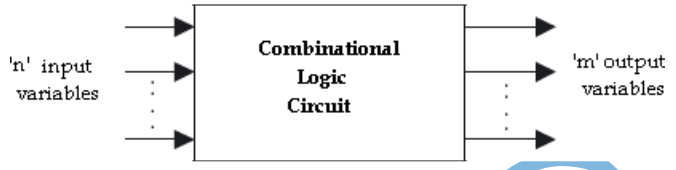
# UNIT-II

## SYNCHRONOUS SEQUENTIAL LOGIC

### INTRODUCTION

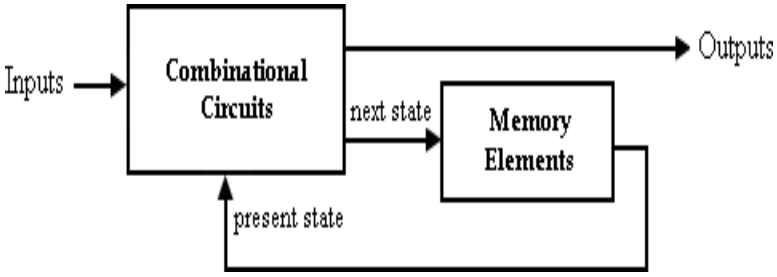
In ***combinational logic circuits***, the outputs at any instant of time depend only on the input signals present at that time. For any change in input, the output occurs immediately.




In ***sequential logic circuits***, it consists of combinational circuits to which storage elements are connected to form a feedback path. The storage elements are devices capable of storing binary information either 1 or 0.

The information stored in the memory elements at any given time defines the present state of the sequential circuit. The present state and the external circuit determine the output and the next state of sequential circuits.

| S.No | Combinational logic | Sequential logic |
|---|---|---|
| 1 | The output variable, at all times depends on the combination of input variables. | The output variable depends not only on the present input but also depend upon the past history of inputs. |
| 2 | Memory unit is not required. | Memory unit is required to store the past history of input variables. |
| 3 | Faster in speed. | Slower than combinational circuits. |
| 4 | Easy to design. | Comparatively harder to design. |
| 5 | Eg. Adder, subtractor, Decoder, Encoders, Magnitude comparator | Eg. Shift registers, Counters |



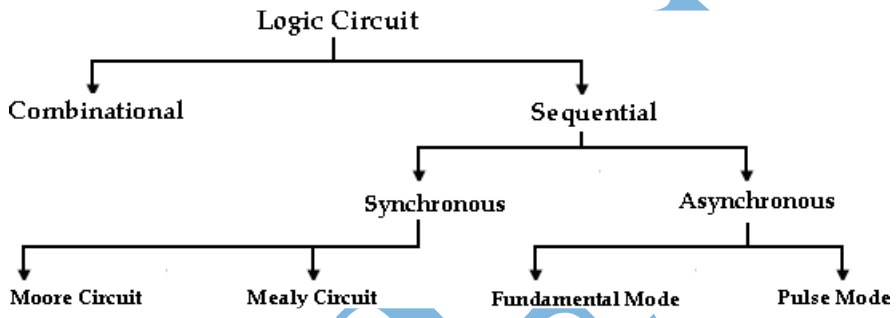

APEC

Thus in sequential circuits, the output variables depend not only on the present input variables but also on the past history of input variables

The rotary channel selected knob on an old-fashioned TV is like a combinational. Its output selects a channel based only on its current input – the position of the knob. The channel-up and channel-down push buttons on a TV is like a sequential circuit. The channel selection depends on the past sequence of up/down pushes.

The comparison between combinational and sequential circuits is given in table below.

## CLASSIFICATION OF LOGIC CIRCUITS




The sequential circuits can be classified depending on the timing of their signals:

- Synchronous sequential circuits
- Asynchronous sequential circuits.

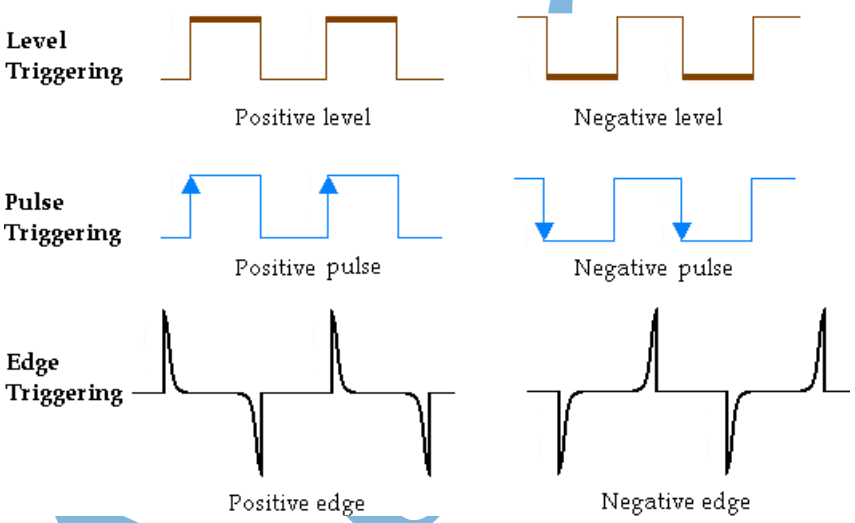
| S.No | Synchronous sequential circuits | Asynchronous sequential circuits |
|---|---|---|
| 1 | Memory elements are clocked Flip-Flops. | Memory elements are either unclocked flip-flops (Latches) or time delay elements. |
| 2 | The change in input signals can affect memory element upon activation of clock signal. | The change in input signals can affect memory element at any instant of time. |
| 3 | The maximum operating speed of clock depends on time delays involved. | Because of the absence of clock, it can operate faster than synchronous circuits. |

| | | |
|---|---|---|
| 4 | Easier to design | More difficult to design |

## TRIGGERING OF FLIP-FLOPS

The state of a Flip-Flop is switched by a momentary change in the input signal. This momentary change is called a trigger and the transition it causes is said to trigger the Flip-Flop. Clocked Flip-Flops are triggered by pulses. A clock pulse starts from an initial value of 0, goes momentarily to 1and after a short time, returns to its initial 0 value.

Latches are controlled by enable signal, and they are level triggered, either positive level triggered or negative level triggered. The output is free to change according to the S and R input values, when active level is maintained at the enable input.




## EDGE TRIGGERED FLIP-FLOPS

Flip-Flops are synchronous bistable devices (has two outputs Q and Q'). In this case, the term synchronous means that the output changes state only at a specified point on the triggering input called the clock (CLK), i.e., changes in the output occur in synchronization with the clock.

An ***edge-triggered Flip-Flop*** changes state either at the positive edge (rising

edge) or at the negative edge (falling edge) of the clock pulse and is sensitive to its inputs only at this transition of the clock. The different types of edge-triggered Flip- Flops are—

S-R Flip-Flop (**S**et – **R**eset)

J-K Flip-Flop (**J**ack **K**ilby)

D Flip-Flop (**D**elay)

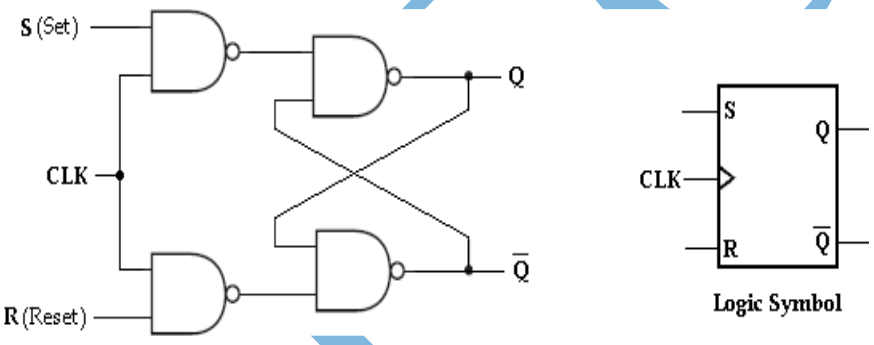
T Flip-Flop (**T**oggle)

Although the S-R Flip-Flop is not available in IC form, it is the basis for the D and J-K Flip-Flops. Each type can be either positive edge-triggered (no bubble at C input) or negative edge-triggered (bubble at C input).

The key to identifying an edge- triggered Flip-Flop by its logic symbol is the small triangle inside the block at the clock (C) input. This triangle is called the **dynamic input indicator**.

## S-R Flip-Flop

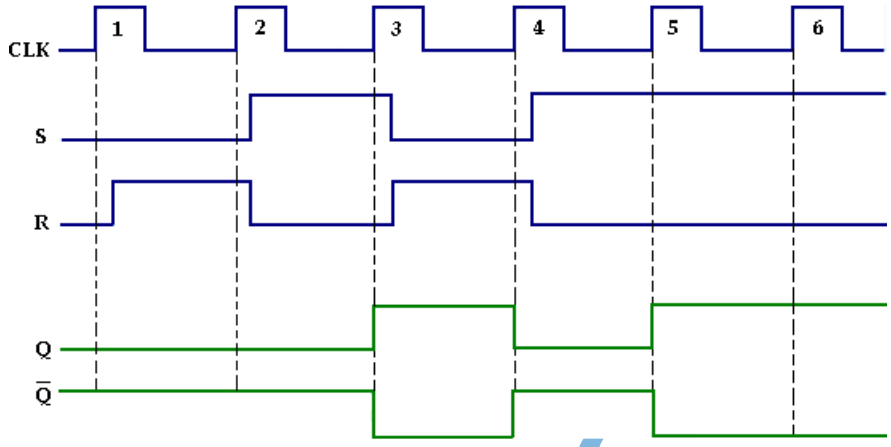
The S and R inputs of the S-R Flip-Flop are called ***synchronous*** inputs because data on these inputs are transferred to the Flip-Flop's output only on the triggering edge of the clock pulse. The circuit is similar to SR latch except enable signal is replaced by clock pulse (CLK). On the positive edge of the clock pulse, the circuit responds to the S and R inputs.




