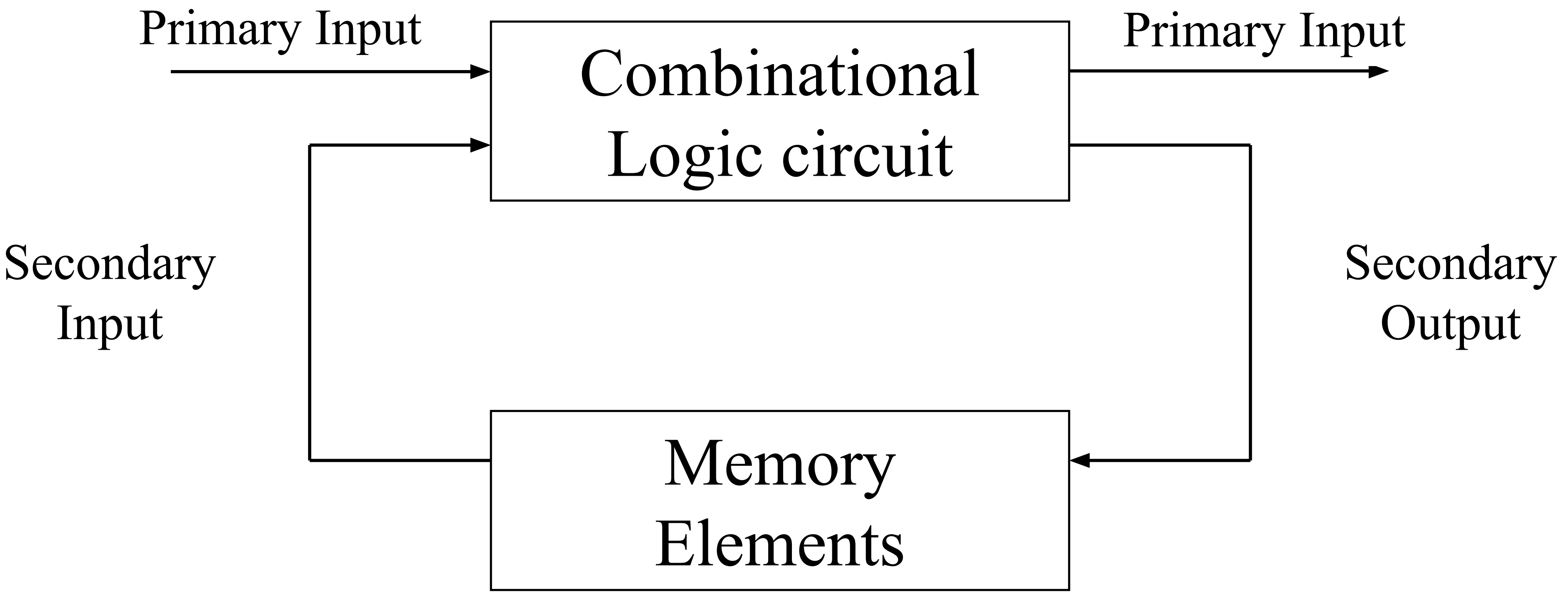
Sequential circuits

- Introduction.
- Latch.
- Flip-Flops.
- Registers.
- Counters

Introduction

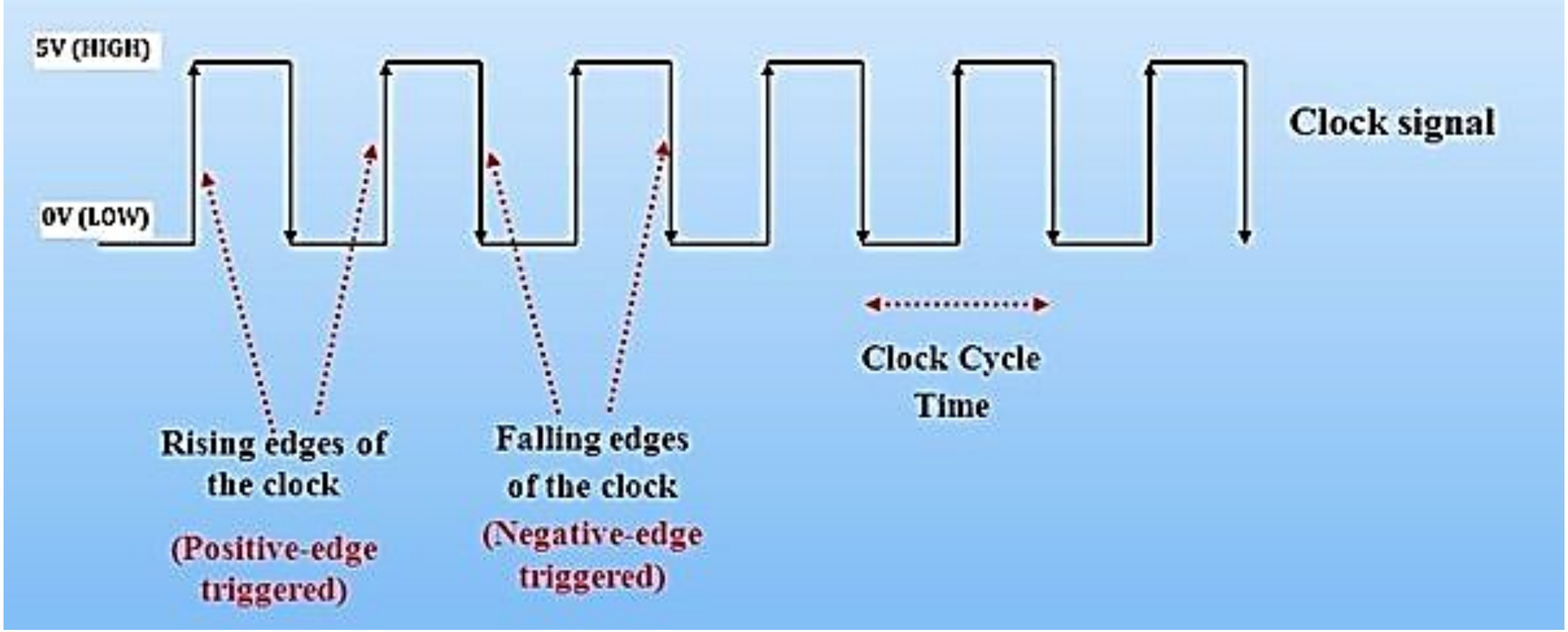
- As studied, Digital circuits without memory components are known as Combinational circuits.
- The Output at any given time in combinational circuits depends only on the input provided.
- **Sequential circuits** are digital circuits wit memory components & that operate in an orderly circuit the outcome is dependent on both the current input & the information kept in a memory component.
- The data kept in the memory component is in relation to previous inputs & outcomes.

Introduction



Introduction

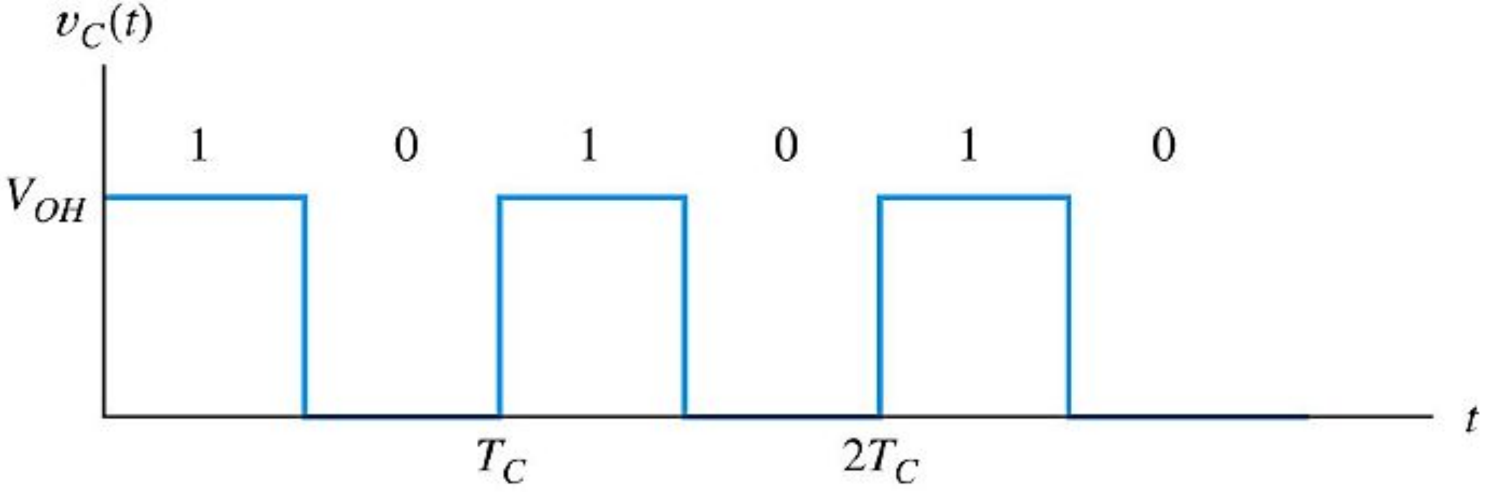
- **Clock Pulse**


5V (HIGH)
0V (LOW)
Clock signal
Rising edges of the clock
(Positive-edge triggered)
Falling edges of the clock
(Negative-edge triggered)
Clock Cycle Time

Introduction

- **<u>Clock Signal</u>**

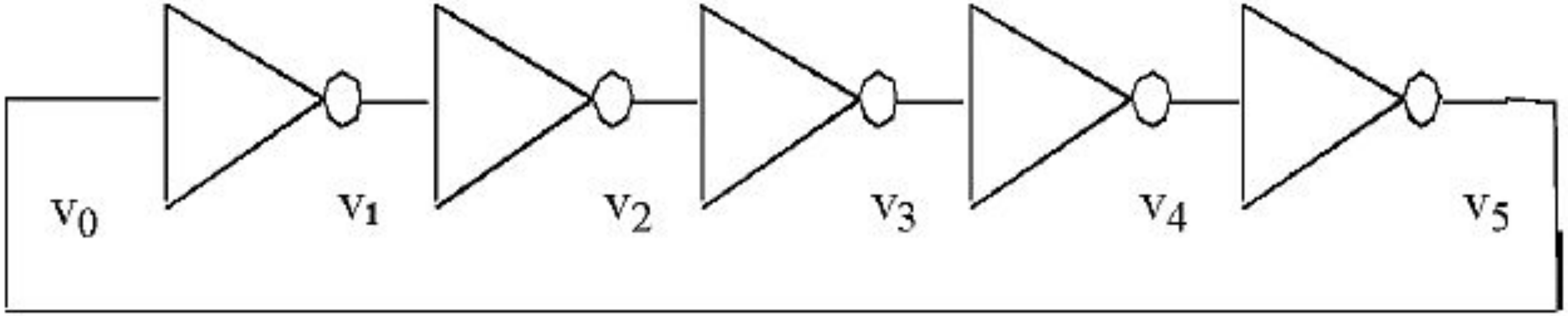
- Sequential logic circuits have **memory**.
- Output is a function of <u>input and present state</u>.
- Sequential circuits are synchronized by a periodic **“clock”** signal



Introduction

- **<u>Clock Signal Generator</u>**

 Clock signals can be generated using odd number of inverters.



The “WHY” slide

- Memory storage elements
  - In order to do fun problems like the door combination lock, we must know the building blocks (like how you had to learn AND and OR before you could do functional things). Be patient --- once you know these elements, you can build a lot of meaningful functions
- State diagrams
  - For combinational logic, truth table was an invaluable visualization tool for a function. For sequential logic, state diagram serves as a way to visualize a function.

Latch

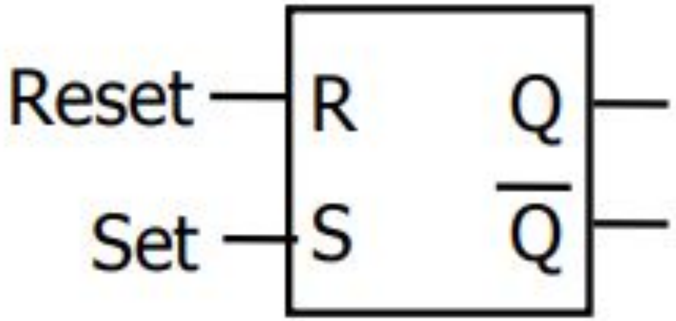
- In digital electronics **latch** is a bistable device capable of staying in either of the two states.
- These two states are **SET** and **RESET.**
- **Flip-Flop** are also the bistable device similar to latches.
- **Latch** & **Flip-Flop** are both fundamental storage components of sequential circuits, and both can store a single bit of data.

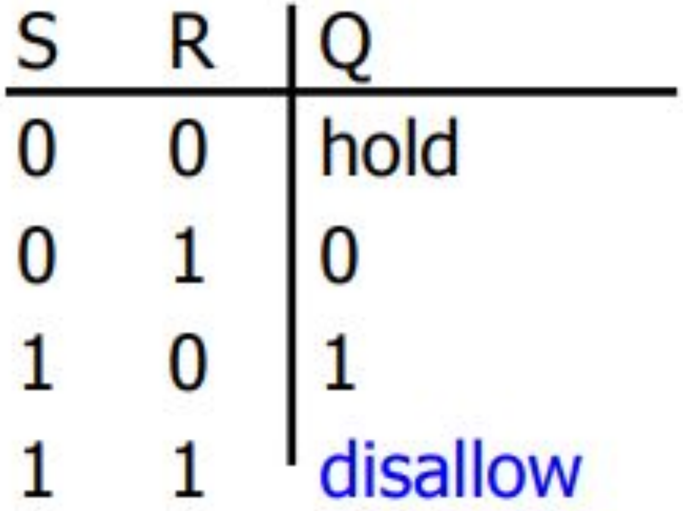
Latch

- Latch is asynchronous, or independent of the pulse of the clock.
- If Enable is set **HIGH**, the output are changed when the inputs are changed.
- When Enabled to **LOW**, the latch maintains its initial state regardless of input changes.

SR Latch

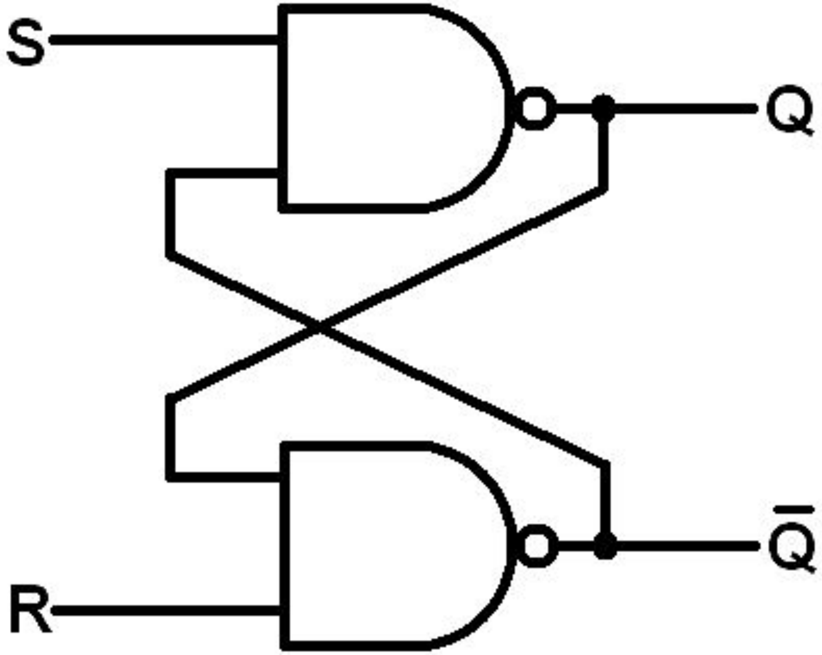
- Cross-coupled NOR gates
  - Can set (S=1, R=0) or reset (R=1, S=0) the output

