clock signal is LOW, the input is never going to affect the output state**. The clock has to be high for the inputs to get active.

## 6. T Flip Flop



## 6. T Flip Flop

- **Truth Table**

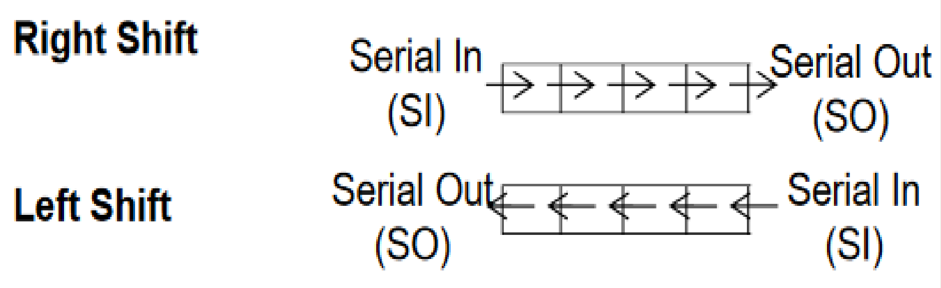


|  | Previous |  | Next |  |
|---|---|---|---|---|
| T | $Q_{Prev}$ | $Q'_{Prev}$ | $Q_{Next}$ | $Q'_{Next}$ |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

## 7. Shift Registers

- One flip-flop can store one-bit of information. In order to store multiple bits of information, we require multiple flip-flops.
- The group of flip-flops, which are used to hold (store) the binary data is known as **register**.
- If the register is capable of shifting bits either towards right hand side or towards left hand side is known as **shift register**.
- An ‘N’ bit shift register contains ‘N’ flip-flops. Following are the four types of shift registers based on applying inputs and accessing of outputs.

## 7. Shift Registers

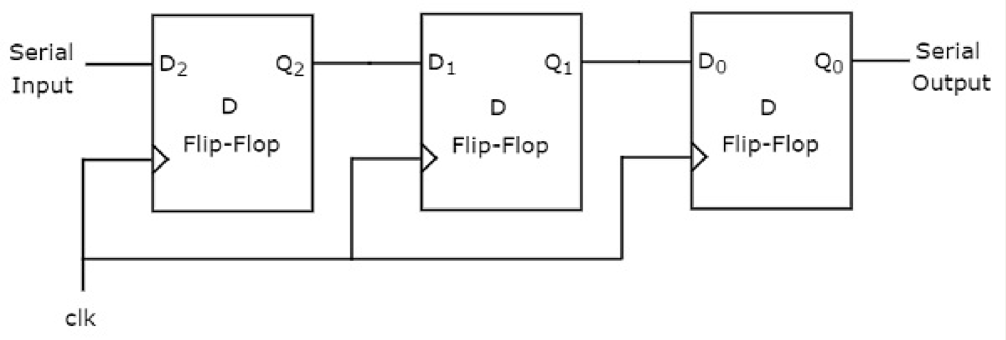


## 7. Shift Registers

- Serial In - Serial Out shift register
- Serial In - Parallel Out shift register
- Parallel In - Serial Out shift register
- Parallel In - Parallel Out shift register
- Bidirectional Shift Register

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

- The shift register, which allows serial input and produces serial output is known as Serial In – Serial Out **(SISO)** shift register. The **block diagram** of 3-bit SISO shift register is shown in the following figure.



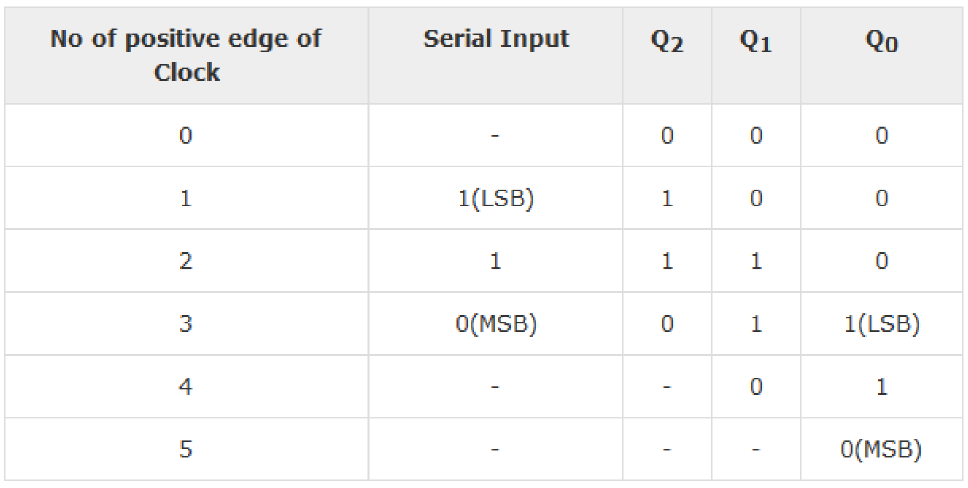
## ❑ Serial In - Serial Out (SISO) Shift Register

- This block diagram consists of three D flip-flops, which are **cascaded**. That means, output of one D flip-flop is connected as the input of next D flip-flop. All these flip-flops are synchronous with each other since, the same clock signal is applied to each one.
- In this shift register, we can send the bits serially from the input of left most D flip-flop. Hence, this input is also called as **serial input**.
- For every positive edge triggering of clock signal, the data shifts from one stage to the next. So, we can receive the bits serially from the output of right most D flip-flop. Hence, this output is also called as **serial output**.

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

- **Example**
- Let us see the working of 3-bit SISO shift register by sending the binary information "011" from LSB to MSB serially at the input.
- Assume, initial status of the D flip-flops from leftmost to rightmost is $Q_2Q_1Q_0$=000
- We can understand the working of 3-bit SISO shift register from the following table.

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



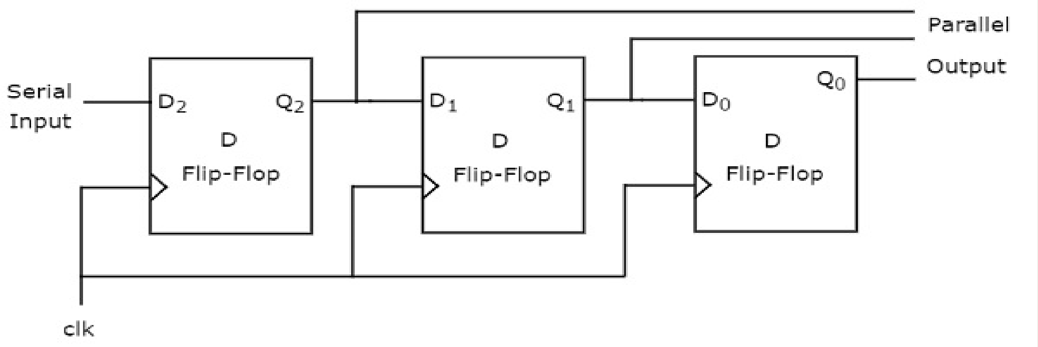
| No of positive edge of Clock | Serial Input | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|
| 0 | - | 0 | 0 | 0 |
| 1 | 1(LSB) | 1 | 0 | 0 |
| 2 | 1 | 1 | 1 | 0 |
| 3 | 0(MSB) | 0 | 1 | 1(LSB) |
| 4 | - | - | 0 | 1 |
| 5 | - | - | - | 0(MSB) |