When S is HIGH and R is LOW, the Q output goes HIGH on the triggering edge of the clock pulse, and the Flip-Flop is SET. When S is LOW and R is HIGH, the Q output goes LOW on the triggering edge of the clock pulse, and the Flip-Flop is RESET. When both S and R are LOW, the output does not change from its prior state. An invalid condition exists when both S and R are HIGH

| CLK | S | R | Qn | Qn+1 | State |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | No Change (NC) |
| 1 | 0 | 0 | 1 | 1 | |
| 1 | 0 | 1 | 0 | 0 | Reset |
| 1 | 0 | 1 | 1 | 0 | |
| 1 | 1 | 0 | 0 | 1 | Set |
| 1 | 1 | 0 | 1 | 1 | |

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | x | Indeterminate |
| 1 | 1 | 1 | 1 | x | * |




**Characteristic table and Characteristic equation:**

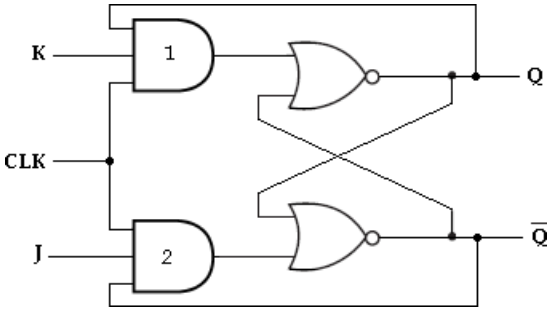
The characteristic table for JK Flip-Flop is shown in the table below. From the table, K-map for the next state transition (Qn+1) can be drawn and the simplified logic expression which represents the characteristic equation of JK Flip-Flop can be found.

| S | R | Qn | Qn +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 x |
| 1 | 1 | 1 | |

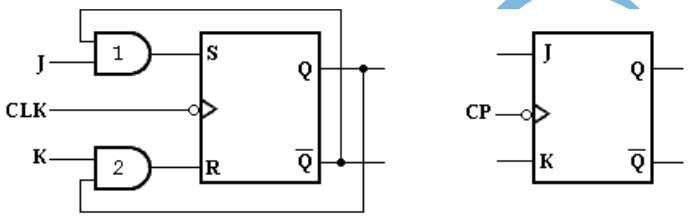
**Qn+1= S+ R'Qn**

**J-K Flip-Flop**

JK means Jack Kilby, Texas Instrument (TI) Engineer, who invented IC in 1958. JK Flip-Flop has two inputs J(set) and K(reset). A JK Flip-Flop can be obtained from the clocked SR Flip-Flop by augmenting two AND gates as shown below.




The data input J and the output Q' are applied o the first AND gate and its output (JQ') is applied to the S input of SR Flip-Flop. Similarly, the data input K and the output Q are applied to the second AND gate and its output (KQ) is applied to the R input of SR Flip-Flop




(a) Using SR flipflop (b) Graphic symbol

**J= K= 0**

When J=K= 0, both AND gates are disabled. Therefore clock pulse have no effect, hence the Flip-Flop output is same as the previous output.

**J= 0, K= 1**

When J= 0 and K= 1, AND gate 1 is disabled i.e., S= 0 and R= 1. This condition will reset the Flip-Flop to 0.

**J= 1, K= 0**

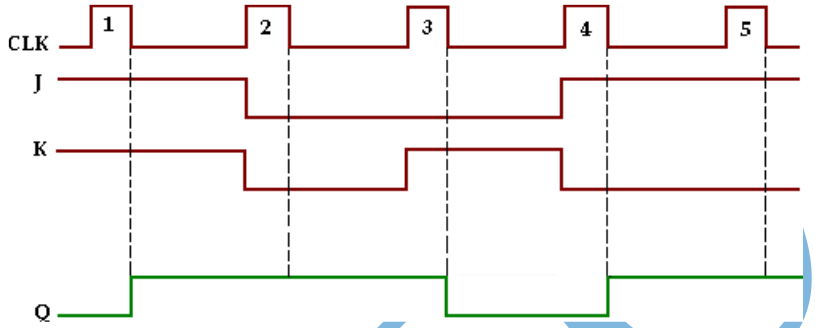
When J= 1 and K= 0, AND gate 2 is disabled i.e., S= 1 and R= 0. Therefore the Flip-Flop will set on the application of a clock pulse.

**J= K= 0**

When J=K= 1, it is possible to set or reset the Flip-Flop. If Q is High, AND gate 2 passes on a reset pulse to the next clock. When Q is low, AND gate 1 passes on a set pulse to the next clock. Eitherway, Q changes to the complement of the last state i.e., toggle. Toggle means to switch to the opposite state

| CLK | Inputs | Output | State |
|---|---|---|---|
| J | K | Qn+1 | |

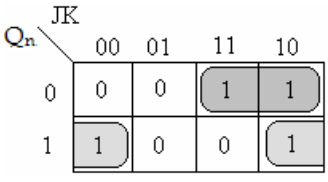
| | | | | |
|---|---|---|---|---|
| 1 | 0 | 0 | Qn | No Change |
| 1 | 0 | 1 | 0 | Reset |
| 1 | 1 | 0 | 1 | Set |
| 1 | 1 | 1 | Qn’ | Toggle |




**Characteristic table and Characteristic equation:**

The characteristic table for JK Flip-Flop is shown in the table below. From the table, K-map for the next state transition (Qn+1) can be drawn and the simplified logic expression which represents the characteristic equation of JK Flip-Flop can be found.

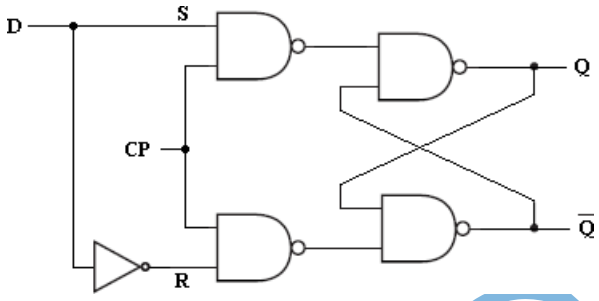
| **Qn** | **J** | **K** | **Qn +1** |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |




**Qn+1= JQ’+ K’Q**

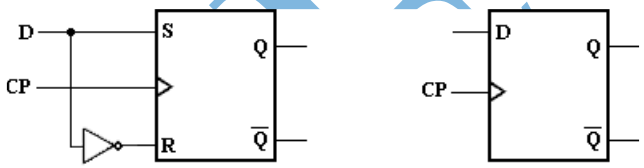
## D Flip-Flop

Like in D latch, in D Flip-Flop the basic SR Flip-Flop is used with complemented inputs. The D Flip-Flop is similar to D-latch except clock pulse is used instead of enable input.



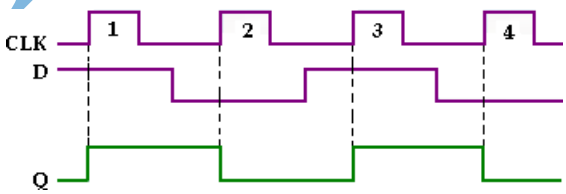

To eliminate the undesirable condition of the indeterminate state in the RS Flip-Flop is to ensure that inputs S and R are never equal to 1 at the same time. This is done by D Flip-Flop. The D (***delay***) Flip-Flop has one input called delay input and clock pulse input.

The D Flip-Flop using SR Flip-Flop is shown below




(a) Using SR flipflop (b) Graphic symbol

| Clock | D | Qn+1 | State |
|---|---|---|---|
| 1 | 0 | 0 | Reset |
| 1 | 1 | 1 | Set |
| 0 | x | Qn | No Change |

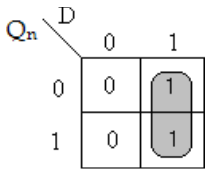



### Characteristic table and Characteristic equation:

The characteristic table for D Flip-Flop shows that the next state of the Flip- Flop is independent of the present state since Qn+1 is equal to D. This means that an input pulse will transfer the value of input D into the output of the Flip-Flop independent of the value of the output before the pulse was applied.

The characteristic equation is derived from K-map.