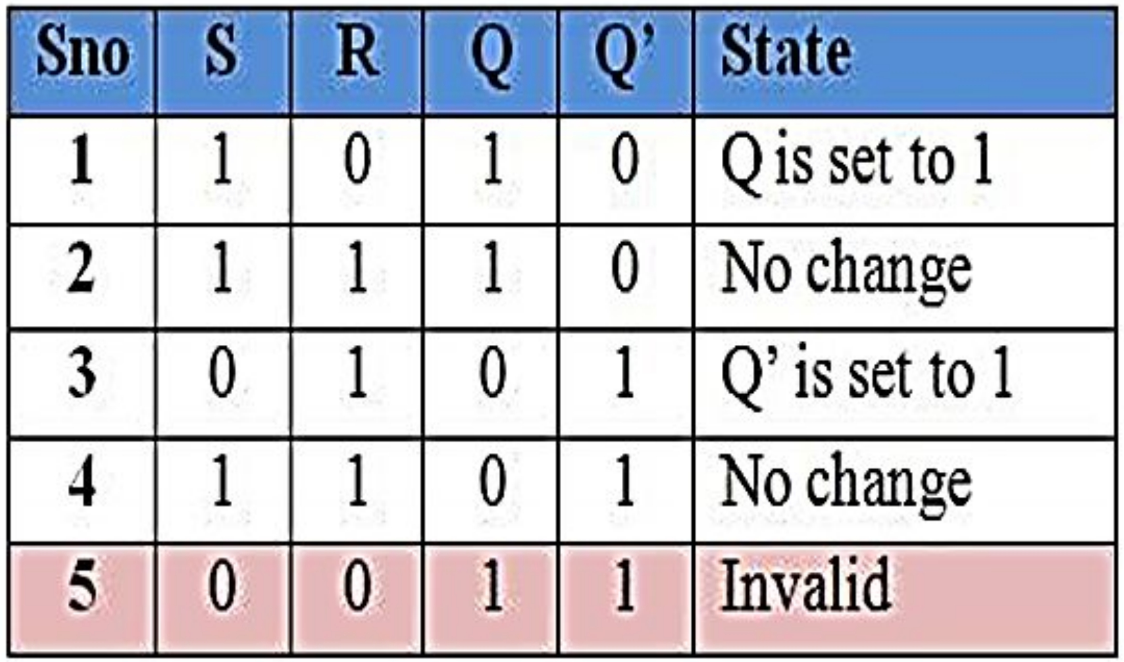




S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	disallow

SR Latch using NAND Gate



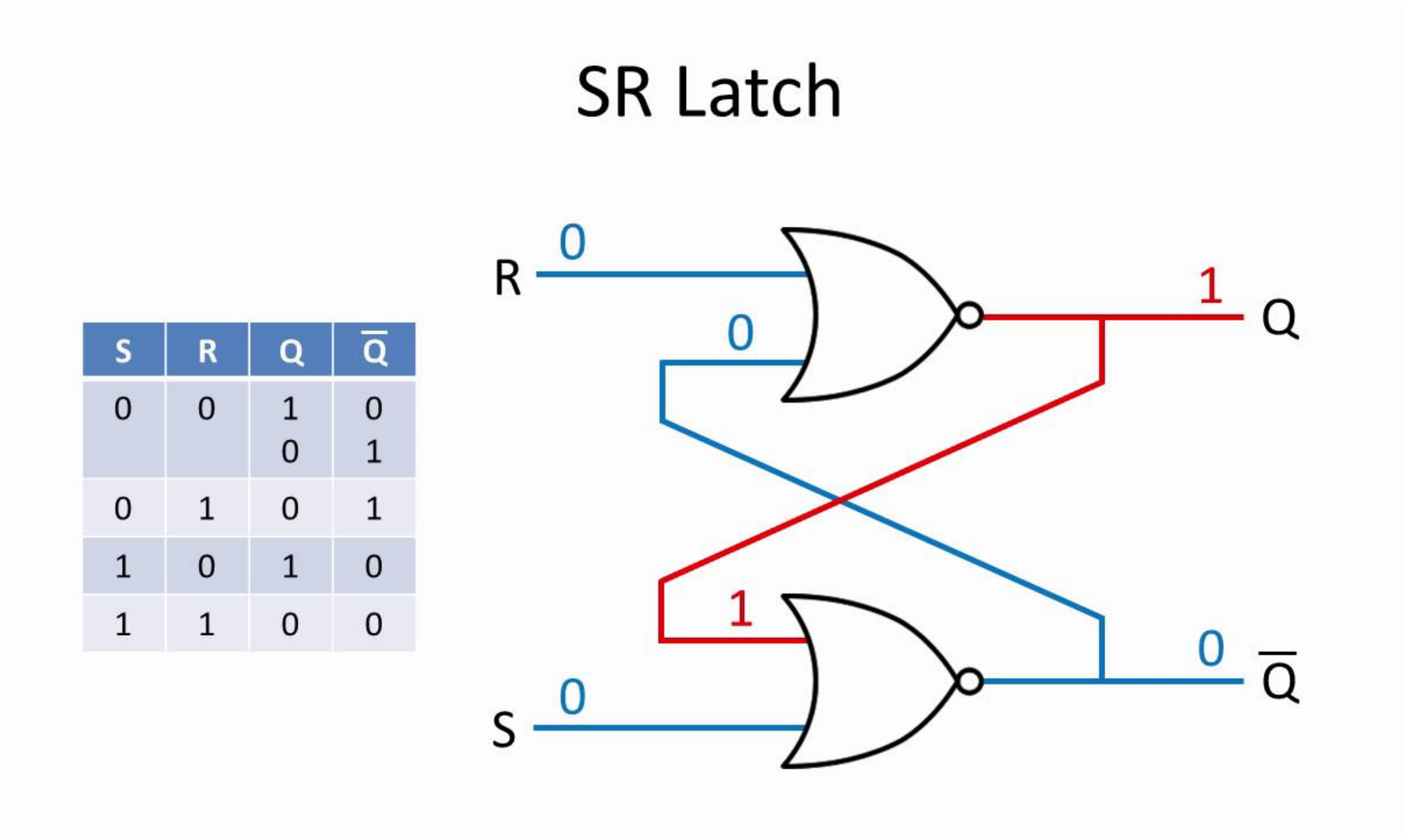


Sno	S	R	Q	Q’	State
1	1	0	1	0	Q is set to 1
2	1	1	1	0	No change
3	0	1	0	1	Q’ is set to 1
4	1	1	0	1	No change
5	0	0	1	1	Invalid

SR Latch using NAND Gate

Input		Output		Description
S	R	Q	Q’	
0	1	1	0	SET
1	1	1	0	Output does not change
1	0	0	1	RESET
1	1	0	1	Output does not change
0	0	1	1	INVALID state

SR latch using NOR Gate



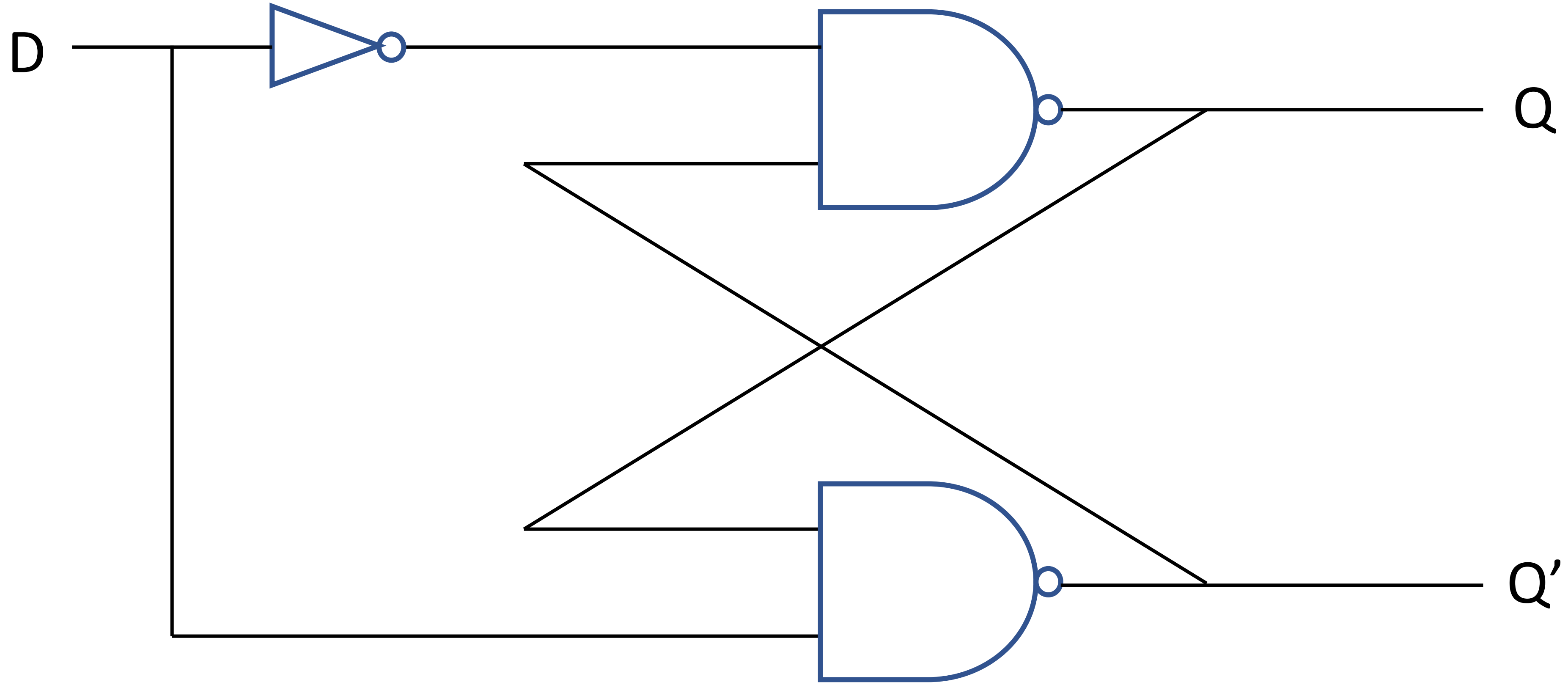
SR Latch

S	R	Q	$\overline{Q}$
0	0	1 0	0 1
0	1	0	1
1	0	1	0
1	1	0	0

SR Latch using NOR Gate

Input		Output		Description
S	R	Q	Q’	
1	0	1	0	SET
0	0	1	0	Output does not change
0	1	0	1	RESET
0	0	0	1	Output does not change
1	1	0	0	INVALID state

D Latch using NAND Gate

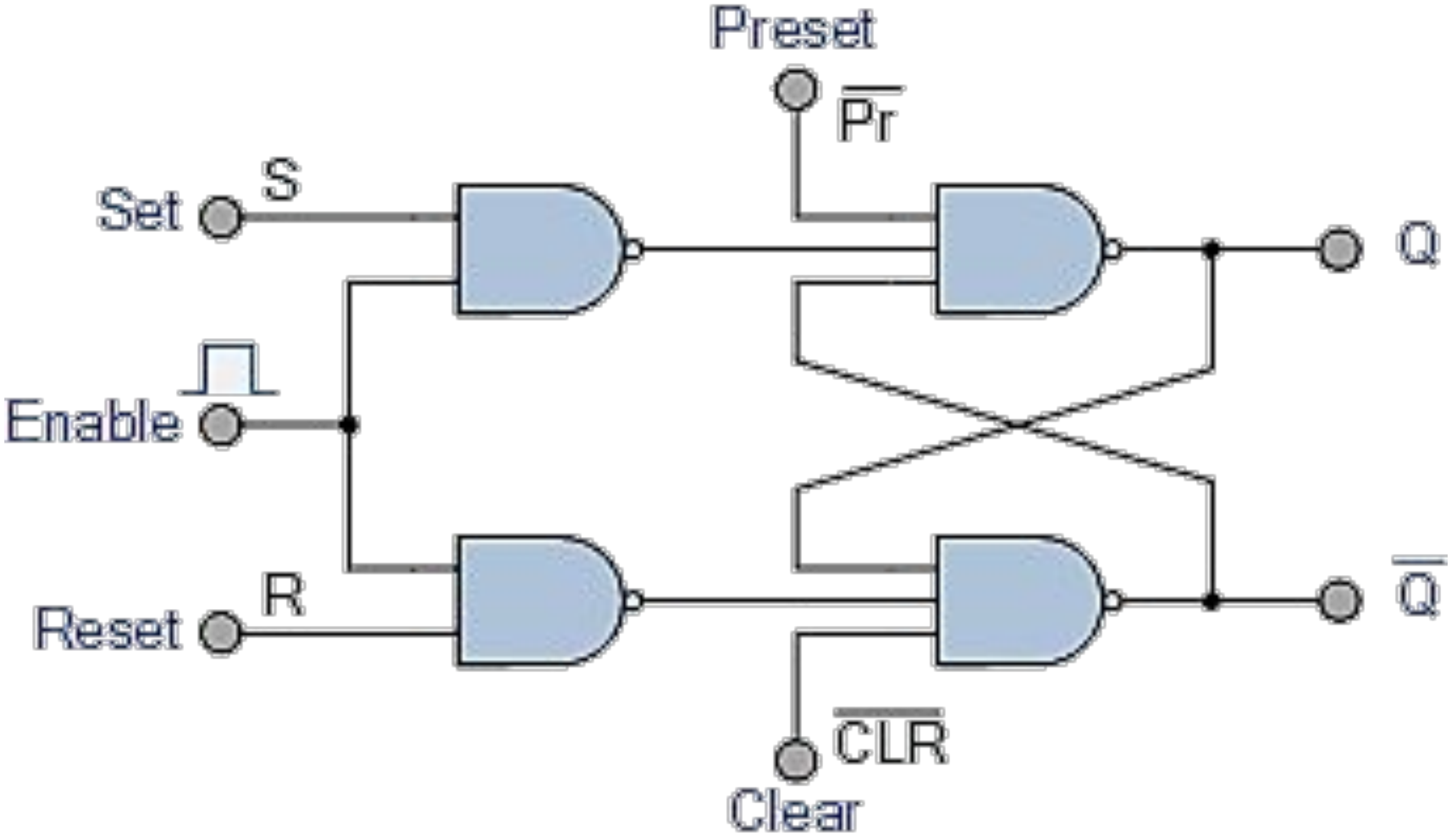


D Latch using NAND Gate

Input	Output		Description
D	**Q**	**Q’**	
1	1	0	SET
0	0	1	RESET

D Latch using NAND Gate

with ENABLE (E) (Gated D Latch)



D Latch with ENABLE

Input		Output		Description
E	**D**	**Q**	**Q’**	
0	x	Q	Q’	No change in output
1	1	1	0	SET state
1	0	0	1	RESET state

D Latch with ENABLE

- This Latch is also called as Gated Latch. When the input ENABLE **(E)** = 0, the latch is disabled & hence the Output **Q** retains its last value Regardless of Input **D**, otherwise the Truth Table shows the values when its ENABLE.
- **PRESET** AND **CLEAR** inputs are the Asynchronous inputs – Used to bring the Flip-Flop to an initial SET state or CLEAR state.
- When PRESET is activated (i. e. PRESET is **1**), **Q** = 1 & **Q’ = 0**
- When CLEAR is activated (i. e. CLEAR is **1**), **Q’** = 1 & **Q = 0**

Level and Edge Triggering

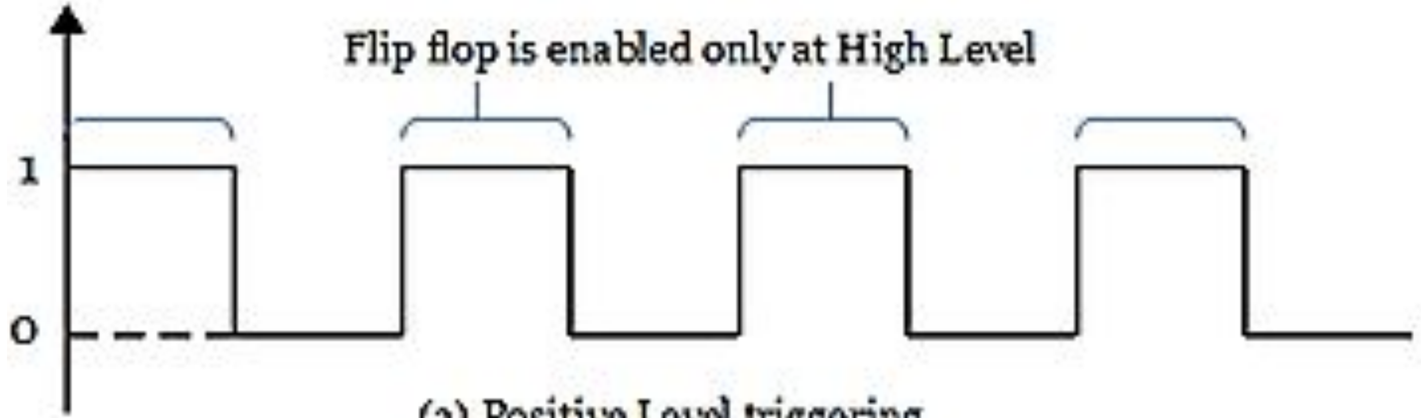
- The basic storage components of sequential circuits are flip-flops and latches.
- **Flipflops** are controlled by the clock signal, while **Latches** are controlled by the Enable signal.
- Latch is thus level triggered or level activated.
- Flipflop is clock edge triggered.

Level Triggering

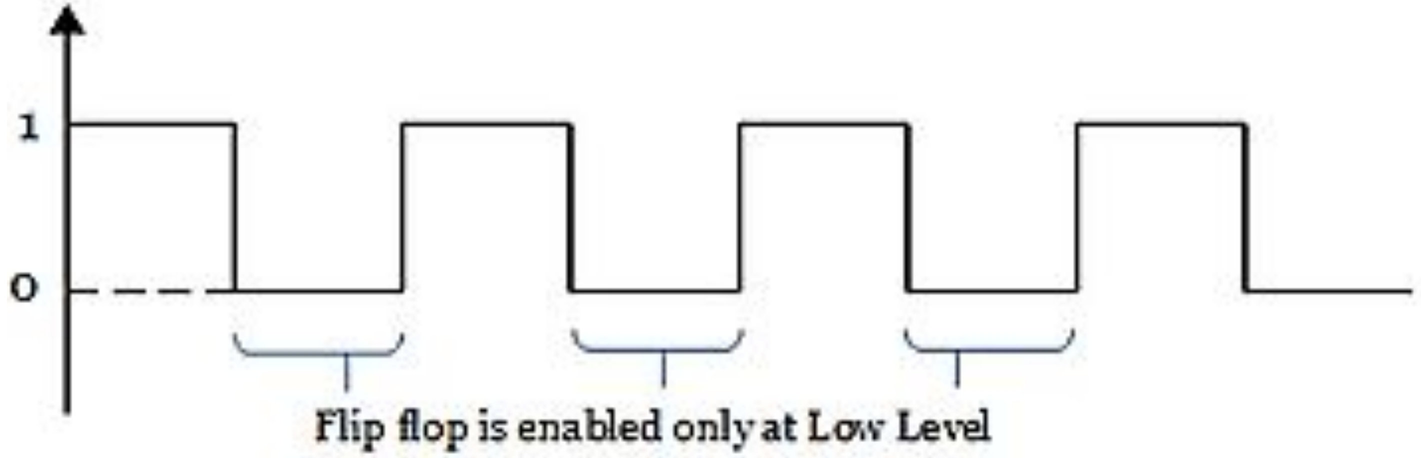
In Level Triggering, the output changes according to the input whenever the Enable signal is active.

