# Module-4
# Sequential Circuits

Ms. P Binduswetha

Assistant Professor- ECE dept

M.Tech., (Ph.D)

binduswetha.ece@gmail.com

# Contents : Sequential Logic design

- Sequential Circuits & types
- Latches
- Flip-flops
- Excitation tables
- Flip-flop conversions
- Registers
- Shift Registers & its types
- Counters :
- BCD Ripple counters
- Synchronous Counters
- Ring Counters
- Johnson Counter

# Types of Logic Circuits

## Combinational Logic circuits

- Combinational Logic Circuits-outputs depend only on current inputs

## Sequential Logic circuits

- Sequential Logic Circuits-outputs depends not only on current inputs but also on the past sequence of inputs

# Combinational Vs Sequential Circuits

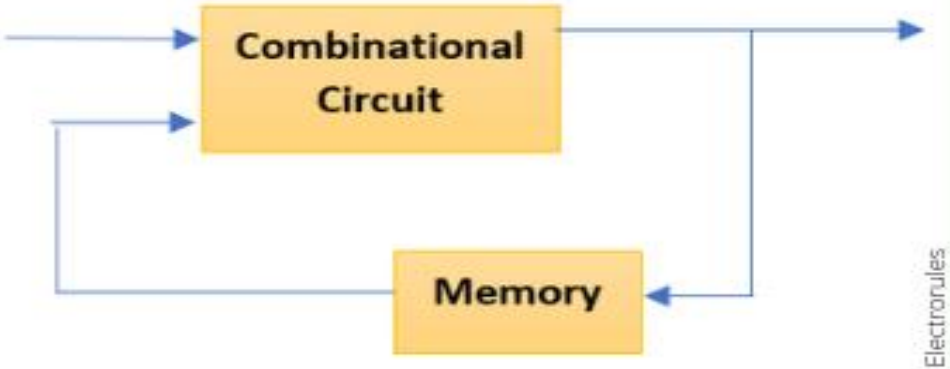
| Combinational Circuit | Sequential Circuit |
| --- | --- |
| Output only depends on the present input | Output depends on present input and past output |
| Memory element is absent | Memory element is present |
| No clock signal is applied | Clock signal is required |
|   |   |
| Example - Half Adder, Full Adder, Multiplexer | Examples - Flipflop, Counters, Registers |

Image Credits: Electronules

# Sequential Circuits

- A combinational circuit produces output solely depending on the current input.
- But a sequential circuit “remembers” its previous state.
- Its output depends on present inputs and previous state.
- Some examples:
  - Latches
  - Registers
  - Memory
  - parallel to serial
  - serial to parallel converters
  - Counters

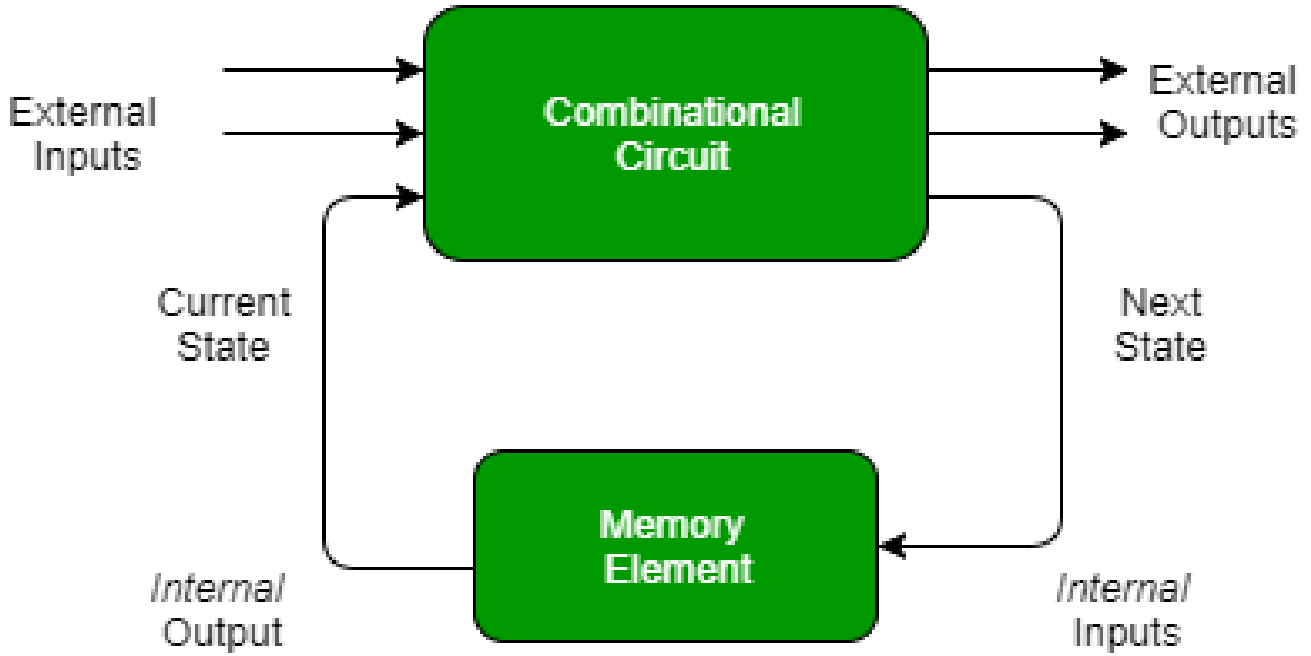
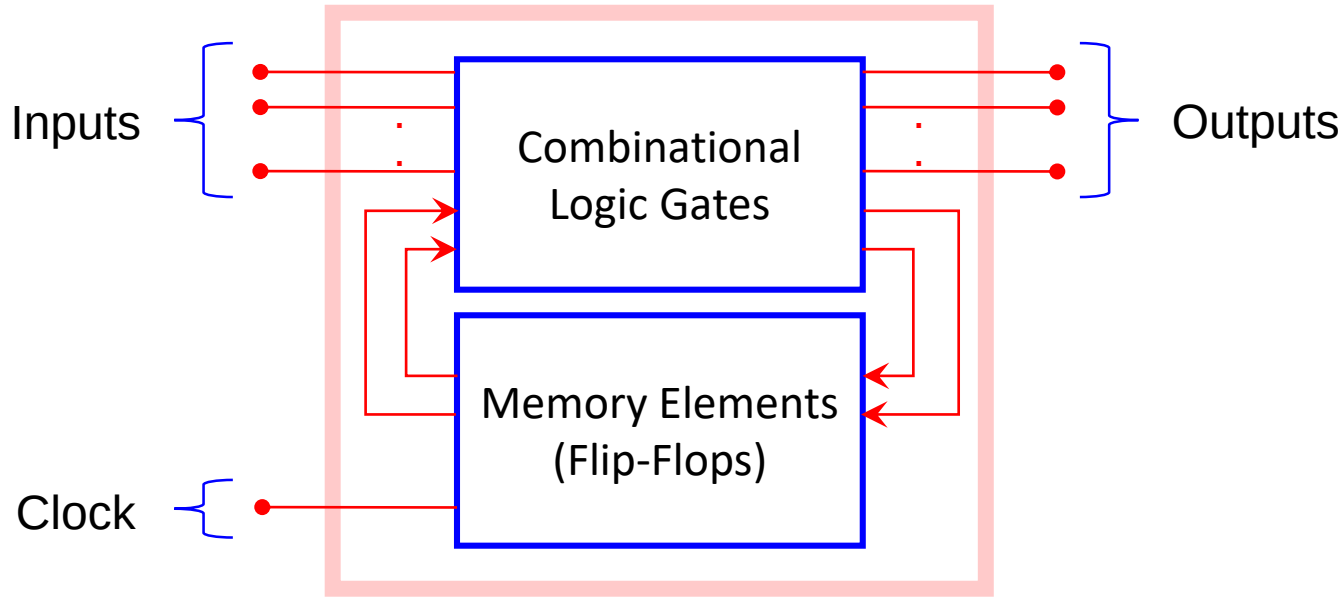



**Figure:** Sequential Circuit

# Sequential Logic circuits & The Flip-Flop




Output depends on present input and past sequence of inputs. Need to remember past history.

The stored data can be changed by applying varying inputs. Flipflops and latches are fundamentally building blocks of digital electronic system. Both are used as data storage elements and it is the basic one in sequential logic.

# Latch vs Flip-flop

- **Latch: Level sensitive device**
  - Positive Latches and Negative latches
  - Can be realized using multiplexers
- **Flip-flop: edge triggered storage element**
  - Can be implemented using latches
  - Cascade a negative latch with a positive latch to obtain a positive edge triggered register
- **Flip flop: bi-stable component formed by the cross coupling of gates.**
- The primary difference between a D flip-flop and D latch is the EN/CLOCK input.
- The flip-flop's CLOCK input is edge sensitive, meaning the flip-flop's output changes on the edge (rising or falling) of the CLOCK input.
- The latch's EN input is level sensitive, meaning the latch's output changes on the level (high or low) of the EN input.

# Latches & Flipflops

- In digital electronics, a latch is one kind of a logic circuit, as a bistable multivibrator.
- It has 2 stable states namely active high & active low.
- It works like a storage device by holding the data through a feedback lane or path.

There are basically 4 main types of latches & flip-flops: SR, D, T & JK. Major differences in these types are the number of inputs they have and how they change state.

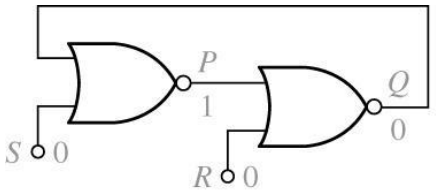
SR flipflop active Low= NAND gates

SR flipflop active HIGH = NOR gates

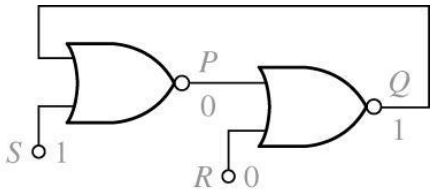
Master-slave edge triggered Flipflos

# S-R Latch

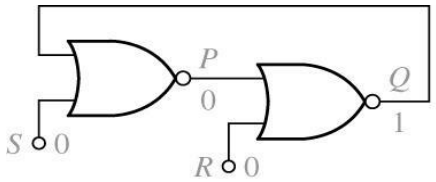
- Set-reset latch
  - Use NOR gate to construct a stable state network



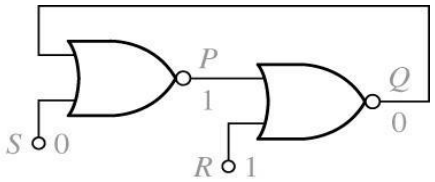

(a)



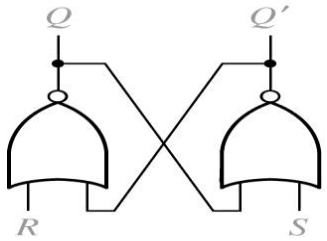

(b)



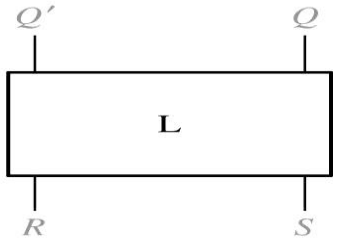

(a)




(b)



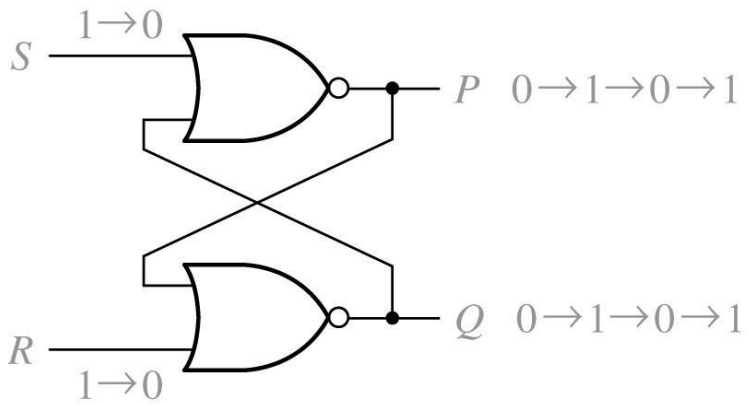

(a)



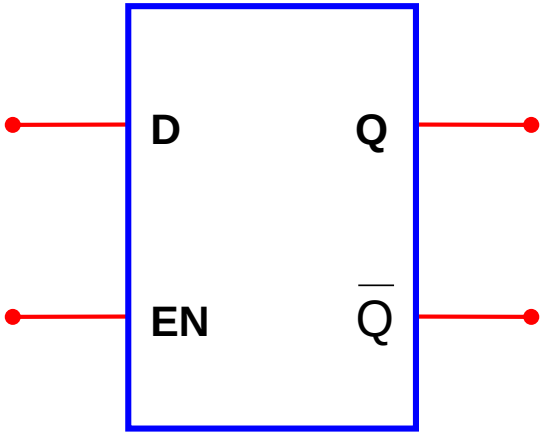

(b)

# S-R Latch (cont.)

- When S=R=1, the S-R latch will not operate properly. (Is it a stable state, if S=R=1?)
  - Q an P are not complementary.
  - If S=R=1 changed to S=R=0, then the network will oscillate assuming both gates have the same delay. (Critical race occurs)



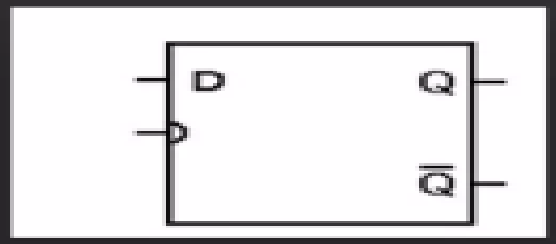
# Transparent D-Latch



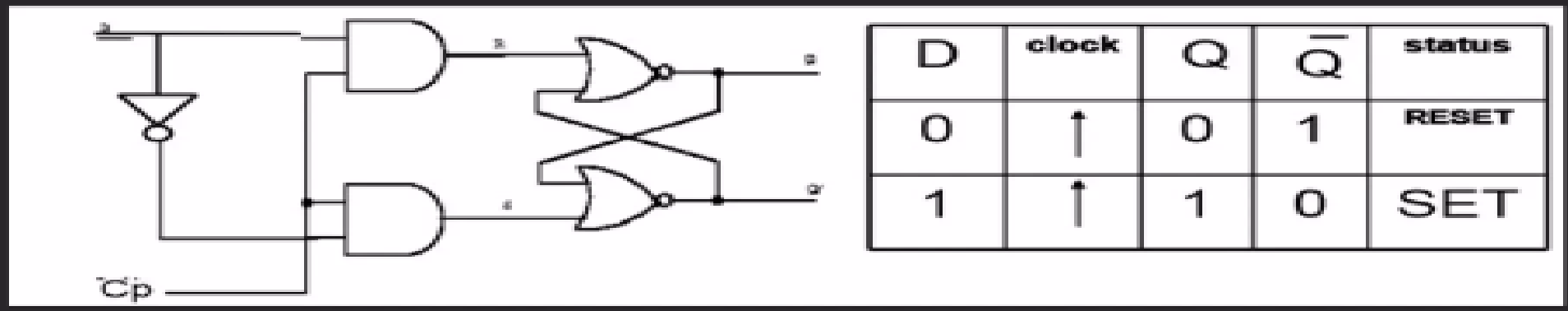

| EN | D | Q | $\overline{Q}$ |
|---|---|---|---|
| 0 | X | $Q_0$ | $\overline{Q}_0$ |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

EN: Enable

# 4. D Flip Flop

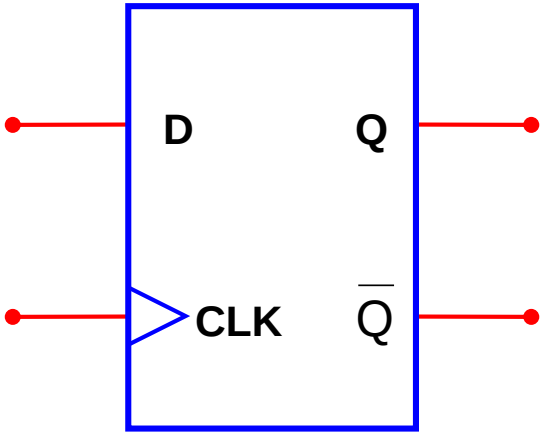







| D | clock | Q | $\overline{Q}$ | status |
|---|---|---|---|---|
| 0 | ↑ | 0 | 1 | RESET |
| 1 | ↑ | 1 | 0 | SET |

# D Flip-Flop: Excitation Table



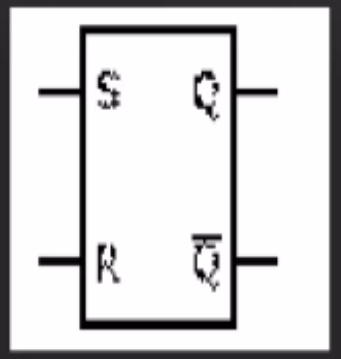

| D | CLK | Q | $\overline{Q}$ |
|---|---|---|---|
| 0 | ↑ | 0 | 1 |
| 1 | ↑ | 1 | 0 |

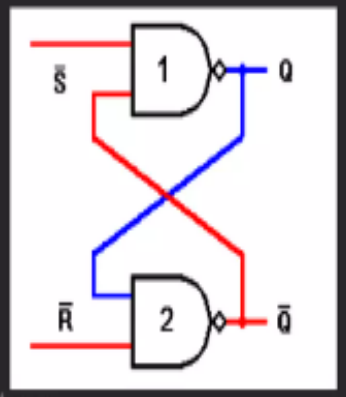
↑ : Rising Edge of Clock

Flip-flop output will go to a state determined by the logic levels present at its synchronous control inputs just prior to the active clock transition.