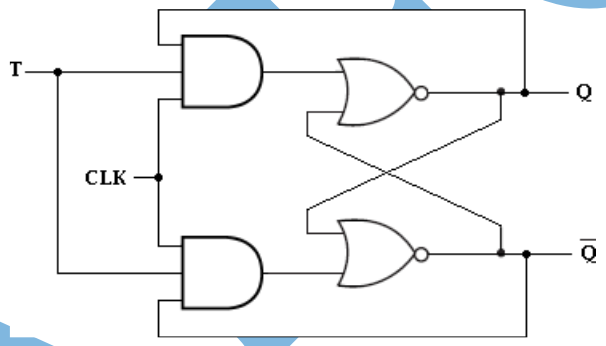
| Qn | D | Qn +1 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |




**Qn+1= D.**

## T Flip-Flop

The T (***Toggle***) Flip-Flop is a modification of the JK Flip-Flop. It is obtained from JK Flip-Flop by connecting both inputs J and K together, i.e., single input. Regardless of the present state, the Flip-Flop complements its output when the clock pulse occurs while input T= 1.

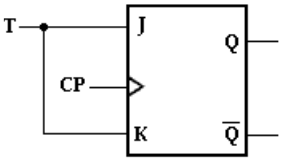



When T= 0, Qn+1= Qn, ie., the next state is the same as the present state and no change occurs.

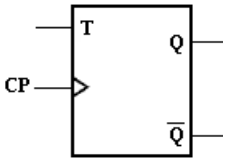
When T= 1, Qn+1= Qn',ie., the next state is the complement of the present state.

**Truth Table:**

| T | Qn+1 | State |
|---|---|---|
| 0 | Qn | No Change |
| 1 | Qn' | Toggle |




(a) Using JK flipflop



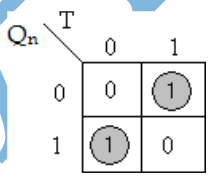

(b) Graphic symbol

| T | Qn+1 | State |
|---|---|---|
| 0 | Qn | No Change |
| 1 | Qn’ | Toggle |

**Characteristic table and Characteristic equation:**

The characteristic table for T Flip-Flop is shown below and characteristic equation is derived using K-map.

| Qn | T | Qn +1 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

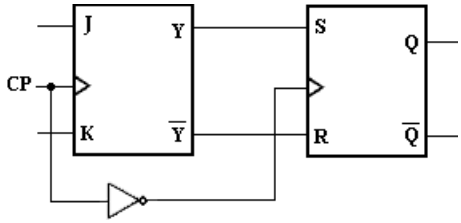



**Qn+1= TQn’+ T’Qn**

**Master-Slave JK Flip-Flop**

A master-slave Flip-Flop consists of clocked JK flip-flop as a master and clocked SR flip-flop as a slave. The output of the master flip-flop is fed as an input to the slave flip-flop. Clock signal is connected directly to the master flip-flop, but is connected through inverter to the slave flip-flop. Therefore, the information present at the J and K inputs is transmitted to the output of master flip-flop on the positive clock pulse and it is held there until the negative clock pulse occurs, after

which it is allowed to pass through to the output of slave flip-flop. The output of the slave flip- flop is connected as a third input of the master JK flip-flop.

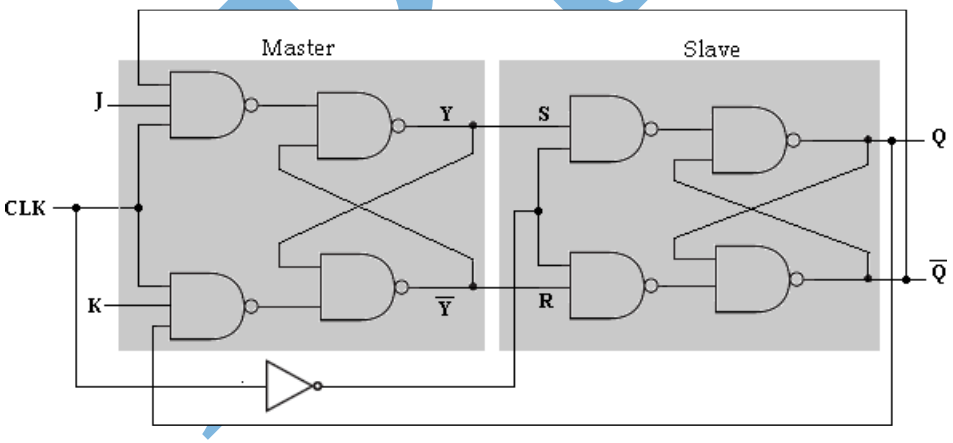


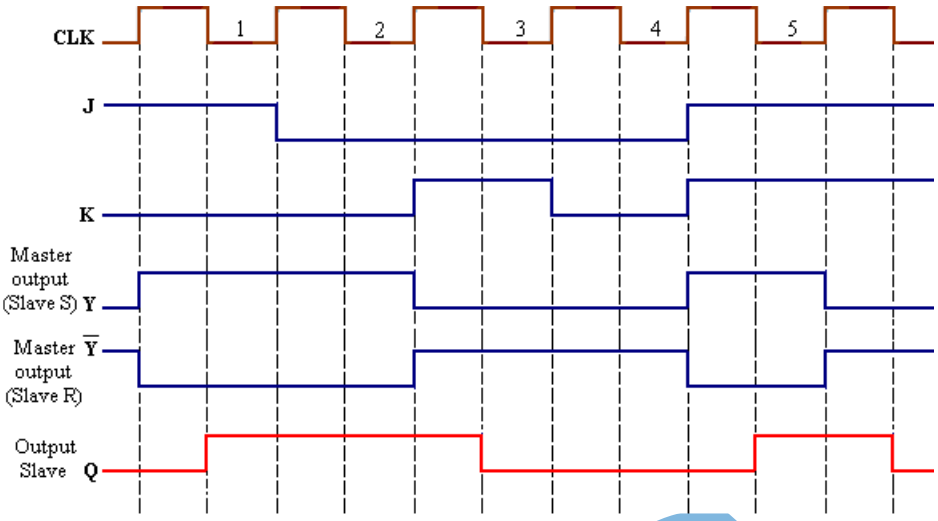

When **J= 1 and K= 0**, the master sets on the positive clock. The high Y output of the master drives the S input of the slave, so at negative clock, slave sets, copying the action of the master.

When **J= 0 and K= 1**, the master resets on the positive clock. The high Y' output of the master goes to the R input of the slave. Therefore, at the negative clock slave resets, again copying the action of the master.

When **J= 1 and K= 1**, master toggles on the positive clock and the output of master is copied by the slave on the negative clock. At this instant, feedback inputs to the master flip-flop are complemented, but as it is negative half of the clock pulse, master flip-flop is inactive. This prevents **race around condition**.

The clocked master-slave J-K Flip-Flop using NAND gate is shown below.






## APPLICATION TABLE (OR) EXCITATION TABLE

The ***characteristic table*** is useful for **analysis** and for defining the operation of the Flip-Flop. It specifies the next state (Qn+1) when the inputs and present state are known.

The ***excitation or application table*** is useful for **design** process. It is used to find the Flip-Flop input conditions that will cause the required transition, when the present state (Qn) and the next state (Qn+1) are known.

| **Present State** | **Inputs** | | **Next State** |
|---|---|---|---|
| **Qn** | **S** | **R** | **Qn+1** |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | x |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | x |

| Present State | Next State | Inputs | Inputs |
|---|---|---|---|
| Qn | Qn+1 | S | R |
| 0 | 0 | 0 | x |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | x |

| Present State | Next State | Inputs |
|---|---|---|
| Qn | Qn+1 | S R |
| 0 | 0 | 0 x |
| 0 | 1 | 1 0 |
| 1 | 0 | 0 1 |
| 1 | 1 | x 0 |

The above table presents the excitation table for SR Flip-Flop. It consists of present state (Qn), next state (Qn+1) and a column for each input to show how the required transition is achieved.

There are 4 possible transitions from present state to next state. The required Input conditions for each of the four transitions are derived from the information available in the characteristic table. The symbol 'x' denotes the don't care condition; it does not matter whether the input is 0 or 1.

| Present State | Inputs | | Next State |
|---|---|---|---|
| Qn | J | K | Qn+1 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

APEC

1 1 1 0

PEC

## D FLIP FLOP

| Present State | Input | Next State |
|---|---|---|
| $Q_n$ | D | $Q_{n+1}$ |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| Present State | Next State | Input |
|---|---|---|
| $Q_n$ | $Q_{n+1}$ | D |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## T FLIP FLOP

| Present State | Input | Next State |
|---|---|---|

| Qn | T | Qn+1 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| Present State Qn | Next State Qn+1 | Input T |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## REALIZATION OF ONE FLIP-FLOP USING OTHER FLIP-FLOPS

It is possible to convert one Flip-Flop into another Flip-Flop with some additional gates or simply doing some extra connection. The realization of one Flip- Flop using other Flip-Flops is implemented by the use of characteristic tables and excitation tables. Let us see few conversions among Flip-Flops.