The level triggering is of two types:

1. **Positive Level Triggering**
2. **Negative Level Triggering**



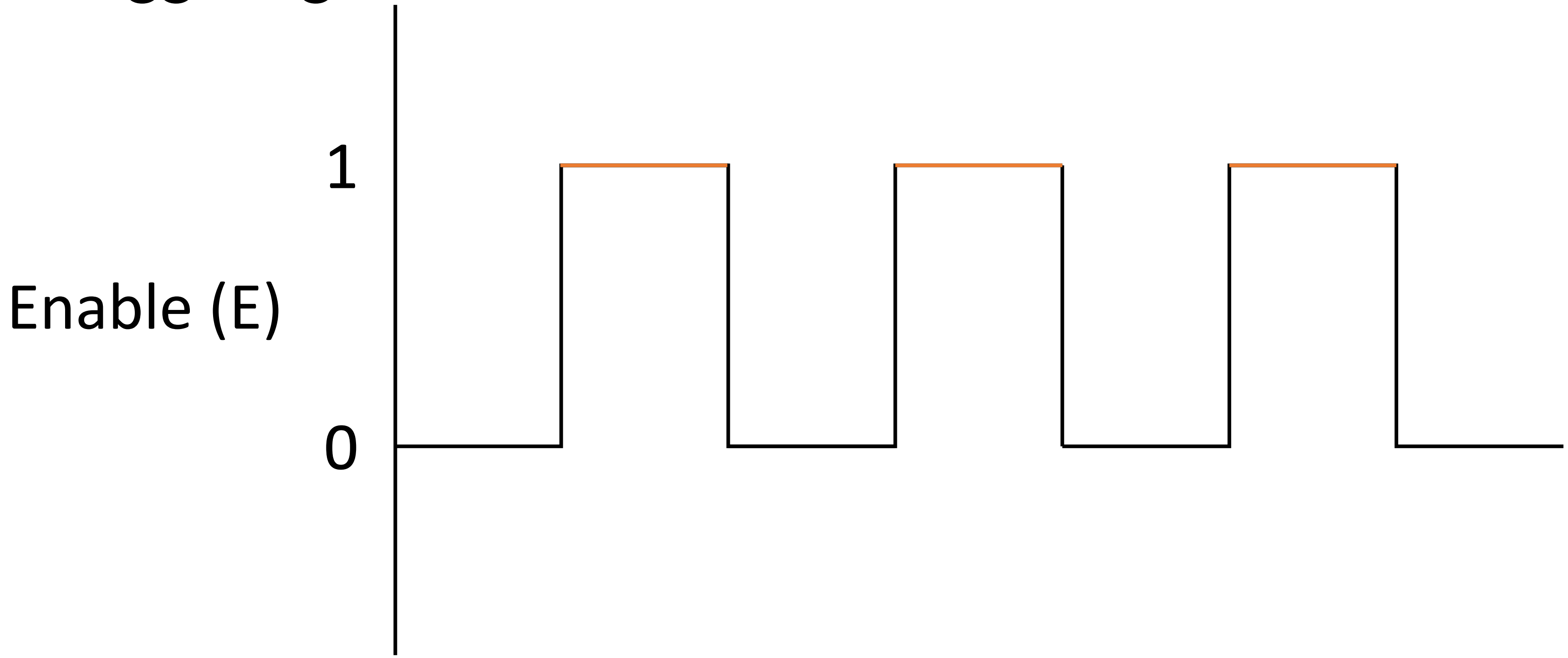
(a) Positive Level triggering



(b) Negative Level triggering

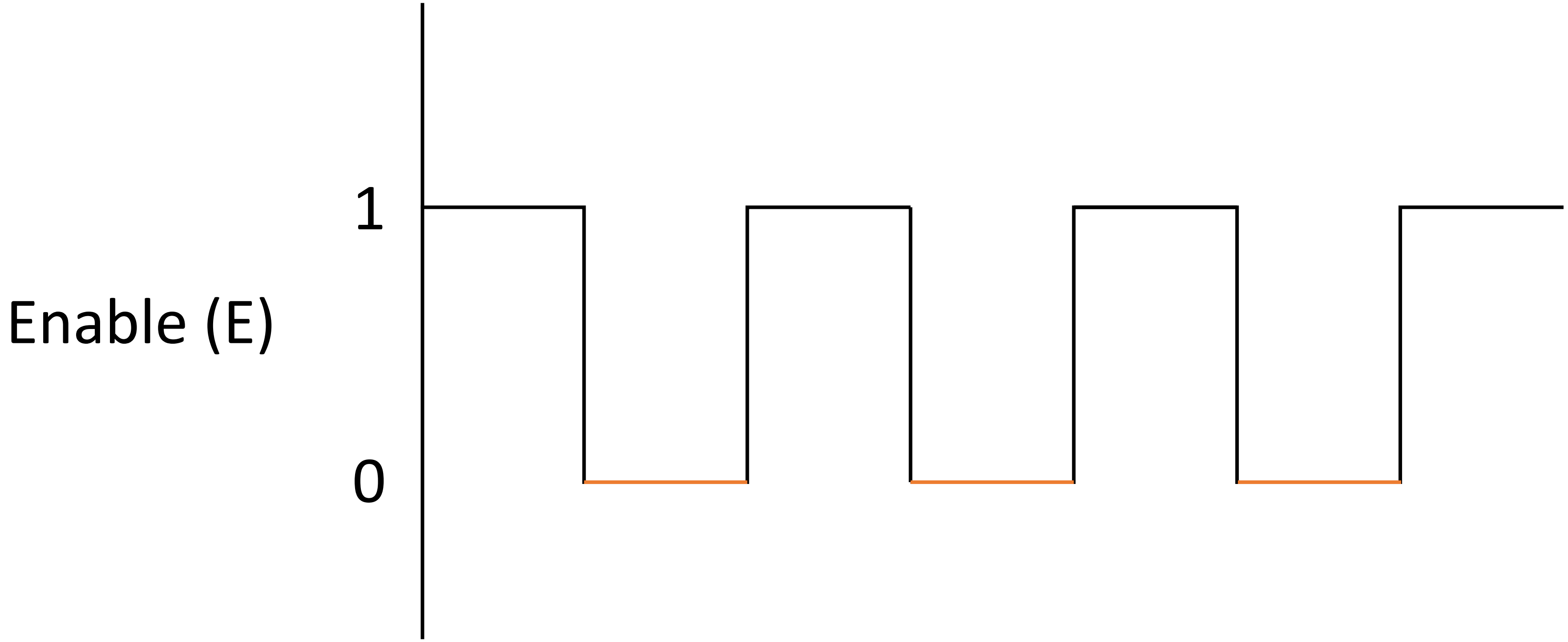
1. Positive Level Triggering

- If the sequential circuit is operated with the Enable signal when it is in **logic high**, then that type of triggering is known as a positive level triggering.



2. Negative Level Triggering

- If the sequential circuit is operated with the Enable signal when it is in **logic low**, then that type of triggering is known as a negative level triggering.



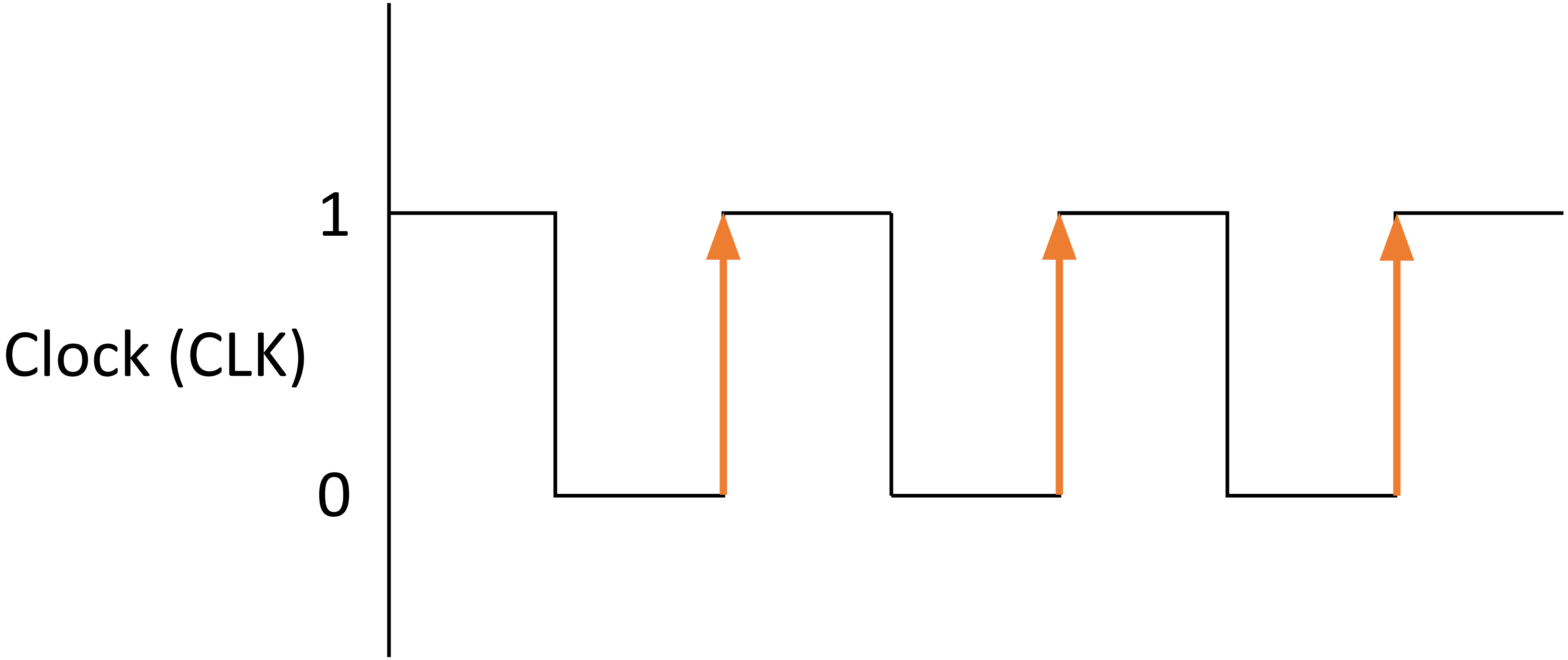
Edge Triggering

- Edge triggering is a type of triggering that allows a circuit to become active at the positive edge or the negative edge of the clock signal.
- Edge triggering is when the flip-flop state is changed as the rising or falling edge of a clock signal passes through a threshold voltage.

1. **Positive Edge Triggering.**
2. **Negative Edge Triggering.**

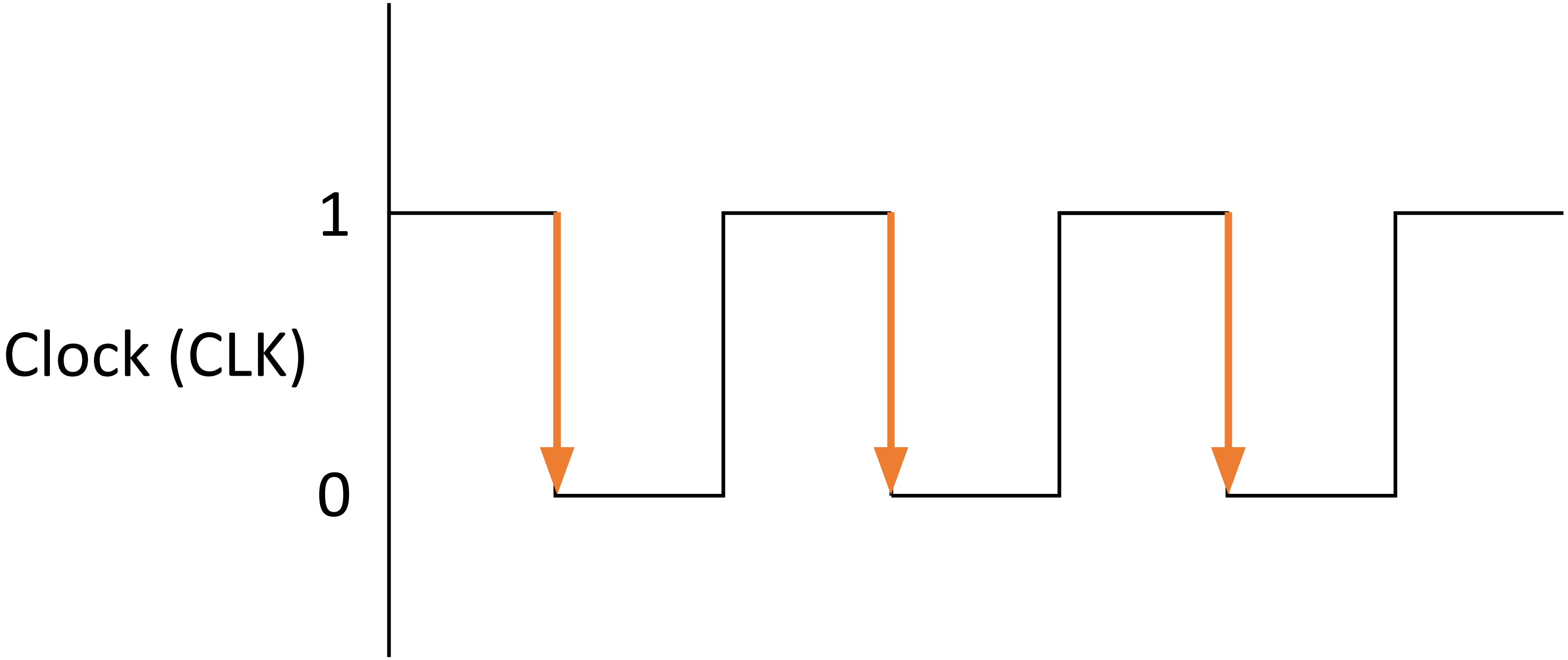
1. Positive Edge Triggering

- A positive edge (or rising edge) is the low-to-high transition. When the clock signal makes a transition from **low to high** i.e. from **0 to 1** it is termed as positive edge triggered.

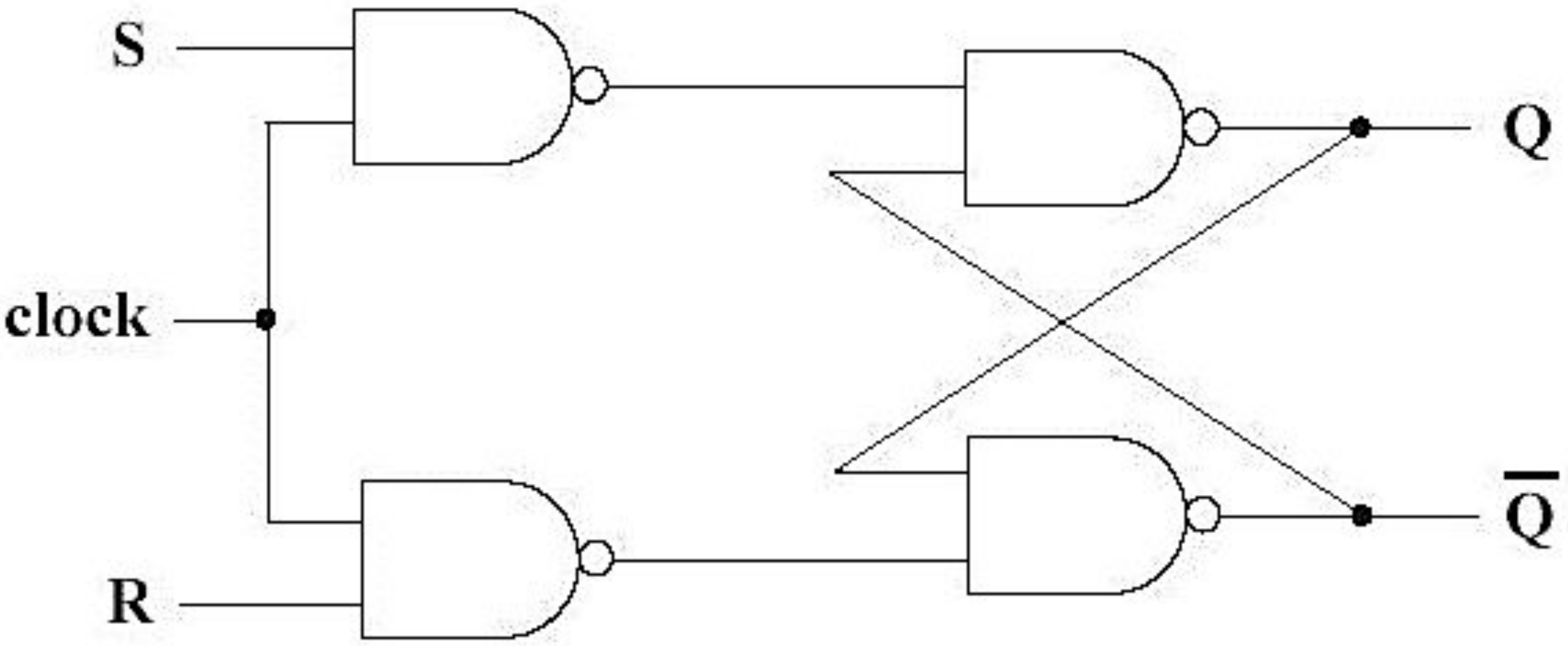


2. Negative Edge Triggering

- A negative edge (or falling edge) is the high-to-low transition. When the clock signal makes a transition from **high to low** i.e. from **1 to 0** it is termed as negative edge triggered.