# 1.SR Flip Flop-

## a.SR Flip Flop Active Low = NAND gates



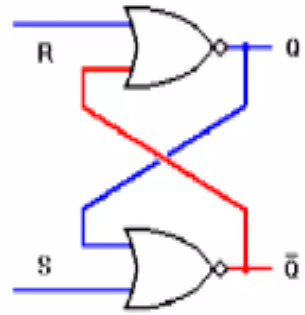

| S | R | Q | $\bar{Q}$ | STATUS |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | INVALID |
| 0 | 1 | 1 | 0 | SET |
| 1 | 0 | 0 | 1 | RESET |
| 1 | 1 | Q | $\bar{Q}$ | HOLD (NoChange) |



# 1.SR Flip Flop-

## a.SR Flip Flop Active High = NOR gates

- NOR GATE LATCH



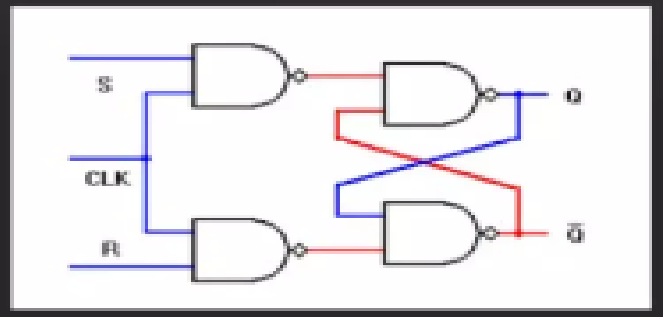

2 Input NOR gate

| A | B | $\overline{A+B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

| S | R | Q | $\bar{Q}$ | STATUS |
|---|---|---|---|---|
| 0 | 0 | Q | $\bar{Q}$ | HOLD (NoChange) |
| 0 | 1 | 0 | 1 | RESET |
| 1 | 0 | 1 | 0 | SET |
| 1 | 1 | 0 | 0 | INVALID |

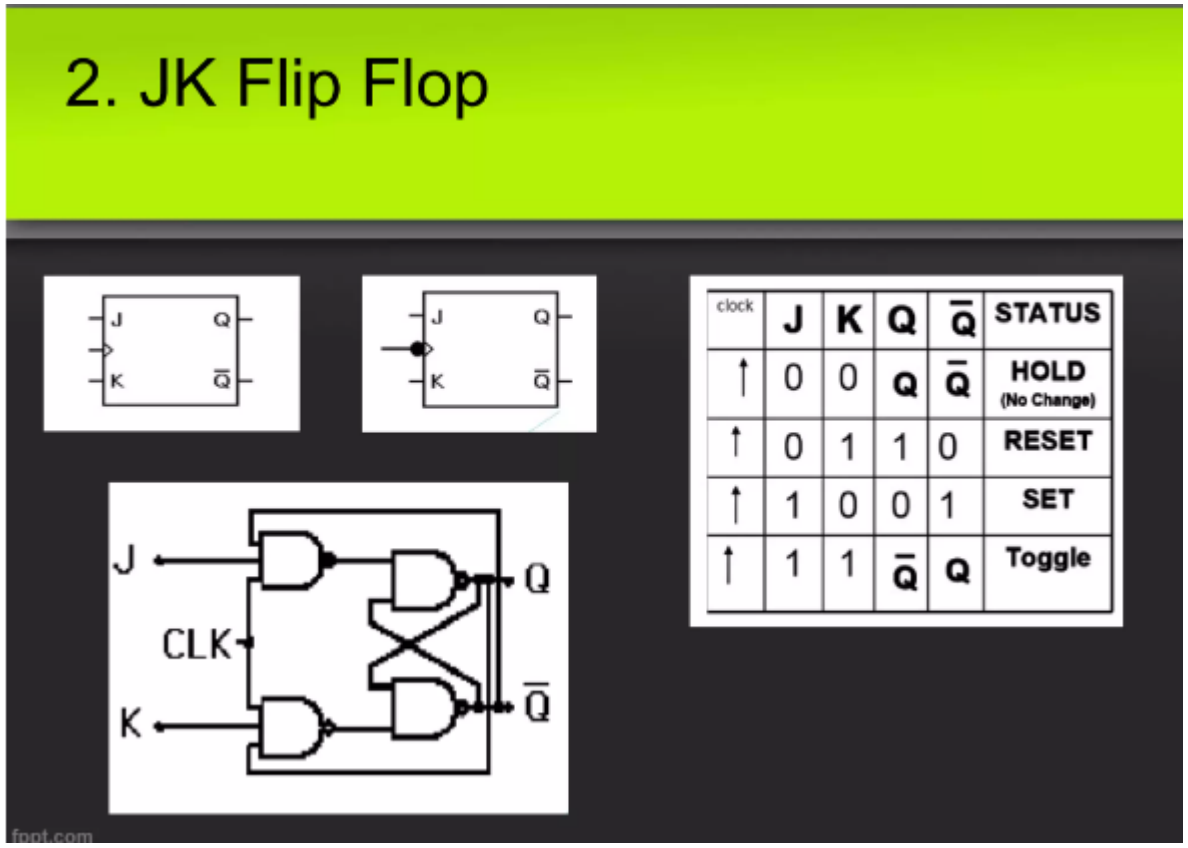
# 1.SR Flip Flop-

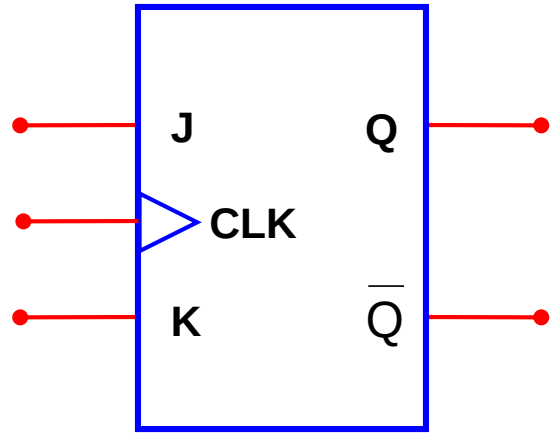
## C. Clocked SR Flip Flop




| S | R | Q | $\bar{Q}$ | STATUS |
|---|---|---|---|---|
| 0 | 0 | Q | $\bar{Q}$ | HOLD (NoChange) |
| 0 | 1 | 0 | 1 | RESET |
| 1 | 0 | 1 | 0 | SET |
| 1 | 1 | 0 | 0 | INVALID |

# J/K Flip-Flop: Excitation Table




| J | K | CLK | Q |
|---|---|---|---|
| 0 | 0 | ↑ | $Q_0$ |
| 0 | 1 | ↑ | 0 |
| 1 | 0 | ↑ | 1 |
| 1 | 1 | ↑ | $\bar{Q}_0$ |

No Change

Clear

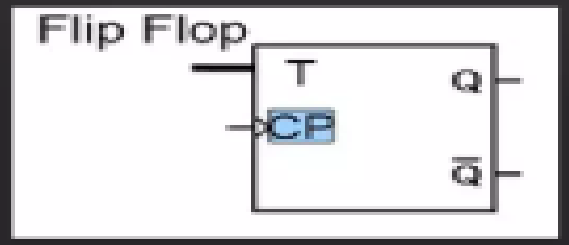
Set

Toggle

↑ : Rising Edge of Clock

$\bar{Q}$ : Complement of Q

# 3. T Flip Flop




| T | clock | Q | $\bar{Q}$ | status |
|---|---|---|---|---|
| 0 | ↑ | Q | $\bar{Q}$ | HOLD |
| 1 | ↑ | $\bar{Q}$ | Q | TOGOL |

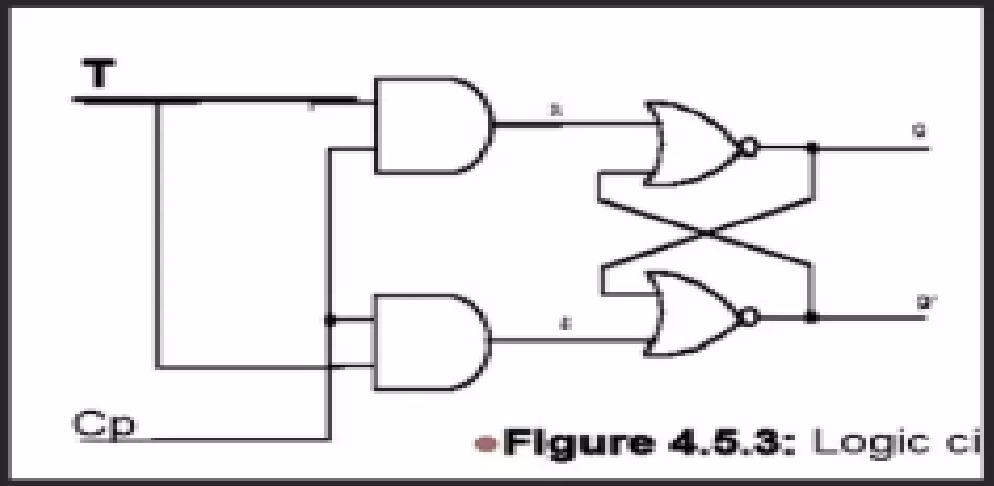
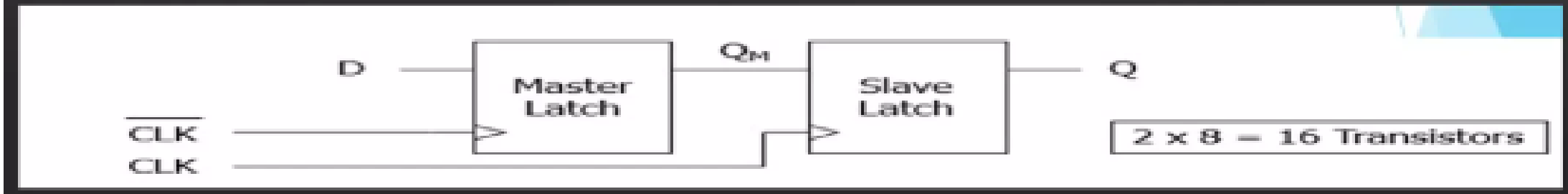
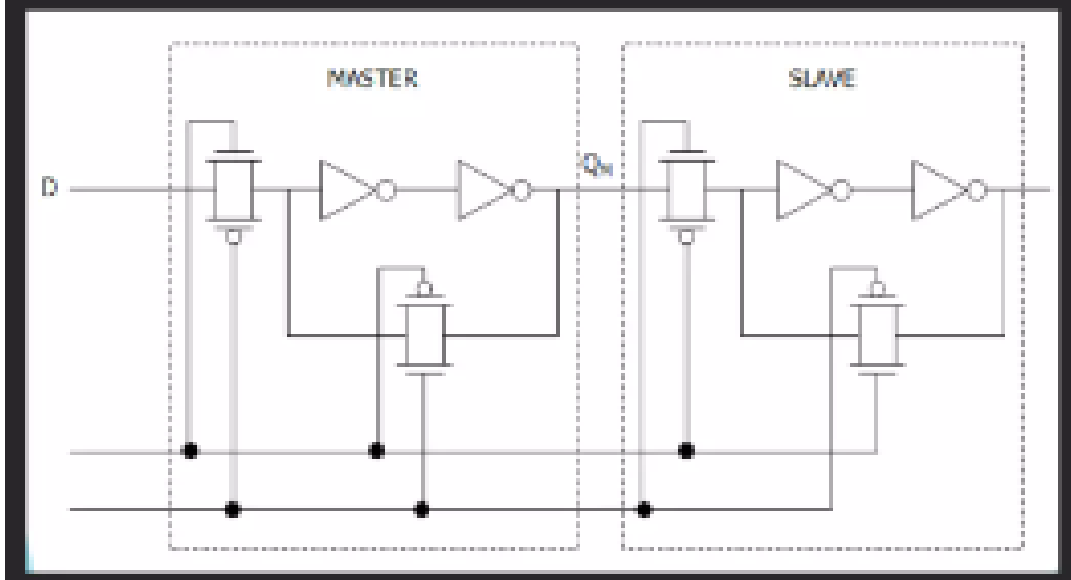
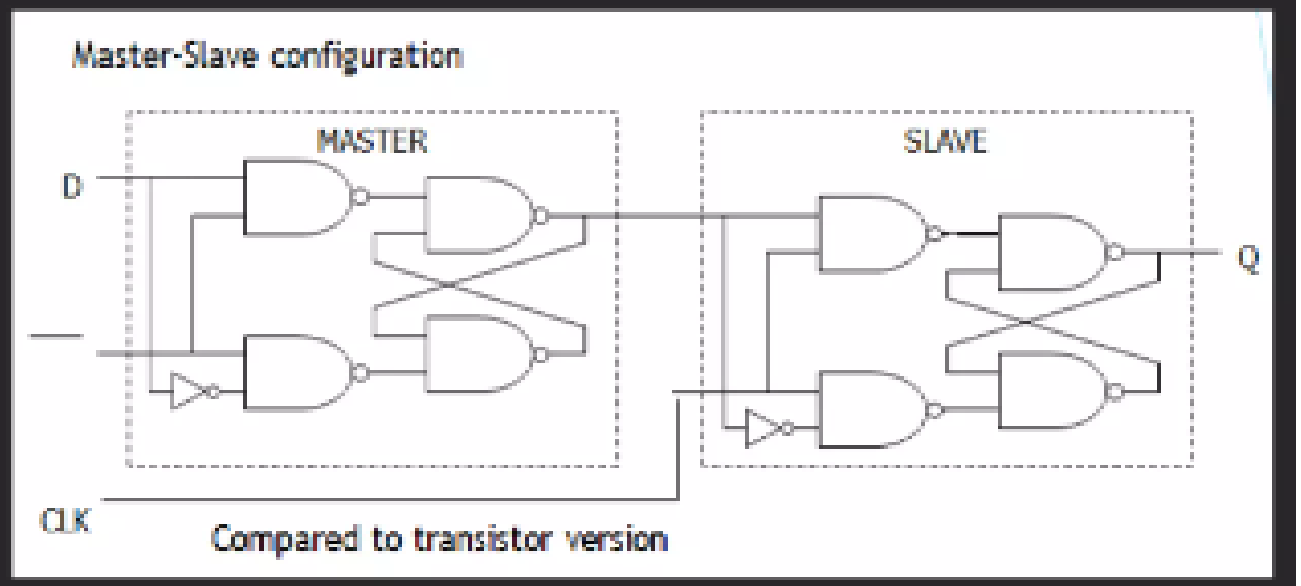



**Figure 4.5.3:** Logic ci

# 5. Master-Slave Edge-Triggered Flip-Flop



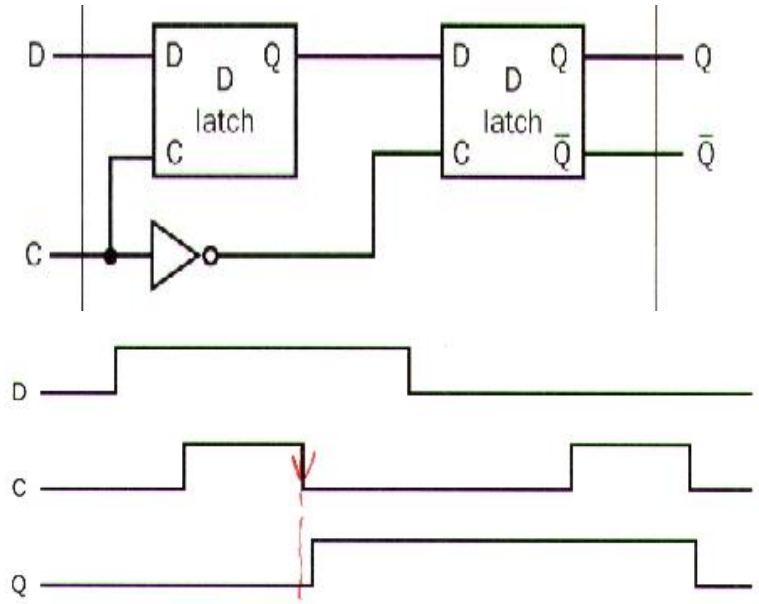



# D Flip Flop with master- slave

- Master latch with a slave latch
  - State changes only at clock edge.
  - Falling edge.

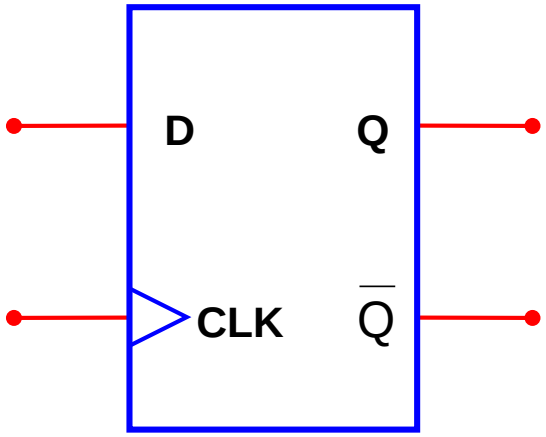
The output of a flipflop will not change state immediately after the application of clk signal or async inputs.

The time interval between the time of application of the triggering edge or async IP and the time at which the output actually makes a transition is called the propagation delay time of the flipflop.



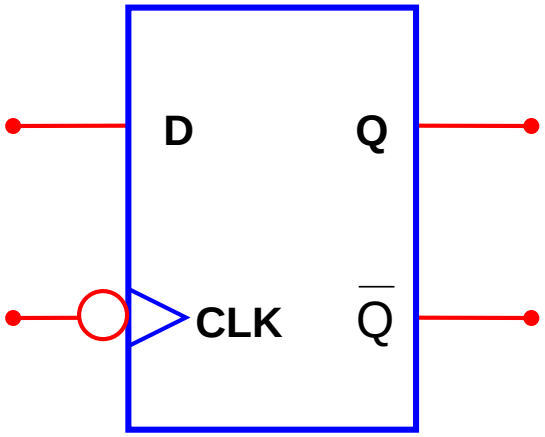
# POS & NEG Edge Triggered D

## Positive Edge Trigger




| D | CLK | Q | $\overline{Q}$ |
|---|---|---|---|
| 0 | ↑ | 0 | 1 |
| 1 | ↑ | 1 | 0 |

↑ : Rising Edge of Clock



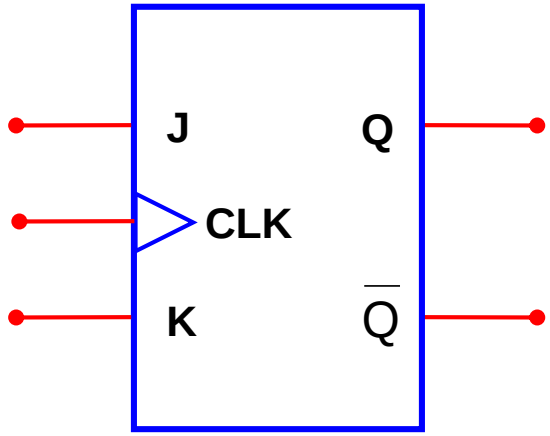

## Negative Edge Trigger

| D | CLK | Q | $\overline{Q}$ |
|---|---|---|---|
| 0 | ↓ | 0 | 1 |
| 1 | ↓ | 1 | 0 |

↓ : Falling Edge of Clock

# POS & NEG Edge Triggered J/K

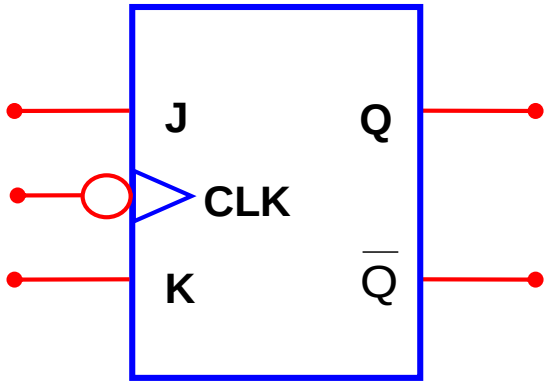
## Positive Edge Trigger




| J | K | CLK | Q |
|---|---|---|---|
| 0 | 0 | ↑ | $Q_0$ |
| 0 | 1 | ↑ | 0 |
| 1 | 0 | ↑ | 1 |
| 1 | 1 | ↑ | $\overline{Q}_0$ |

↑ : Rising Edge of Clock

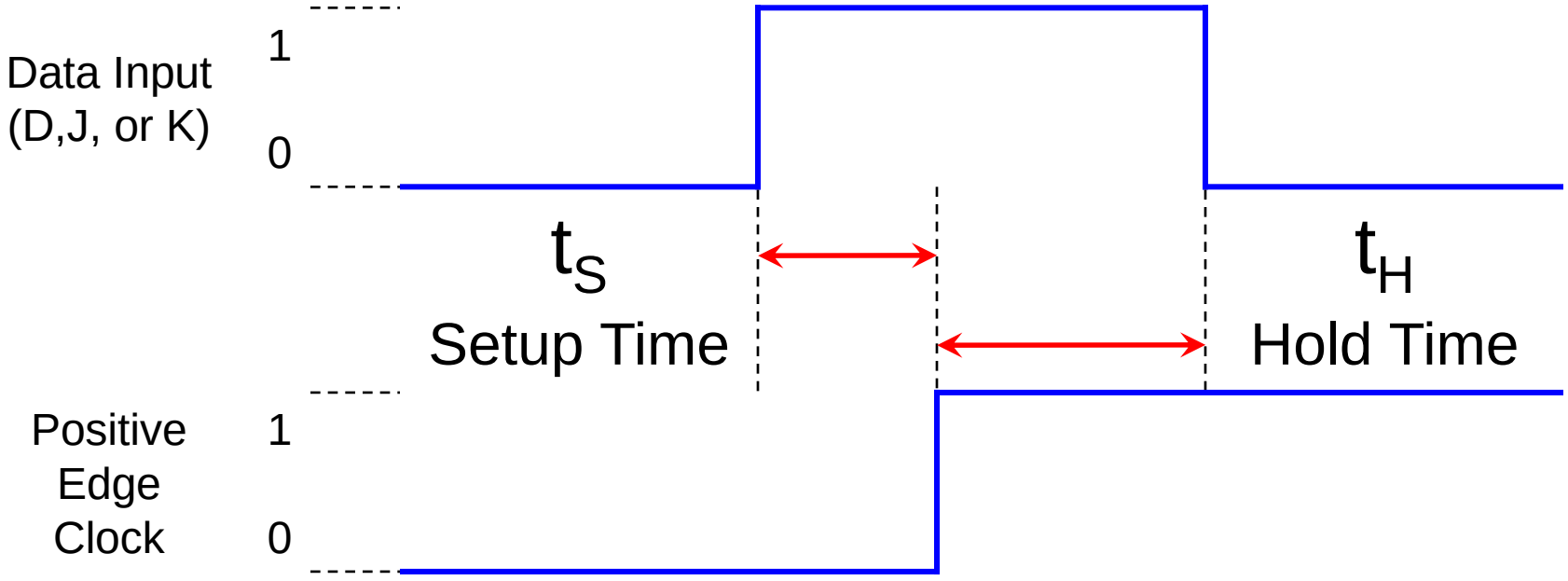
## Negative Edge Trigger




| J | K | CLK | |
|---|---|---|---|
| $Q_0$ | 0 | ↓ | |
| $Q_0$ | 1 | ↓ | 0 |
| 1 | 0 | ↓ | 1 |
| 1 | 1 | ↓ | |

$\overline{Q}_0$↓ : Rising Edge of Clock

# Flip-Flop Timing




**Setup Time ($t_S$)**: The time interval before the active transition of the clock signal during which the data input (D, J, or K) must be maintained.