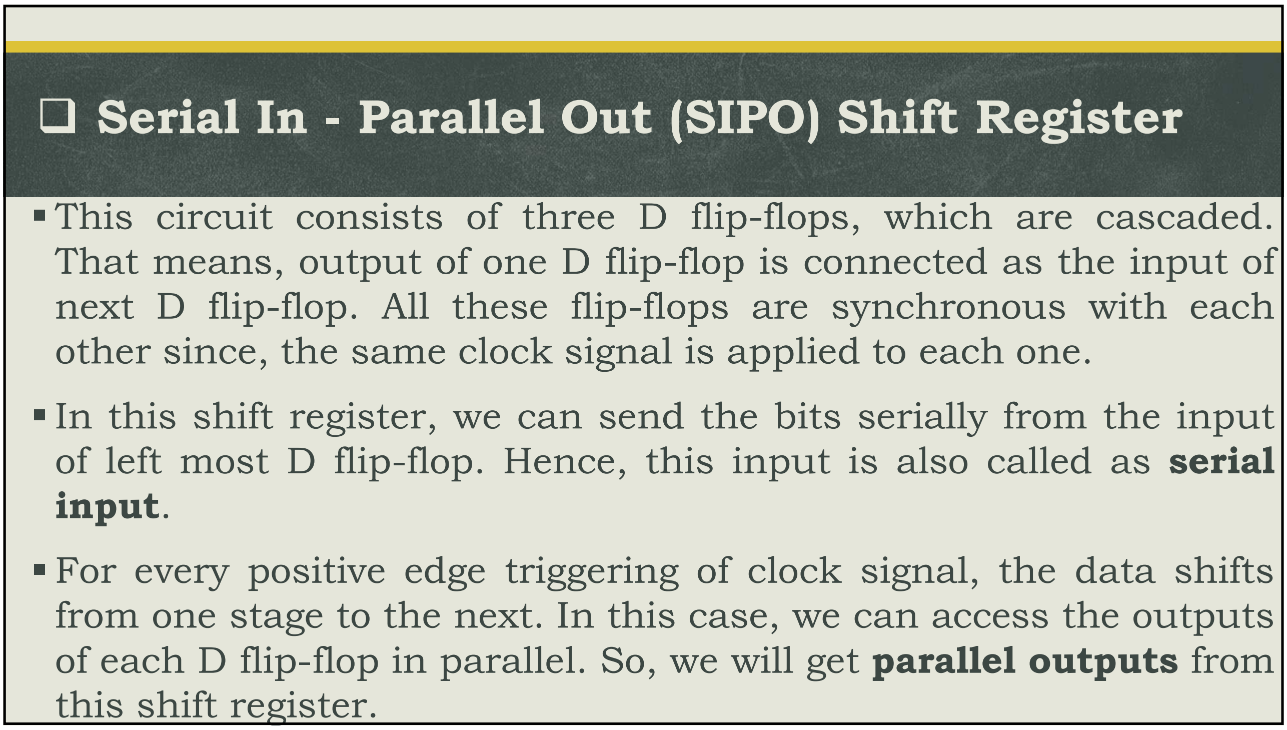
## ❑ Serial In - Serial Out (SISO) Shift Register

- The initial status of the D flip-flops in the absence of clock signal is $Q_2Q_1Q_0$=000.
- Here, the serial output is coming from $Q_0$So, the LSB (1) is received at 3rd positive edge of clock and the MSB (0) is received at 5th positive edge of clock.
- Therefore, the 3-bit SISO shift register requires five clock pulses in order to produce the valid output. Similarly, the N-bit SISO shift register requires 2N-1 clock pulses in order to shift 'N' bit information.

## ❑ Serial In - Parallel Out (SIPO) Shift Register

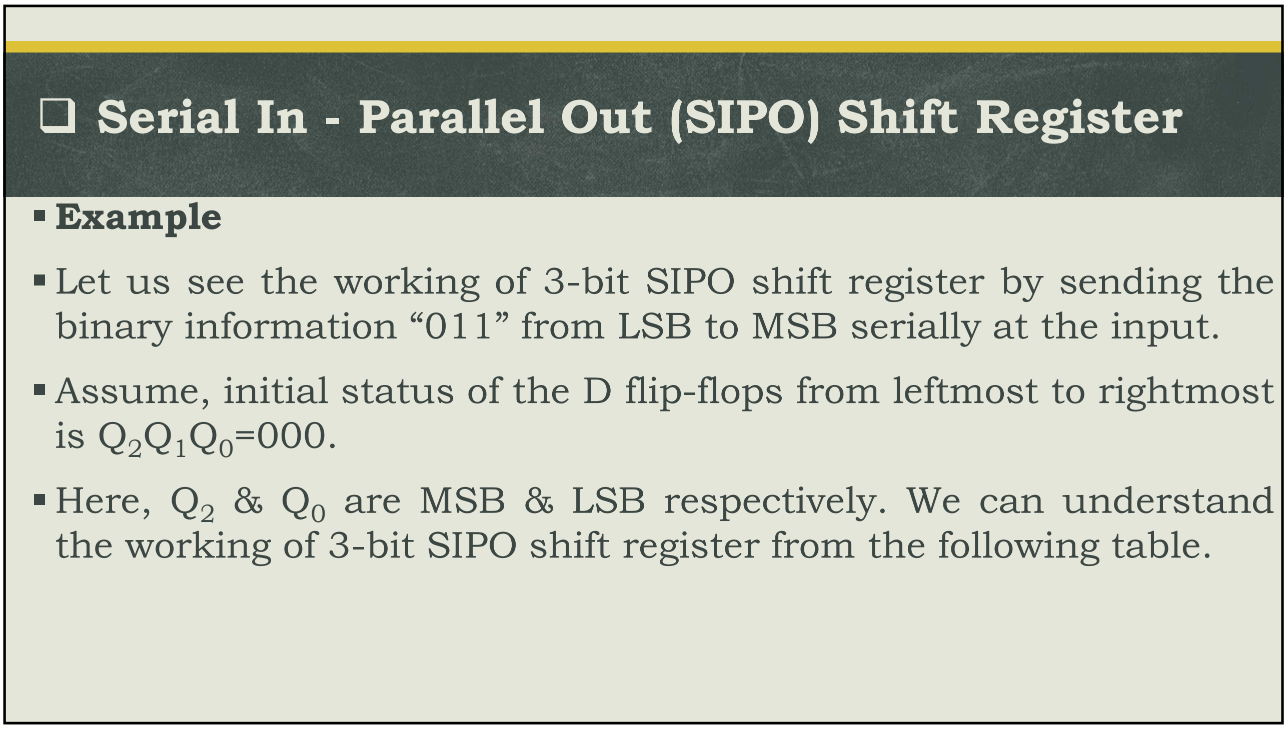
- The shift register, which allows serial input and produces parallel output is known as Serial In – Parallel Out **(SIPO)** shift register. The **block diagram** of 3-bit SIPO shift register is shown in the following figure.





## ❑ Serial In - Parallel Out (SIPO) Shift Register

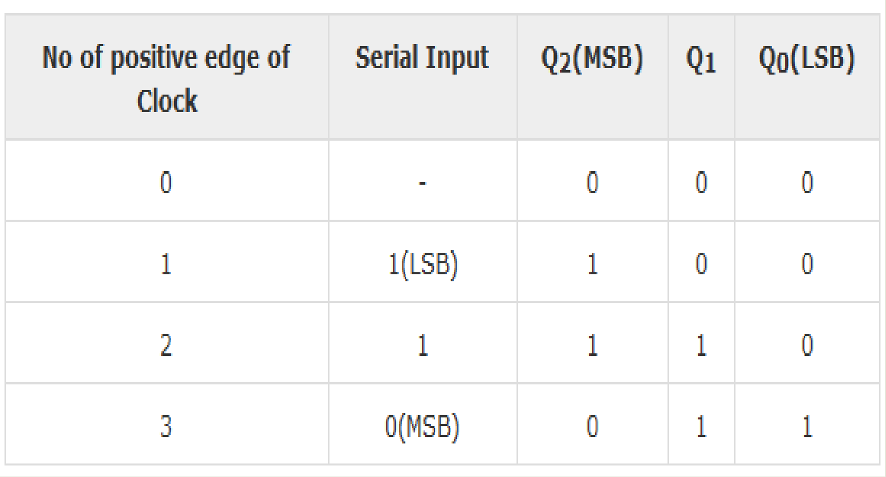
- This circuit consists of three D flip-flops, which are cascaded. That means, output of one D flip-flop is connected as the input of next D flip-flop. All these flip-flops are synchronous with each other since, the same clock signal is applied to each one.
- In this shift register, we can send the bits serially from the input of left most D flip-flop. Hence, this input is also called as **serial input**.
- For every positive edge triggering of clock signal, the data shifts from one stage to the next. In this case, we can access the outputs of each D flip-flop in parallel. So, we will get **parallel outputs** from this shift register.



## ❑ Serial In - Parallel Out (SIPO) Shift Register

- **Example**
- Let us see the working of 3-bit SIPO shift register by sending the binary information “011” from LSB to MSB serially at the input.
- Assume, initial status of the D flip-flops from leftmost to rightmost is $Q_2Q_1Q_0$=000.
- Here, $Q_2$ & $Q_0$ are MSB & LSB respectively. We can understand the working of 3-bit SIPO shift register from the following table.