- SR Flip-Flop to D Flip-Flop
- SR Flip-Flop to JK Flip-Flop
- SR Flip-Flop to T Flip-Flop
- JK Flip-Flop to T Flip-Flop
- JK Flip-Flop to D Flip-Flop
- D Flip-Flop to T Flip-Flop
- T Flip-Flop to D Flip-Flop

**SR Flip-Flop to D Flip-Flop:**

- Write the characteristic table for required Flip-Flop (D Flip-Flop).
- Write the excitation table for given Flip-Flop (SR Flip-Flop).
- Determine the expression for given Flip-Flop inputs (S & R) by using K- map.
- Draw the Flip-Flop conversion logic diagram to obtain the required flip-flop (D Flip-Flop) by using the above obtained expression.

| Required Flip-Flop (D) | | | Given Flip-Flop (SR) | |
|---|---|---|---|---|
| Input | Present state | Next state | Flip-Flop Inputs | |
| D | Qn | Qn+1 | S | R |

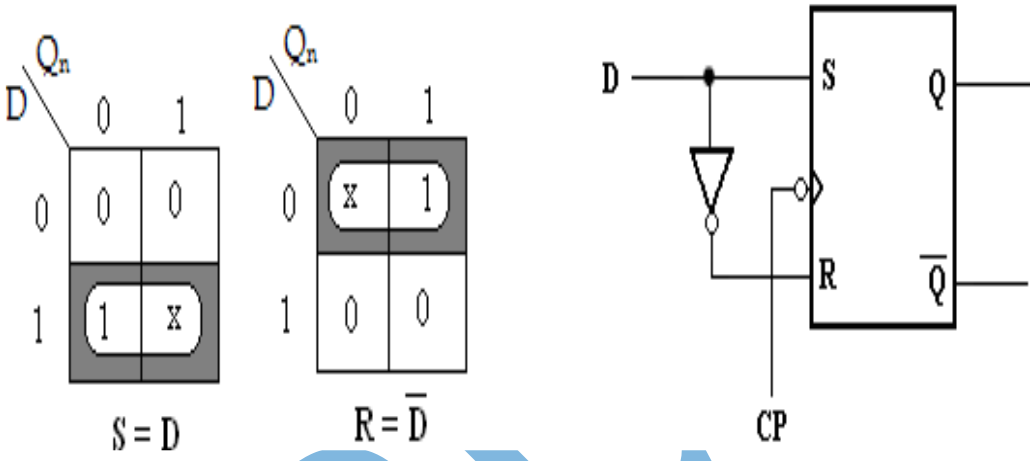
| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | x |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | x | 0 |

K-map simplification

Logic diagram

For S

For R




**SR Flip-Flop to JK Flip-Flop**

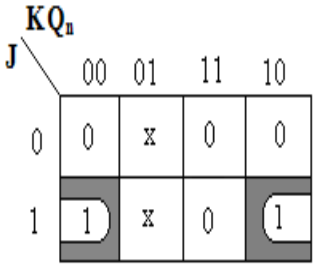
The excitation table for the above conversion is, Qn

| **Inputs** | | **Present state** | **Next state** | **Flip-Flop Inputs** | |
|---|---|---|---|---|---|
| **J** | **K** | **Qn** | **Qn+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 |

APL

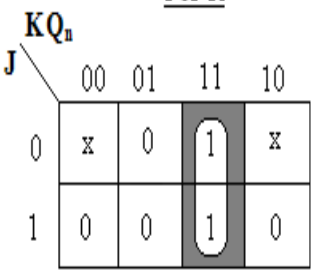
K-map simplification

For S



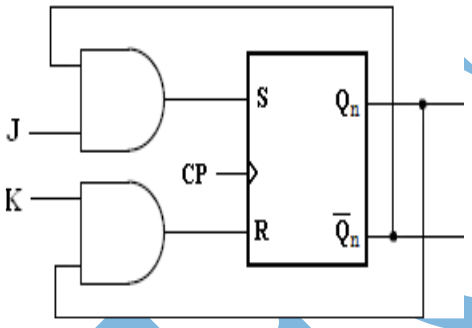

$S = J\overline{Q}_n$

For R




$R = KQ_n$

Logic diagram



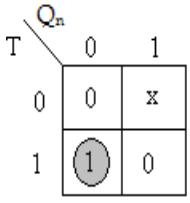

## SR Flip-Flop to T Flip-Flop

The excitation table for the above conversion is

| Input | Present state | Next state | Flip-Flop Inputs | |
|---|---|---|---|---|
| **T** | **Qn** | **Qn+1** | **S** | **R** |
| 0 | 0 | 0 | 0 | x |
| 0 | 1 | 1 | x | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

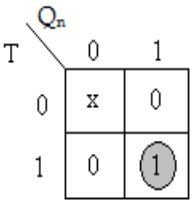
K-map simplification

For S



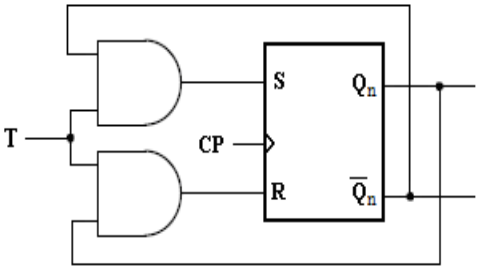

$S = T\overline{Q}_n$

For R




$R = TQ_n$

Logic diagram

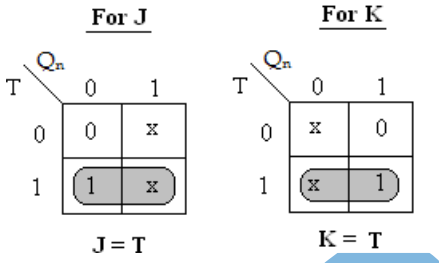


## JK Flip-Flop to T Flip-Flop

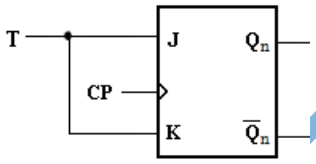
The excitation table for the above conversion is

| Input | Present state | Next state | Flip-Flop Inputs | |
|---|---|---|---|---|
| **T** | **Qn** | **Qn+1** | **J** | **K** |
| 0 | 0 | 0 | 0 | x |
| 0 | 1 | 1 | x | 0 |
| 1 | 0 | 1 | 1 | x |
| 1 | 1 | 0 | x | 1 |

K-map simplification




Logic diagram

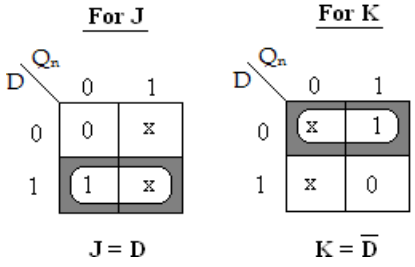



## JK Flip-Flop to D Flip-Flop

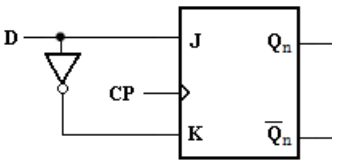
The excitation table for the above conversion is

| Input | Present state | Next state | Flip-Flop Inputs | |
|---|---|---|---|---|
| **D** | **Qn** | **Qn+1** | **J** | **K** |
| 0 | 0 | 0 | 0 | x |
| 0 | 1 | 0 | x | 1 |
| 1 | 0 | 1 | 1 | x |
| 1 | 1 | 1 | x | 0 |

K-map simplification




Logic diagram

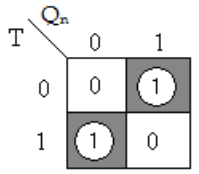



## D Flip-Flop to T Flip-Flop

The excitation table for the above conversion is

| Input | Present state | Next state | Flip-Flop Input |
|---|---|---|---|
| **T** | **Qn** | **Qn+1** | **D** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

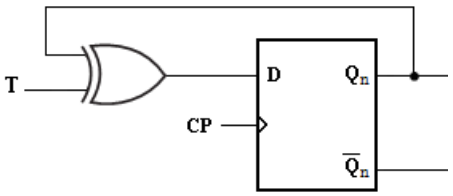
K-map simplification




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

$= T \oplus Q_n$

Logic diagram

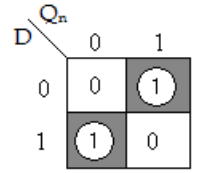



## T Flip-Flop to D Flip-Flop

The excitation table for the above conversion is

| Input | Present state | Next state | Flip-Flop Input |
|---|---|---|---|
| **D** | **Qn** | **Qn+1** | **T** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 |

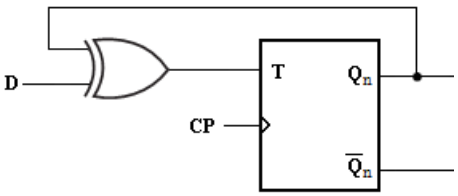
K-map simplification




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

$= D \oplus Q_n$

Logic diagram




## CLASSIFICATION OF SYNCHRONOUS SEQUENTIAL CIRCUIT

In synchronous or clocked sequential circuits, clocked Flip-Flops are used as memory elements, which change their individual states in synchronism with the periodic clock signal. Therefore, the change in states of Flip-Flop and change in state of the entire circuits occur at the transition of the clock signal.

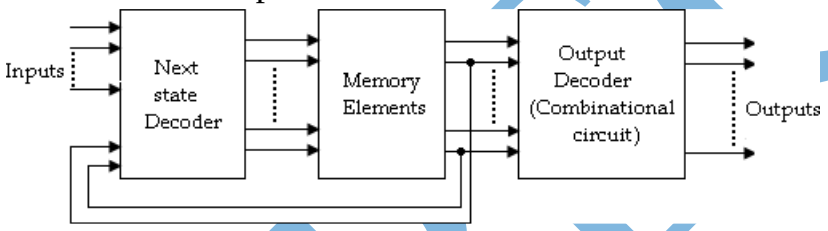
The synchronous or clocked sequential networks are represented by two models.