SR Flip-Flop using NAND Gate



SR Flip-Flop using NAND Gate

Input			Output		Description
Clock	S	R	Q	Q’	
1	0	0	Q	Q’	Output does not change
1	0	1	0	1	RESET
1	1	0	1	0	SET
1	1	1	x	x	INVALID STATE
0	x	x	Q	Q’	Output does not change

SR Flip-Flop using NOR Gate – same as NAND gate

Input			Output		Description
Clock	S	R	Q	Q’	
1	0	0	Q	Q’	Output does not change (HOLD)
1	0	1	0	1	RESET
1	1	0	1	0	SET
1	1	1	x	x	INVALID STATE
0	x	x	Q	Q’	Output does not change

D Flip-Flop using NAND Gate

D Flip-Flop

Input		Output		Description
Clock(clk)	D	Q	Q’	
0	x	Q	Q’	No change in output
1	1	1	0	SET state
1	0	0	1	RESET state

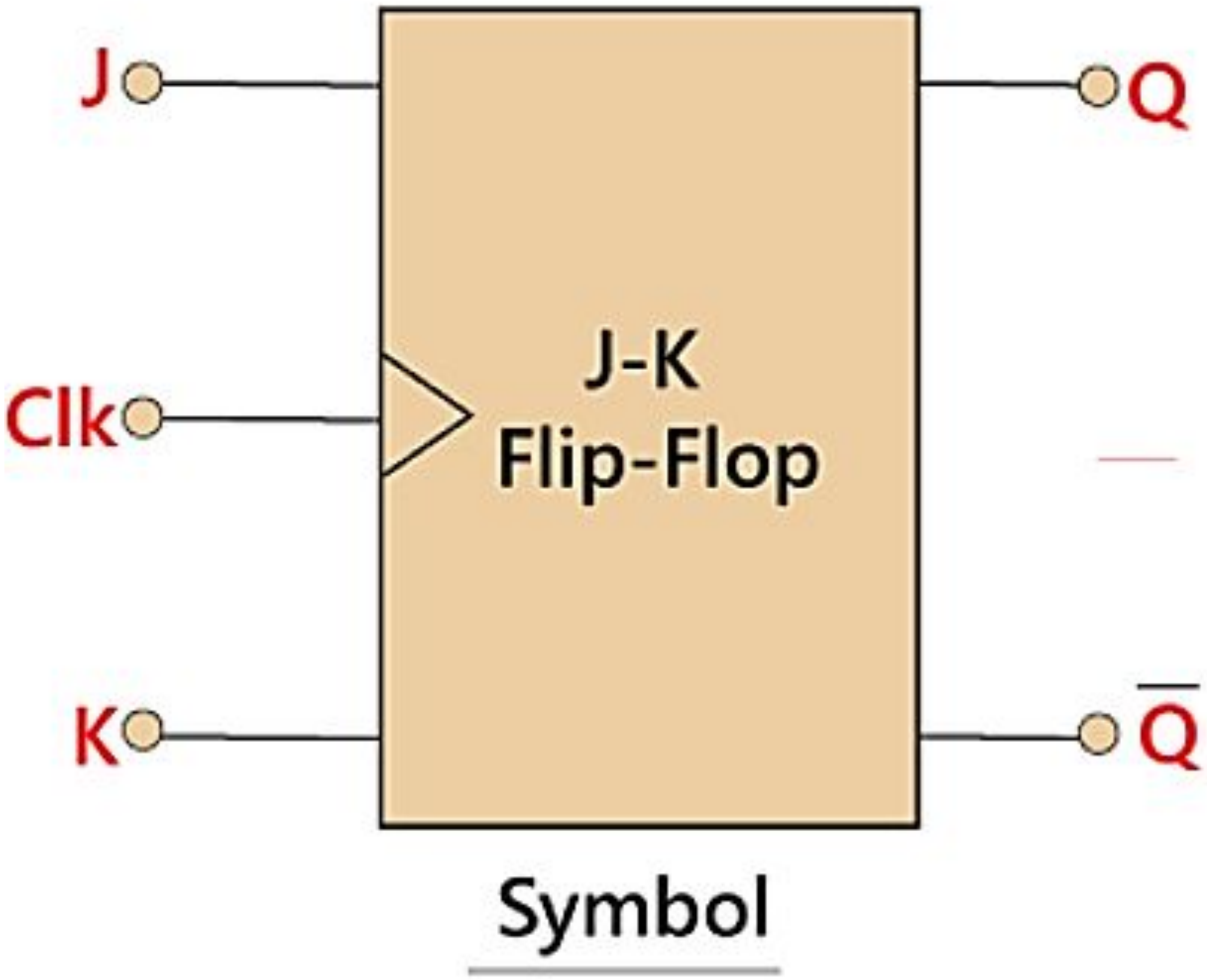
JK Flip-Flop

- The JK flip flop is one of the most used flip flops in digital circuits.
- The JK flip flop is a universal flip flop having two inputs 'J' and 'K'. In SR flip flop, the 'S' and 'R' are the shortened abbreviated letters for Set and Reset, but J and K are not. The J and K are themselves autonomous letters which are chosen to distinguish the flip flop design from other types.
- The JK flip flop work in the same way as the SR flip flop work.
- The JK flip flop has 'J' and 'K' flip flop instead of 'S' and 'R'.

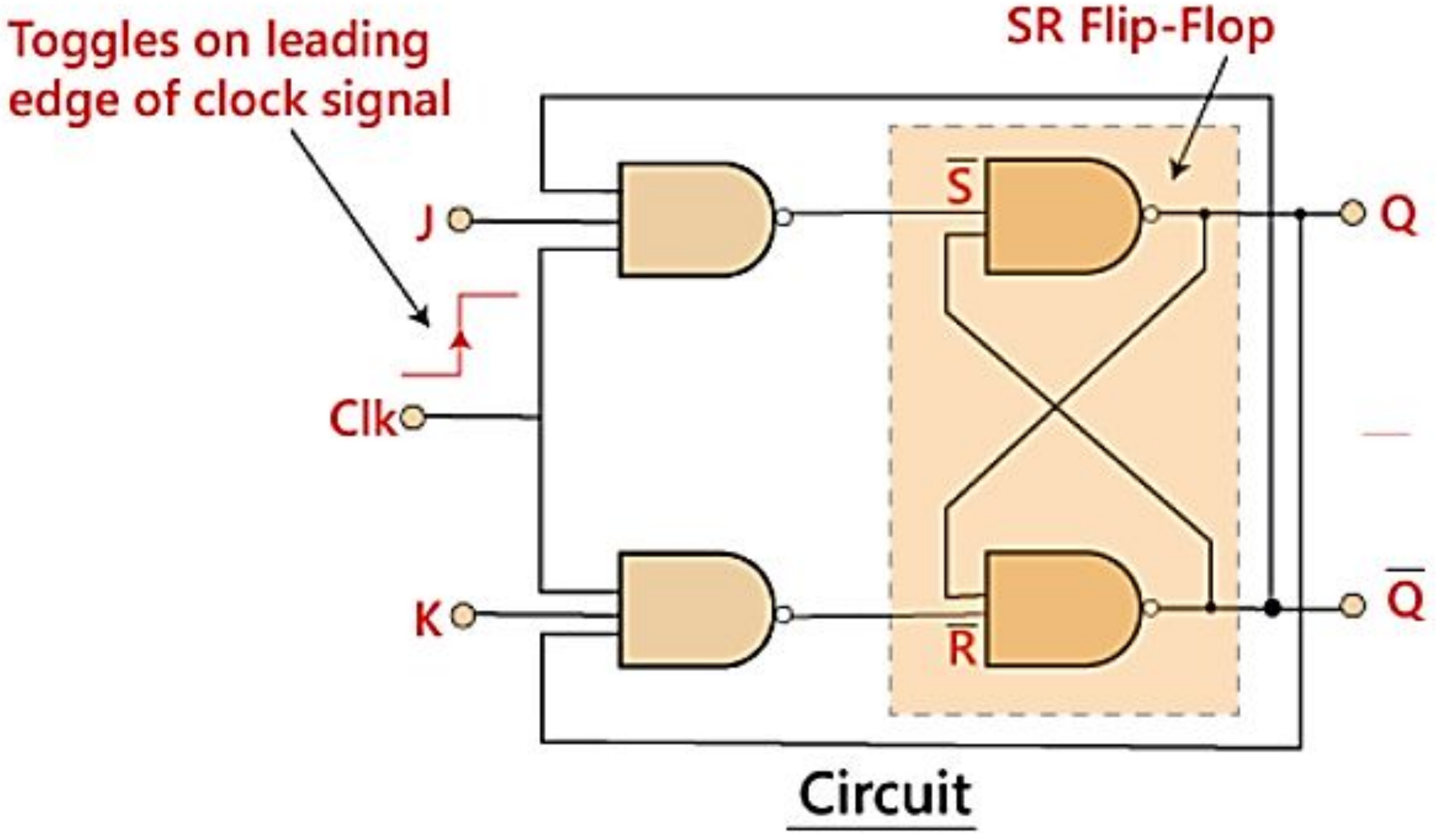
JK Flip-Flop

- The only difference between JK flip flop and SR flip flop is that when both inputs of SR flip flop is set to 1, the circuit produces the invalid states as outputs, but in case of JK flip flop, there are no invalid states even if both 'J' and 'K' flip flops are set to 1.

JK Flip-Flop



JK Flip-Flop


Toggles on leading edge of clock signal
SR Flip-Flop
J
$\overline{S}$
Q
Clk
K
$\overline{R}$
$\overline{Q}$
Circuit

JK Flip-Flop

- In SR flip flop, both the inputs 'S' and 'R' are replaced by two inputs J and K. It means the J and K input equates to S and R, respectively.
- The two 2-input AND gates are replaced by two 3-input NAND gates. The third input of each gate is connected to the outputs at Q and Q'. The cross-coupling of the SR flip-flop permits the previous invalid condition of (S = "1", R = "1") to be used to produce the "toggle action" as the two inputs are now interlocked.

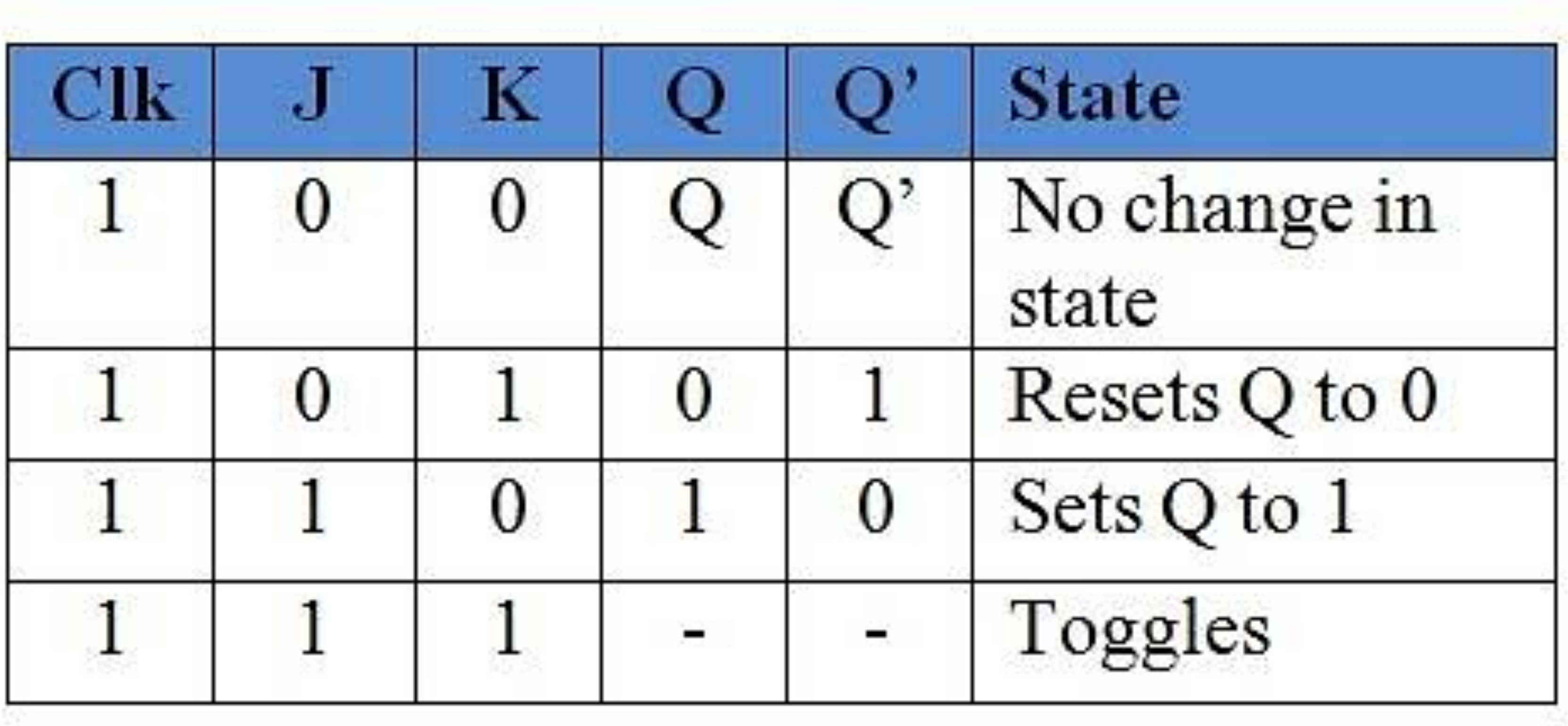
JK Flip-Flop

- If the circuit is "set", the J input is interrupted from the "0" position of Q' through the lower NAND gate. If the circuit is "RESET", K input is interrupted from 0 positions of Q through the upper NAND gate. Since Q and Q' are always different, we can use them to control the input. When both inputs 'J' and 'K' are set to 1, the JK toggles the flip flop.

Positive Edge Triggered - JK Flip-Flop

- In SR flip flop there arise invalid state when **S = 1 & R = 1** because both output **Q** & **Q’** becomes same and this is not possible as **Q’** is complement of **Q**. To avoid this condition SR flip flop is converted to JK flip flop.
- Hence J corresponds to S (SET) input & K corresponds to R (RESET) input of SR flip flop.

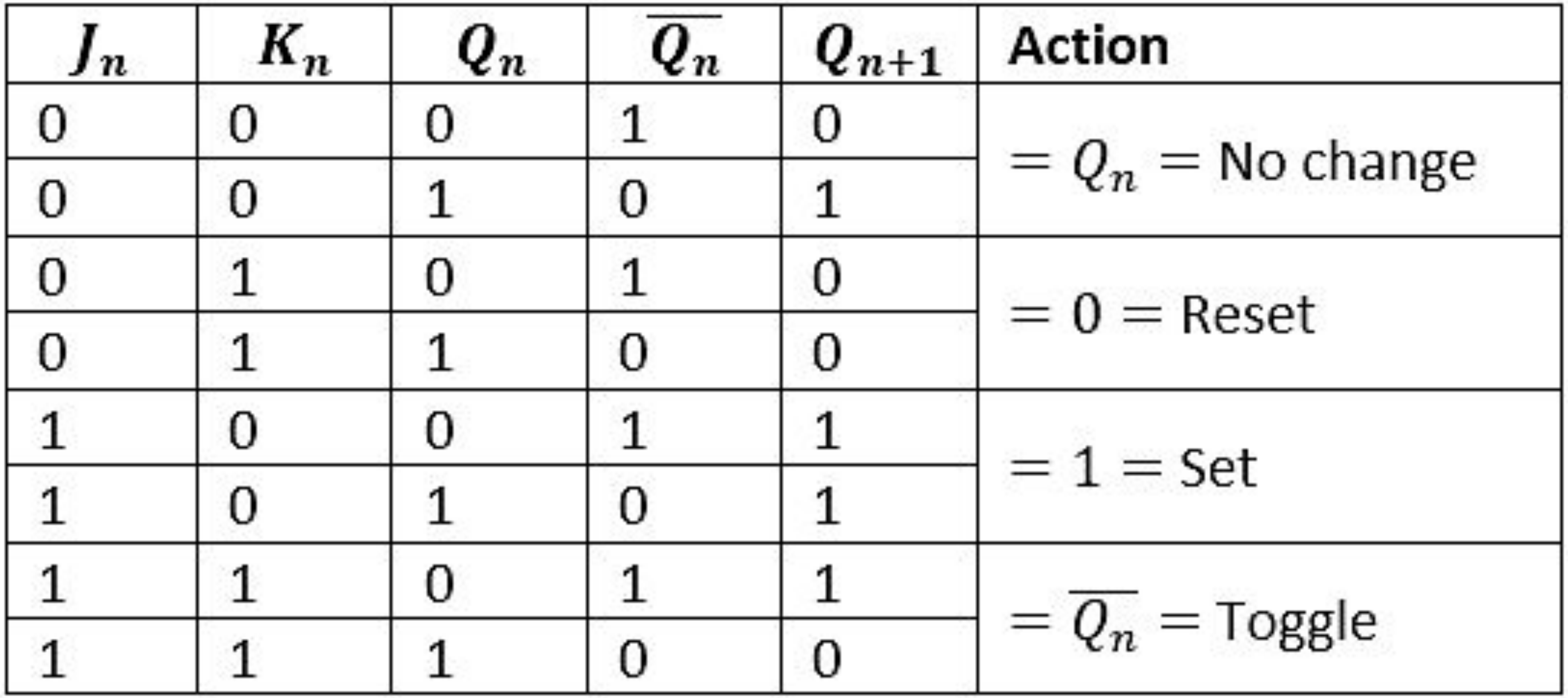
Positive Edge Triggered - JK Flip-Flop



Clk	J	K	Q	Q’	State
1	0	0	Q	Q’	No change in state
1	0	1	0	1	Resets Q to 0
1	1	0	1	0	Sets Q to 1
1	1	1	-	-	Toggles