## ❑ Serial In - Parallel Out (SIPO) Shift Register



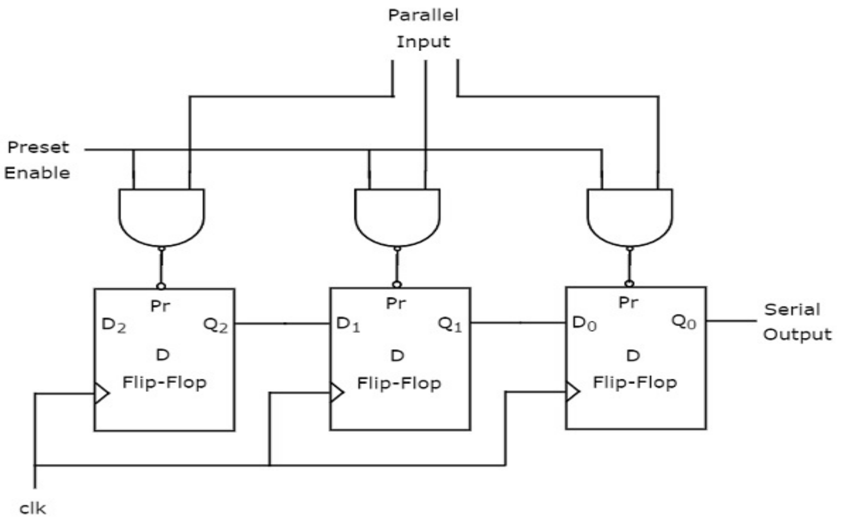
| No of positive edge of Clock | Serial Input | $Q_2$(MSB) | $Q_1$ | $Q_0$(LSB) |
|---|---|---|---|---|
| 0 | - | 0 | 0 | 0 |
| 1 | 1(LSB) | 1 | 0 | 0 |
| 2 | 1 | 1 | 1 | 0 |
| 3 | 0(MSB) | 0 | 1 | 1 |

## ❑ Serial In - Parallel Out (SIPO) Shift Register

- The initial status of the D flip-flops in the absence of clock signal is $Q_2Q_1Q_0$=000.
- The binary information “011” is obtained in parallel at the outputs of D flip-flops for third positive edge of clock.
- So, the 3-bit SIPO shift register requires three clock pulses in order to produce the valid output. Similarly, the N-bit SIPO shift register requires N clock pulses in order to shift ‘N’ bit information.

## ❑ Parallel In - Serial Out (PISO) Shift Register

- The shift register, which allows parallel input and produces serial output is known as Parallel In – Serial Out **(PISO)** shift register. The **block diagram** of 3-bit PISO shift register is shown in the following figure.



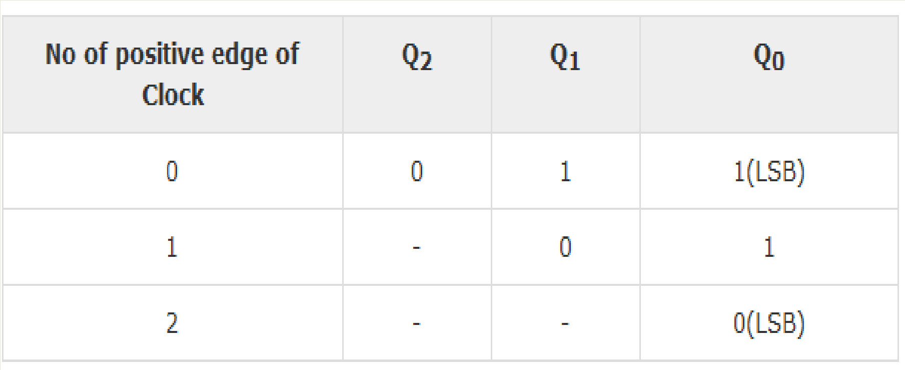
## ❑ Parallel In - Serial Out (PISO) Shift Register

- This circuit consists of three D flip-flops, which are cascaded. That means, output of one D flip-flop is connected as the input of next D flip-flop.
- All these flip-flops are synchronous with each other since, the same clock signal is applied to each one.
- In this shift register, we can apply the **parallel inputs** to each D flip-flop by making Preset Enable to 1.
- For every positive edge triggering of clock signal, the data shifts from one stage to the next. So, we will get the **serial output** from the right most D flip-flop.

## ❑ Parallel In - Serial Out (PISO) Shift Register

- **Example**
- Let us see the working of 3-bit PISO shift register by applying the binary information “011” in parallel through preset inputs.
- Since the preset inputs are applied before positive edge of Clock, the initial status of the D flip-flops from leftmost to rightmost will be $Q_2Q_1Q_0$=011.
- We can understand the working of 3-bit PISO shift register from the following table.

## ❑ Parallel In - Serial Out (PISO) Shift Register



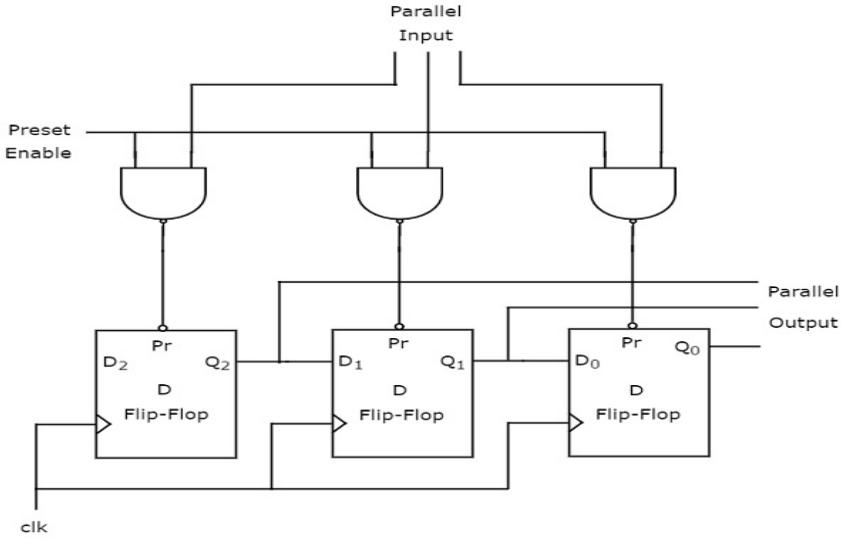
| No of positive edge of Clock | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|
| 0 | 0 | 1 | 1(LSB) |
| 1 | - | 0 | 1 |
| 2 | - | - | 0(LSB) |

## ❑ Parallel In - Serial Out (PISO) Shift Register

- Here, the serial output is coming from Q0. So, the LSB (1) is received before applying positive edge of clock and the MSB (0) is received at 2nd positive edge of clock.
- Therefore, the 3-bit PISO shift register requires two clock pulses in order to produce the valid output. Similarly, the N-bit PISO shift register requires N-1 clock pulses in order to shift 'N' bit information.

## ❑ Parallel In - Parallel Out (PIPO) Shift Register

- The shift register, which allows parallel input and produces parallel output is known as Parallel In − Parallel Out **(PIPO)** shift register. The **block diagram** of 3-bit PIPO shift register is shown in the following figure.



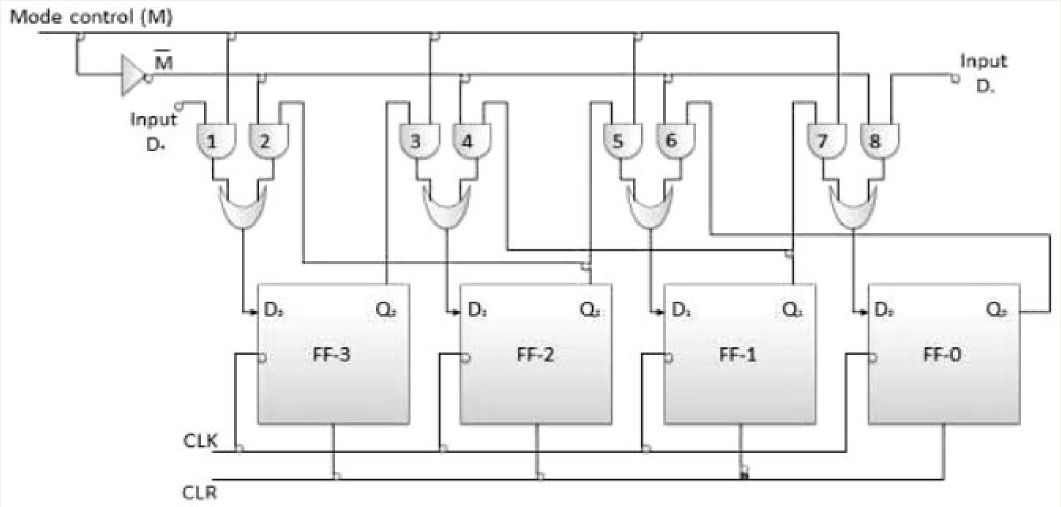
## ❑ Parallel In - Parallel Out (PIPO) Shift Register

- This circuit consists of three D flip-flops, which are cascaded. That means, output of one D flip-flop is connected as the input of next D flip-flop. All these flip-flops are synchronous with each other since, the same clock signal is applied to each one.
- In this shift register, we can apply the **parallel inputs** to each D flip-flop by making Preset Enable to 1. We can apply the parallel inputs through preset or clear.
- These two are asynchronous inputs. That means, the flip-flops produce the corresponding outputs, based on the values of asynchronous inputs. In this case, the effect of outputs is independent of clock transition. So, we will get the **parallel outputs** from each D flip-flop.