**Moore model:** The output depends only on the present state of the Flip-Flops.

**Mealy model:** The output depends on both the present state of the Flip-Flops and on the inputs.

**Moore model:**

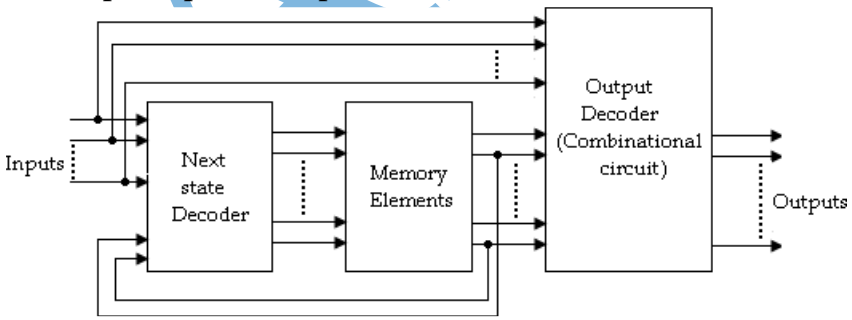
In the Moore model, the outputs are a function of the present state of the Flip- Flops only. The output depends only on present state of Flip-Flops, it appears only after the clock pulse is applied, i.e., it varies in synchronism with the clock input




**Mealy model:**

**Moore model**

In the Mealy model, the outputs are functions of both the present state of the Flip-Flops and inputs.



| S.No | Moore model | Mealy model |
|---|---|---|
| 1 | Its output is a function of present state only. | Its output is a function of present state as well as present input. |
| 2 | An input change does not affect the output. | Input changes may affect the output of the circuit. |
| 3 | It requires more number of states for implementing same function. | It requires less number of states for implementing same function. |

## ANALYSIS OF SYNCHRONOUS SEQUENTIAL CIRCUIT

### ANALYSIS PROCEDURE:

The synchronous sequential circuit analysis is summarizes as given below:

1. Assign a state variable to each Flip-Flop in the synchronous sequential circuit.

2. Write the excitation input functions for each Flip-Flop and also write the Moore/ Mealy output equations.

3. Substitute the excitation input functions into the bistable equations for the Flip-Flops to obtain the next state output equations.

4. Obtain the state table and reduced form of the state table.

5. Draw the state diagram by using the second form of the state table.

### ANALYSIS OF MEALY MODEL:

1. A sequential circuit has two JK Flip-Flops A and B, one input (x) and one output (y). the Flip-Flop input functions are,

**JA= B+ x JB= A'+ x'**

**KA= 1 KB= 1**

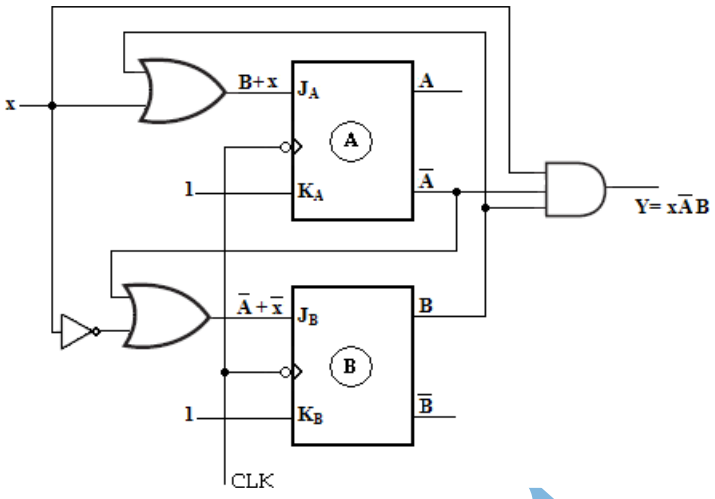
and the circuit output function, **Y= xA'B**.

a) Draw the logic diagram of the Mealy circuit,

b) Tabulate the state table,

c) Draw the state diagram.

APEC

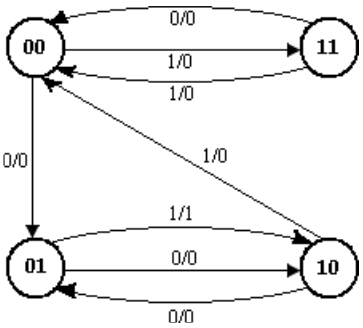



| Presentstate | | Input | Flip-FlopInputs | | | Nextstate | | Output | |
|---|---|---|---|---|---|---|---|---|---|
| **A** | **B** | **x** | **JA=B+x** | **KA=1** | **JB=A'+x'** | **KB=1** | **A(t+1)** | **B(t+1)** | **Y=xA'B** |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |

| Presentstate | | Nextstate | | | | Output | |
|---|---|---|---|---|---|---|---|
| **x=0** | | **x=1** | | **x=0** | | **x=1** | |
| **A** | **B** | **A** | **B** | **A** | **B** | **y** | **y** |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**State Diagram:**

APEC




**2. A sequential circuit with two 'D' Flip-Flops A and B, one input (x) and one output (y). The Flip-Flop input functions are:**

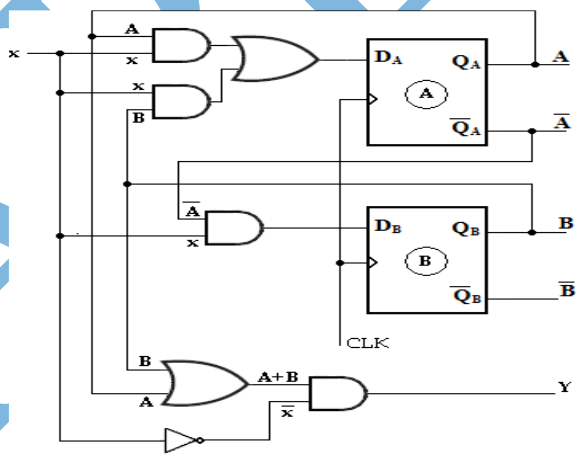
**DA= Ax+ Bx DB= A'x**

and the circuit output function is, **Y= (A+ B) x'**.

(a) Draw the logic diagram of the circuit,

(b) Tabulate the state table,

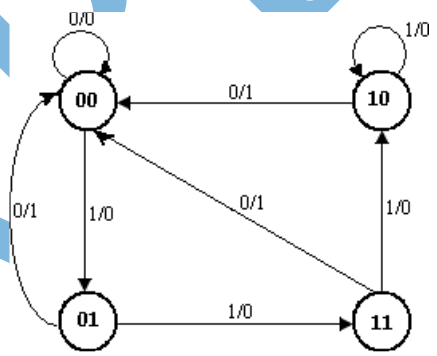
(c) Draw the state diagram



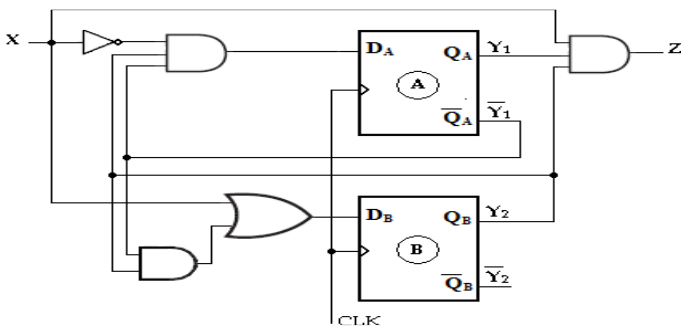

| Present state | | Input | Flip-Flop Inputs | | Next state | | Output |
|---|---|---|---|---|---|---|---|
| **A** | **B** | **x** | **DA= Ax+Bx** | **DB= A'x** | **A(t+1)** | **B(t+1)** | **Y= (A+B) x'** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

| **Present state** | | **Next state** | | | | **Output** | |
|---|---|---|---|---|---|---|---|
| **x= 0** | | **x= 1** | | **x= 0** | | **x= 1** | |
| **A** | **B** | **A** | **B** | **A** | **B** | **Y** | **Y** |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |




**Analyze the synchronous Mealy machine and obtain its state diagram.**



The given synchronous Mealy machine consists of two D Flip-Flops, one inputs and one output. The Flip-Flop input functions are,

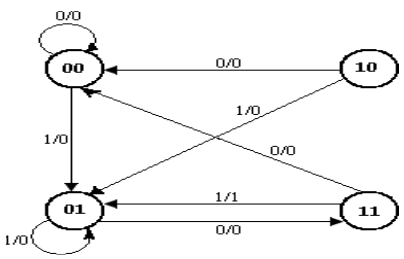
**DA= Y1'Y2X' DB= X+ Y1'Y2**

The circuit output function is, **Z= Y1Y2X.**

| **Present state** | | **Input** | **Flip-Flop Inputs** | | **Next state** | | **Output** |
|---|---|---|---|---|---|---|---|
| **Y1** | **Y2** | **X** | **DA= Y1'Y2 X'** | **DB= X+ Y1'Y2** | **Y1 (t+1)** | **Y2 (t+1)** | **Z= Y1Y2X** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |

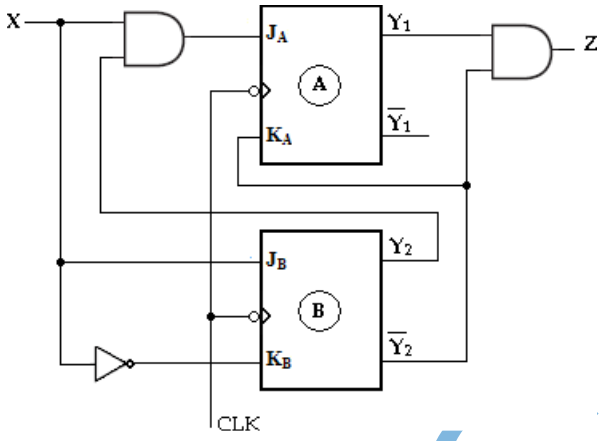
**REDUCED STATE TABLE**

| **Present state** | | **Next state** | | | | **Output** | |
|---|---|---|---|---|---|---|---|
| **X= 0** | | **X= 1** | | **X= 0** | | **X= 1** | |
| **Y1** | **Y2** | **Y1** | **Y2** | **Y1** | **Y2** | **Z** | **Z** |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |



## ANALYSIS OF MOORE MODEL

**Analyze the synchronous Moore circuit and obtain its state diagram**




Using the assigned variable Y1 and Y2 for the two JK Flip-Flops, we can write the four excitation input equations and the Moore output equation as follows:

**JA= Y2X ; KA= Y2’ JB= X ; KB= X’**

and output function**, Z= Y1Y2’**

| **Present state** | | **Input** | **Flip-Flop Inputs** | | | **Next state** | | **Output** |
|---|---|---|---|---|---|---|---|---|
| **Y1** | **Y2** | **X** | **JA= Y2X** | **KA= Y2’** | **JB= X** | **KB= X’** | **Y1 (t+1)** | **Y2 (t+1)** | **Z= Y1Y2’** |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

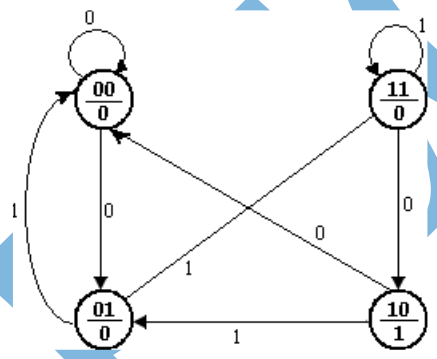
**Present state Next state Output**

APEC

| X= 0 | | X= 1 | | Y | |
|---|---|---|---|---|---|
| **Y1** | **Y2** | **Y1** | **Y2** | **Y1** | **Y2** |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 |

**State Diagram:**

Here the output depends on the present state only and is independent of the input. The two values inside each circle separated by a slash are for the present state and output.




## STATE REDUCTION/ MINIMIZATION

The state reduction is used to avoid the redundant states in the sequential circuits. The reduction in redundant states reduces the number of required Flip- Flops and logic gates, reducing the cost of the final circuit.

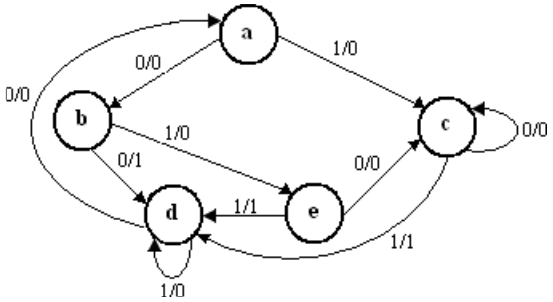
The two states are said to be redundant or equivalent, if every possible set of inputs generate exactly same output and same next state. When two states are equivalent, one of them can be removed without altering the input-output relationship.

Since ‘n’ Flip-Flops produced 2n state, a reduction in the number of states may result in a reduction in the number of Flip-Flops.

The need for state reduction or state minimization is explained with one example.

**Examples**:

1. Reduce the number of states in the following state diagram and draw the reduced state diagram




**Step 1: Determine the state table for given state diagram**

| Present state | Next state | | Output | |
|---|---|---|---|---|
| **X= 0** | **X= 1** | **X= 0** | **X= 1** | |
| a | b | c | 0 | 0 |
| b | d | e | 1 | 0 |
| c | c | d | 0 | 1 |
| d | a | d | 0 | 0 |
| e | c | d | 0 | 1 |

**Step 2: Find equivalent states**

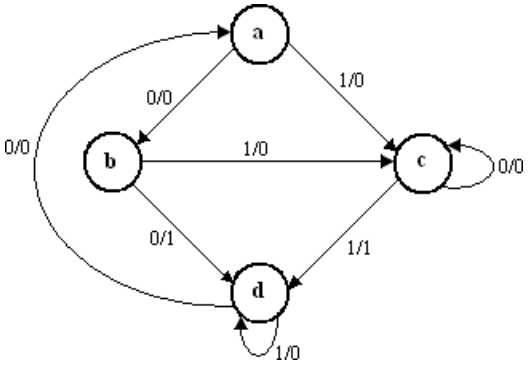
**State table**

From the above state table **c** and **e** generate exactly same next state and same output for every possible set of inputs. The state **c** and **e** go to next states **c** and **d** and have outputs 0 and 1 for x=0 and x=1 respectively. Therefore state **e** can be removed and replaced by **c**. The final reduced state table is shown below.

| Present state | Next state | | Output | |
|---|---|---|---|---|
| **X= 0** | **X= 1** | **X= 0** | **X= 1** | |
| a | b | c | 0 | 0 |
| b | d | c | 1 | 0 |
| c | c | d | 0 | 1 |
| d | a | d | 0 | 0 |

**Step 3: Draw state diagram**

APEC




## DESIGN OF SYNCHRONOUS SEQUENTIAL CIRCUITS:

A synchronous sequential circuit is made up of number of Flip-Flops and combinational gates. The design of circuit consists of choosing the Flip-Flops and then finding a combinational gate structure together with the Flip-Flops. The number of Flip-Flops is determined from the number of states needed in the circuit. The combinational circuit is derived from the state table.

**Design procedure:**

1. The given problem is determined with a state diagram.

2. From the state diagram, obtain the state table.

3. The number of states may be reduced by state reduction methods (if applicable).

4. Assign binary values to each state (Binary Assignment) if the state table contains letter symbols.

5. Determine the number of Flip-Flops and assign a letter symbol (A, B, C,…) to each.

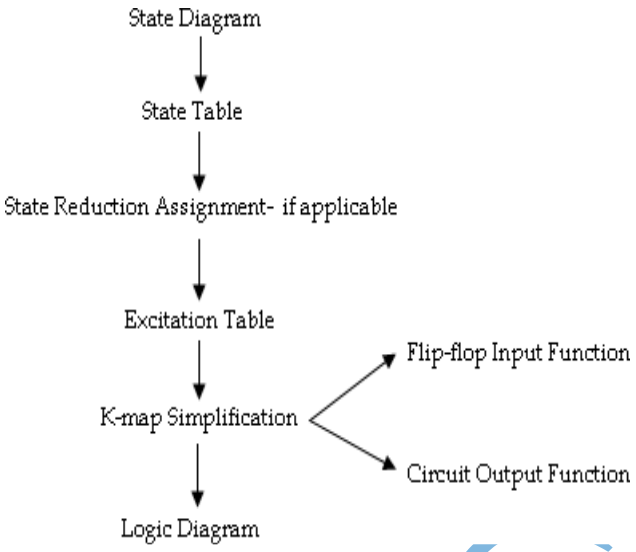
6. Choose the type of Flip-Flop (SR, JK, D, T) to be used.

7. From the state table, circuit excitation and output tables.

8. Using K-map or any other simplification method, derive the circuit output functions and the Flip-Flop input functions.

9. Draw the logic diagram.

APEC




The type of Flip-Flop to be used may be included in the design specifications or may depend what is available to the designer. Many digital systems are constructed with JK Flip-Flops because they are the most versatile available. The selection of inputs is given as follows.

| **Flip-Flop** | **Application** |
|---|---|
| JK | General Applications |
| D | Applications requiring transfer of data (Ex: Shift Registers) |
| T | Application involving complementation (Ex: Binary Counters) |

**Excitation Tables:**

Before going to the design examples for the clocked synchronous sequential circuits we revise Flip-Flop excitation tables.

| **Present State** | **Next State** | **Inputs** | |
|---|---|---|---|
| **Qn** | **Qn+1** | **S** | **R** |

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | x |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | x | 0 |

**Excitation table for SR Flip-Flop**

| **Present State** | **Next State** | **Inputs** | |
|---|---|---|---|
| **Qn** | **Qn+1** | **J** | **K** |
| 0 | 0 | 0 | x |
| 0 | 1 | 1 | x |
| 1 | 0 | x | 1 |
| 1 | 1 | x | 0 |

**Excitation table for JK Flip-Flop**

| **Present State** | **Next State** | **Input** |
|---|---|---|
| **Qn** | **Qn+1** | **T** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Excitation table for T Flip-Flop**