**Hold Time ($t_H$)**: The time interval after the active transition of the clock signal during which the data input (D, J, or K) must be maintained.

# Asynchronous Inputs

Asynchronous inputs (Preset & Clear) are used to override the clock/data inputs and force the outputs to a predefined state.

The Preset (PR) input forces the output to:

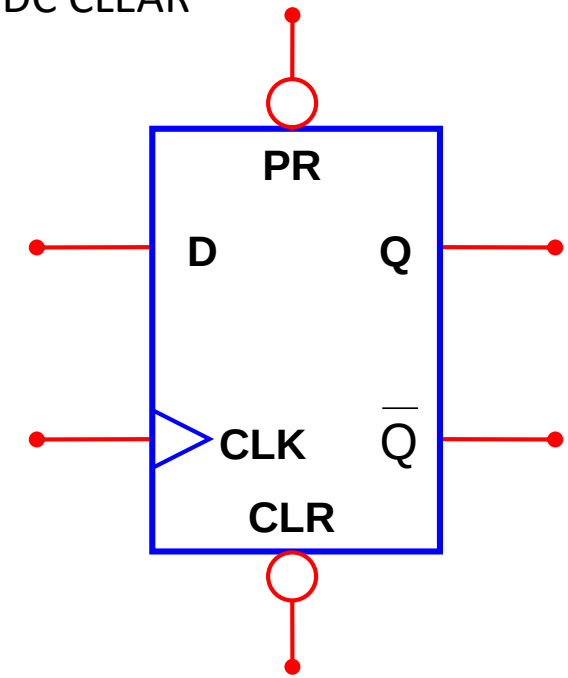
$$Q = 1 \ \& \ \overline{Q} = 0$$

The Clear (CLR) input forces the output to:

$$Q = 0 \ \& \ \overline{Q} = 1$$

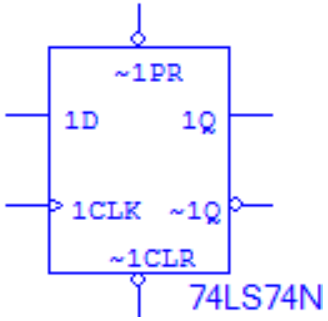
| PR PRESET | CLR CLEAR | CLK CLOCK | D DATA | $Q$ | $\overline{Q}$ | |
|---|---|---|---|---|---|---|
| 1 | 1 | ↑ | 0 | 0 | 1 | |
| 1 | 1 | ↑ | 1 | 1 | 0 | |
| 0 | 1 | X | X | 1 | 0 | Asynchronous Preset |
| 1 | 0 | X | X | 0 | 1 | Asynchronous Clear |
| 0 | 0 | X | X | 1 | 1 | ILLEGAL CONDITION |

Signals : PRESET/ Direct SET Sd/ DC SET and
CLEAR / CLR or direct RESET Rd or DC CLEAR

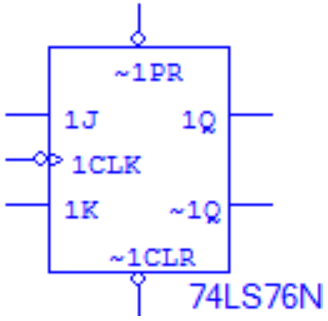



Async inputs are override inputs, used to override all other inputs in order to place flipflop in one state or other.

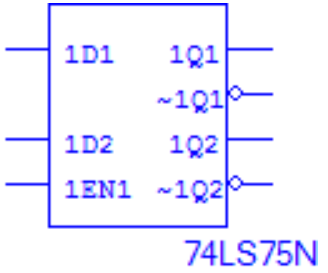
# Flip-Flops & Latches




74LS74
Dual Positive-Edge-Triggered D Flip-Flops with Preset, Clear, and Complementary Outputs

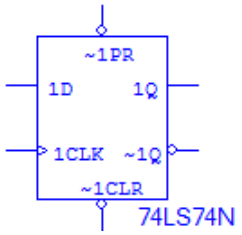



74LS76
Dual Negative-Edge-Triggered J-K Flip-Flops with Preset, Clear, and Complementary Outputs




74LS75
Quad Latch

# 74LS74: D Flip-Flop



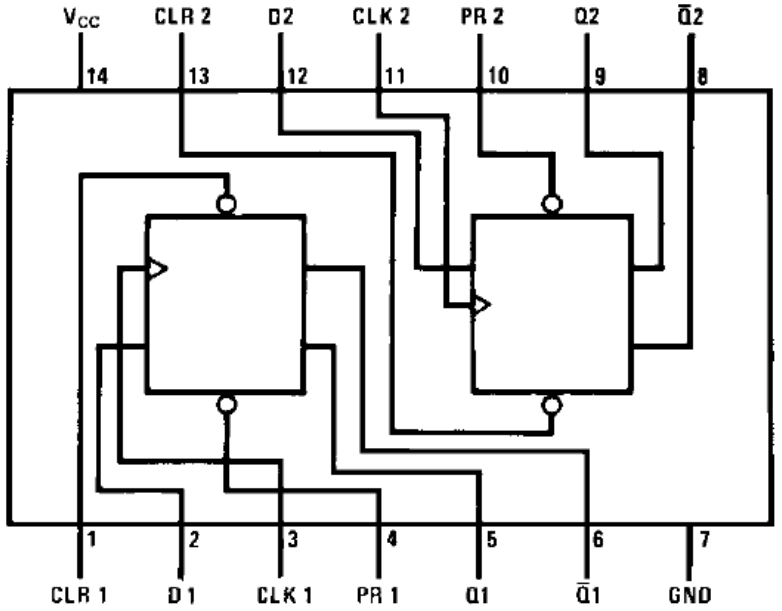

## Function Table

| Inputs | | | | Outputs | |
|---|---|---|---|---|---|
| PR | CLR | CLK | D | Q | $\overline{Q}$ |
| L | H | X | X | H | L |
| H | L | X | X | L | H |
| L | L | X | X | H (Note 1) | H (Note 1) |
| H | H | ↑ | H | H | L |
| H | H | ↑ | L | L | H |
| H | H | L | X | $Q_0$ | $\overline{Q}_0$ |

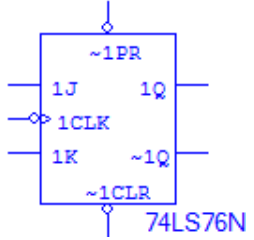
H = HIGH Logic Level
X = Either LOW or HIGH Logic Level
L = LOW Logic Level
↑ = Positive-going Transition
$Q_0$ = The output logic level of Q before the indicated input conditions were established.

**Note 1:** This configuration is nonstable; that is, it will not persist when either the preset and/or clear inputs return to their inactive (HIGH) level.

## Connection Diagram



# 74LS76: J/K Flip-Flop




## Function Table

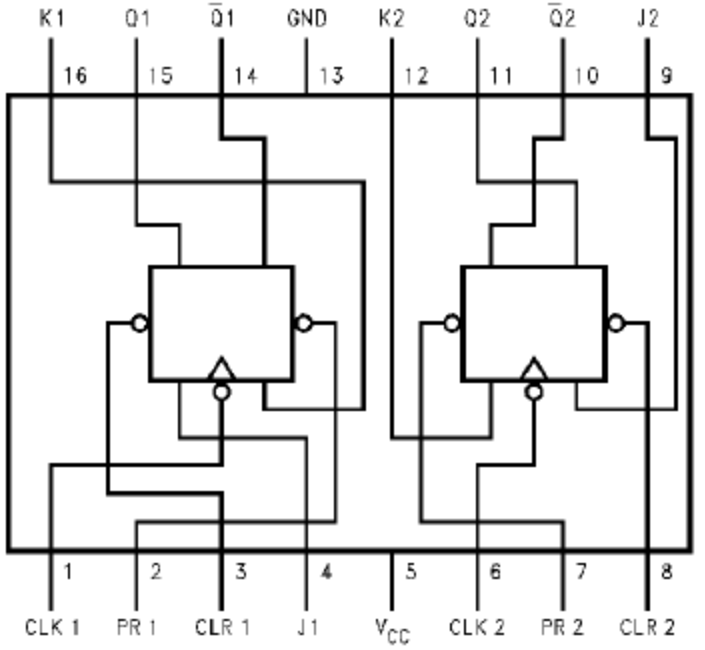
| Inputs | | | | | Outputs | |
|---|---|---|---|---|---|---|
| PR | CLR | CLK | J | K | Q | $\overline{Q}$ |
| L | H | X | X | X | H | L |
| H | L | X | X | X | L | H |
| L | L | X | X | X | H (Note 1) | H (Note 1) |
| H | H | ⎍ | L | L | $Q_0$ | $\overline{Q}_0$ |
| H | H | ⎍ | H | L | H | L |
| H | H | ⎍ | L | H | L | H |
| H | H | ⎍ | H | H | Toggle | |

H = High Logic Level
L = Low Logic Level
X = Either Low or High Logic Level
⎍ = Positive pulse data. The J and K inputs must be held constant while the clock is high. Data is transfered to the outputs on the falling edge of the clock pulse.
$Q_0$ = The output logic level before the indicated input conditions were established.
Toggle = Each output changes to the complement of its previous level on each complete active high level clock pulse.

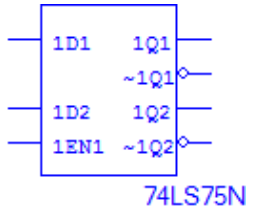
**Note 1:** This configuration is nonstable; that is, it will not persist when the preset and/or clear inputs return to their inactive (high) level.

## Connection Diagram

Dual-In-Line Package



# 74LS75: D Latch

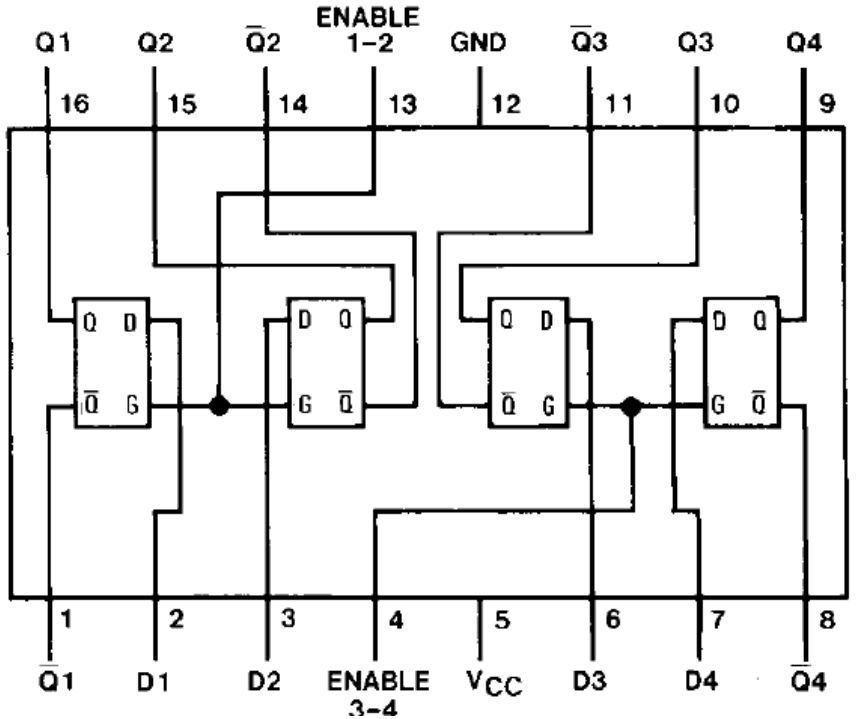



## Function Table (Each Latch)

| Inputs | | Outputs | |
|---|---|---|---|
| D | Enable | Q | $\overline{Q}$ |
| L | H | L | H |
| H | H | H | L |
| X | L | $Q_0$ | $\overline{Q}_0$ |

H = HIGH Level
L = LOW Level
X = Don't Care
$Q_0$ = The Level of Q Before the HIGH-to-LOW Transition of ENABLE

## Connection Diagram



# Flip-flop tables & equations

- Truth table : It describes how the outputs are obtained for given input combinations.
- Characteristic table:  It shows the next state when input and present states are known.
- Excitation table:  It explains the input required to get given combination of present and next state.

# Asynchronous inputs

- Why makes it a synchronous input?
  - Used to change state of a system
  - If not synchronized, the signals may violate the setup time or hold time of a receiving device.
- Metastable behavior
  - State in the middle of 1 and 0.

**Significance of Race around condition?**

# FLIPFLOP EXCITATION TABLE

- Excitation tables of a flipflops can be obtained from the truth tables of the same.
- It indicates that inputs required to be applied to the flipflop to take it from the present state to next state.

| S | R | Present state $Q_n$ | Next state $Q_{n+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | X |
| 1 | 1 | 1 | X |

Invalid states

*Truth table of SR flip flop*

| $Q_n$ | $Q_{n+1}$ | S | R |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | X | 0 |

*Excitation table of SR flip flop*

# FLIPFLOP EXCITATION TABLES

| SR Flip-flop | | | |
|---|---|---|---|
| Q(t) | Q(t+1) | S | R |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | X | 0 |

| D Flip-flop | | |
|---|---|---|
| Q(t) | Q(t+1) | DR |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| JK flip-flop | | | |
|---|---|---|---|
| Q(t) | Q(t+1) | J | K |
| 0 | 0 | 0 | x |
| 0 | 1 | 1 | x |
| 1 | 0 | x | 1 |
| 1 | 1 | x | 0 |

| T flip-flop | | |
|---|---|---|
| Q(t) | Q(t+1) | DR |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Excitation Table

- Excitation table describes the inputs required to achieve a desired state change.
  - i.e. if Q is the present state of memory and we wish to change it to Q+, what values must we assign to the external inputs of the flip-flop for the change to occur?

e.g. D flip-flop

| Q | Q+ | D |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

e.g. JK flip-flop

| Q | Q+ | J | K |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |

Transition from 0 to 0
Achieve by "hold" $JK = 00$
or "reset" $JK = 01$

"set" $JK = 10$
or "toggle" $JK = 11$

# Characteristic equation: It specifies the next state of a device as a function of its excitation (inputs)

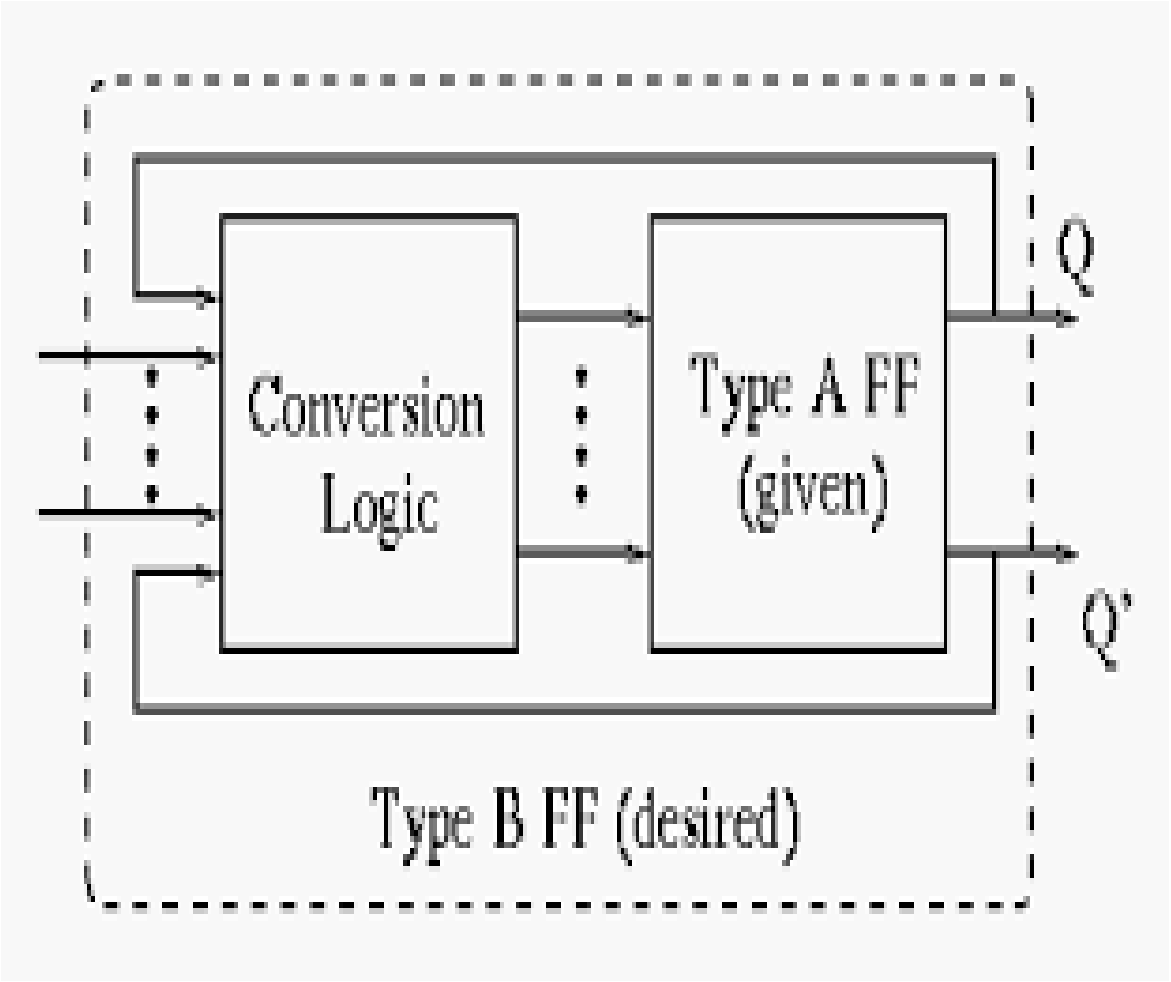
Flip-flop characteristic equations:

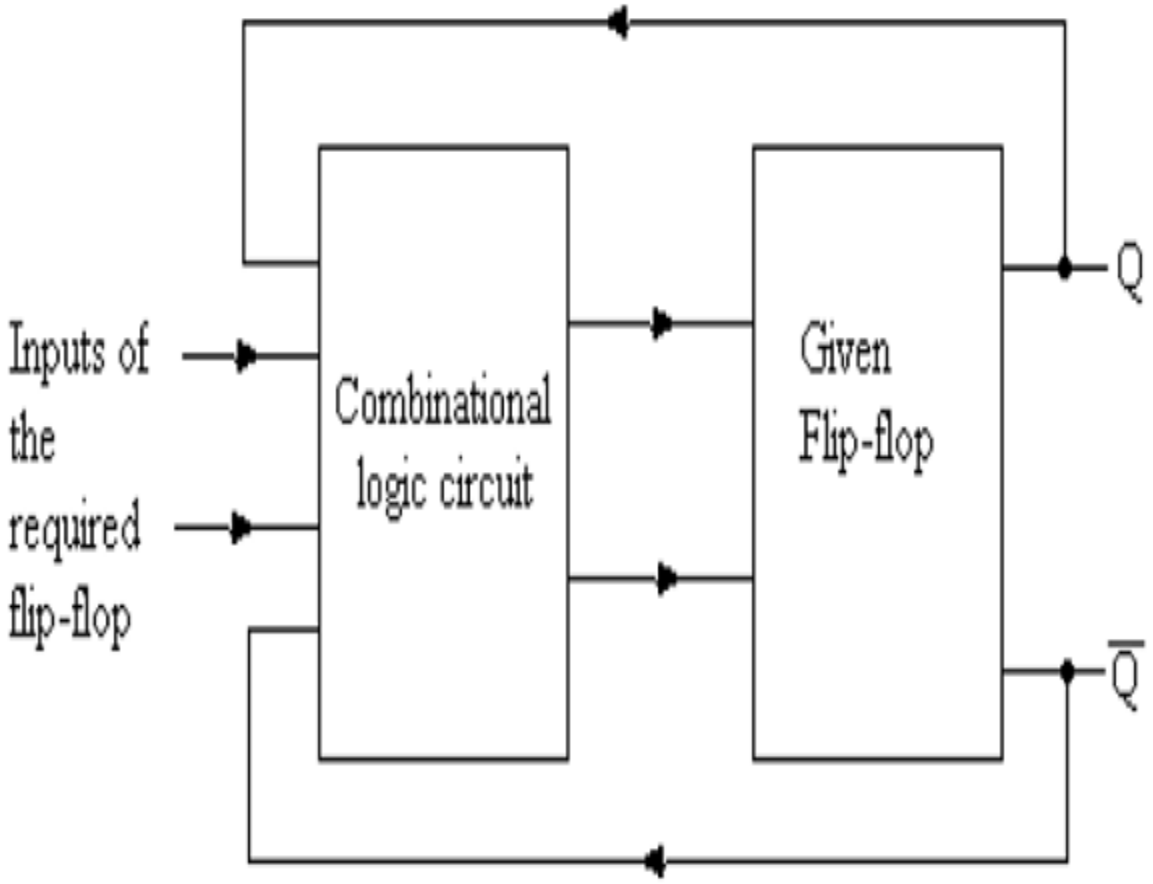
| | |
|---|---|
| D | Q+ = D |
| D with enable | Q+ = D • EN + Q • EN' |
| JK | Q+ = J • Q' + K' • Q |
| T | Q+ = Q' |
| T with enable | Q+ = Q' • EN + Q • EN' |

# FLIP-FLOP CONVERSIONS

- So, there will be total of **twelve** flip-flop conversions.
- Follow these steps for converting one flip-flop to the other.
- Consider the characteristic table of desired flip-flop.
- Fill the excitation values inputs of given flip-flop for each combination of present state and next state.
- For eg, The conversion of flip-flops to a JK flip-flop is to **cross connect the Q and Q outputs with the S and R inputs through additional 3-input AND gates** .
- If the J and K inputs are both HIGH, logic “1” then the Q output will change state (Toggle) for as long as the clock input, (CLK) is HIGH.