## ❑ Parallel In - Parallel Out (PIPO) Shift Register

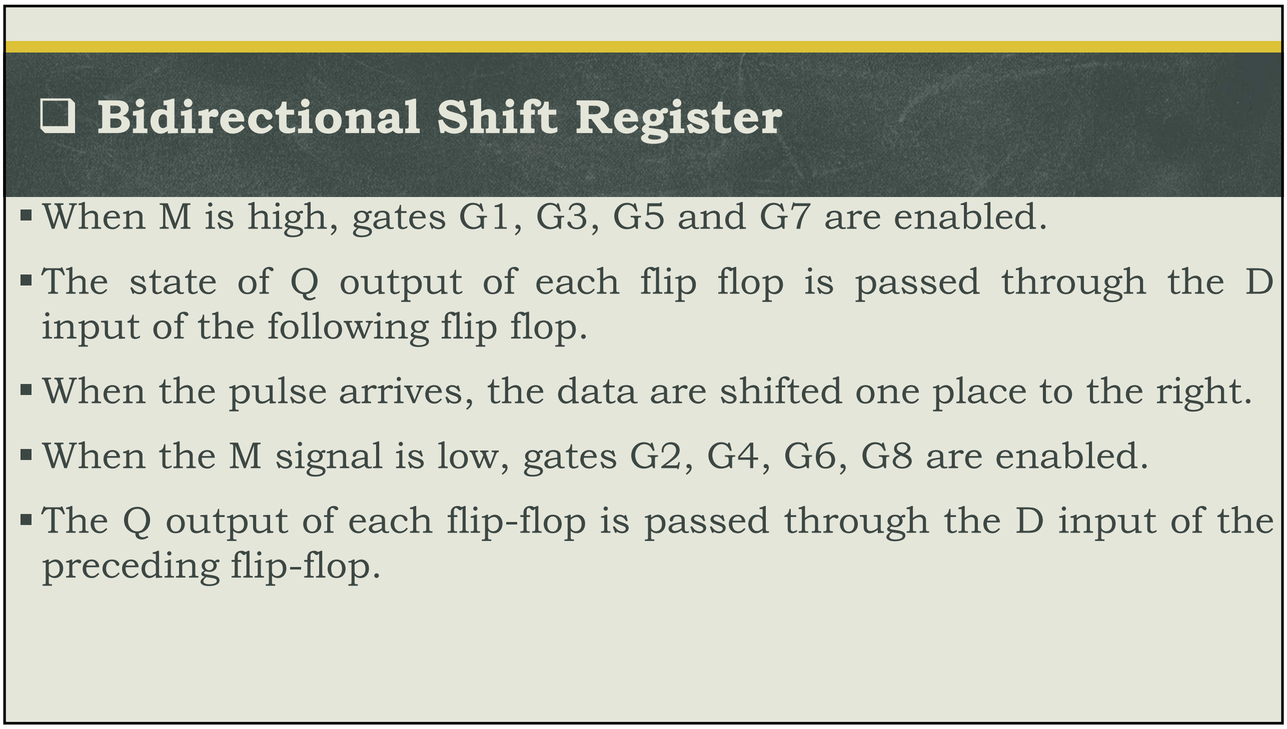
- **Example**
- Let us see the working of 3-bit PIPO shift register by applying the binary information “011” in parallel through preset inputs.
- Since the preset inputs are applied before positive edge of Clock, the initial status of the D flip-flops from leftmost to rightmost will be $Q_2Q_1Q_0$=011.
- So, the binary information “011” is obtained in parallel at the outputs of D flip-flops before applying positive edge of clock.
- Therefore, the 3-bit PIPO shift register requires one clock pulses in order to produce the valid output.

## ❑ Bidirectional Shift Register



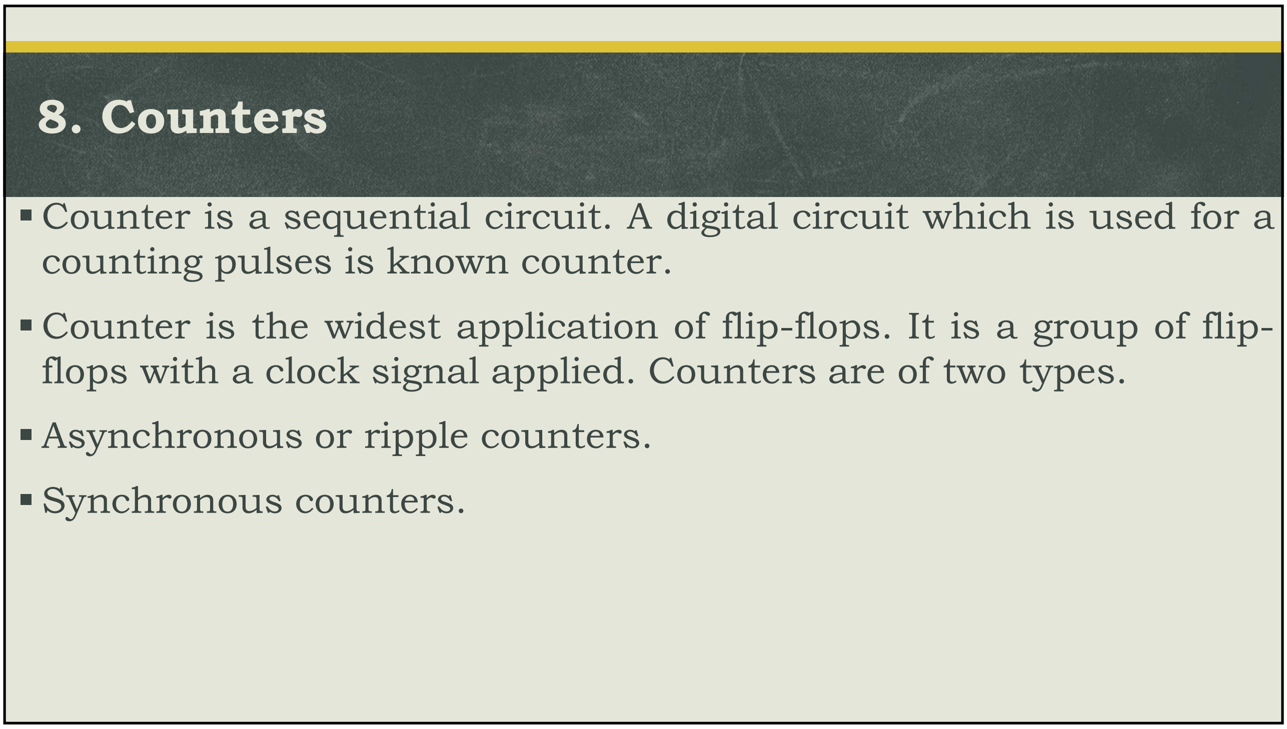
## ❑ Bidirectional Shift Register

- Bidirectional shift register allows shifting of data either to left or to the right side.
- It can be implemented using logic gates circuitry that enables the transfer of data from one stage to the next stage to the right or to the left, depend on the level of control line.
- The M is the control input signal which allows data shifting either towards right or towards left.
- A high on this line enables the shifting of data towards right and low enables it towards left.



## ❑ Bidirectional Shift Register

- When M is high, gates G1, G3, G5 and G7 are enabled.
- The state of Q output of each flip flop is passed through the D input of the following flip flop.
- When the pulse arrives, the data are shifted one place to the right.
- When the M signal is low, gates G2, G4, G6, G8 are enabled.
- The Q output of each flip-flop is passed through the D input of the preceding flip-flop.