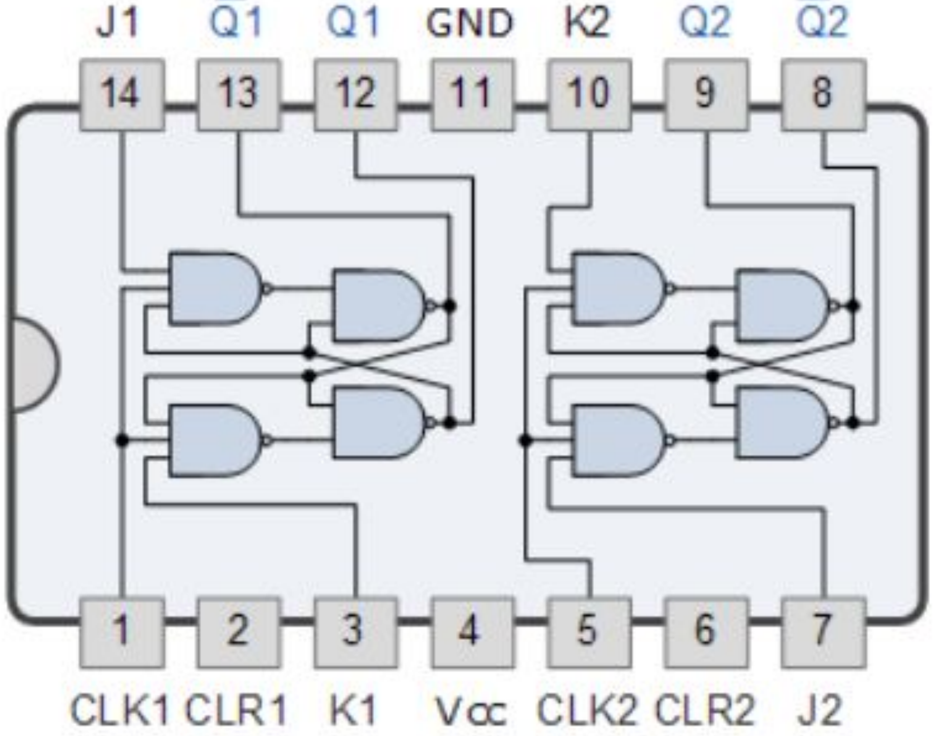
Positive Edge Triggered - JK Flip-Flop

Characteristic table

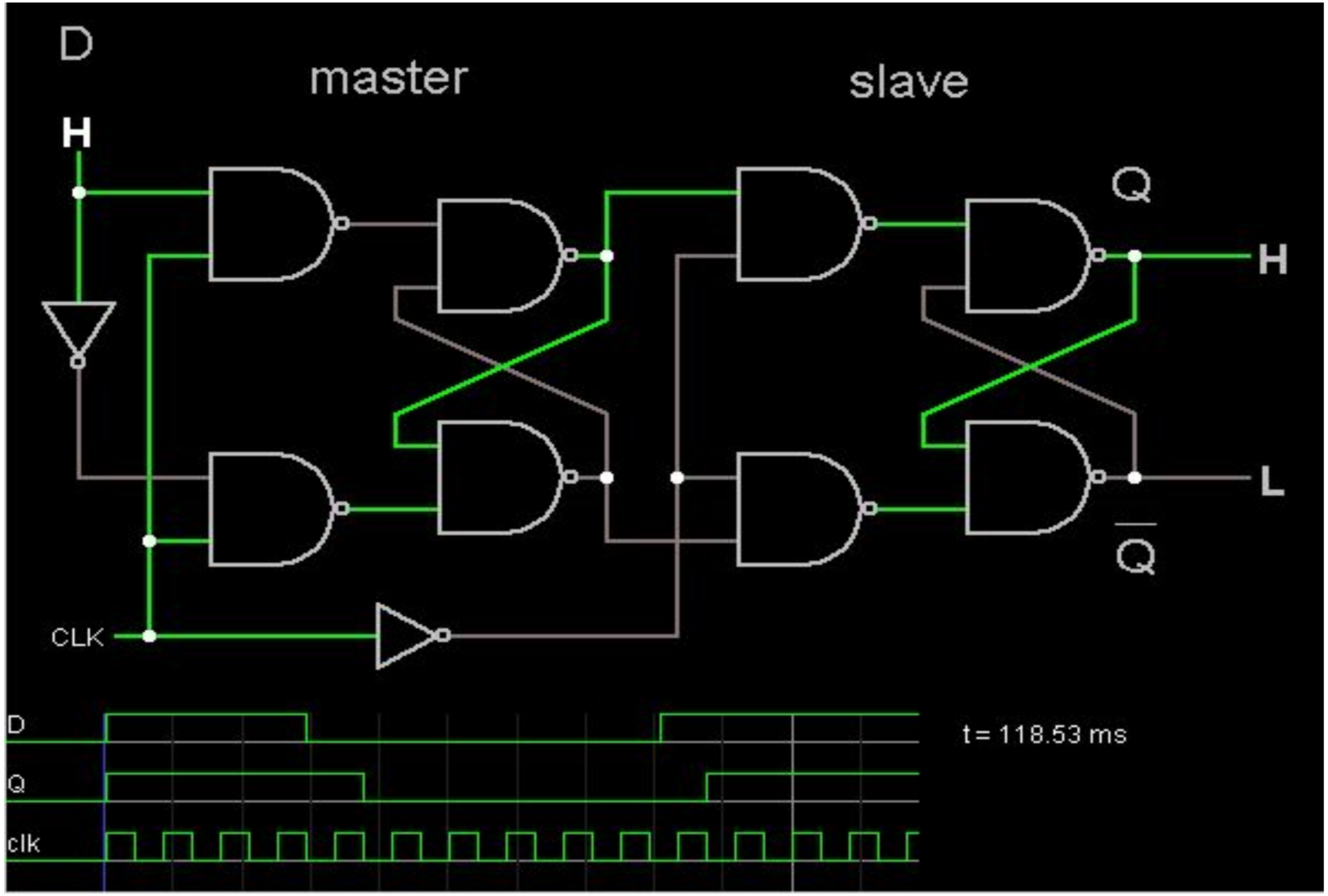


J_n	K_n	Q_n	$\overline{Q_n}$	Q_{n+1}	Action
0	0	0	1	0	$= Q_n =$ No change
0	0	1	0	1	
0	1	0	1	0	$= 0 =$ Reset
0	1	1	0	0	
1	0	0	1	1	$= 1 =$ Set
1	0	1	0	1	
1	1	0	1	1	$= \overline{Q_n} =$ Toggle
1	1	1	0	0	

PIN diagram - 7473 - JK Flip-Flop



Master Slave D Flip-Flop



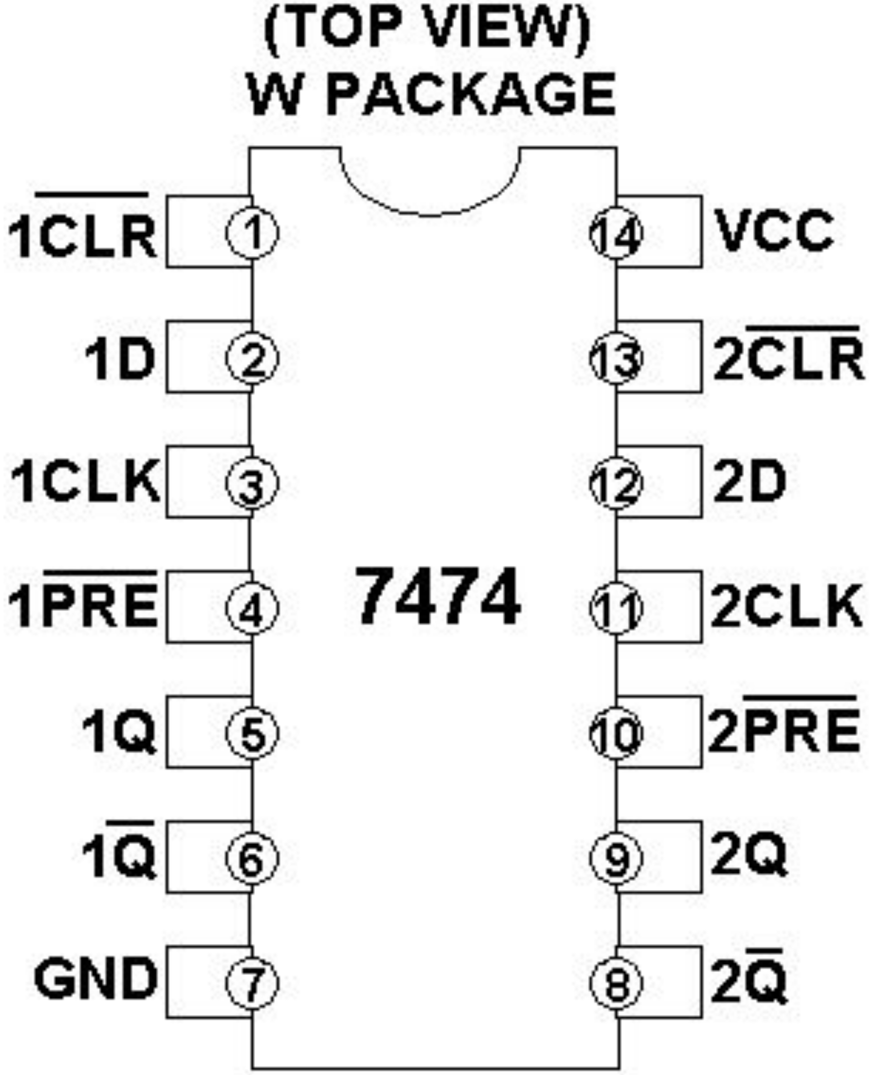
Master Slave D Flip-Flop

- The basic D-type flip flop can be improved further by adding a second SR flip-flop to its output that is activated on the complementary clock signal to produce a “Master-Slave D-type flip flop”. On the leading edge of the clock signal (LOW-to-HIGH) the first stage, the “master” latches the input condition at D, while the output stage is deactivated.

Master Slave D Flip-Flop

- On the trailing edge of the clock signal (HIGH-to-LOW) the second "slave" stage is now activated, latching on to the output from the first master circuit. Then the output stage appears to be triggered on the negative edge of the clock pulse. "Master-Slave D-type flip flops" can. be constructed by the cascading together of two latches with opposite clock.

PIN diagram - 7474 D Flip-Flop with PRESET & SET



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.
- The registers which will shift the bits to left are called “Shift left registers”.
- The registers which will shift the bits to right are called “Shift right registers”.

Shift Registers

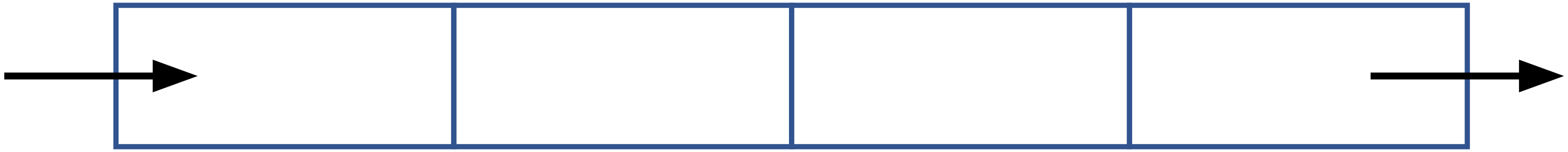
- Shift registers are basically of 4 types. These are:

- **Serial In Serial Out shift register (SISO)**
- **Serial In parallel Out shift register (SIPO)**
- **Parallel In Serial Out shift register (PISO)**
- **Parallel In parallel Out shift register (PIPO)**

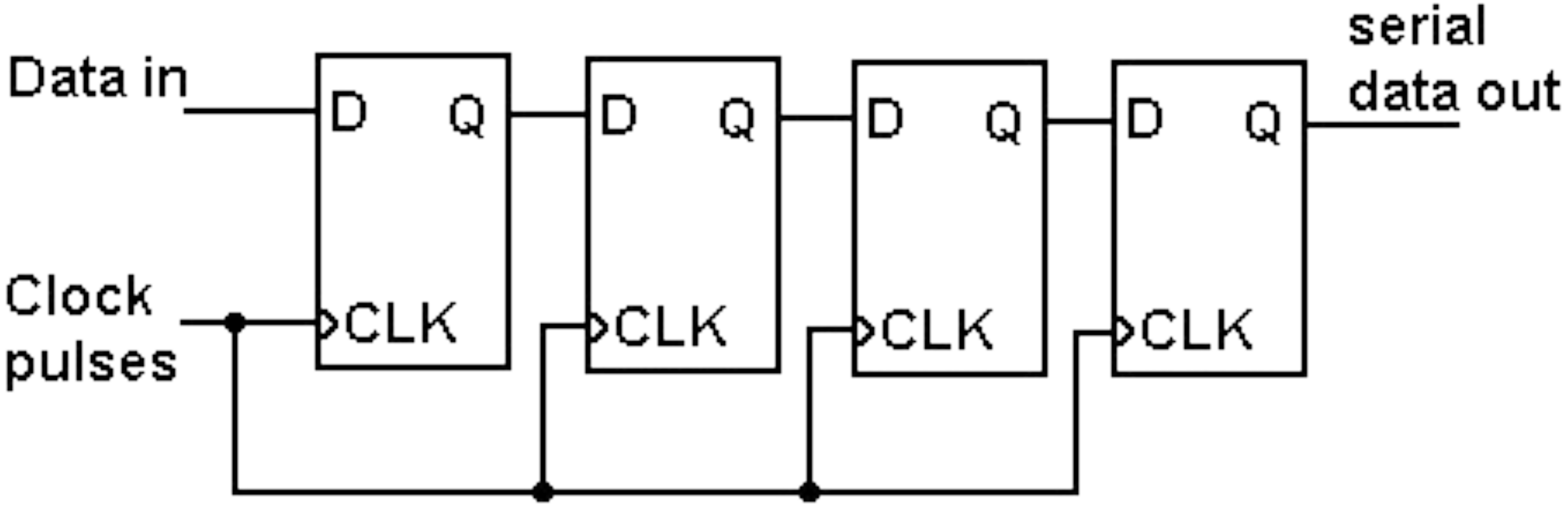
Serial In Serial Out shift register (SISO)

- 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. Since there is only one output, the data leaves the shift register one bit at a time in a serial pattern, thus the name Serial-In Serial-Out Shift Register.
- The logic circuit given below shows a 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.

Serial In Serial Out shift register (SISO)



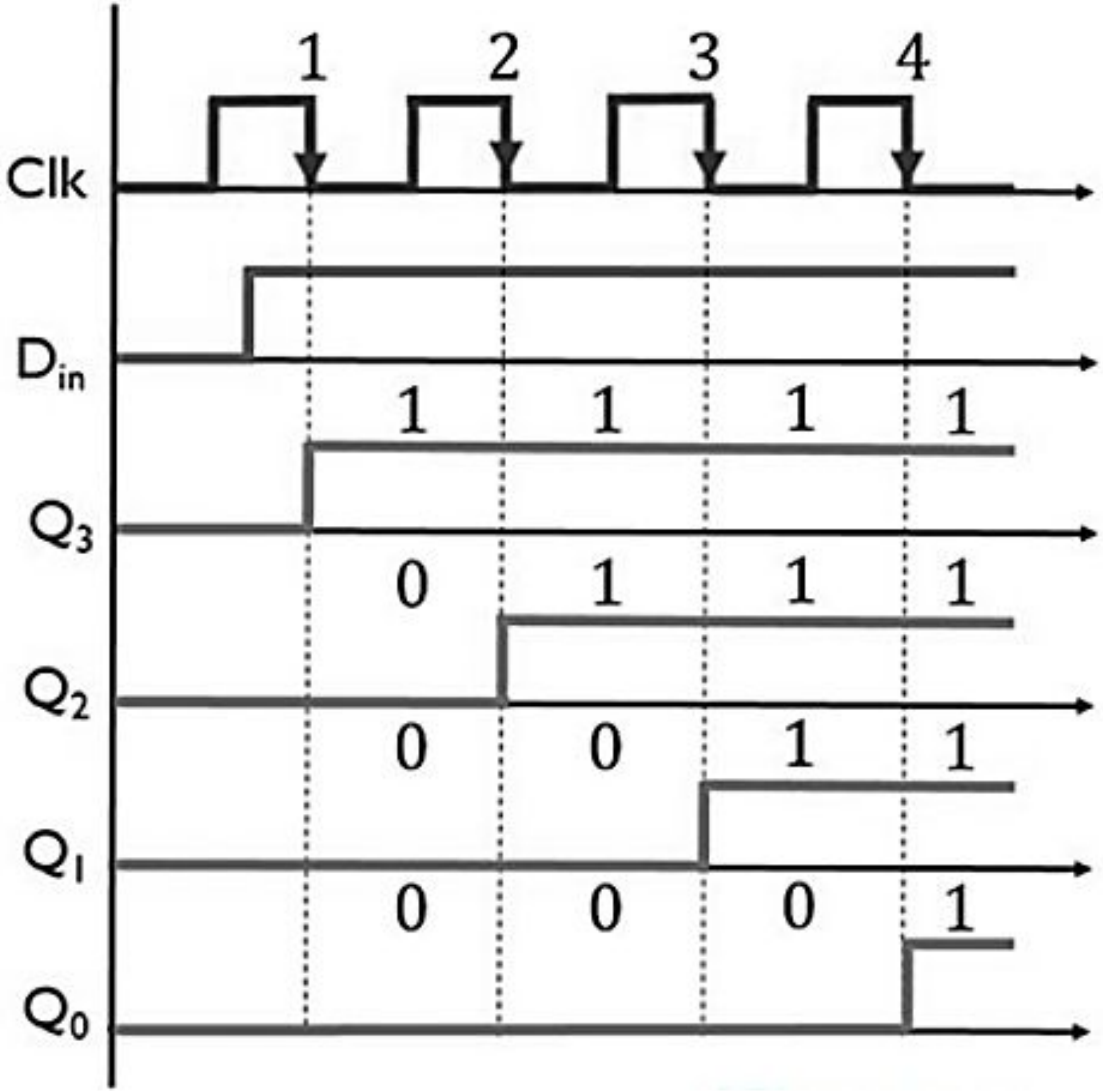
Serial In Serial Out shift register (SISO)



Serial In Serial Out shift register (SISO)

CLK	‘Q3’	‘Q2’	‘Q1’	‘Q0’
Initially (Reset)	0	0	0	0
1st Falling Edge	1	0	0	0
2nd Falling Edge	1	1	0	0
3rd Falling Edge	1	1	1	0
4th Falling Edge	1	1	1	1

Serial In Serial Out shift register (SISO)



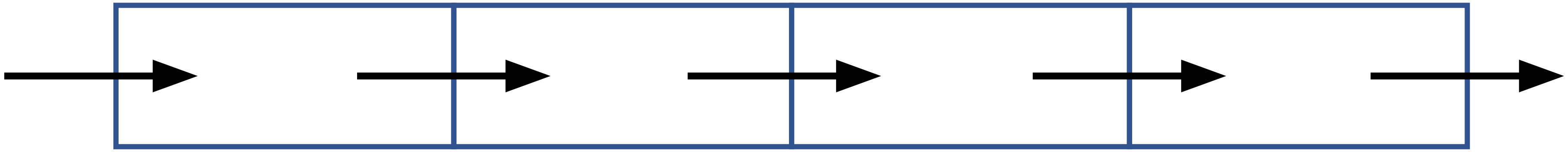
Serial In Serial Out shift register (SISO)

- The SISO shift register applications include the following.

1. The SISO shift register is mainly used to generate time delays in digital logic circuits.
2. These shift registers are used to transfer manipulation and store the data.
3. SISO register is used efficiently to decrease the no. of wires connecting the different systems within the design.
4. SISO shift register delays data through a single CLK time for every stage & they will store a data bit for every register.
5. These types of registers are mainly used especially for time delays.