# Flip-flop Conversion block diagram



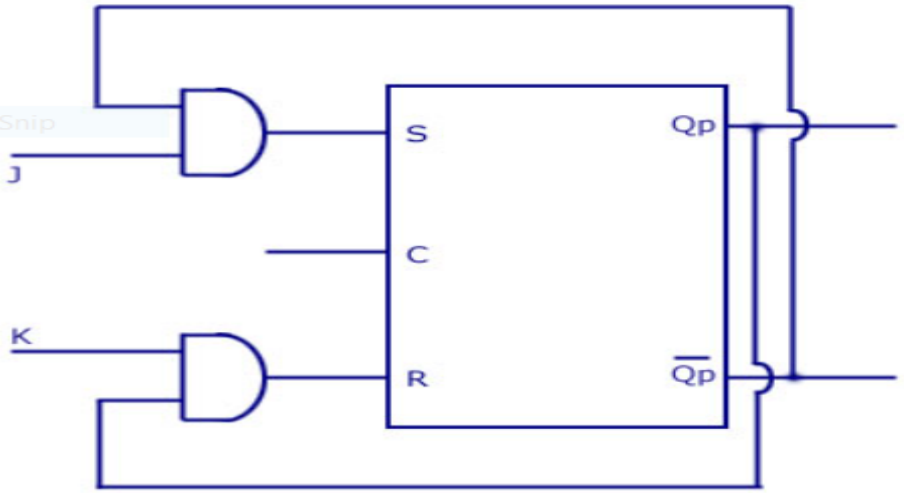
# SR flip-flop to JK flip-flop

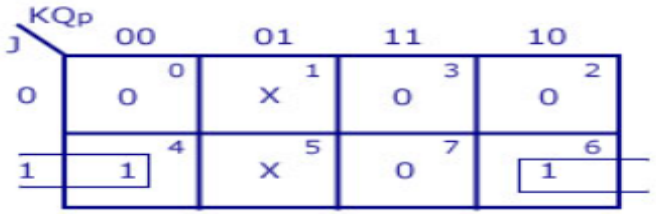
## S-R Flip Flop to J-K Flip Flop

### Conversion Table

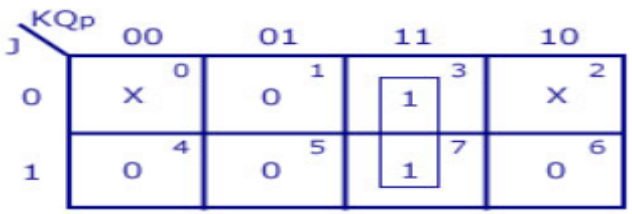
| J-K Inputs | | Outputs | | S-R Inputs | |
|---|---|---|---|---|---|
| J | K | Qp | Qp+1 | S | R |
| 0 | 0 | 0 | 0 | 0 | X |
| 0 | 0 | 1 | 1 | X | 0 |
| 0 | 1 | 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | X | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |

### Logic Diagram




$S = \overline{J}Qp$




$R = KQp$

# SR flip-flop to T flip-flop

## Conversion of SR Flip Flop to T Flip Flop

### 1. Truth Table for T flip-flop

| Input | Outputs | |
|---|---|---|
| T | $Q_n$ | $Q_{n+1}$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

### 2. Excitation Table for SR flip-flop

| Outputs | | Inputs | |
|---|---|---|---|
| $Q_n$ | $Q_{n+1}$ | S | R |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | X | 0 |

### 3. Conversion Table

| T | $Q_n$ | $Q_{n+1}$ | S | R |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | X |
| 0 | 1 | 1 | X | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

### 4. K-map Simplification

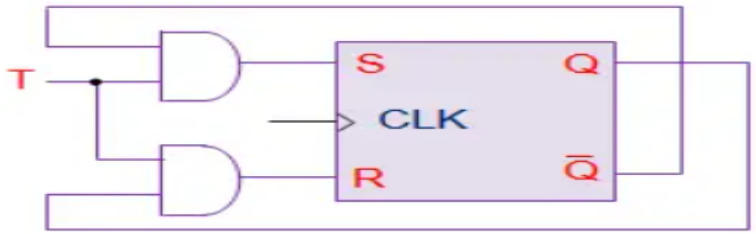
| T \ $Q_n$ | 0 | 1 |
|---|---|---|
| 0 | $0^0$ | $X^1$ |
| 1 | $1^2$ | $0^3$ |

$S = T\bar{Q}_n$

| T \ $Q_n$ | 0 | 1 |
|---|---|---|
| 0 | $X^0$ | $0^1$ |
| 1 | $0^2$ | $1^3$ |

$R = TQ_n$

### 5. Circuit Design



# SR flip-flop to D flip-flop

## Conversion of SR Flip Flop to D Flip Flop

### 1. Truth Table for D Flip Flop

| Input | Outputs | |
|---|---|---|
| D | $Q_n$ | $Q_{n+1}$ |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

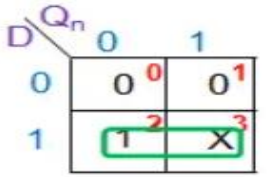
### 2. Excitation Table for SR Flip Flop

| Outputs | | Inputs | |
|---|---|---|---|
| $Q_n$ | $Q_{n+1}$ | S | R |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | X | 0 |

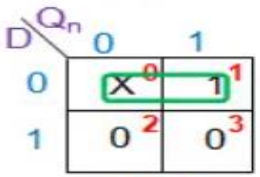
### 3. Conversion Table

| D | $Q_n$ | $Q_{n+1}$ | S | R |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | X |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | X | 0 |

### 4. K-map Simplification

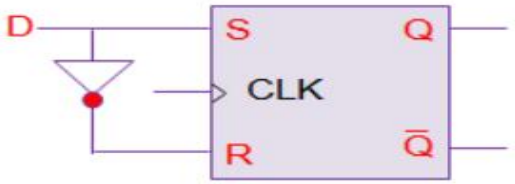



S = D




R = $\overline{D}$

### 5. Circuit Design



# Conversion of SR Flip Flop to JK Flip Flop

## 1. Truth Table for JK flip-flop

| Inputs | | Outputs | |
|---|---|---|---|
| J | K | $Q_n$ | $Q_{n+1}$ |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

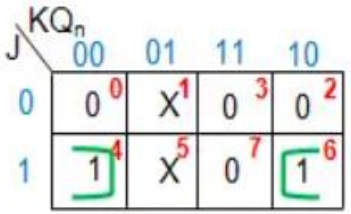
## 2. Excitation Table for SR flip-flop

| Outputs | | Inputs | |
|---|---|---|---|
| $Q_n$ | $Q_{n+1}$ | S | R |
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | X | 0 |

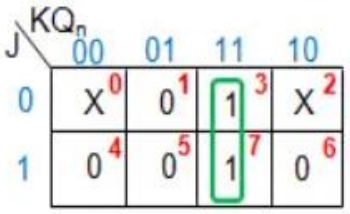
## 3. Conversion Table

| J | K | $Q_n$ | $Q_{n+1}$ | S | R |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | X |
| 0 | 0 | 1 | 1 | X | 0 |
| 0 | 1 | 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | X | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |

## 4. K-map Simplification

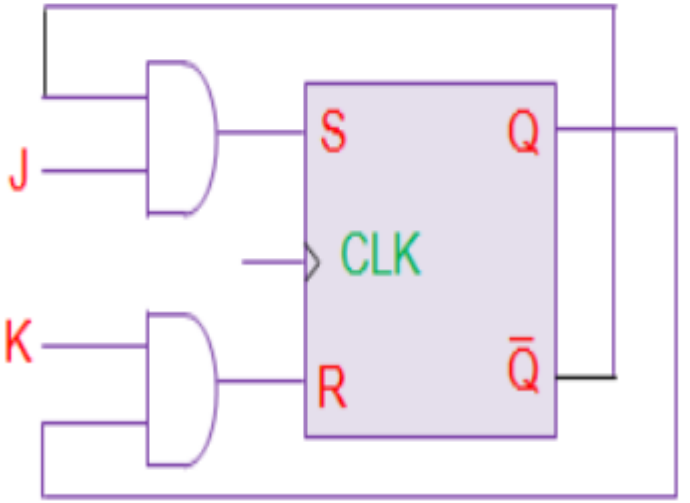



$S = J\bar{Q}_n$




$R = KQ_n$

## 5. Circuit Design



# Conversion of D Flip Flop to JK Flip Flop

## 1. Truth Table for JK flip-flop

| Inputs | | Outputs | |
|---|---|---|---|
| J | K | $Q_n$ | $Q_{n+1}$ |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

## 2. Excitation Table for D flip-flop

| Outputs | | Input |
|---|---|---|
| $Q_n$ | $Q_{n+1}$ | D |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## 3. Conversion Table

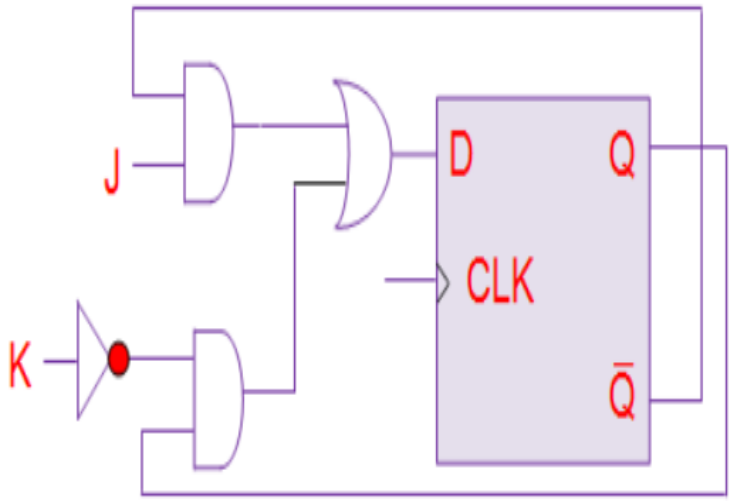
| J | K | $Q_n$ | $Q_{n+1}$ | D |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 |

## 4. K-map Simplification

| J \ $KQ_n$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 [0] | 1 [1] | 0 [3] | 0 [2] |
| 1 | 1 [4] | 1 [5] | 0 [7] | 1 [6] |

$$D = J\bar{Q}_n + \bar{K}Q_n$$

## 5. Circuit Design



# Conversion of D Flip Flop to SR Flip Flop

## 1. Truth Table for SR flip-flop

| S | R | $Q_n$ | $Q_{n+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | invalid | |
| 1 | 1 | invalid | |

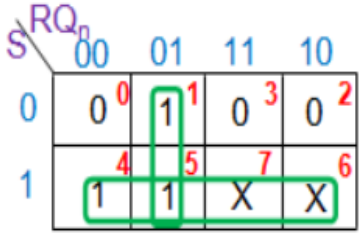
## 2. Excitation Table for D flip-flop

| Outputs | | Input |
|---|---|---|
| $Q_n$ | $Q_{n+1}$ | D |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## 3. Conversion Table

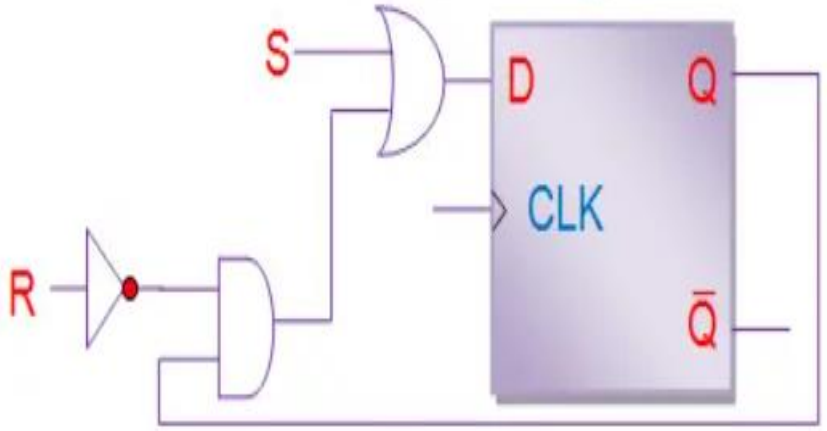
| S | R | $Q_n$ | $Q_{n+1}$ | D |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | invalid | | X |
| 1 | 1 | invalid | | X |

## 4. K-map Simplification




$$D = S + \bar{R}Q_n$$

## 5. Circuit Design



# Conversion of D Flip Flop to T Flip Flop

## 1. Truth Table for T Flip Flop

| Input | Outputs | |
|---|---|---|
| T | $Q_n$ | $Q_{n+1}$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

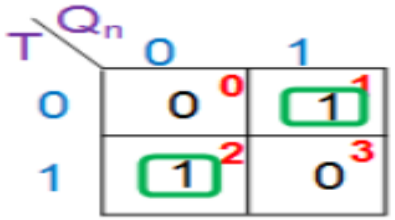
## 2. Excitation Table for D Flip Flop

| Outputs | | Input |
|---|---|---|
| $Q_n$ | $Q_{n+1}$ | D |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## 3. Conversion Table

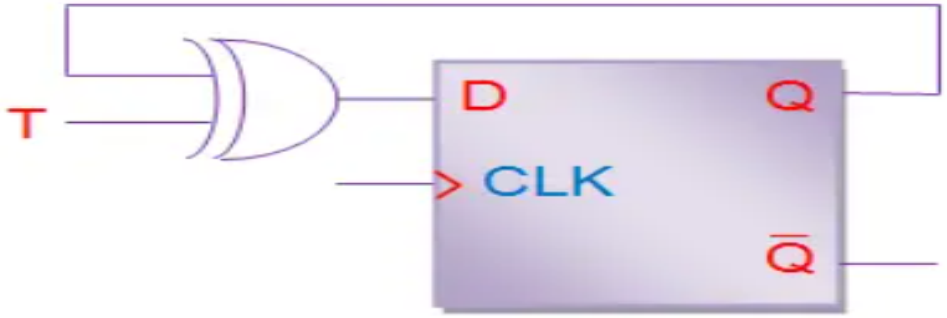
| T | $Q_n$ | $Q_{n+1}$ | D |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

## 4. K-map Simplification




$$D = T\overline{Q}_n + \overline{T}Q_n$$
$$= T \oplus Q_n$$

## 5. Circuit Design



# Conversion of T Flip Flop to JK Flip Flop

## 1. Truth Table for JK Flip Flop

| Inputs | | Outputs | |
|---|---|---|---|
| J | K | $Q_n$ | $Q_{n+1}$ |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

## 2. Excitation Table for T Flip Flop

| Outputs | | Input |
|---|---|---|
| $Q_n$ | $Q_{n+1}$ | T |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## 3. Conversion Table

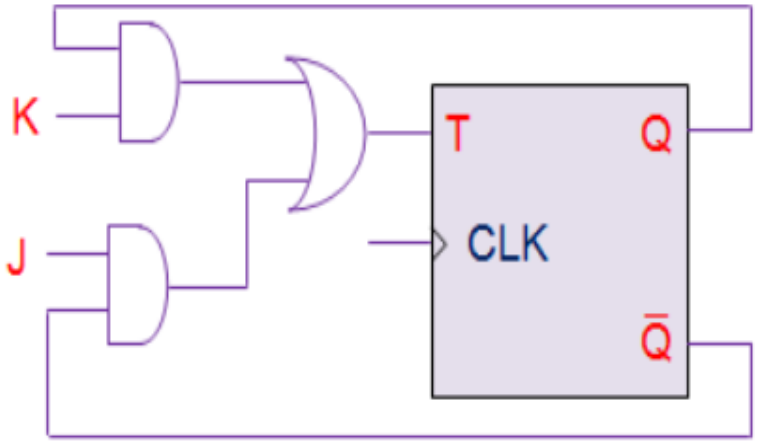
| J | K | $Q_n$ | $Q_{n+1}$ | T |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |

## 4. K-map Simplification

| J \ $KQ_n$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 [0] | 0 [1] | 1 [3] | 0 [2] |
| 1 | 1 [4] | 0 [5] | 1 [7] | 1 [6] |

$$T = J\bar{Q}_n + KQ_n$$

## 5. Circuit Design



# Conversion of T Flip Flop to SR Flip Flop

## 1. Truth Table for SR Flip Flop

| S | R | $Q_n$ | $Q_{n+1}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | invalid | |
| 1 | 1 | invalid | |

## 2. Excitation Table for T Flip Flop

| Outputs | | Input |
|---|---|---|
| $Q_n$ | $Q_{n+1}$ | T |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## 3. Conversion Table