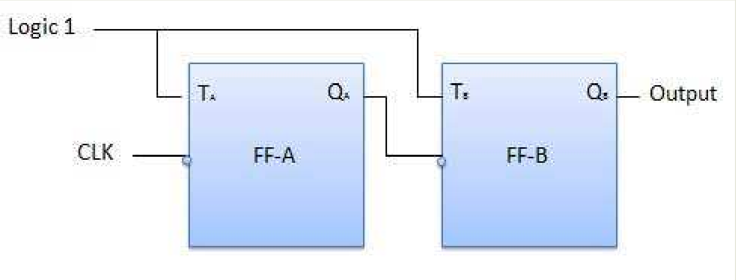


## 8. Counters

- Counter is a sequential circuit. A digital circuit which is used for a counting pulses is known counter.
- Counter is the widest application of flip-flops. It is a group of flip-flops with a clock signal applied. Counters are of two types.
- Asynchronous or ripple counters.
- Synchronous counters.

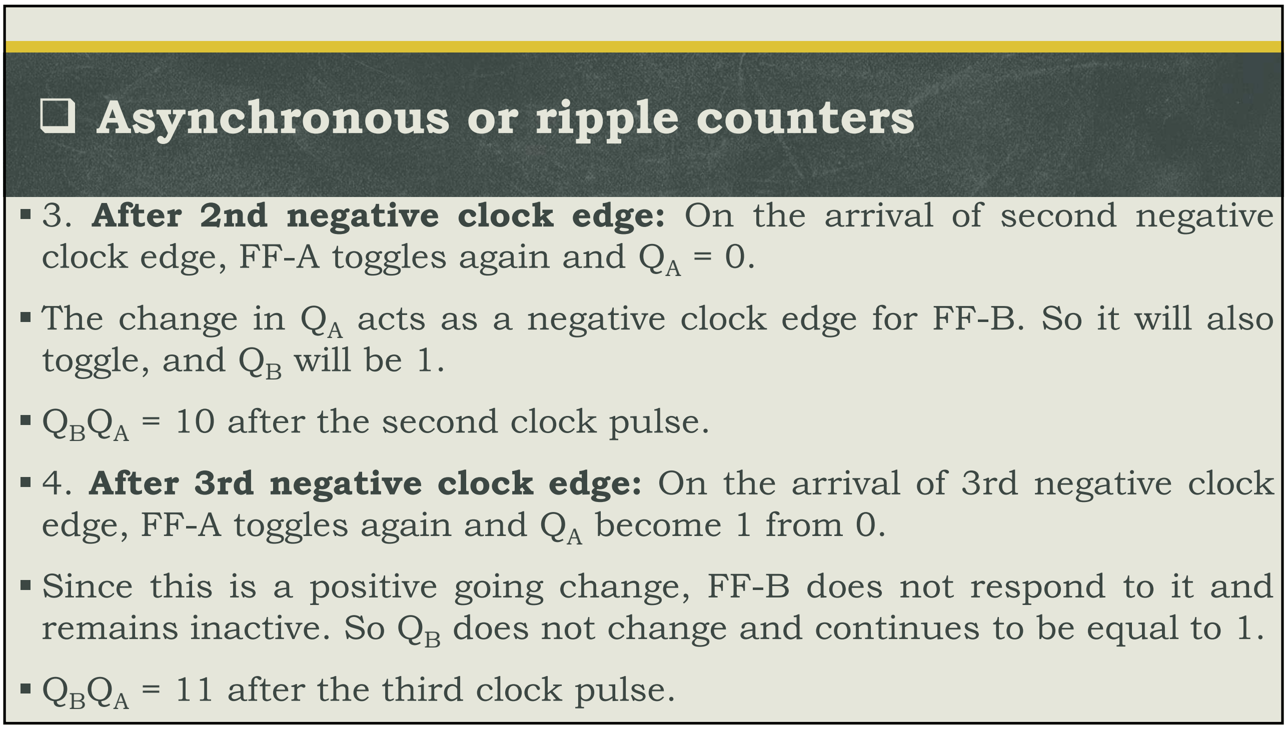
## ❑ Asynchronous or ripple counters

- The logic diagram of a 2-bit ripple up counter is shown in figure. The toggle (T) flip-flop are being used. But we can use the JK flip-flop also with J and K connected permanently to logic 1.
- 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.



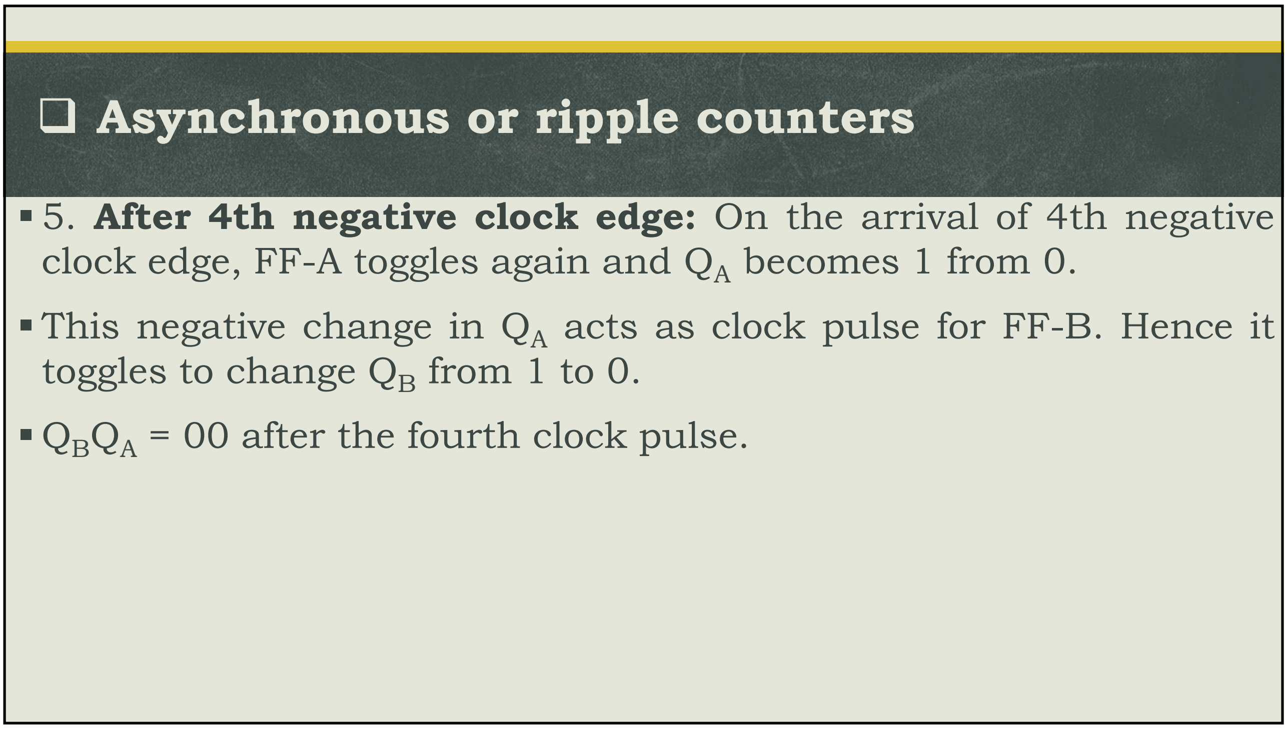
## ❑ Asynchronous or ripple counters

- **Operation**
- 1. **Initially let both the FFs be in the reset state:** $Q_BQ_A = 00$ initially
- 2. **After 1st negative clock edge:** 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.
- $Q_BQ_A = 01$ after the first clock pulse.



## ❑ Asynchronous or ripple counters

- 3. **After 2nd negative clock edge:** 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.
- $Q_BQ_A = 10$ after the second clock pulse.
- 4. **After 3rd negative clock edge:** On the arrival of 3rd 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 the third clock pulse.

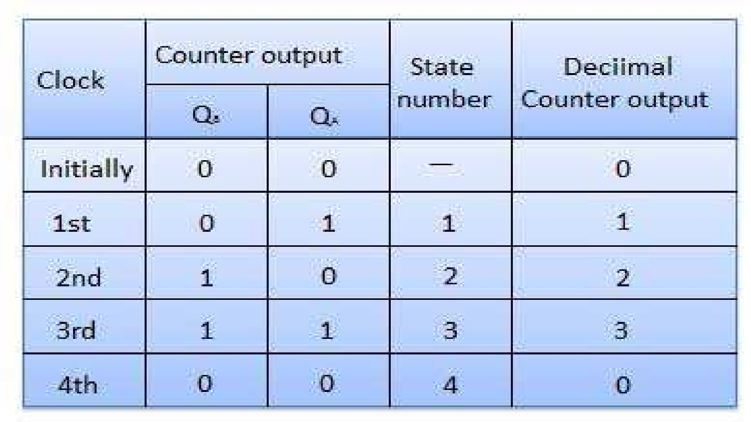


## ❑ Asynchronous or ripple counters

- 5. **After 4th negative clock edge:** On the arrival of 4th negative clock edge, FF-A toggles again and $Q_A$ becomes 1 from 0.
- 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.