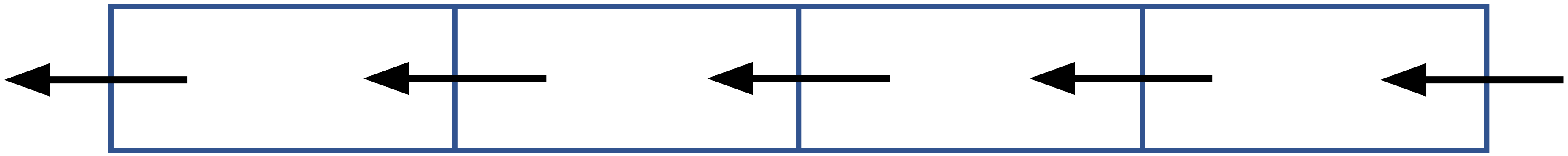
Serial In Serial Out shift register (SISO)

Shift Right Register



Serial In Serial Out shift register (SISO)

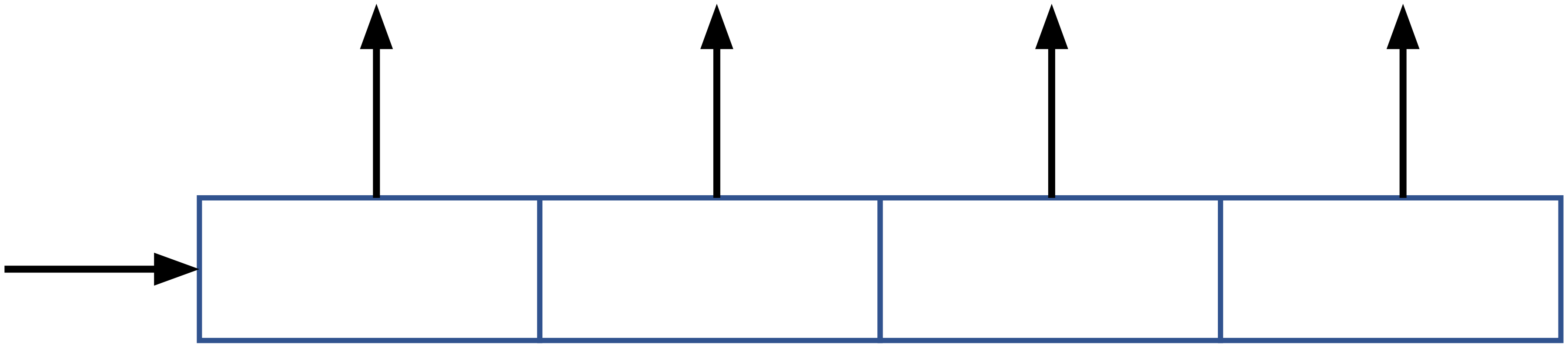
Shift Left Register



Serial In Parallel Out shift register (SIPO)

- A serial-in, parallel-out shift register is similar to the serial-in, serial-out shift register in that it shifts data into internal storage elements and shifts data out at the serial-out, data-out, pin.
- It is different in that it makes all the internal stages available as outputs. Therefore, a serial-in, parallel-out shift register converts data from serial format to parallel format.
- Serial In Parallel Out **(SIPO)** shift register accepts data in serial form and generates output in parallel form. Here data is entered one bit-by-bit serially and once the complete data enters the registers (shifting one bit-by-bit) then output can be obtained in parallel form. i.e., complete data is available at the same time instead of one bit-by-bit.

Serial In Parallel Out shift register (SIPO)

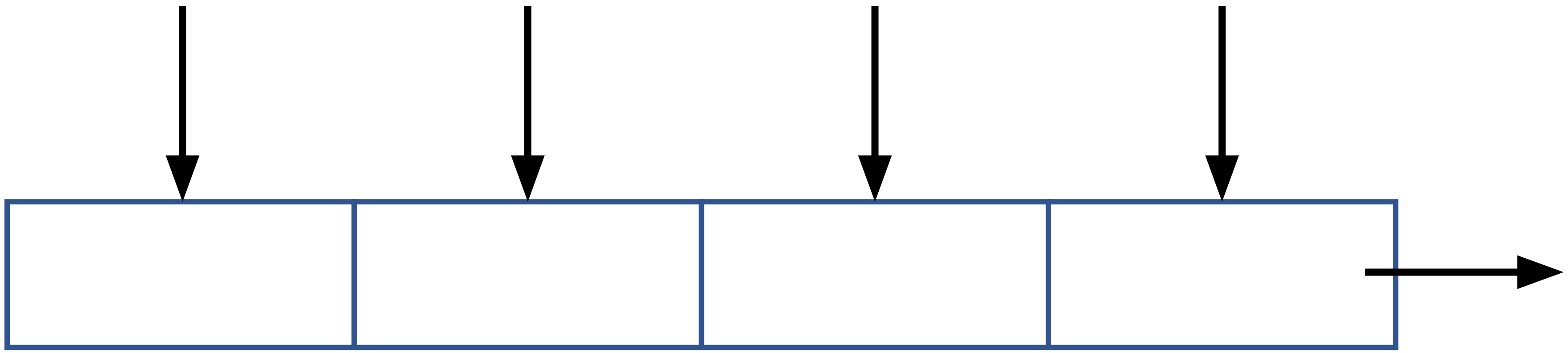


Serial In Parallel Out shift register (SIPO)

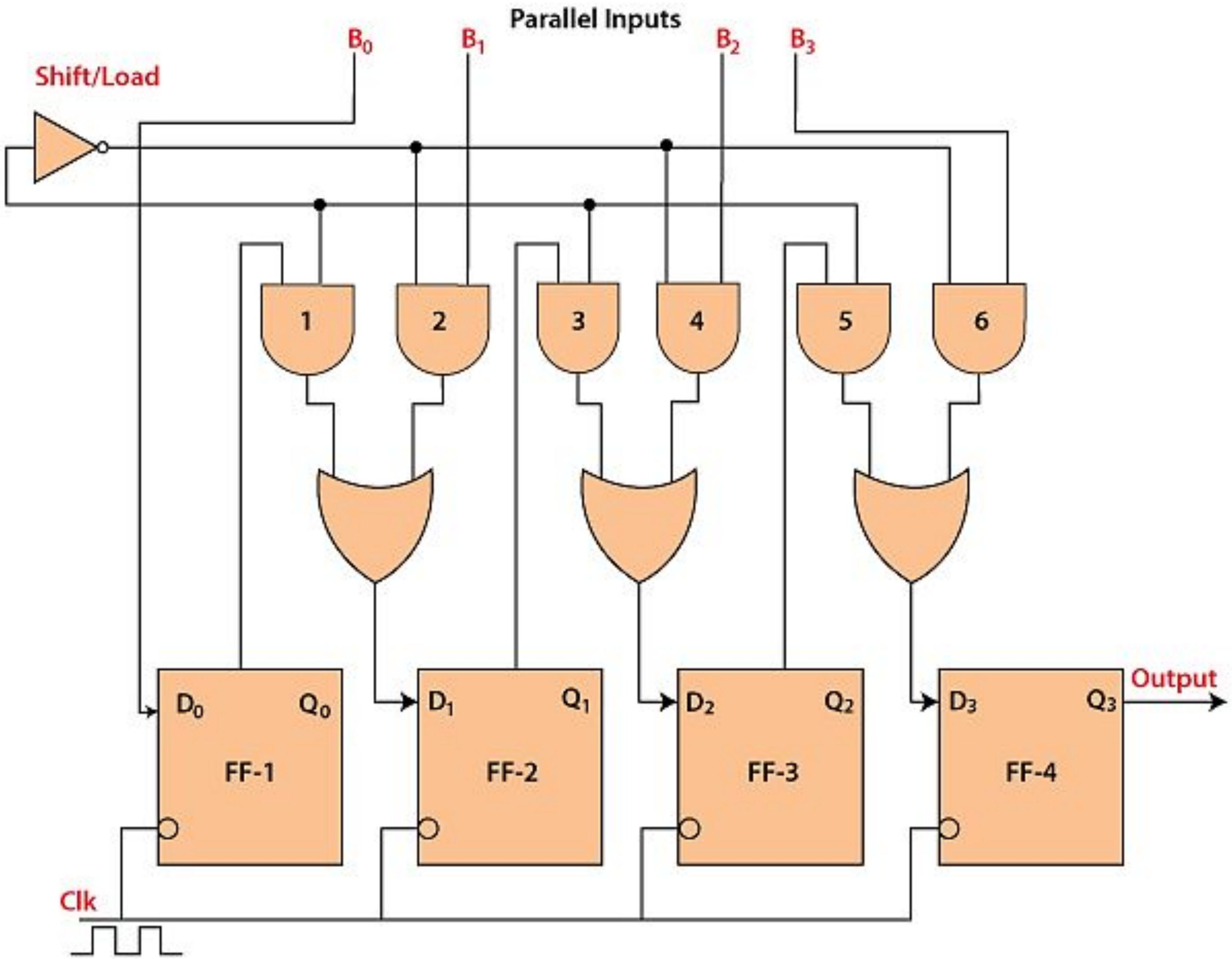
Parallel In Serial Out shift register (PISO)

- The shift register which uses parallel input and generates serial output is known as the parallel input serial output shift register or PISO shift register. This shift register works in a reverse way to the SIPO shift register. In this type of shift register, the input data enters a parallel way and comes out serially. So the i/p of the second FF is the o/p of the first flip flop.

Parallel In Serial Out shift register (PISO)



Parallel In Serial Out shift register (PISO)



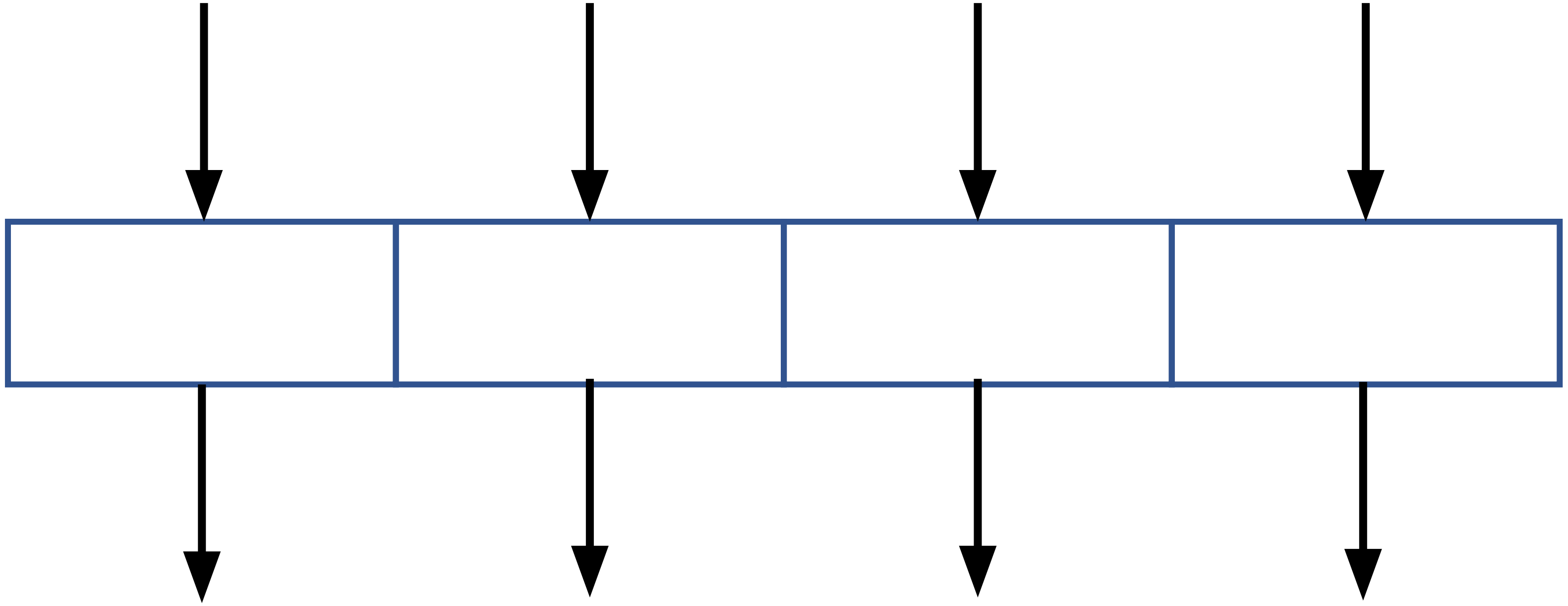
Parallel In Parallel Out shift register (PIPO)

- Parallel In Parallel Out (PIPO) shift registers are the type of storage devices in which both data loading as well as data retrieval processes occur in parallel mode.
- PIPO shift register accepts data in parallel form and generates output in parallel form. This means all the data bits are entered into their respective flip flop simultaneously instead of bit-by-bit.

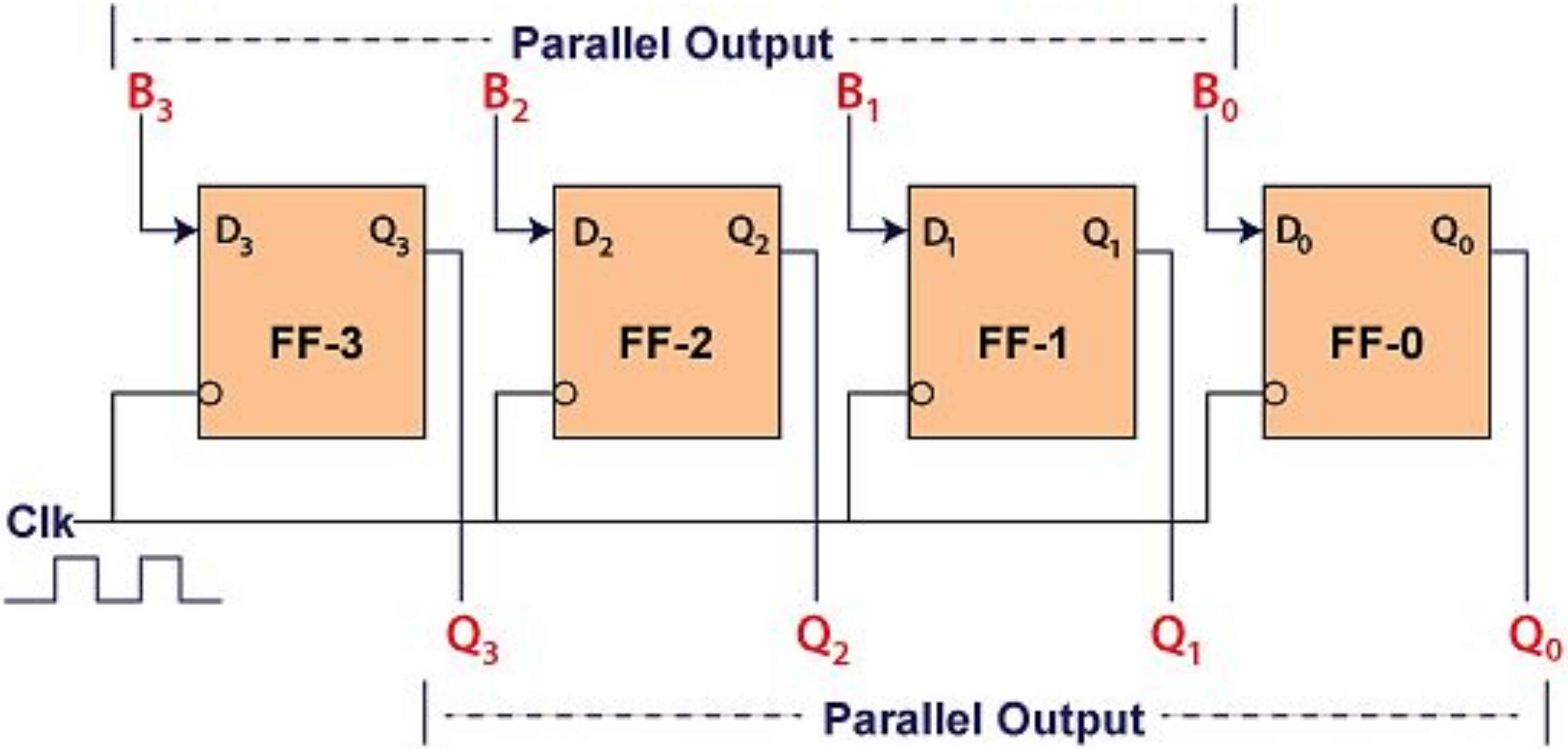
Parallel In Parallel Out shift register (PIPO)

- In "Parallel IN Parallel OUT", the inputs and the outputs come in a parallel way in the register. The inputs A0, A1, A2, and A3, are directly passed to the data inputs D0, D1, D2, and D3 of the respective flip flop. The bits of the binary input is loaded to the flip flops when the negative clock edge is applied. The clock pulse is required for loading all the bits. At the output side, the loaded bits appear.