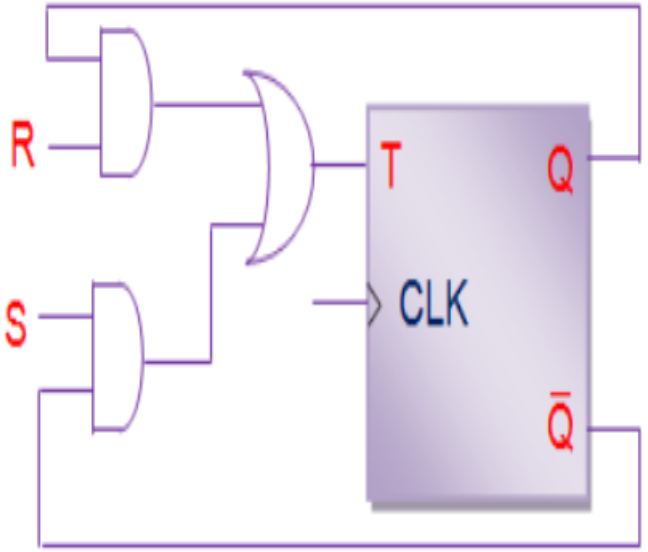
| S | R | $Q_n$ | $Q_{n+1}$ | T |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | invalid | | X |
| 1 | 1 | invalid | | X |

## 4. K-map Simplification

| S \ $RQ_n$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 [0] | 0 [1] | 1 [3] | 0 [2] |
| 1 | 1 [4] | 0 [5] | X [7] | X [6] |

$$T = S\bar{Q}_n + RQ_n$$

## 5. Circuit Design



# Conversion of T Flip Flop to D Flip Flop

## 1. Truth Table for D Flip Flop

| Input | Outputs | |
|---|---|---|
| D | $Q_n$ | $Q_{n+1}$ |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

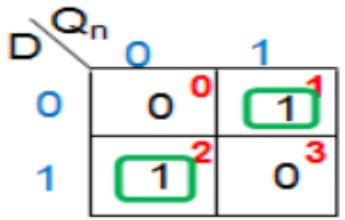
## 2. Excitation Table for T Flip Flop

| Outputs | | Input |
|---|---|---|
| $Q_n$ | $Q_{n+1}$ | T |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## 3. Conversion Table

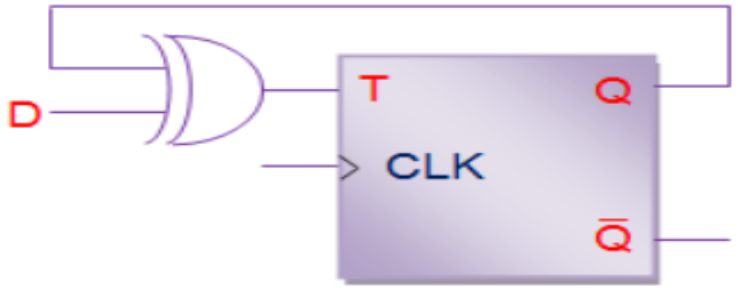
| D | $Q_n$ | $Q_{n+1}$ | T |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

## 4. K-map Simplification




$$D = D\bar{Q}_n + \bar{D}Q_n$$
$$= D \oplus Q_n$$

## 5. Circuit Design



# Registers & shift Registers

- In order to store multiple bits of information, we require multiple flip-flops. The group of flip-flops, which are used to hold/ store/new data into register serial to parallel OR shift the data with in register(left or right) OR retrieve 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.
- Register's output depends on the past and present states of the inputs. **The device which follows these properties is termed as a sequential circuit**.
- They are a group of flip-flops connected in a chain so that the output from one flip-flop becomes the input of the next flip-flop.
- **A register is a group of flip-flops** used to store a binary word. One flip-flop is needed for each bit in the data word.

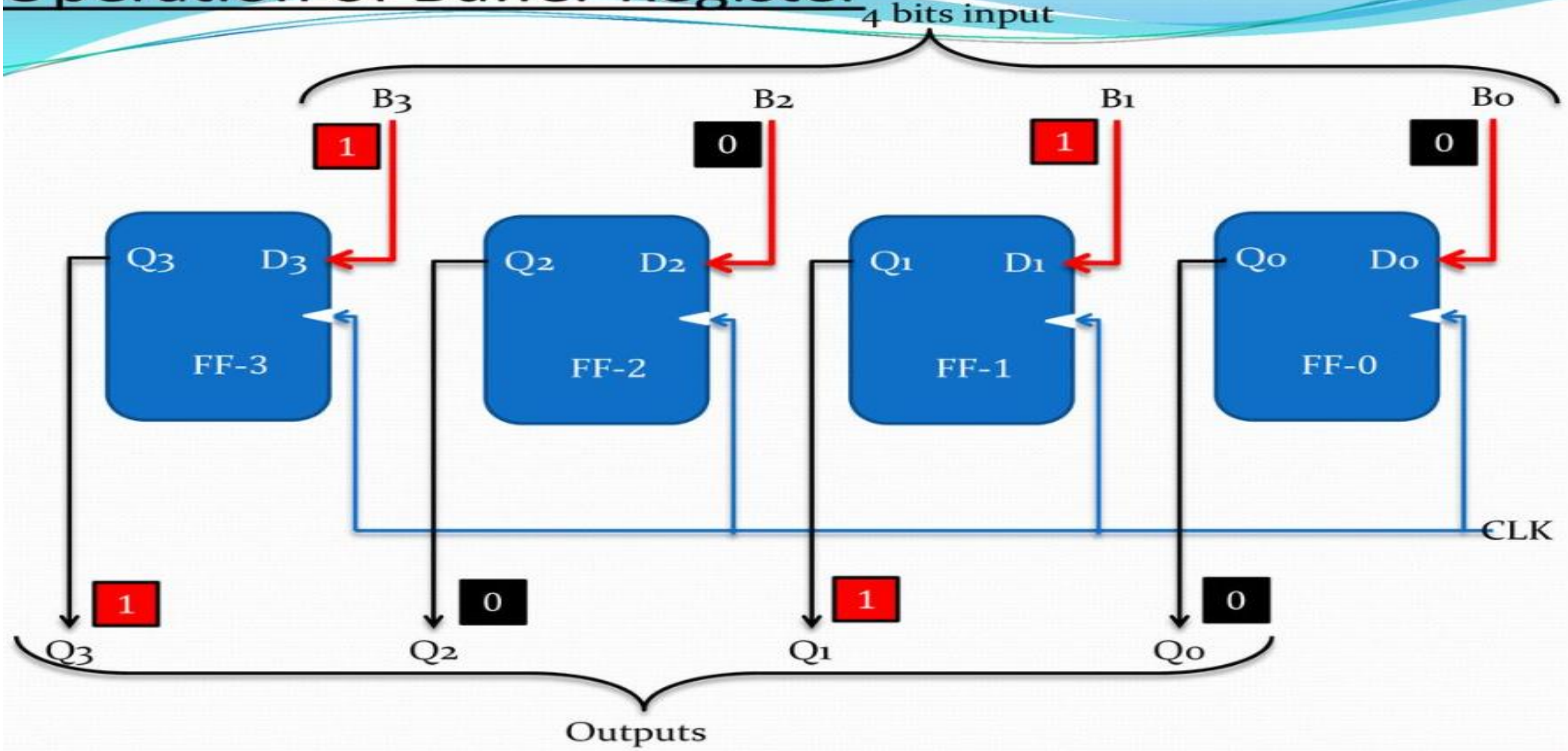
# Shift Registers

- A shift register is composed of a cascade of flip-flops in which the output (Q) of each flip-flop is connected to the data (D) input of the next flip-flop in the chain. A serial input (SI) is applied to the data (D) input of the first flip-flop.
- A Shift Register can shift the bits either to the left or to the right. A Shift Register, which shifts the bit to the left, is known as "Shift left register", and it shifts the bit to the right, known as "Right Shift register".
- Loading a register: Transferring new information into the register. Requires a load control input.
- Parallel loading:  all bits are loaded simultaneously.

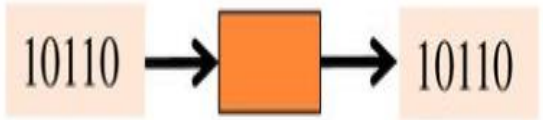
# Types of Shift Registers are :

- Serial In Serial Out [SISO]
- Serial In Parallel Out [SIPO]
- Parallel In Serial Out [PISO]
- Parallel In Parallel Out [PIPO]
- Bi-directional Shift Register
- Universal Shift Register

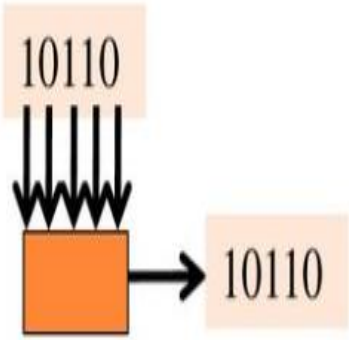
# Operation of Buffer Register



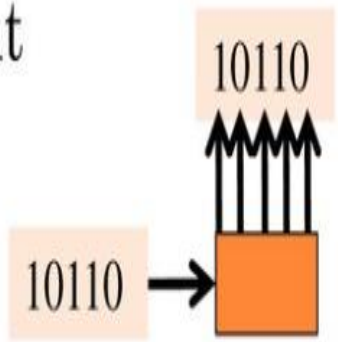
- SISO: Serial In, Serial Out



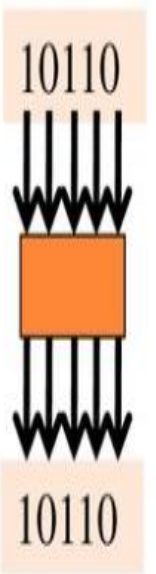

- PISO: Parallel In, Serial Out

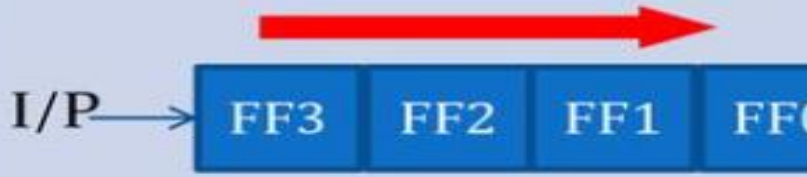
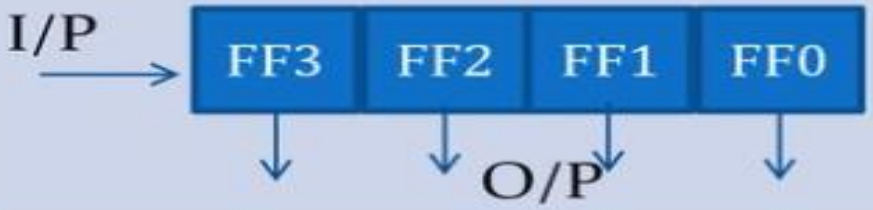
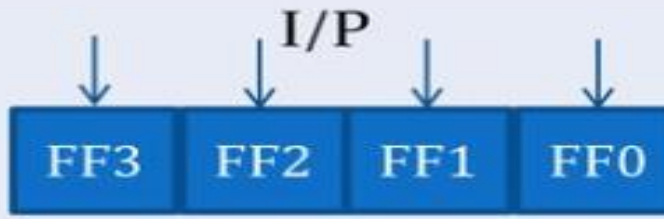



- SIPO: Serial In, Parallel Out

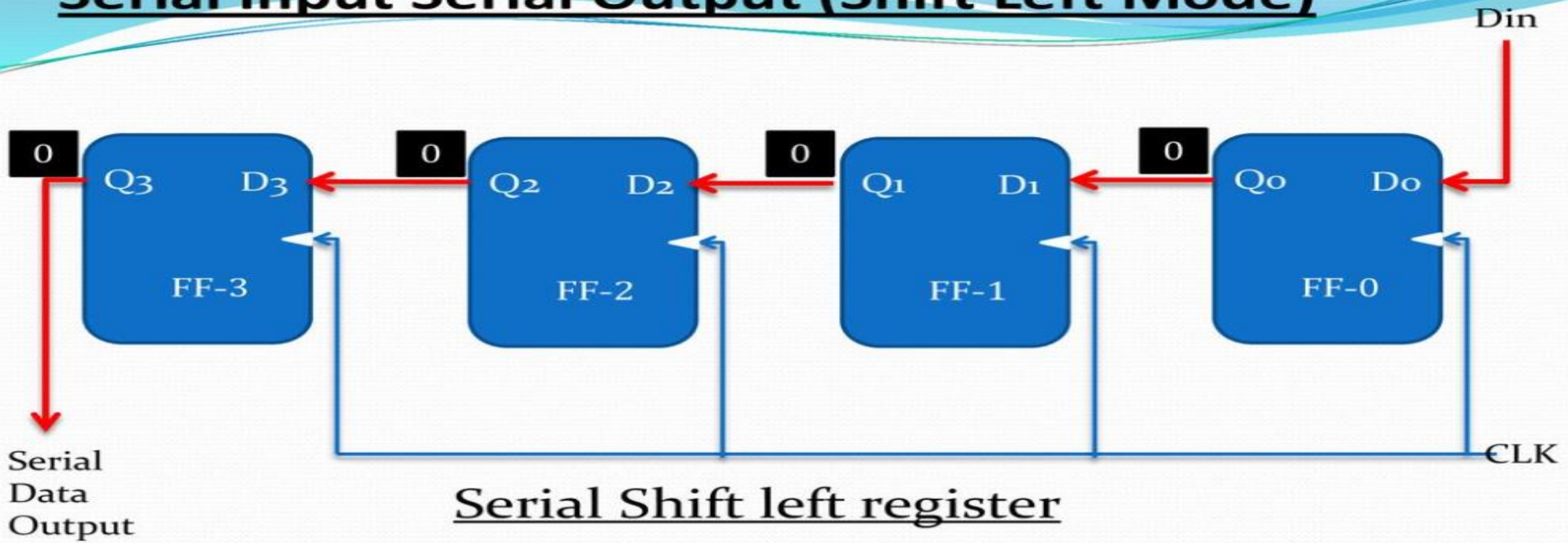



- PIPO: Parallel In, Parallel Out



| Sr. No. | Mode | Illustrative Diagram | Comments |
|---|---|---|---|
| 1. | SISO (Shift Right) |   | Data bits shift from Left to Right by 1 position per clock cycle. |
| 2. | SISO (Shift Left) |   | Data bits shift from Right to Left by 1 position per clock cycle. |
| 3. | SIPO |   | All o/p bits are made aval. simult. after 4-clcok pulse |
| 4. | PISO |   | All i/p bits are applied simult and. After 4-clk pulse the required o/p is available serially. |

# Serial input Serial Output (Shift Left Mode)



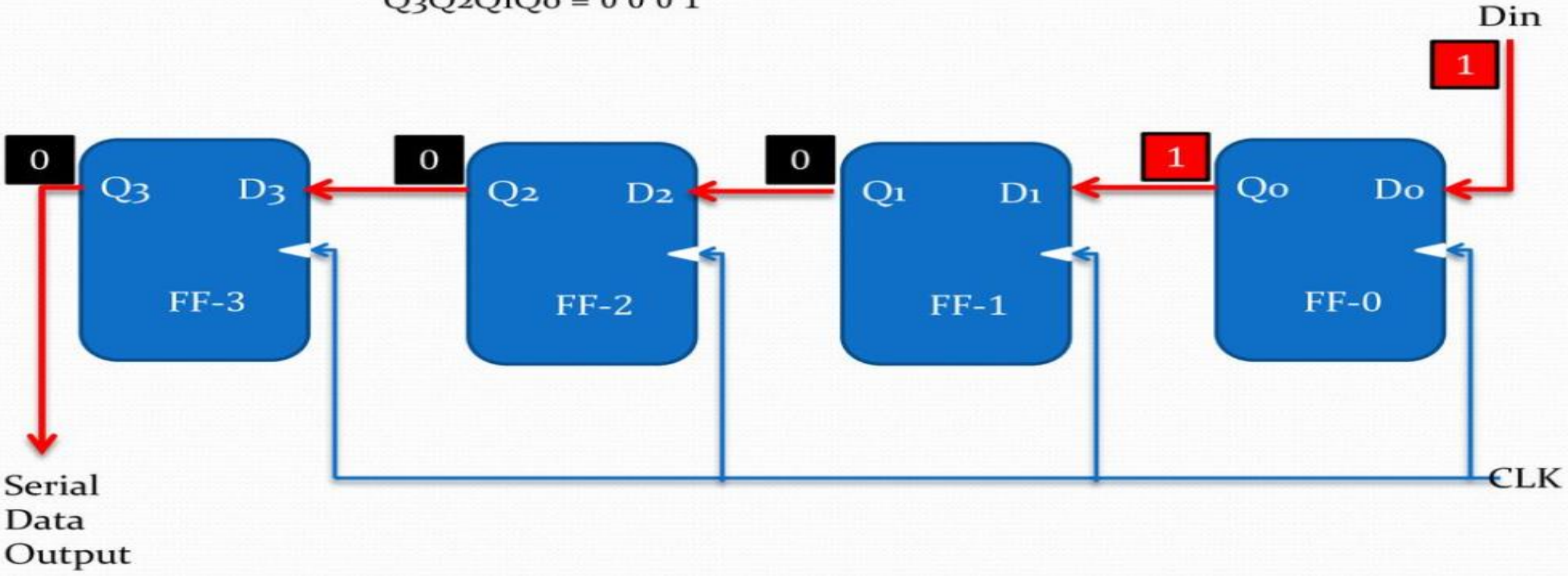

## Serial Shift left register

- Before application of clock let assume all outputs are zero and apply MSB bit of the number to entered to Din. So Din = Do = 1.

# Serial input Serial Output (Shift Left Mode)

- Apply the clock . On the first falling edge of clock, the FF-0 is SET and the stored data in the register is

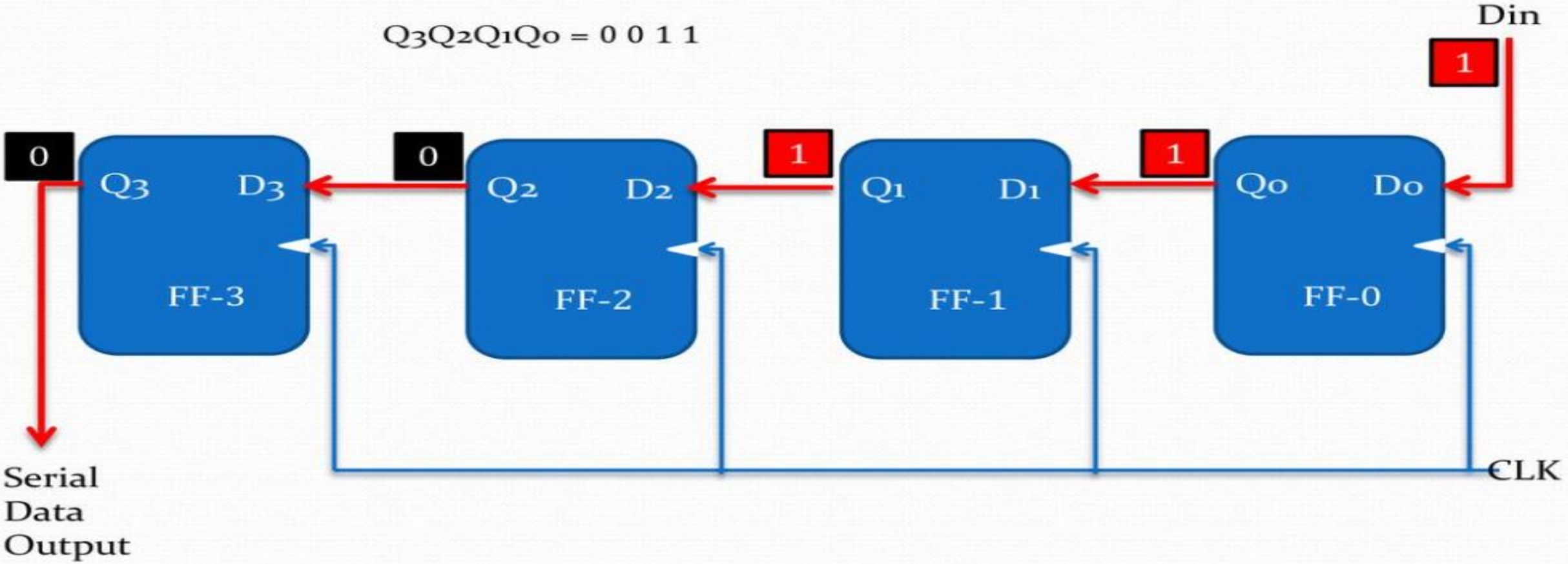
$Q_3Q_2Q_1Q_0 = 0\ 0\ 0\ 1$



# Serial input Serial Output (Shift Left Mode)

- Apply the NEXT bit to Din . So if Din =1.
- As soon as the next positive edge of the clock hits. FF- 1 will SET and the stored data changes to,