## ❑ Asynchronous or ripple counters

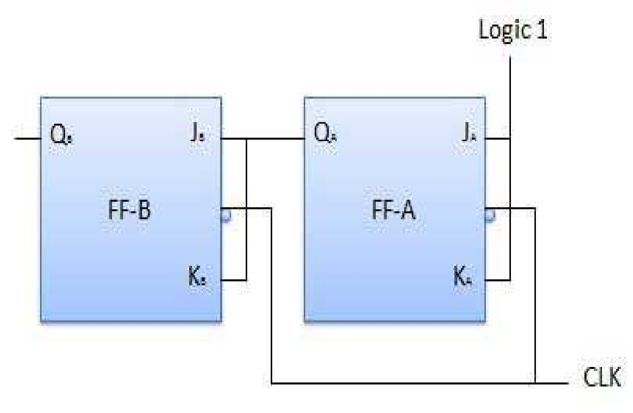
**Truth Table**



| Clock | Counter output | | State number | Deciimal Counter output |
|---|---|---|---|---|
| | $Q_B$ | $Q_A$ | | |
| Initially | 0 | 0 | — | 0 |
| 1st | 0 | 1 | 1 | 1 |
| 2nd | 1 | 0 | 2 | 2 |
| 3rd | 1 | 1 | 3 | 3 |
| 4th | 0 | 0 | 4 | 0 |

## ❑ Synchronous counters

- If the "clock" pulses are applied to all the flip-flops in a counter simultaneously, then such a counter is called as synchronous counter.
- **2-bit Synchronous up counter**
- The $J_A$ and $K_A$ inputs of FF-A are tied to logic 1. So FF-A will work as a toggle flip-flop. The $J_B$ and $K_B$ inputs are connected to $Q_A$.



## ❑ Synchronous counters

- **Operation**
- 1. **Initially let both the FFs be in the reset state :** $Q_B Q_A$ = 00 initially.
- 2. **After 1st negative clock edge:** As soon as the first negative clock edge is applied, FF-A will toggle and $Q_A$ will change from 0 to 1.
- But at the instant of application of negative clock edge, $Q_A$ , $J_B$ = $K_B$ = 0. Hence FF-B will not change its state. So $Q_B$ will remain 0.
- $Q_B Q_A$ = 01 after the first clock pulse.

## ❑ Synchronous counters

- 3. **After 2nd negative clock edge:** On the arrival of second negative clock edge, FF-A toggles again and $Q_A$ changes from 1 to 0.
- But at this instant $Q_A$ was 1. So $J_B$ = $K_B$= 1 and FF-B will toggle. Hence $Q_B$ changes from 0 to 1.
- $Q_B Q_A$ = 10 after the second clock pulse.
- 4. **After 3rd negative clock edge:** On application of the third falling clock edge, FF-A will toggle from 0 to 1 but there is no change of state for FF-B.
- $Q_B Q_A$ = 11 after the third clock pulse.

## ❑ Synchronous counters

- 5. **After 4th negative clock edge:** On application of the next clock pulse, $Q_A$ will change from 1 to 0 as $Q_B$ will also change from 1 to 0.
- $Q_BQ_A$ = 00 after the fourth clock pulse.

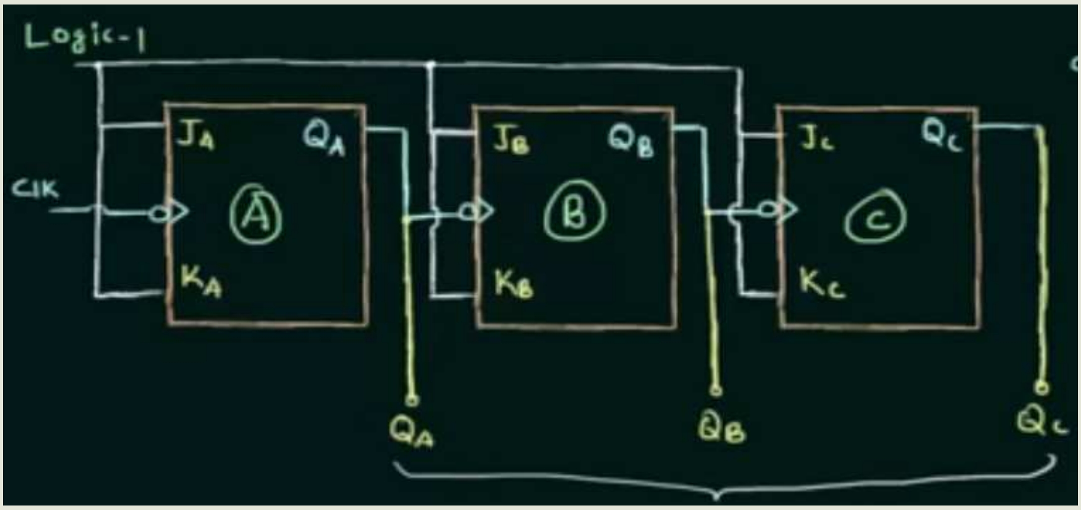
## ❑ Classification of counters

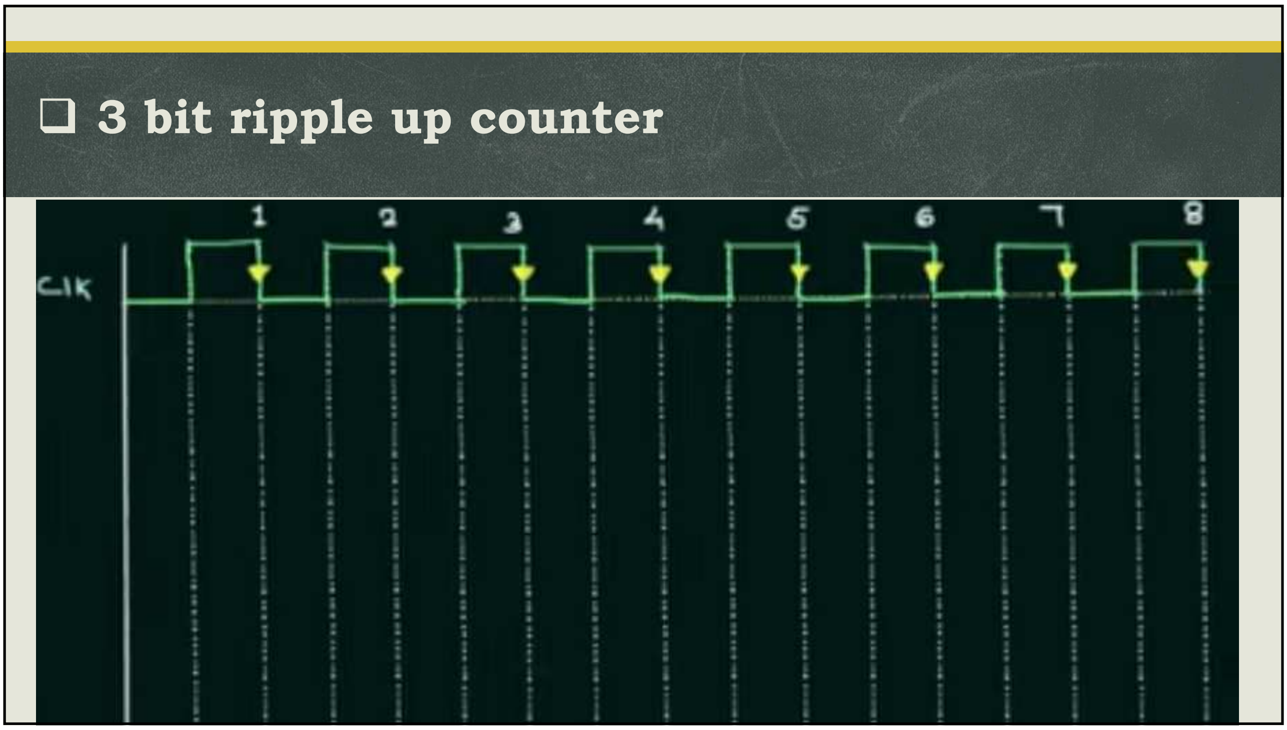
- Depending on the way in which the counting progresses, the synchronous or asynchronous counters are classified as follows −
- Up counters
- Down counters
- Up/Down counters