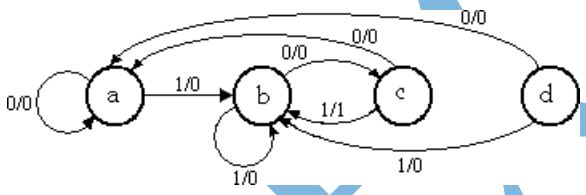
| **Present State** | **Next State** | **Input** |
|---|---|---|

APEC

| Qn | Qn+1 | D |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**Excitation table for D Flip-Flop**

**Design a clocked sequential machine using JK Flip-Flops for the state diagram shown in the figure. Use state reduction if possible. Make proper state assignment.**




**State Table:**

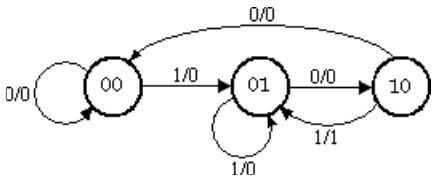
| Present state | Next state | | Output | |
|---|---|---|---|---|
| **X= 0** | **X= 1** | **X= 0** | | **X= 1** |
| a | a | b | 0 | 0 |
| b | c | b | 0 | 0 |
| c | a | b | 0 | 1 |
| d | a | b | 0 | 0 |

**Reduced State Table:**

| Present state | Next state | | Output | |
|---|---|---|---|---|
| **X= 0** | **X= 1** | **X= 0** | | **X= 1** |
| a | a | b | 0 | 0 |
| b | c | b | 0 | 0 |
| c | a | b | 0 | 1 |

**Binary Assignment:**

Now each state is assigned with binary values. Since there are three states, number of Flip-Flops required is two and 2 binary numbers are assigned to the states. **a= 00; b= 0; and c= 10.**

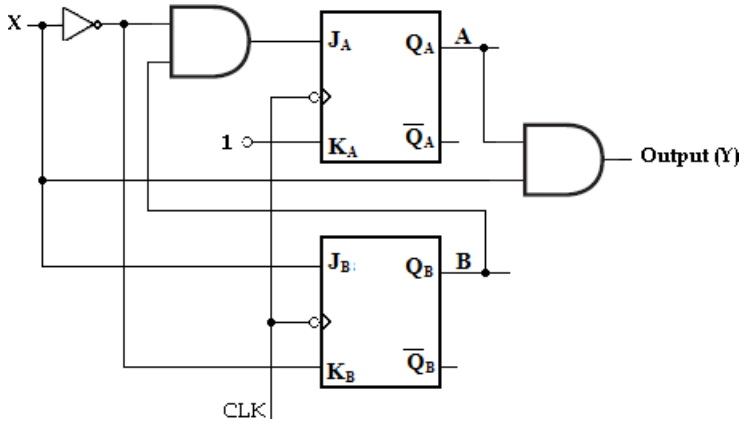



**Excitation Table**

| Present State | Next State | Inputs | |
|---|---|---|---|
| **Qn** | **Qn+1** | **J** | **K** |
| 0 | 0 | 0 | x |
| 0 | 1 | 1 | x |
| 1 | 0 | x | 1 |
| 1 | 1 | x | 0 |

| Input | Present state | | Next state | | Flip-Flop Inputs | | | Output | |
|---|---|---|---|---|---|---|---|---|---|
| **X** | **A** | **B** | **A** | **B** | **JA** | **KA** | **JB** | **KB** | **Y** |
| 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | x | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | x | 1 | x | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | x | x | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | x | x | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | x | 1 | 0 | x | 0 |
| 1 | 1 | 0 | 0 | 1 | x | 1 | 1 | x | 1 |
| 0 | 1 | 1 | x | x | x | x | x | x | x |
| 1 | 1 | 1 | x | x | x | x | x | x | x |

APEC




## SHIFT REGISTERS

A register is simply a group of Flip-Flops that can be used to store a binary number. There must be one Flip-Flop for each bit in the binary number. For instance, a register used to store an 8-bit binary number must have 8 Flip-Flops.

The Flip-Flops must be connected such that the binary number can be entered (shifted) into the register and possibly shifted out. A group of Flip-Flops connected to provide either or both of these functions is called a ***shift register***.

The bits in a binary number (data) can be removed from one place to another

in either of two ways. The first method involves shifting the data one bit at a time in a serial fashion, beginning with either the most significant bit (MSB) or the least significant bit (LSB). This technique is referred to as *serial shifting*. The second

method involves shifting all the data bits simultaneously and is referred to as *parallel*

*shifting*.

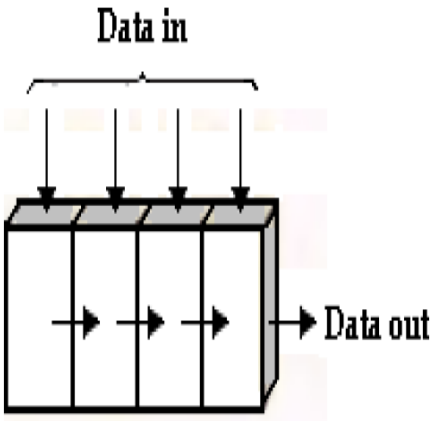
There are two ways to shift into a register (serial or parallel) and similarly two ways to shift the data out of the register. This leads to the construction of four basic register types—

i. Serial in- serial out,

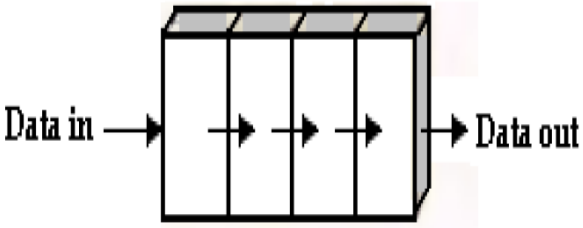
ii. Serial in- parallel out,

iii. Parallel in- serial out,

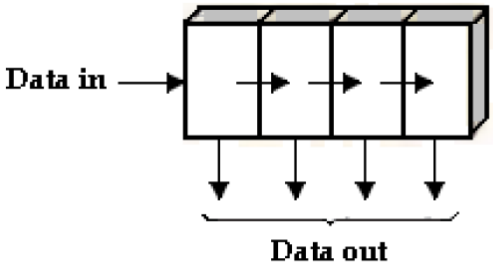
iv. Parallel in- parallel out.



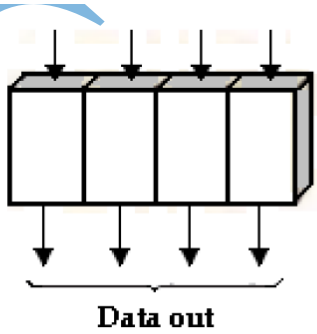

(iii) Parallel in- serial out




(i) Serial in- serial out



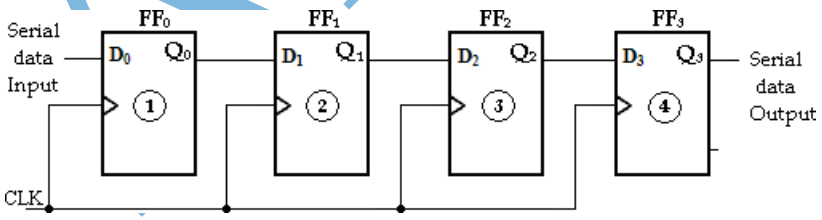

(iii) Serial in- parallel out




(iv) Parallel in- parallel out

**Serial-In Serial-Out Shift Register:**

The serial in/serial out shift register accepts data serially, i.e., one bit at a time on a single line. It produces the stored information on its output also in serial form.




The entry of the four bits 1010 into the register is illustrated below, beginning with the right-most bit. The register is initially clear. The 0 is put onto the data input line, making D=0 for FF0. When the first clock pulse is applied, FF0 is reset, thus storing the 0.

Next the second bit, which is a 1, is applied to the data input, making D=1 for FF0 and D=0 for FF1 because the D input of FF1 is connected to the Q0 output. When the second clock pulse occurs, the 1 on the data input is shifted into FF0, causing FF0 to set; and the 0 that was in FF0 is shifted into FFl.

The third bit, a 0, is now put onto the data-input line, and a clock pulse is applied. The 0 is entered into FF0, the 1 stored in FF0 is shifted into FFl, and the 0 stored in FF1 is shifted into FF2.

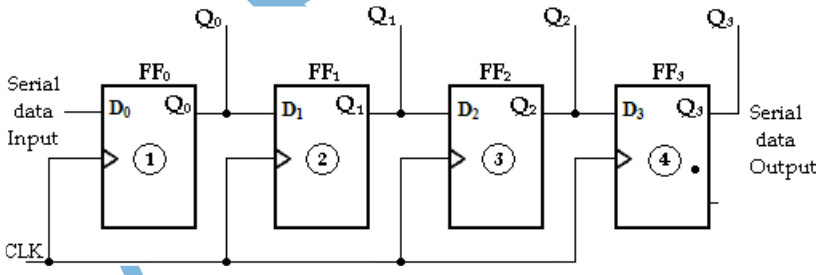
The last bit, a 1, is now applied to the data input, and a clock pulse is applied. This time the 1 is entered into FF0, the 0 stored in FF0 is shifted into FFl, the 1 stored in FF1 is shifted into FF2, and the 0 stored in FF2 is shifted into FF3. This completes the serial entry of the four bits into the shift register, where they can be stored for any length of time as long as the Flip-Flops have dc power.