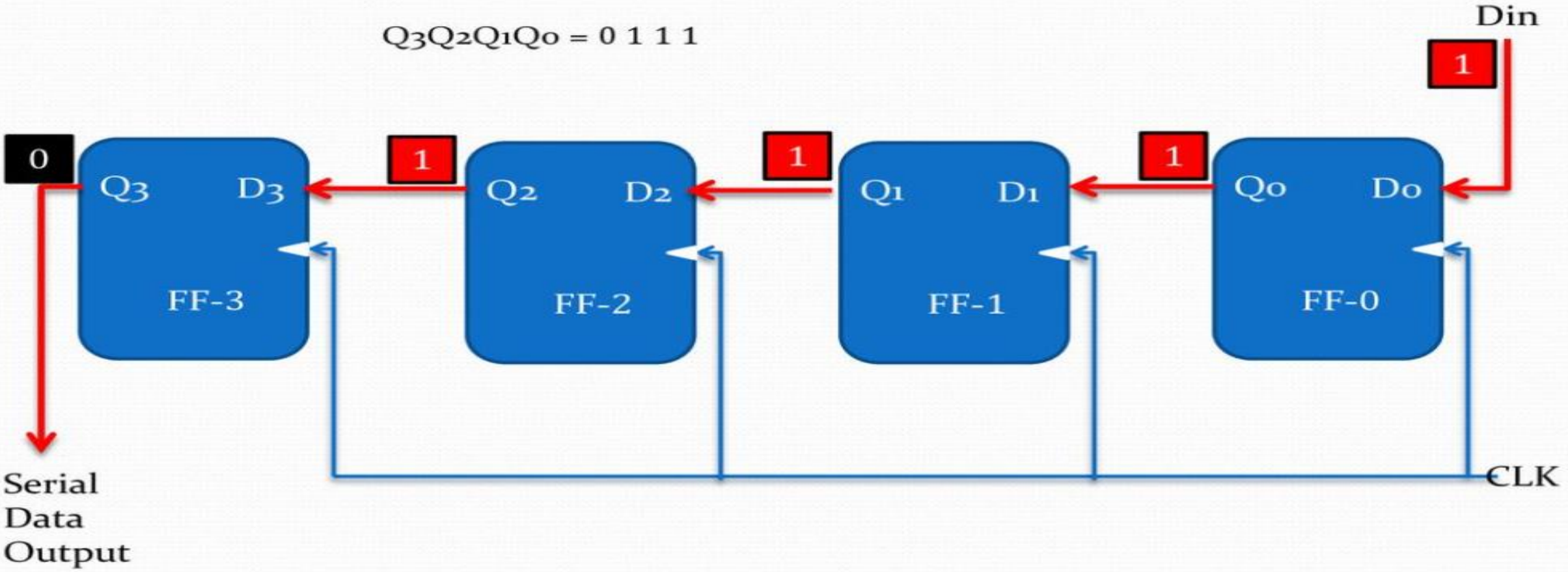
$Q_3Q_2Q_1Q_0 = 0\ 0\ 1\ 1$



# Serial input Serial Output (Shift Left Mode)

- Apply the NEXT bit to Din . So if Din =1.
- As soon as the next positive edge of the clock hits. FF- 2will SET and the stored data changes to,

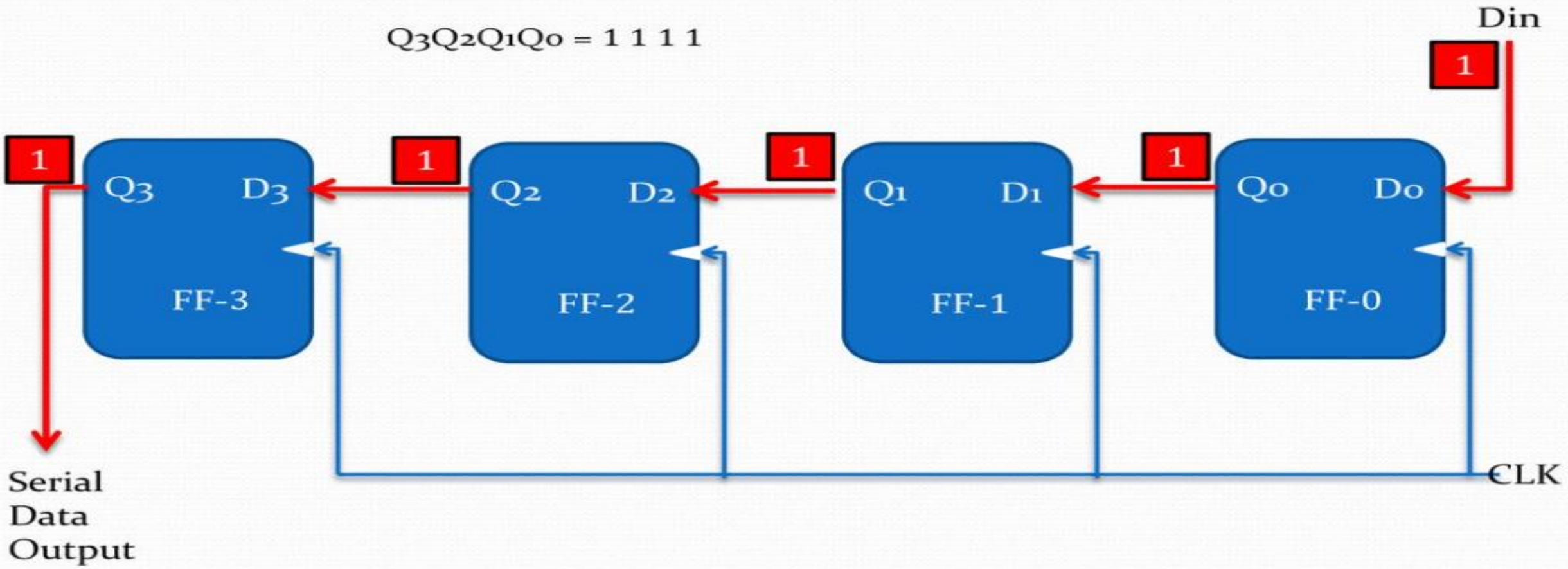
$Q_3Q_2Q_1Q_0 = 0\ 1\ 1\ 1$



# Serial input Serial Output (Shift Left Mode)

- Apply the NEXT bit to Din . So if Din =1.
- As soon as the next positive edge of the clock hits. FF- 3will SET and the stored data changes to,

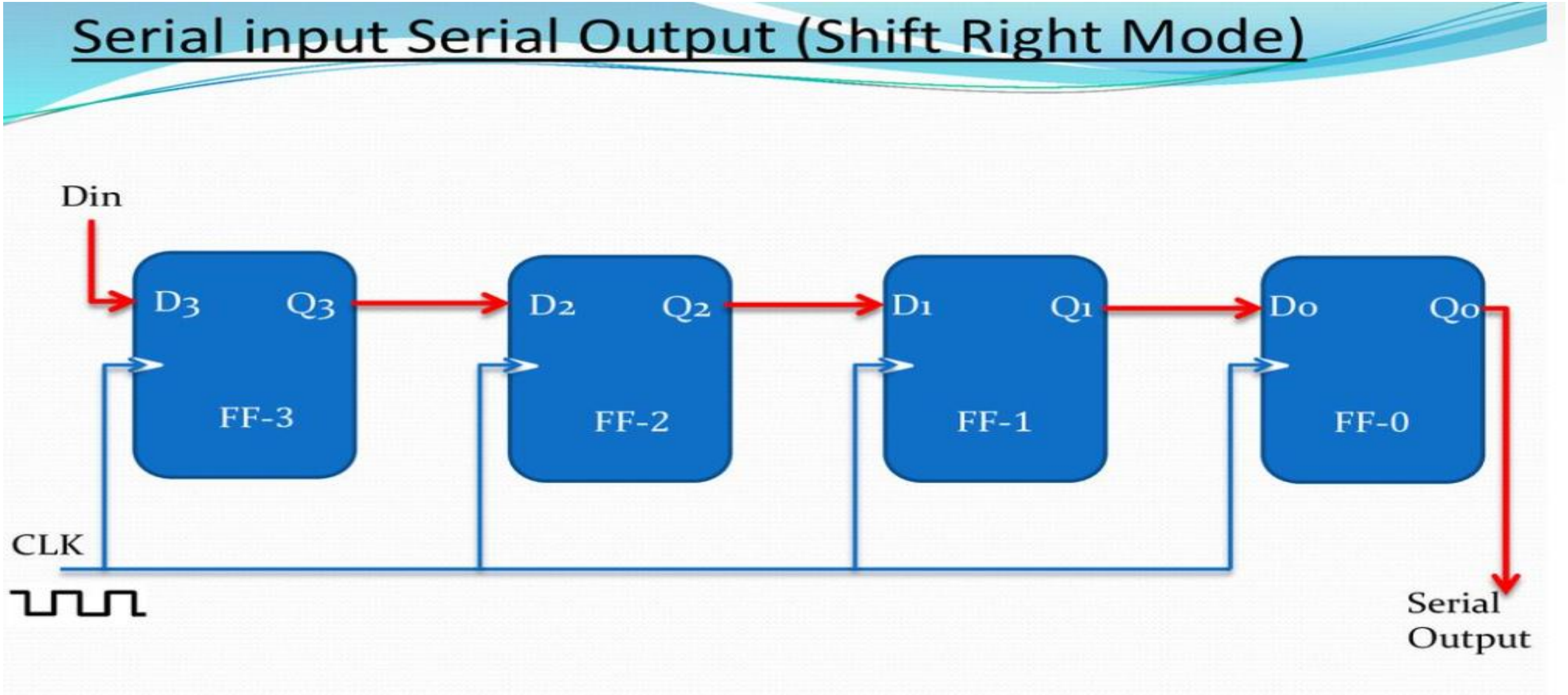
$Q_3Q_2Q_1Q_0 = 1\ 1\ 1\ 1$



# Summary:

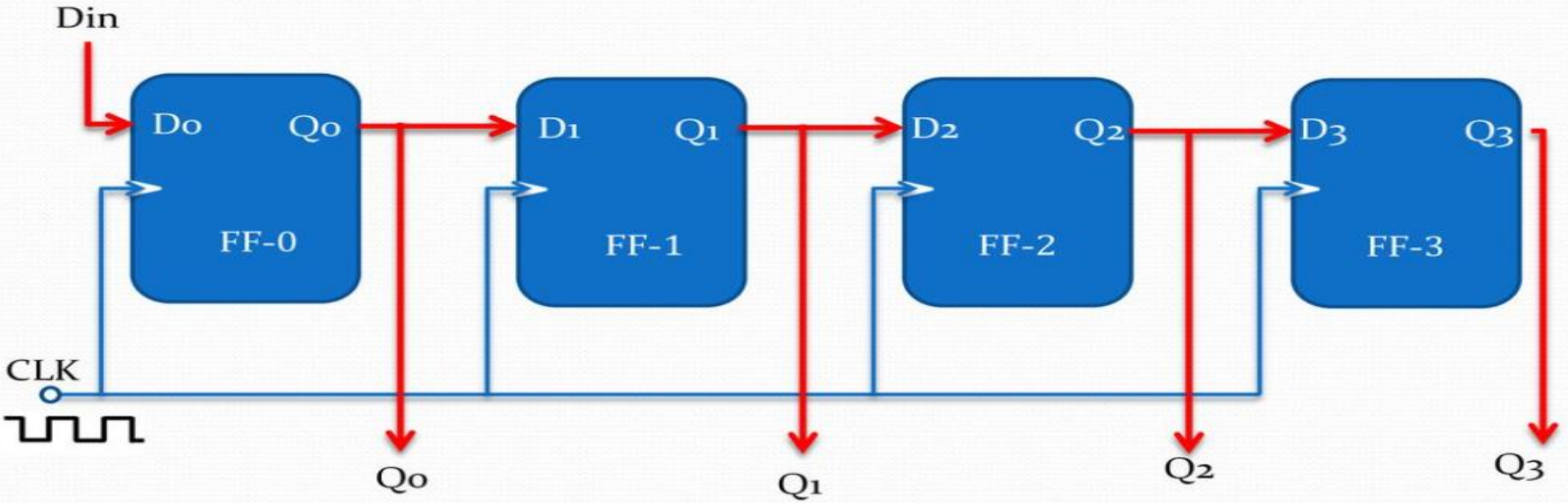
| CLK | $Q_3$ | $Q_2=D_3$ | $Q_1=D_2$ | $Q_0=D_1$ | Serial input Din = Do |
|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | |
| ↑ | 0 | 0 | 0 | 1 ← | 1 |
| ↑ | 0 | 0 | 1 | 1 ← | 1 |
| ↑ | 0 | 1 | 1 | 1 ← | 1 |
| ↑ | 1 | 1 | 1 | 1 ← | 1 |

# SISO shift right mode



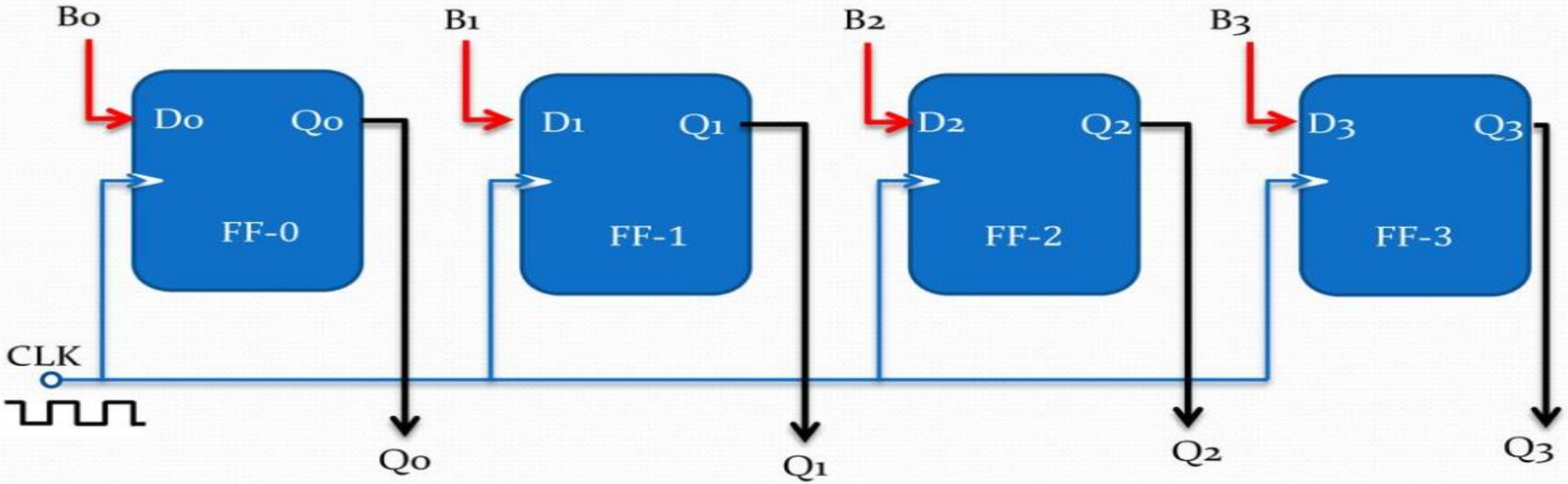
# Serial input Parallel Output (SIPO)

- In this operation the data is entered serially and taken out in parallel.
- That means first the data is loaded bit by bit. The output are disabled as the loading is taking place.
- Number of clock cycles required to load a four bits data is 4. Hence the speed of operation of SIPO mode is same as that of SISO mode.

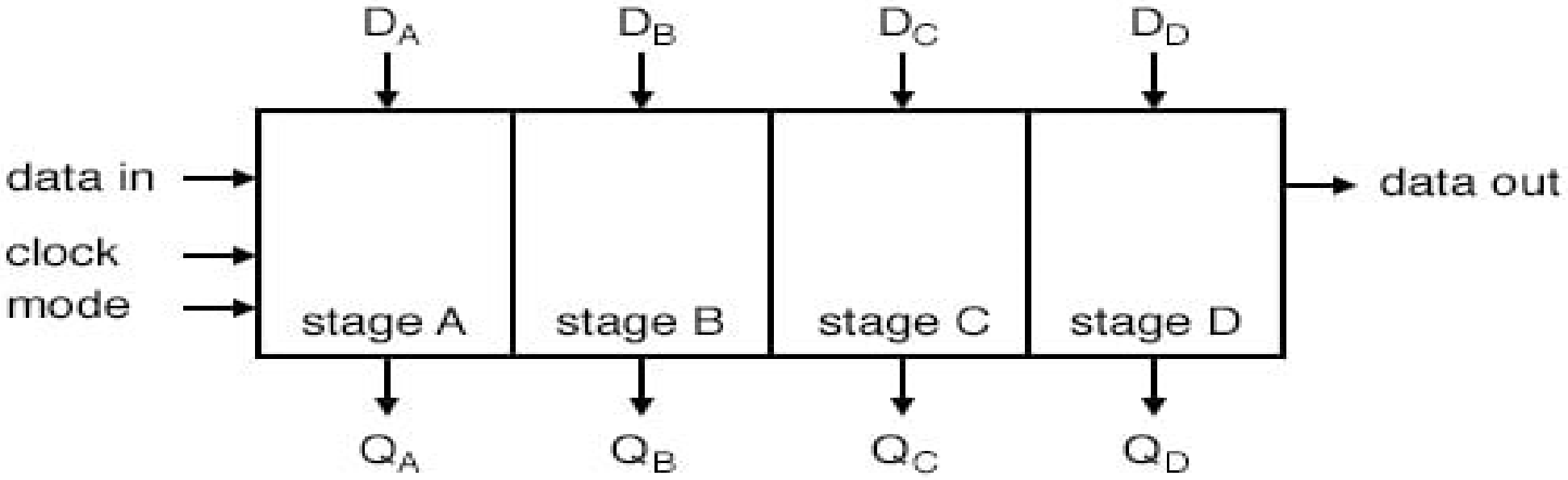


# Parallel input Parallel Output (PIPO)

- In this operation the data are entered parallel.
- The 4-bit binary input $B_0$, $B_1$, $B_2$, $B_3$ is applied to data inputs $D_0$, $D_1$, $D_2$ and $D_3$ respectively of the four flip-flops.
- As soon as a positive clock edge is applied, the input binary bits will be loaded into the flip-flops simultaneously.
- The loaded bits will appear simultaneously to the output side. **ONLY ONE CLOCK IS ESSENTIAL TO LOAD ALL THE BITS.**