## ❑ Classification of counters

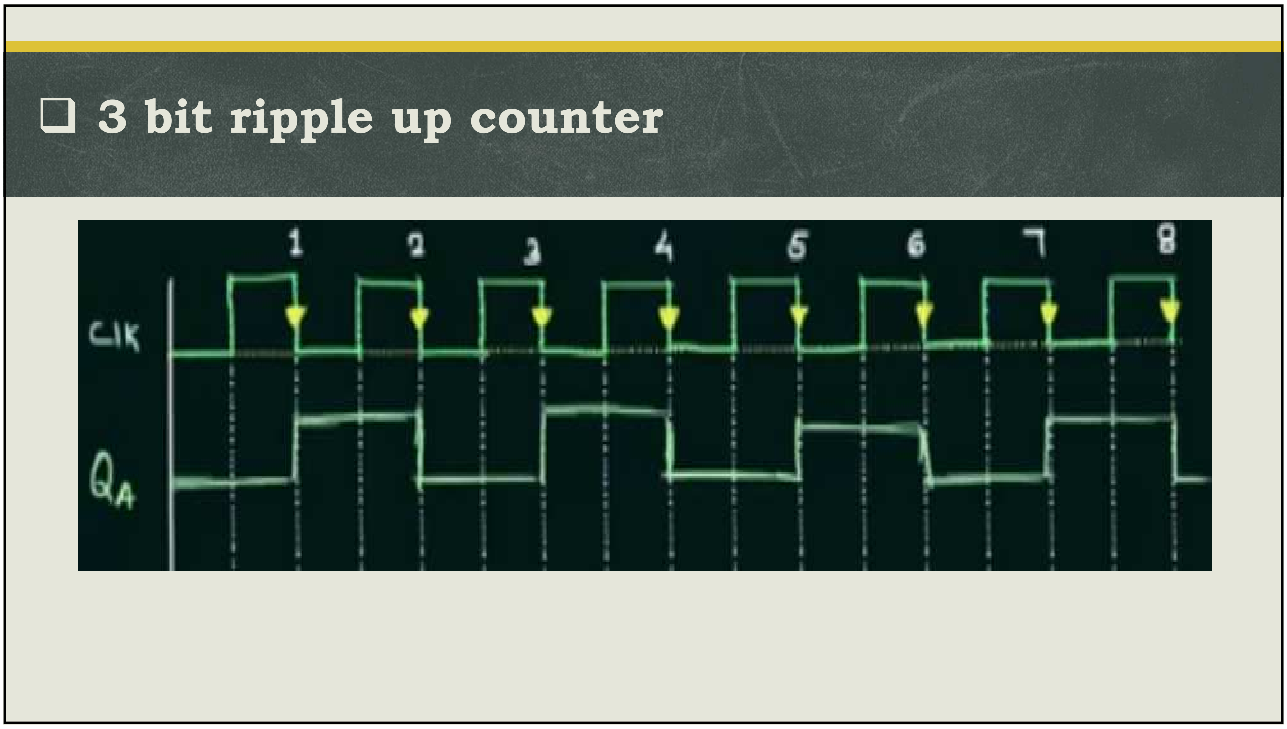
- **UP/DOWN Ripple Counters**
- In the UP/DOWN ripple counter all the FFs operate in the toggle mode. So either T flip-flops or JK flip-flops are to be used.
- **UP counting mode (M=0)** – The Q output of the preceding FF is connected to the clock of the next stage if up counting is to be achieved. For this mode, the mode select input M is at logic 0 (M=0).
- **DOWN counting mode (M=1)** – If M = 1, then the Q bar output of the preceding FF is connected to the next FF. This will operate the counter in the counting mode.