Parallel In Parallel Out shift register (PIPO)



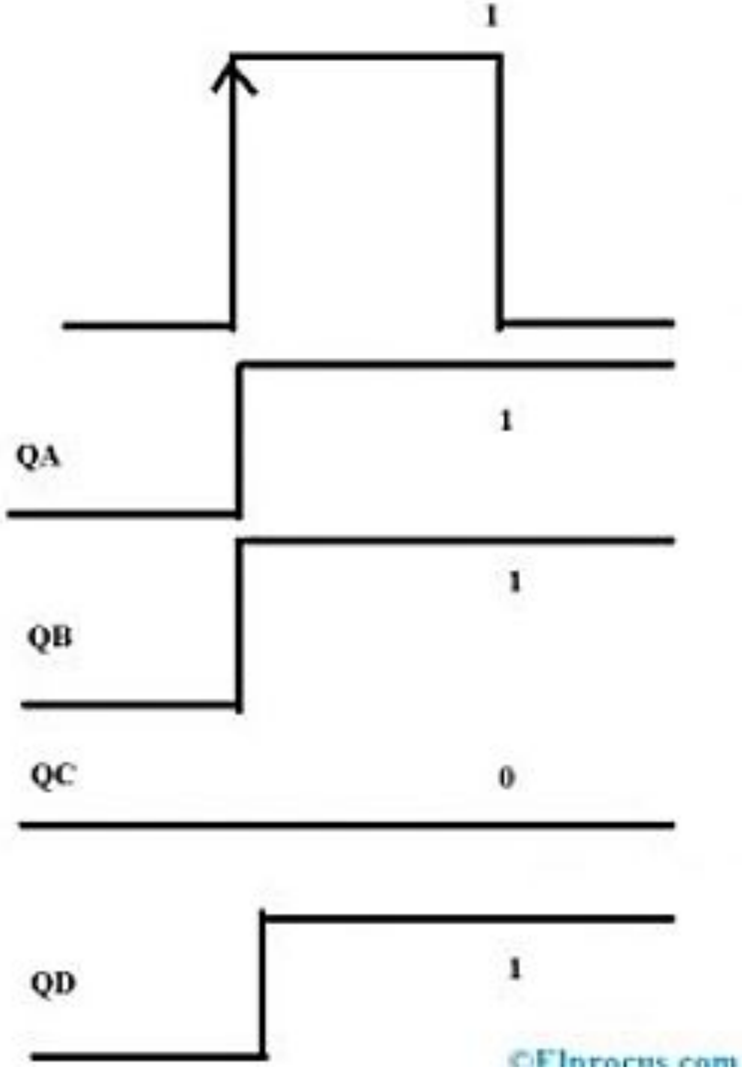
Parallel In Parallel Out shift register (PIPO)



Parallel In Parallel Out shift register (PIPO)

CLK Pulse	QA	QB	QC	QD
0	0	0	0	0
1	1	1	0	1

Parallel In Parallel Out shift register (PIPO)



Parallel In Parallel Out shift register (PIPO)

The advantages of the PIPO shift register include the following.

1. These shift registers are very easy and fast to utilize.
2. They work very fast as compared to logic circuits while converting data.
3. The Sequence number of Pseudo Noise within CDMA is generated through them.
4. It occupies less space and decreases the usage of wiring.
5. Data conversion is easy in shift registers.
6. Its design is very simple.
7. Data encryption & decryption is possible by using shift registers.
8. Data tracking is also possible by using them.

Parallel In Parallel Out shift register (PIPO)

The applications of the PIPO shift register include the following.

1. The PIPO shift register is mainly used to add time delay to digital circuits.
2. Shift registers are used for converting data and also to shift the data from left to right and right to left.
3. It is used for storing the data.
4. These types of shift registers are also used for data storage, manipulation & data transfer.
5. This is a temporary storage device where both data loading & retrieval processes take place within the parallel mode.

Counters

- A Counter is a device which has structure similar to Register. The counter like register also consists of cascaded flip-flops.
- The Register is used only to store the data binary data, the counter is used to count the number of pulses applied at its clock input.
- Counters can also count time interval between any two clock pulses or can be broadly classified as –**synchronous counters, asynchronous/ripple counters and MOD counters.**

Counters

- **Asynchronous/Ripple counter:** The flip-flops are not clocked simultaneously, instead the output of each cascaded flip-flop is connected to clock input of next higher order flip-flop. Thus, the output of each flip-flop drives the next higher order flip-flop in series.
- **This counter is also called serial counter.**
- The first flip-flop is driven by clock pulse and rest all other cascaded flip-flops are driven by output of their previous flip-flop.

Counters – Asynchronous Counters

Various types of asynchronous(ripple) counters are:

- **Asynchronous Binary Up counter**
    i. **2-bit asynchronous binary up counter.**
    ii. **3-bit asynchronous binary up counter.**
    iii. **4-bit asynchronous binary up counter.**

- **Asynchronous Binary down counter**
    i. **2-bit asynchronous binary down counter.**
    ii. **3-bit asynchronous binary down counter.**
    iii. **4-bit asynchronous binary down counter.**

Counters – Asynchronous Counters

- <u>**Asynchronous Binary up/down counter**</u>

i) 3-bit binary up/down counter.

- **<u>BCD ripple counter</u> (Decade counter)**

Counters – Synchronous Counters

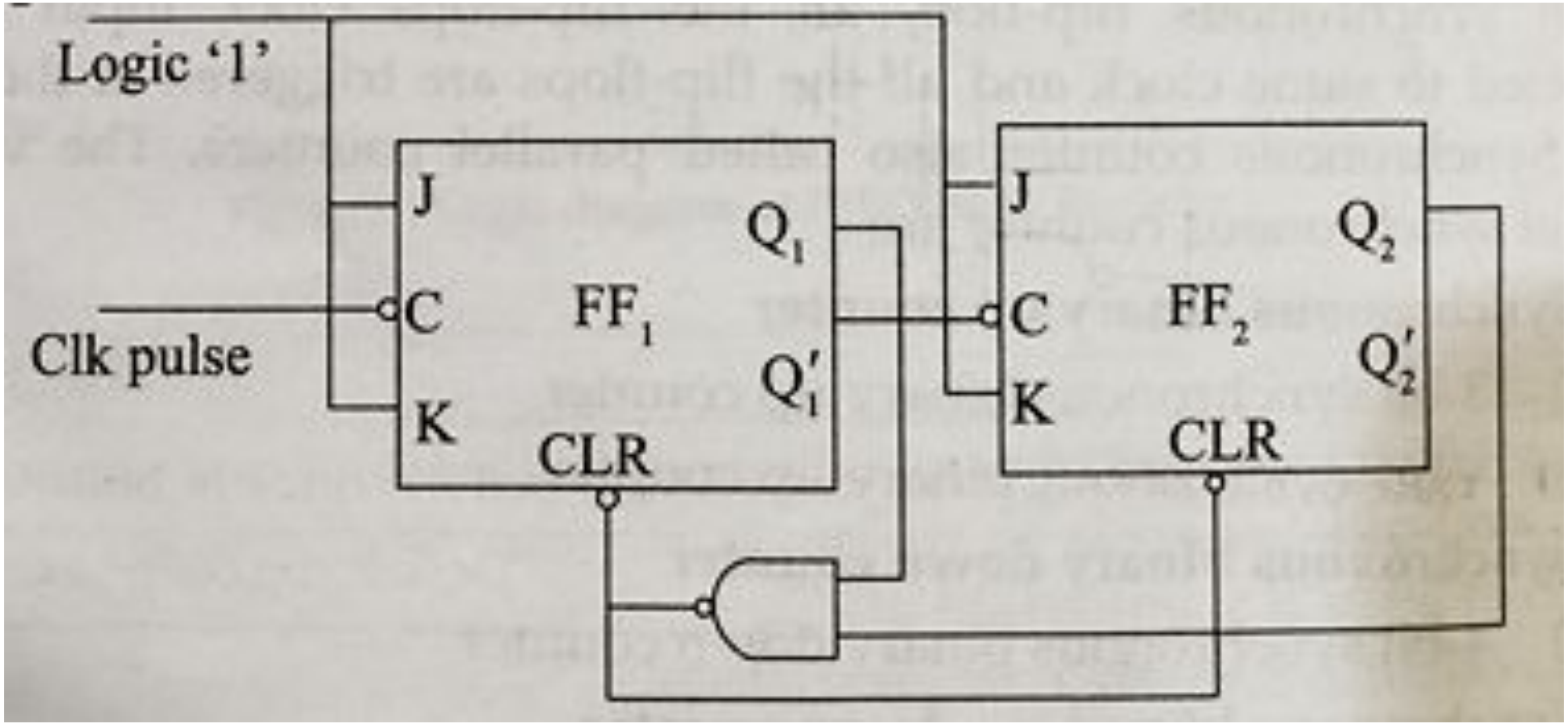
- **Synchronous binary up counter**
  1. 3-bit synchronous binary up counter.
  2. 4-bit synchronous binary up counter.
- **Synchronous binary down counter**
  1. 4-bit synchronous binary down counter.
- **Synchronous binary up down counter**
  1. 3-bit synchronous binary up down counter.

- **BCD Counter**

Counters – MOD-n counter

- MOD-n counter is also called divide-by **n** Counter. These counters are defined on the basis of number of states which a Counter is capable of counting.
- For example, MOD-10 counter can count 10 states. Similarly as MOD-4 counter is a divide-by-4 Counter capable of Counting 4 states.
- The modulus of a counter is the number of unique logic states that counter goes through before the counter comes back to its initial state and start repeating the sequence.
- For example, a **'n'** bit counter has a modulus of **2 to power n**, where 'n' is the number of flip-flops in the counter. Modulus of a counter can be define for both asynchronous and synchronous counter.

Counters – MOD-3 counter

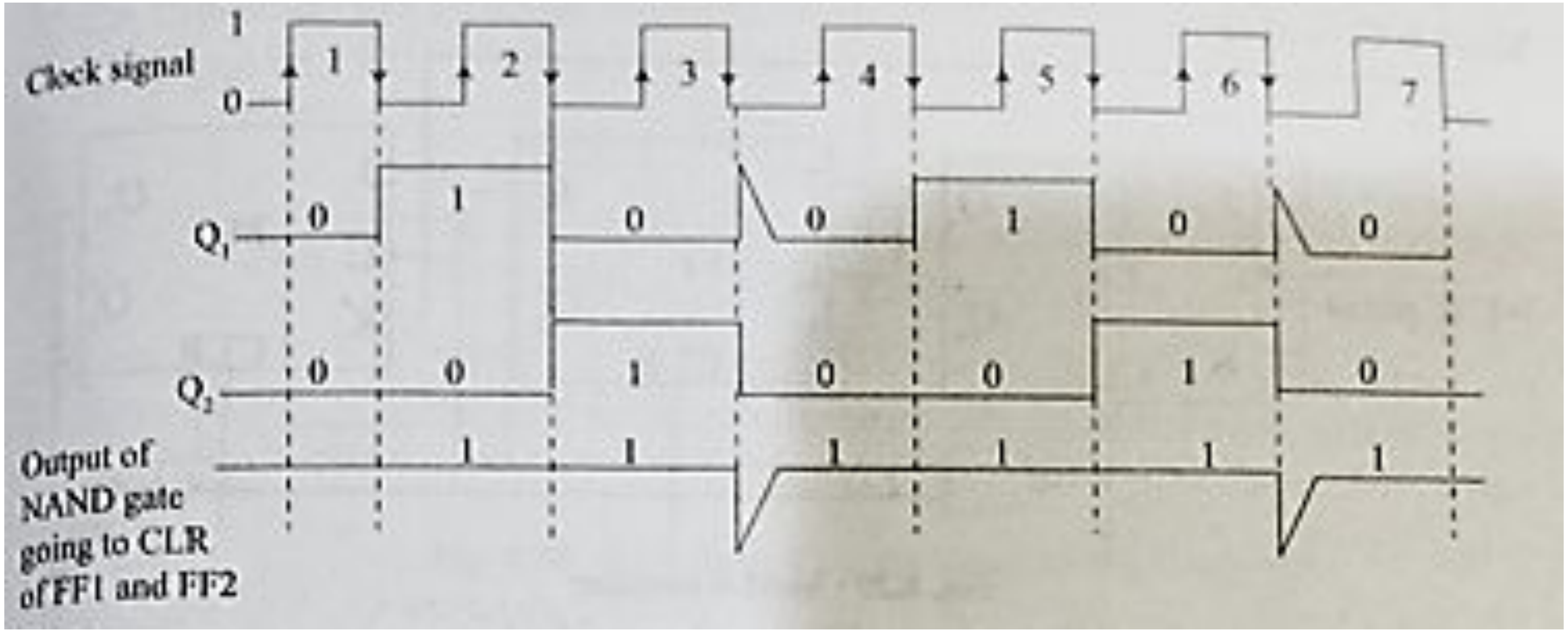


Counters – MOD-3 counter

CLK	Q2	Q1	Description
0	0	0	Initial State
1	0	1	Only **Q1** changes
2	1	0	**Q1** & **Q2** changes
3	0	0	Back to initial State

Counters – MOD-3 counter

Timing Diagram



Counters –

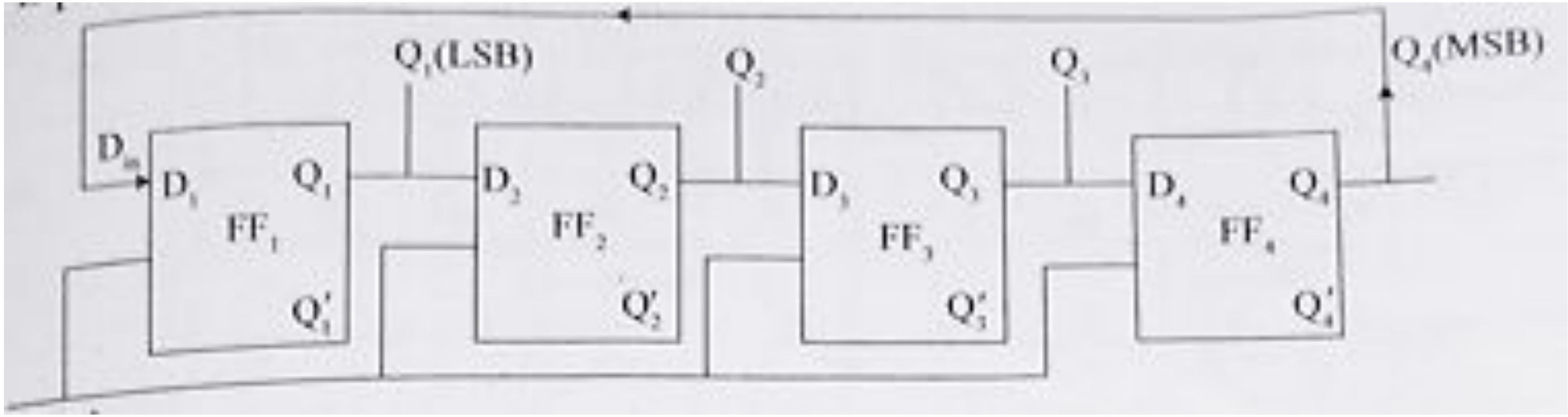
- **Ring Counter**
- **Johnson Counter**

- Both are basically **Registers** but since they generate particular sequences of states they are called **counters**.

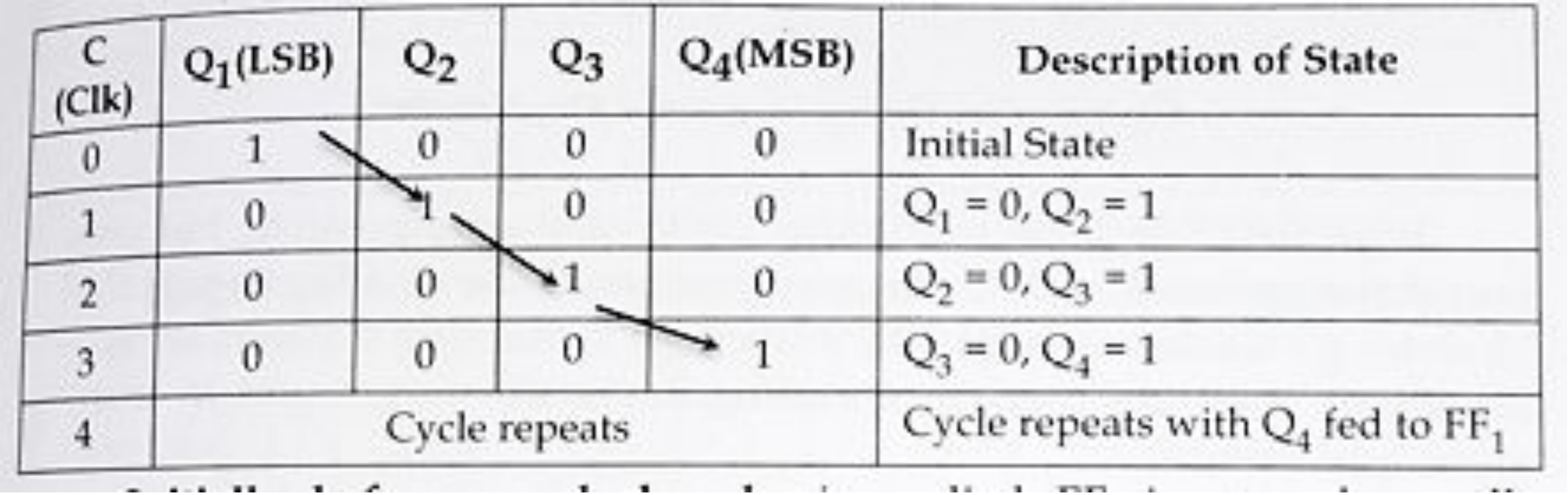
Counters – Ring Counter

- Ring Counter is made by connecting **D flip-flop** in cascaded form similar to Shift Registers.
- All Clock pulses are Synchronous i.e. connected in simultaneously.
- The Output **Q4** is connected to **‘D’** input of **FF1** again.