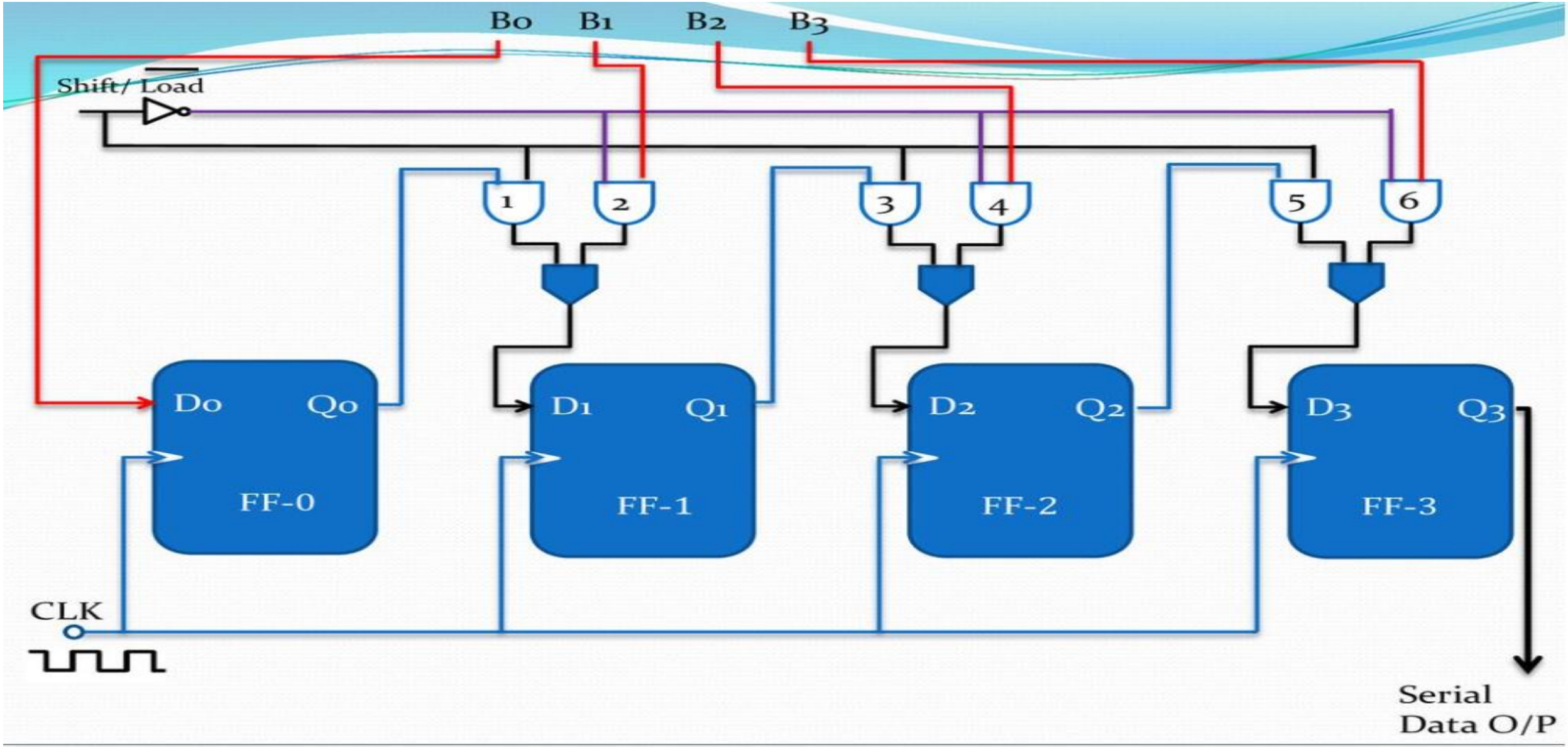


# PIPO Shift Register




Parallel-in parallel-out shift register with 4 stages

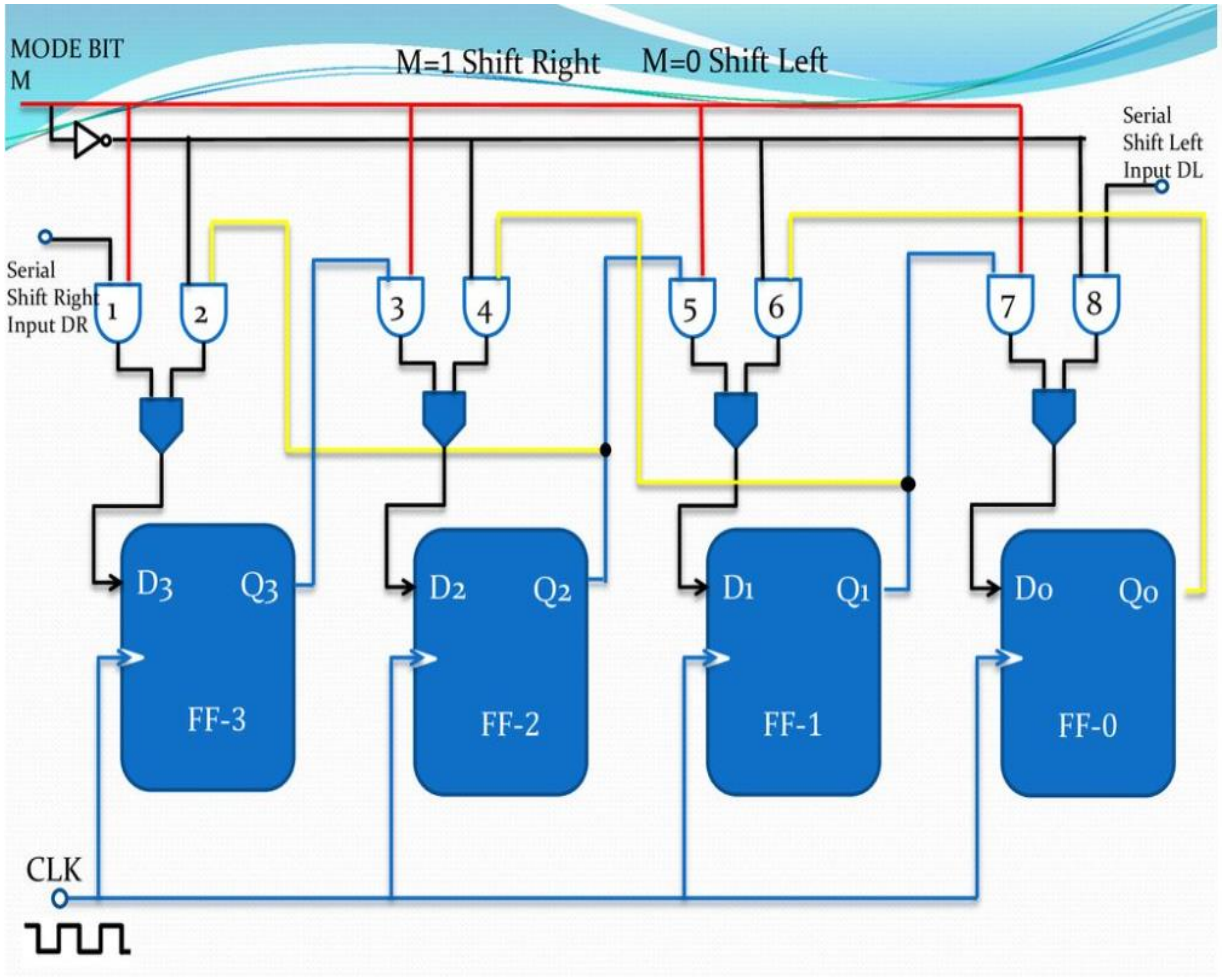
# Parallel Input Serial Output (PISO)

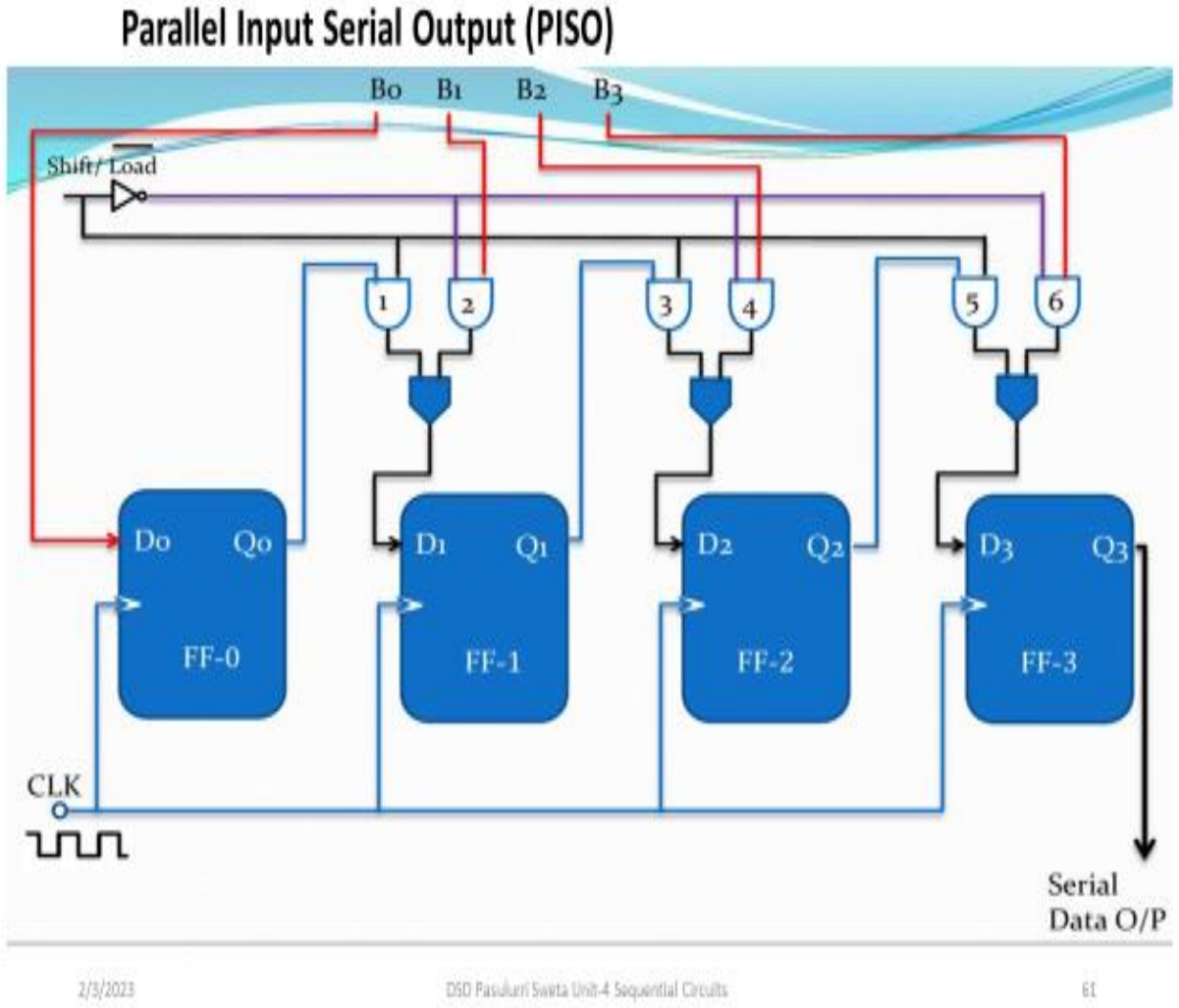


# PISO

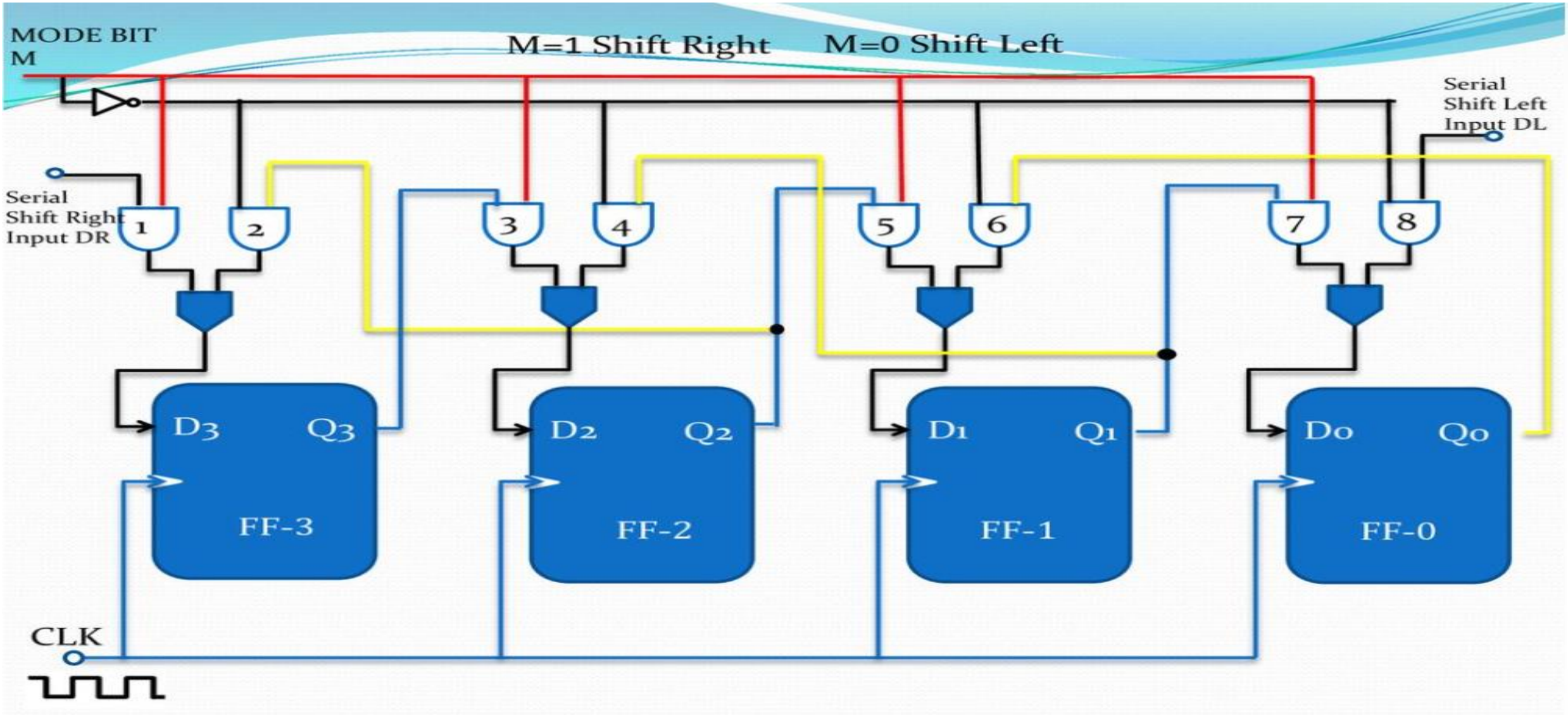
- In this operation the data are entered parallel.
- Output of previous FF is connected to the input of the next via a combinational circuit.
- The binary input data B0,B1, B2 & B3 is applied through the same combinational circuit.
- There are 2 modes: Shift mode  OR Load mode

# Bi-directional & PISO registers



# Bi directional Shift register



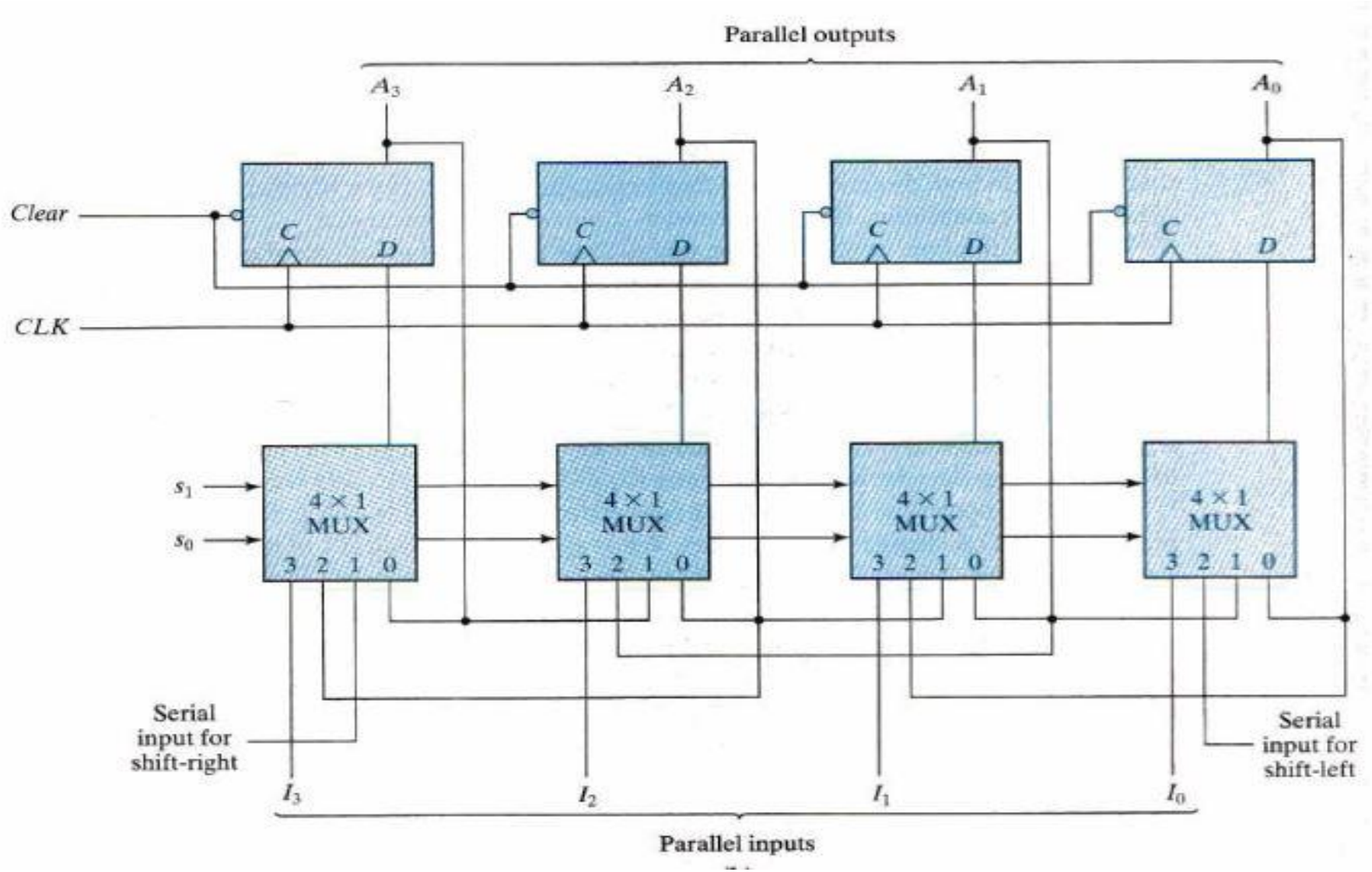
# 4-bit Universal Shift Register

| Mode Control $S_1$ | Mode Control $S_0$ | Register Operation |
|---|---|---|
| 0 | 0 | No change |
| 0 | 1 | Shift Right |
| 1 | 0 | Shift Left |
| 1 | 1 | Parallel load |

# Universal Shift Registers

- This register is capable of performing :
- Left shifting
- Right shifting
- Parallel loading
- Universal shift registers are used as memory elements in computers. Unidirectional shift register is capable of shifting in only one direction.
- A Reg is capable of shifting in one direction only = Unidirectional S.R.
- A Reg is capable of shifting in both direction= Bi-directional S.R
- If a Reg has both shifts and parallel load capabilities= Universal S.R.
- Whose IP& OP= either serial or parallel form

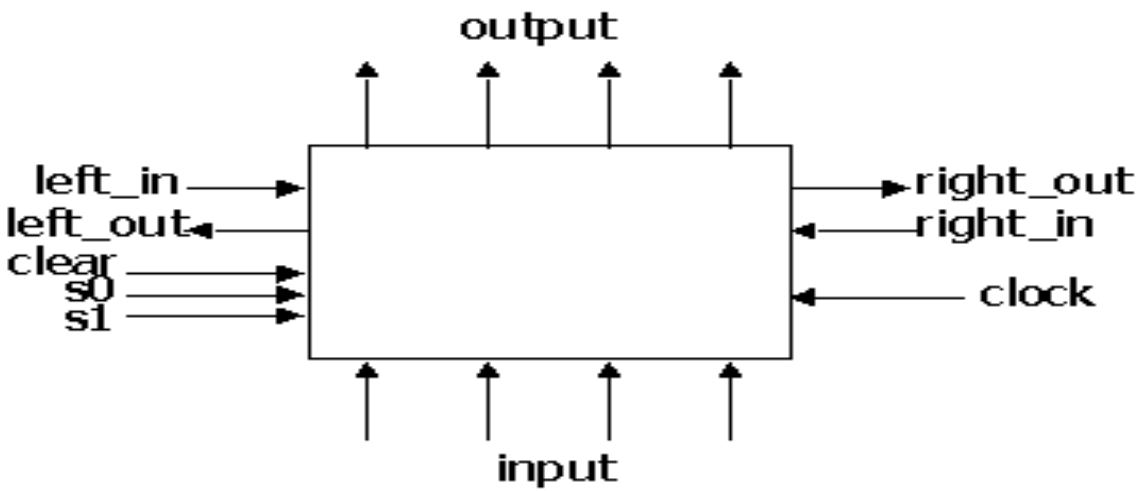
# Universal Shift register




| Mode Control $S_1$ | Mode Control $S_0$ | Register Operation |
|---|---|---|
| 0 | 0 | No change |
| 0 | 1 | Shift Right |
| 1 | 0 | Shift Left |
| 1 | 1 | Parallel load |