## ❑ 3 bit ripple up counter



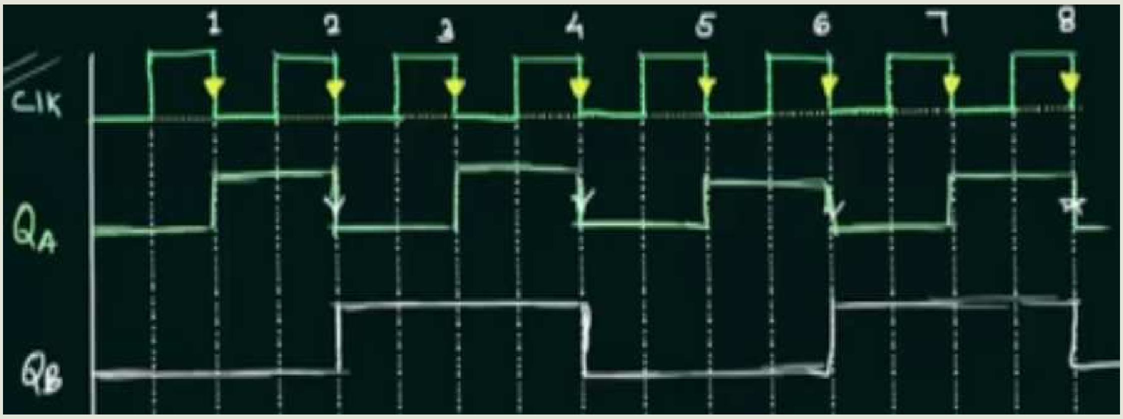


## ❑ 3 bit ripple up counter

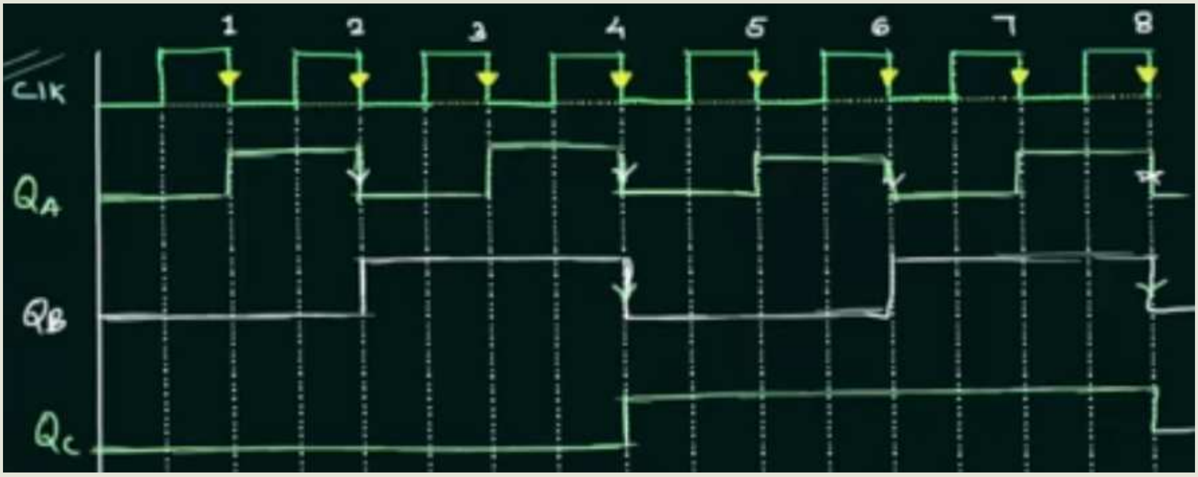


## ❑ 3 bit ripple up counter

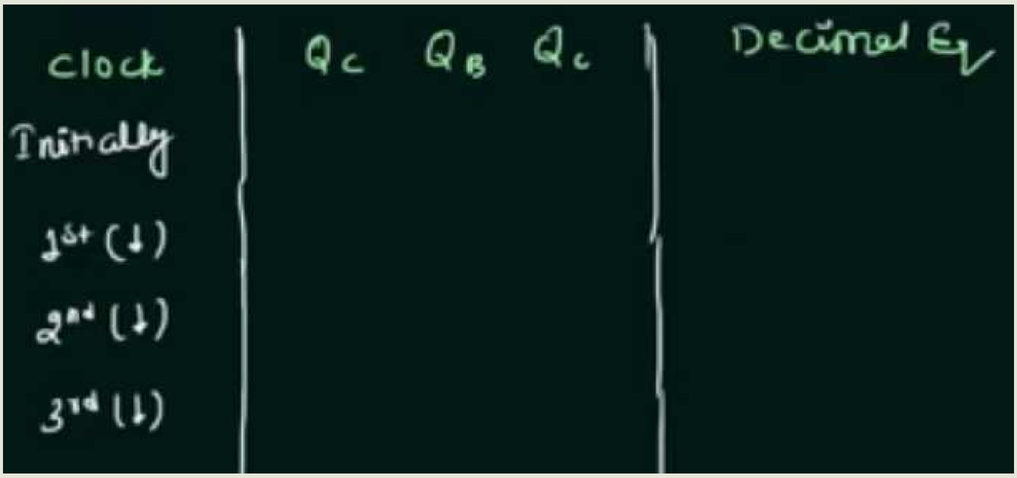
## ❑ 3 bit ripple up counter



## ❑ 3 bit ripple up counter

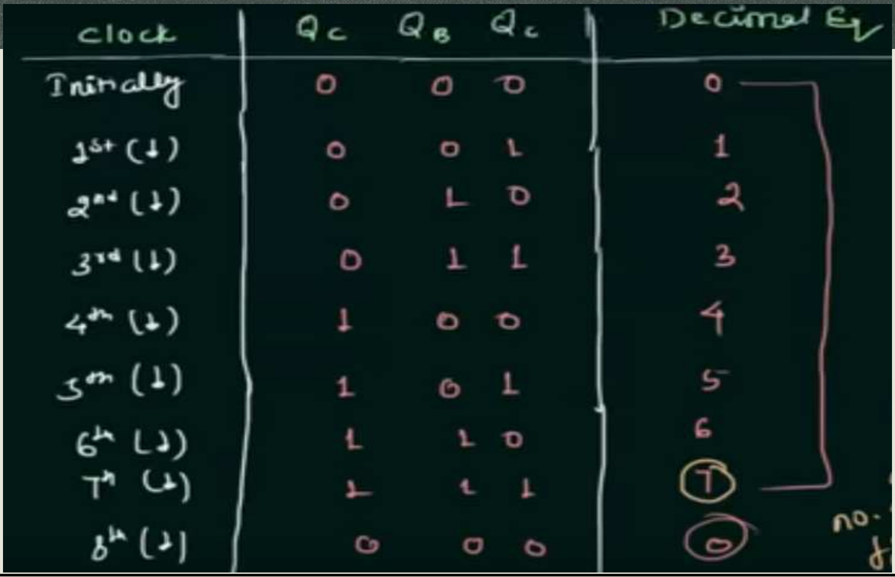


## ❑ 3 bit ripple up counter



| clock | $Q_C$ | $Q_B$ | $Q_c$ | Decimal Eq |
|---|---|---|---|---|
| Initially | | | | |
| 1st (↓) | | | | |
| 2nd (↓) | | | | |
| 3rd (↓) | | | | |

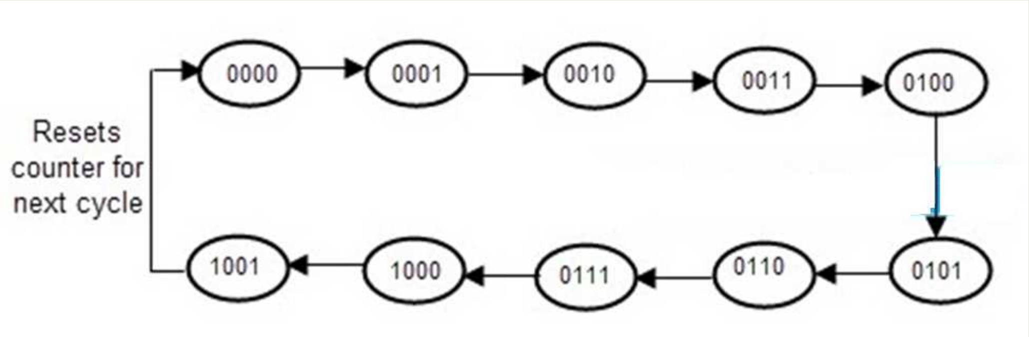
## ❑ 3 bit ripple up counter



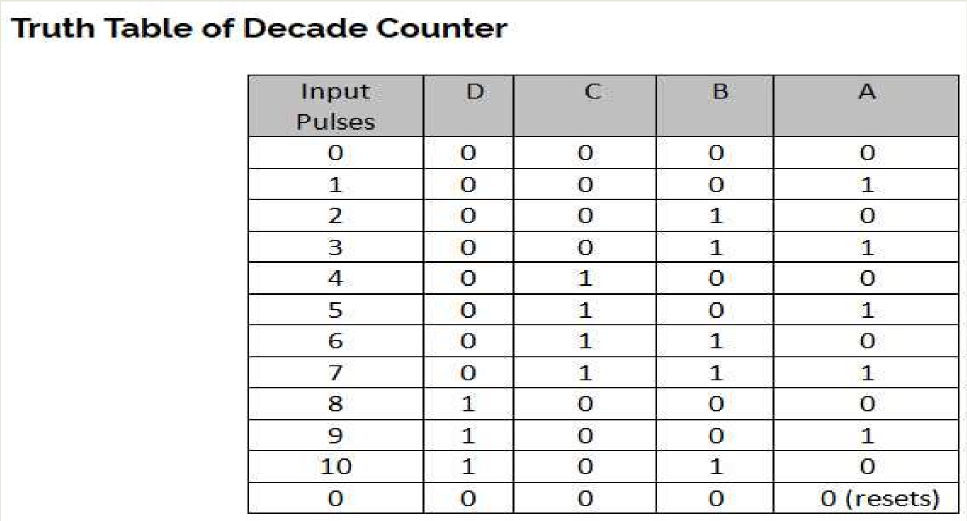
| clock | $Q_C$ | $Q_B$ | $Q_c$ | Decimal Eq |
|---|---|---|---|---|
| Initially | 0 | 0 | 0 | 0 |
| 1st (↓) | 0 | 0 | 1 | 1 |
| 2nd (↓) | 0 | 1 | 0 | 2 |
| 3rd (↓) | 0 | 1 | 1 | 3 |
| 4th (↓) | 1 | 0 | 0 | 4 |
| 5th (↓) | 1 | 0 | 1 | 5 |
| 6th (↓) | 1 | 1 | 0 | 6 |
| 7th (↓) | 1 | 1 | 1 | 7 |
| 8th (↓) | 0 | 0 | 0 | 0 |

## ❑ BCD or Decade Counter

- A binary coded decimal (BCD) is a serial digital counter that counts ten digits .And it resets for every new clock input. As it can go through 10 unique combinations of output, it is also called as "Decade counter".



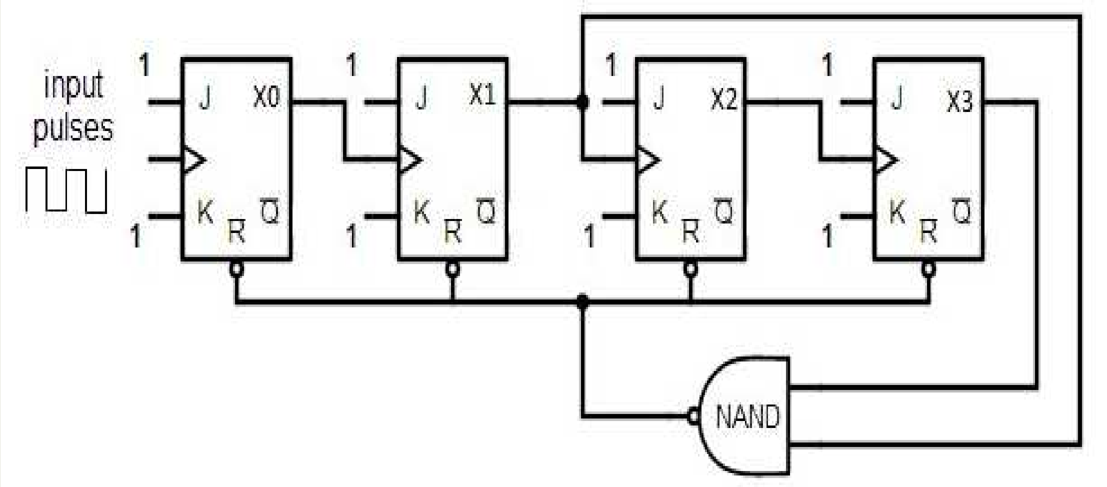
## ❑ BCD or Decade Counter



Truth Table of Decade Counter

| Input Pulses | D | C | B | A |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 |
| 6 | 0 | 1 | 1 | 0 |
| 7 | 0 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 |
| 10 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 (resets) |

## ❑ BCD or Decade Counter


input pulses
1
J
X0
K
R
Q̄
1
1
J
X1
K
R
Q̄
1
1
J
X2
K
R
Q̄
1
1
J
X3
K
R
Q̄
1
NAND