Counters – Ring Counter

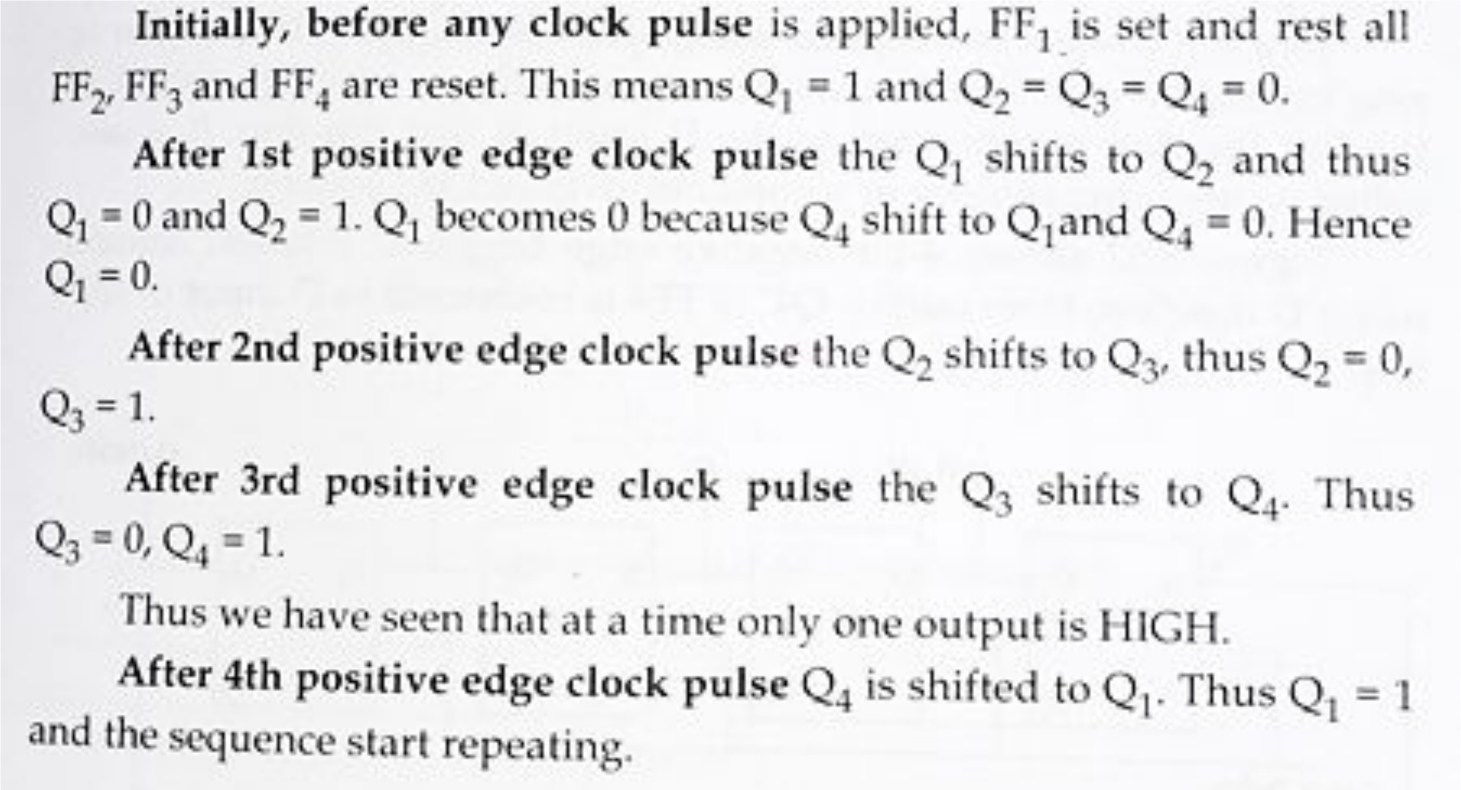


Counters – Ring Counter



C (Clk)	Q_1(LSB)	Q_2	Q_3	Q_4(MSB)	Description of State
0	1	0	0	0	Initial State
1	0	1	0	0	$Q_1 = 0, Q_2 = 1$
2	0	0	1	0	$Q_2 = 0, Q_3 = 1$
3	0	0	0	1	$Q_3 = 0, Q_4 = 1$
4	Cycle repeats				Cycle repeats with Q_4 fed to FF_1

Counters – Ring Counter



Initially, before any clock pulse is applied, FF_1 is set and rest all FF_2, FF_3 and FF_4 are reset. This means $Q_1 = 1$ and $Q_2 = Q_3 = Q_4 = 0$.

After 1st positive edge clock pulse the Q_1 shifts to Q_2 and thus $Q_1 = 0$ and $Q_2 = 1$. Q_1 becomes 0 because Q_4 shift to $Q_1$and $Q_4 = 0$. Hence $Q_1 = 0$.

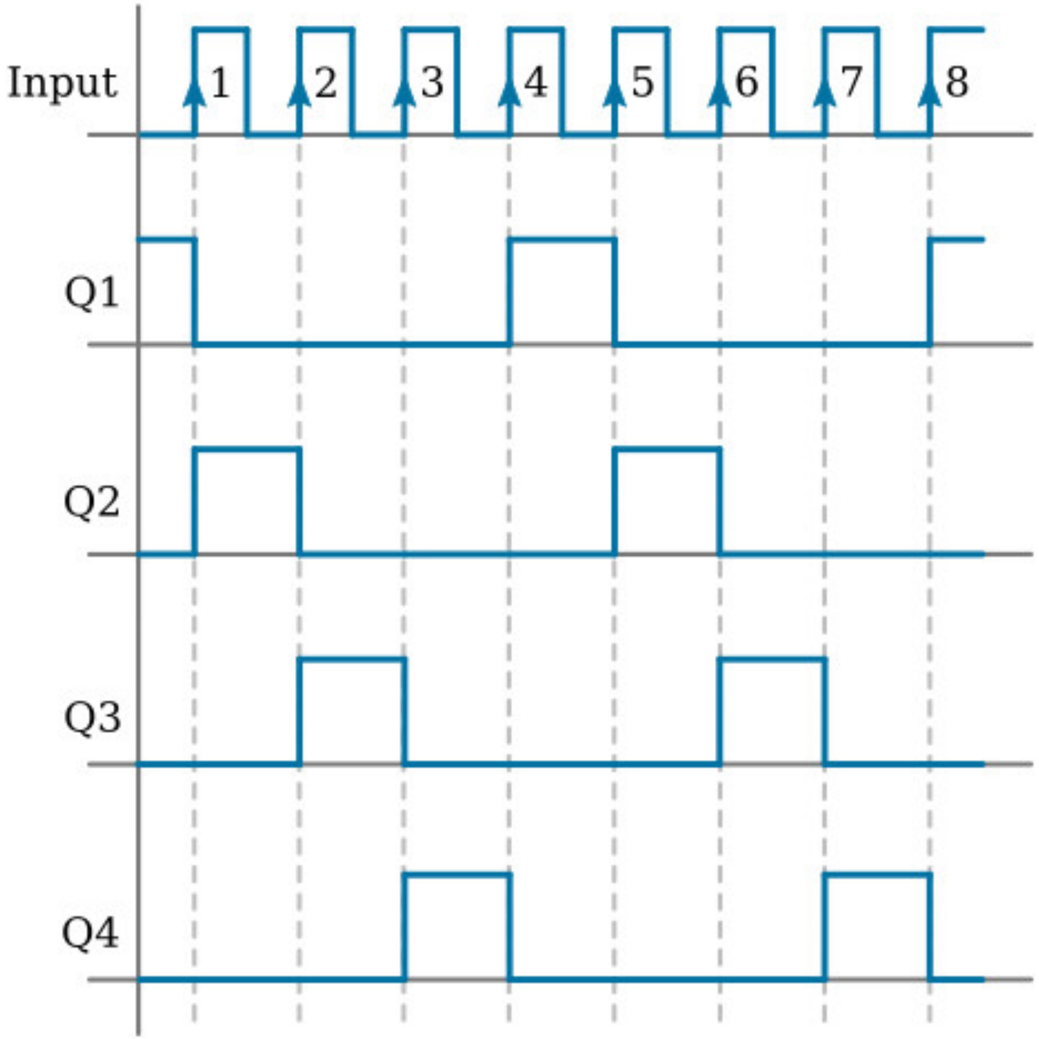
After 2nd positive edge clock pulse the Q_2 shifts to Q_3, thus $Q_2 = 0$, $Q_3 = 1$.

After 3rd positive edge clock pulse the Q_3 shifts to Q_4. Thus $Q_3 = 0$, $Q_4 = 1$.

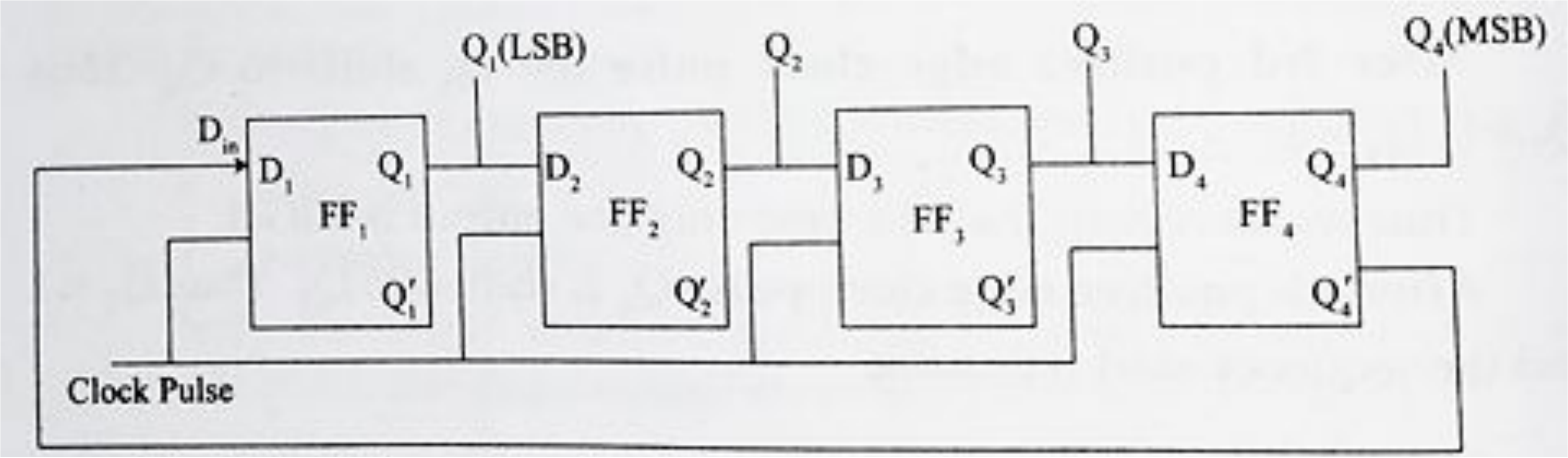
Thus we have seen that at a time only one output is HIGH.

After 4th positive edge clock pulse Q_4 is shifted to Q_1. Thus $Q_1 = 1$ and the sequence start repeating.

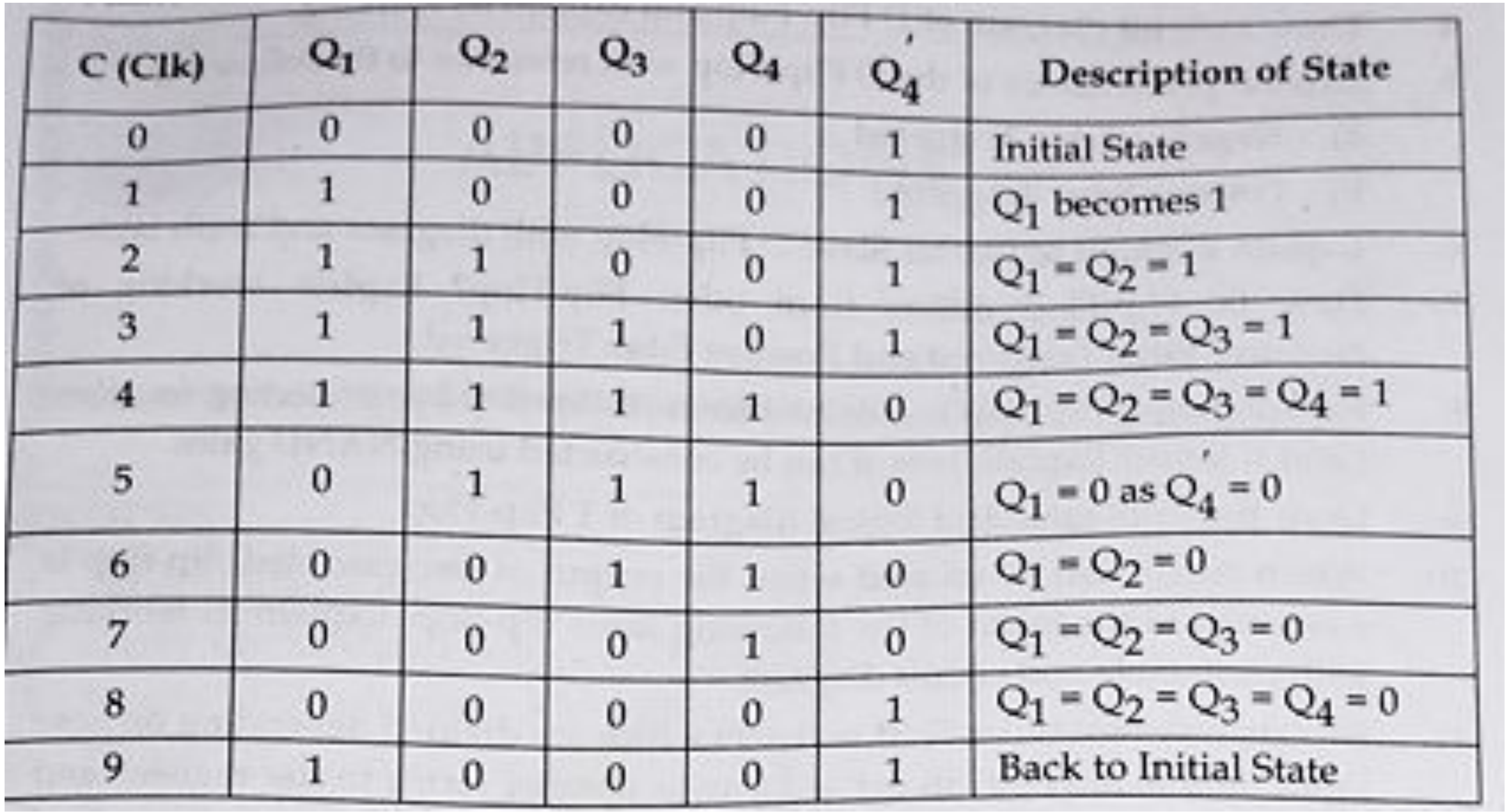
Counters – Ring Counter



Counters – Johnson Counter

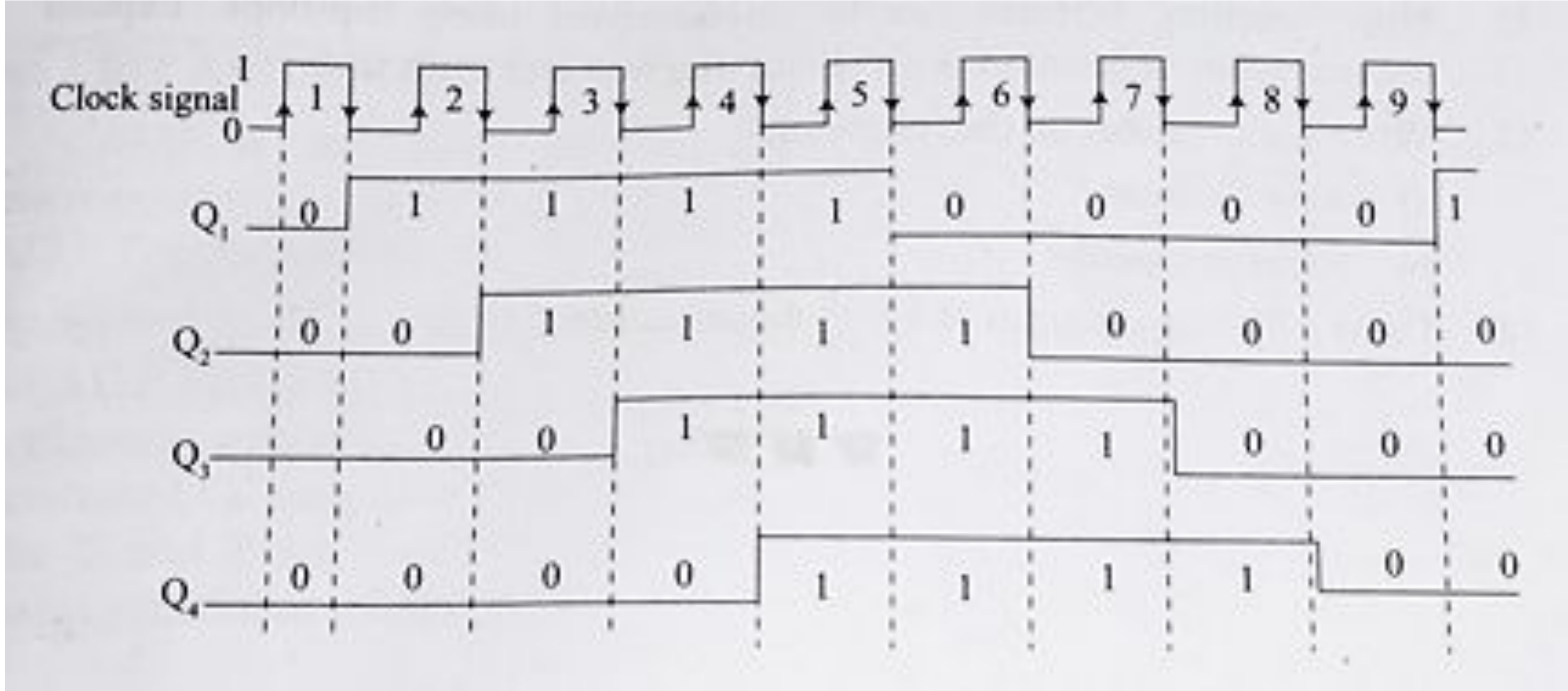


Counters – Johnson Counter



C (Clk)	Q_1	Q_2	Q_3	Q_4	Q_4'	Description of State
0	0	0	0	0	1	Initial State
1	1	0	0	0	1	Q_1 becomes 1
2	1	1	0	0	1	$Q_1 = Q_2 = 1$
3	1	1	1	0	1	$Q_1 = Q_2 = Q_3 = 1$
4	1	1	1	1	0	$Q_1 = Q_2 = Q_3 = Q_4 = 1$
5	0	1	1	1	0	$Q_1 = 0$ as $Q_4' = 0$
6	0	0	1	1	0	$Q_1 = Q_2 = 0$
7	0	0	0	1	0	$Q_1 = Q_2 = Q_3 = 0$
8	0	0	0	0	1	$Q_1 = Q_2 = Q_3 = Q_4 = 0$
9	1	0	0	0	1	Back to Initial State

Counters – Johnson Counter


Clock signal
1
0
1 2 3 4 5 6 7 8 9
Q_1 0 1 1 1 1 0 0 0 0 1
Q_2 0 0 1 1 1 1 0 0 0 0
Q_3 0 0 1 1 1 1 0 0 0
Q_4 0 0 0 0 1 1 1 1 0 0