# 4-BIT UNIVERSAL SHIFT REGISTER

- Holds 4 values
  - serial or parallel inputs
  - serial or parallel outputs
  - permits shift left or right
  - shift in new values from left or right

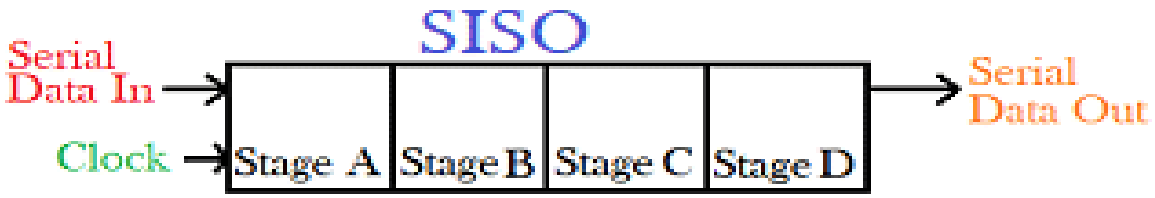


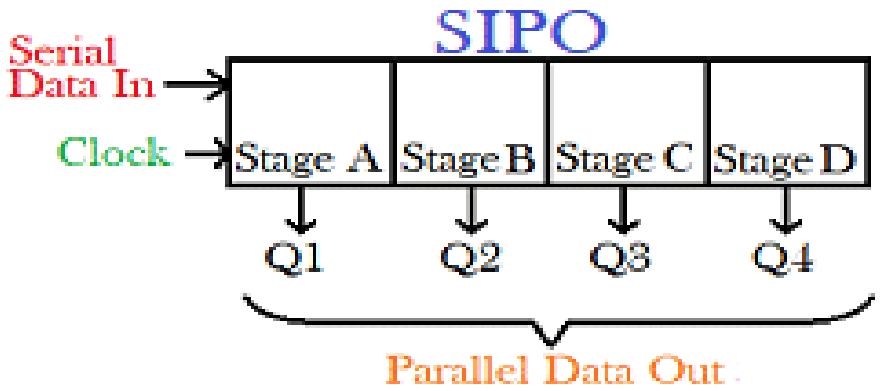

clear sets the register contents and output to 0

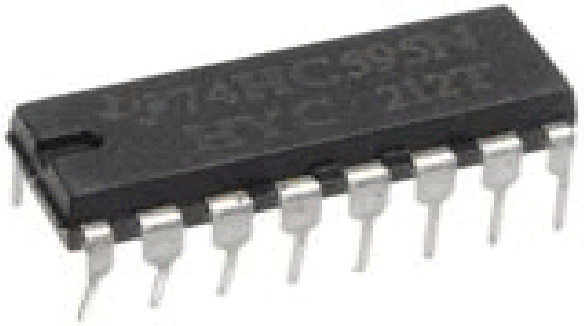
s1 and s0 determine the shift function

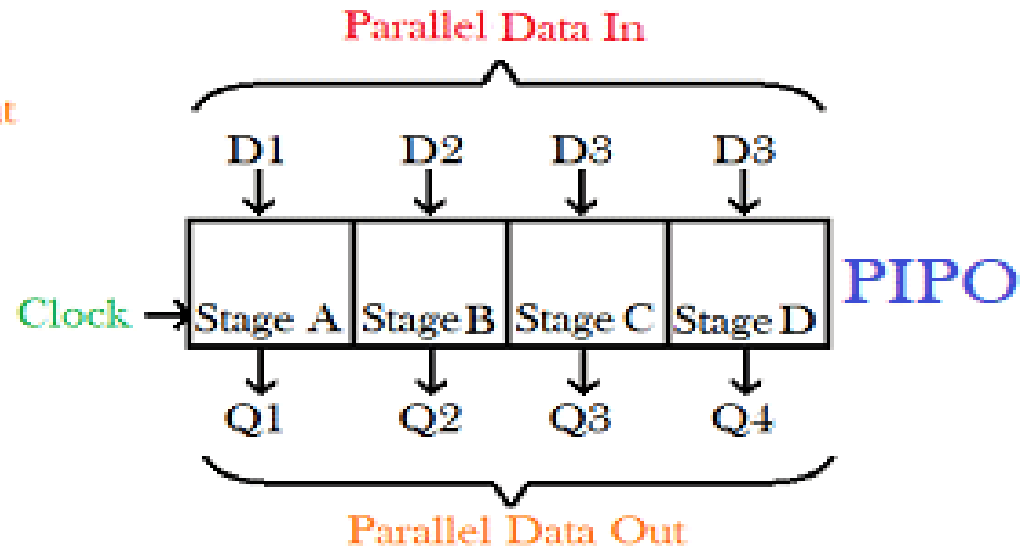
| s0 | s1 | function |
|---|---|---|
| 0 | 0 | hold state |
| 0 | 1 | shift right |
| 1 | 0 | shift left |
| 1 | 1 | load new input |

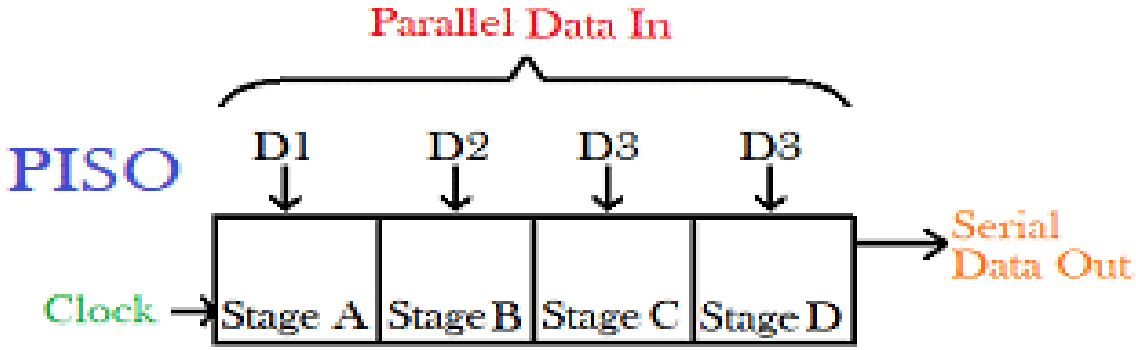
# Shift Registers quick summary





# Applications of Shift Registers

1. For Temporary data storage.
2. For multiplication and division
3. As a delay line
4. Ring counters
5. Parallel to serial converter

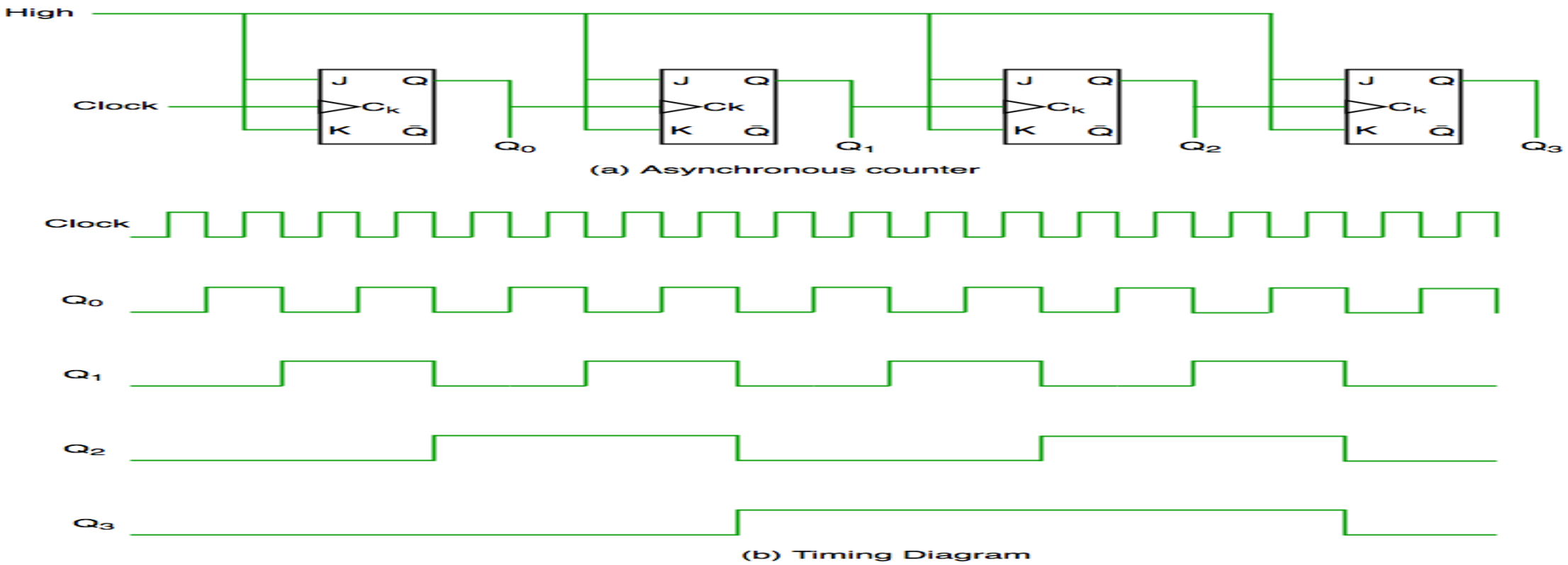
# Counters

- A **Counter** is a device which stores (and sometimes displays) the number of times a particular event or process has occurred, often in relationship to a clock signal.
- Counters are used in digital electronics for counting purpose, they can count specific event happening in the circuit.
- For example, in UP counter a counter increases count for every rising edge of clock. Not only counting, a counter can follow the certain sequence based on our design like any random sequence 0,1,3,2… .They can also be designed with the help of flip flops.
- They are used as frequency dividers where the frequency of given pulse waveform is divided. Counters are sequential circuit that count the number of pulses can be either in binary code or BCD form.
- The main properties of a counter are timing , sequencing , and counting. Counter works in two modes
- Up counter
- Down counter

# Counters

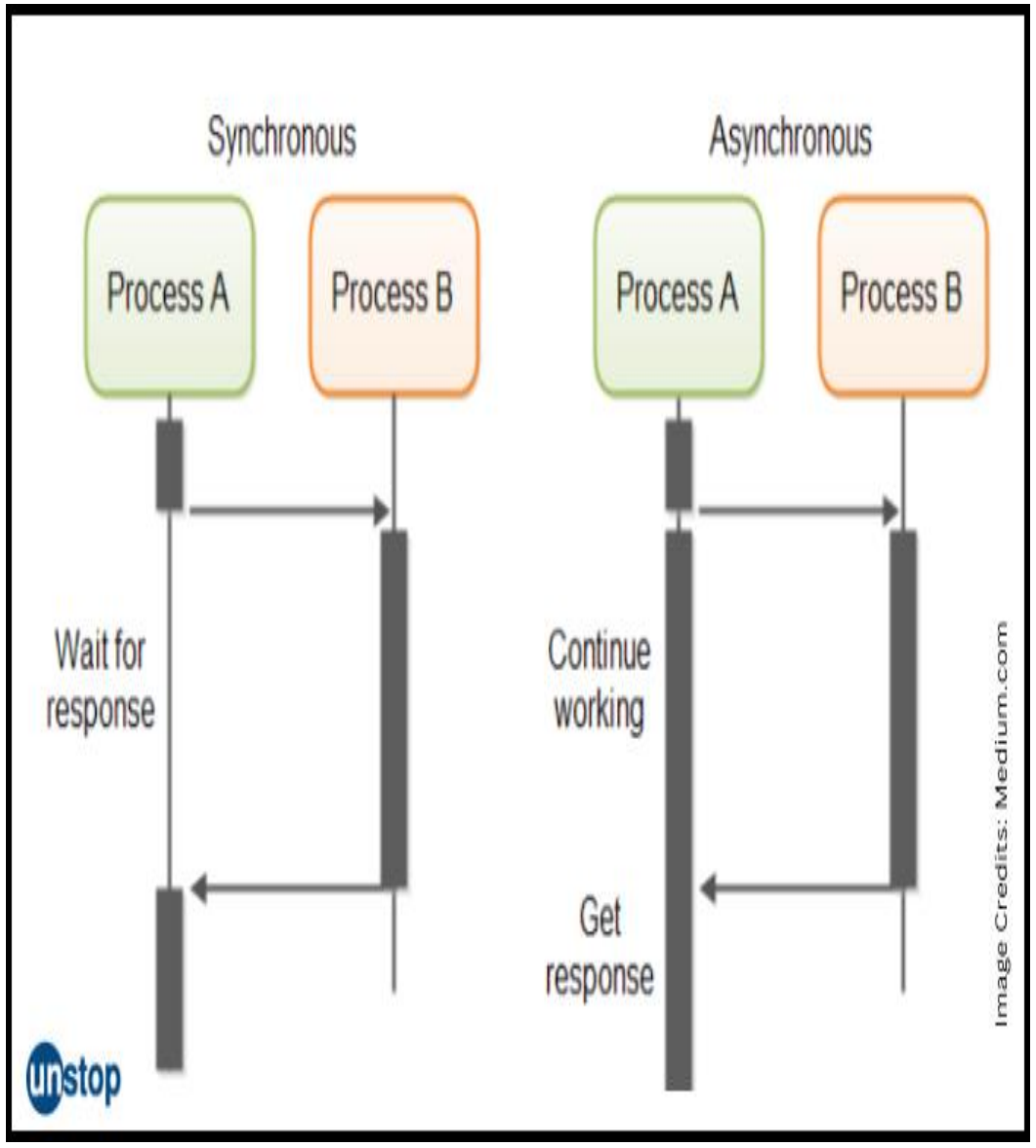
- Counters are broadly divided into two categories
- Asynchronous counter: In asynchronous counter we don’t use universal clock, only first flip flop is driven by main clock and the clock input of rest of the following flip flop is driven by output of previous flip flops.
- Synchronous counter: Unlike the asynchronous counter, synchronous counter has one global clock which drives each flip flop so output changes in parallel. The one advantage of synchronous counter over asynchronous counter is, it can operate on higher frequency than asynchronous counter as it does not have cumulative delay because of same clock is given to each flip flop.

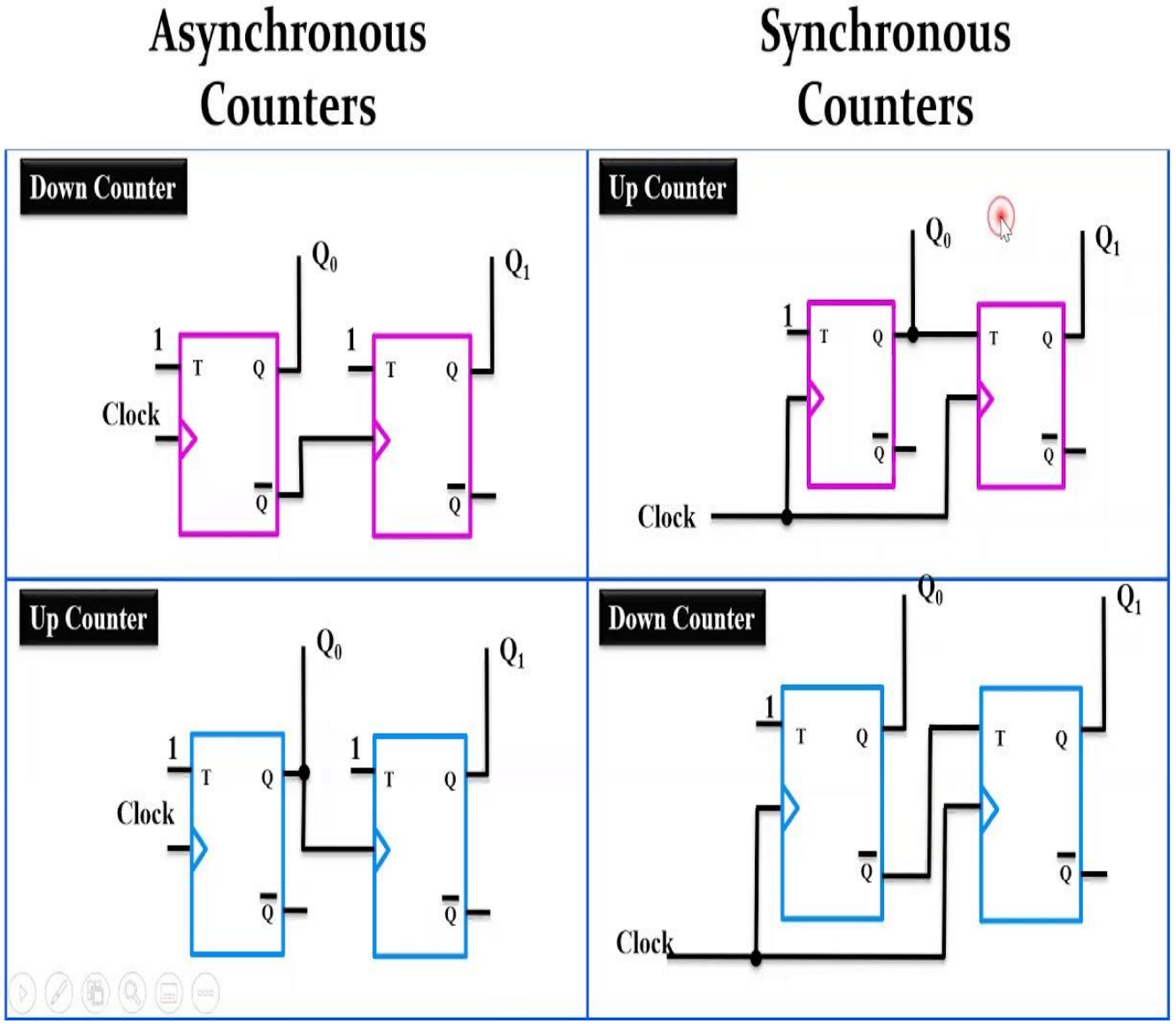
# Asynchronous Counter




It is evident from timing diagram that Q0 is changing as soon as the rising edge of clock pulse is encountered, Q1 is changing when rising edge of Q0 is encountered(because Q0 is like clock pulse for second flip flop) and so on. In this way ripples are generated through Q0,Q1,Q2,Q3 hence it is also called **RIPPLE counter.** A ripple counter is a cascaded arrangement of flip flops where the output of one flip flop drives the clock input of the following flip flop

# Async & Sync counter differences



# Sync Vs Async counters

| Synchronous Counter | Asynchronous Counter |
| --- | --- |
| The counter is clocked in such a way that each flip-flop in the counter is triggered by the same clock signal at the same time. | Only the first flip-flop is clocked by an external clock which in turn drives the clock output of the following flip-flop. |
| All the flip-flops in the counter change state at the same time in sync with the input clock signal. | The clock input of the flip-flops is not driven by the same clock signal. |
| It can operate at much higher clock frequencies than its asynchronous counterpart. | It is slower in operation than synchronous counters. |
| The design involves a complex logic circuit as the number of states increases. | Logic circuit is quite simple even for more number of states. |
| There is no inherent propagation delay in synchronous counters. | A subsequent propagation delay is encountered from one flip-flop to another. |

- Kindly Refer suggested Digital Electronics course Text Books and other online like NPTEL resources for more design practice with full focus…
- **Kumar, Anand. A., Fundamentals of Digital Circuits.**
- **Digital design by M. Morris mano**
- **Fundamentals of Logic Design by Charles H. Roth**
- **Digital design by John F. Wakerly**
- **Fletcher, W.L., An Engineering Approach to Digital design**
- **Taub, H and D. Schilling, Digital Integrated Electronics & many more**
- **Try to study Literatures to grab more stuff & knowledge…**

- *It's the right time to strain your nerves and show your Potentials…*
- Happy & Safe Learning…
- Pasulurri Sweta
- Binduswetha.ece@gmail.com

Thanking You