To get the data out of the register, the bits must be shifted out serially and taken off the Q3 output. After CLK4, the right-most bit, 0, appears on the Q3 output.

When clock pulse CLK5 is applied, the second bit appears on the Q3 output. Clock pulse CLK6 shifts the third bit to the output, and CLK7 shifts the fourth bit to the output. While the original four bits are being shifted out, more bits can be shifted in. All zeros are shown being shifted out, more bits can be shifted in.

**Serial-In Parallel-Out Shift Register:**

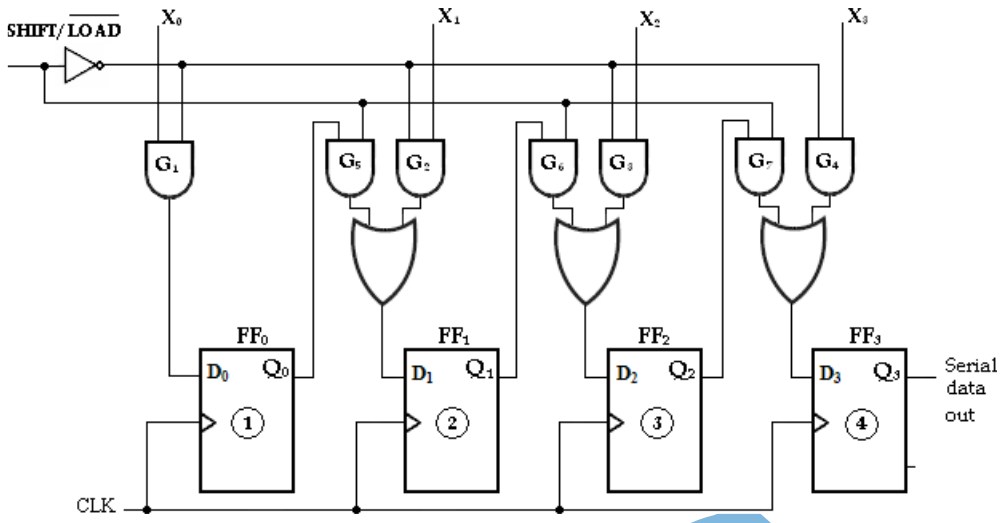
In this shift register, data bits are entered into the register in the same as serial-in serial-out shift register. But the output is taken in parallel. Once the data are stored, each bit appears on its respective output line and all bits are available simultaneously instead of on a bit-by-bit.




**Parallel-In Serial-Out Shift Register:**

In this type, the bits are entered in parallel i.e., simultaneously into their respective stages on parallel lines.

A 4-bit parallel-in serial-out shift register is illustrated below. There are four data input lines, X0, X1, X2 and X3 for entering data in parallel into the register. SHIFT/ LOAD input is the control input, which allows four bits of data to **load** in parallel into the register.

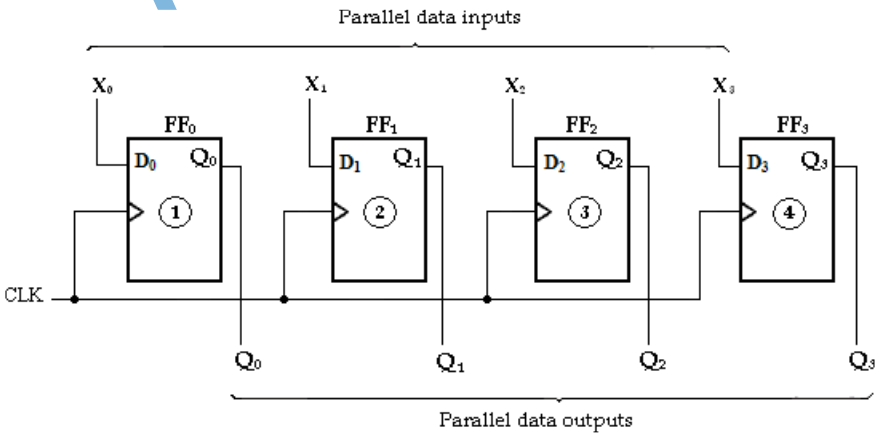



When SHIFT/LOAD is LOW, gates G1, G2, G3 and G4 are enabled, allowing each data bit to be applied to the D input of its respective Flip-Flop. When a clock pulse is applied, the Flip-Flops with D = 1 will **set** and those with D = 0 will **reset,** thereby storing all four bits simultaneously.

When SHIFT/LOAD is HIGH, gates G1, G2, G3 and G4 are disabled and gates G5, G6 and G7 are enabled, allowing the data bits to shift right from one stage to the next. The OR gates allow either the normal shifting operation or the parallel data- entry operation, depending on which AND gates are enabled by the level on the SHIFT/LOAD input.

**Parallel-In Parallel-Out Shift Register:**

In this type, there is simultaneous entry of all data bits and the bits appear on parallel outputs simultaneously.




**UNIVERSAL SHIFT REGISTERS**

If the register has shift and parallel load capabilities, then it is called a shift register with parallel load or ***universal shift register***. Shift register can be used for converting serial data to parallel data, and vice-versa. If a parallel load capability is added to a shift register, the data entered in parallel can be taken out in serial fashion by shifting the data stored in the register.

The functions of universal shift register are:

1. A clear control to clear the register to 0.

2. A clock input to synchronize the operations.

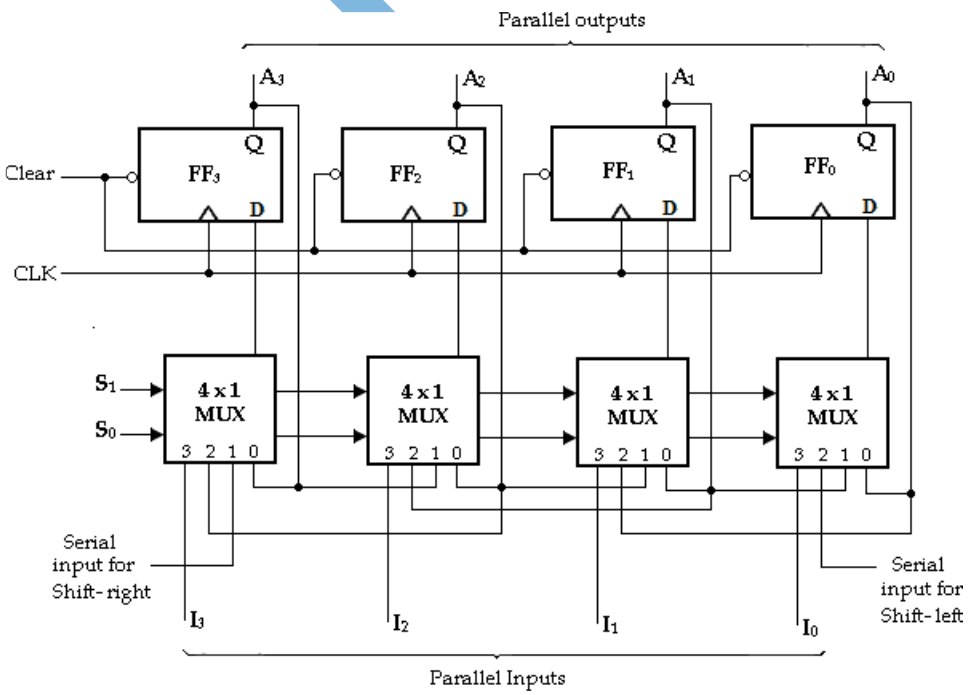
3. A shift-right control to enable the shift right operation and the serial input and output lines associated with the shift right.

4. A shift-left control to enable the shift left operation and the serial input and output lines associated with the shift left.

5. A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.

6. 'n' parallel output lines.

7. A control line that leaves the information in the register unchanged even though the clock pulses re continuously applied.



It consists of four D-Flip-Flops and four 4 input multiplexers (MUX). S0 and S1 are the two selection inputs connected to all the four multiplexers. These two selection inputs are used to select one of the four inputs of each multiplexer

The input 0 in each MUX is selected when S1S0= 00 and input 1 is selected when S1S0= 01. Similarly inputs 2 and 3 are selected when S1S0= 10 and S1S0= 11 respectively. The inputs S1 and S0 control the mode of the operation of the register.

When S1S0= 00, the present value of the register is applied to the D-inputs of the Flip-Flops. This is done by connecting the output of each Flip-Flop to the 0 input of the respective multiplexer. The next clock pulse transfers into each Flip-Flop, the binary value is held previously, and hence no change of state occurs.

When S1S0= 01, terminal 1 of the multiplexer inputs has a path to the D inputs of the Flip-Flops. This causes a shift-right operation with the lefter serial input transferred into Flip-Flop FF3.

When S1S0= 10, a shift-left operation results with the right serial input going into Flip-Flop FF1.

Finally when S1S0= 11, the binary information on the parallel input lines (I1, I2, I3 and I4) are transferred into the register simultaneously during the next clock pulse.

The function table of bi-directional shift register with parallel inputs and parallel outputs is shown below.

| **Mode Control** | | |
|---|---|---|
| **S1** | **S0** | **Operation** |
| 0 | 0 | No change |
| 0 | 1 | Shift-right |
| 1 | 0 | Shift-left |
| 1 | 1 | Parallel load |

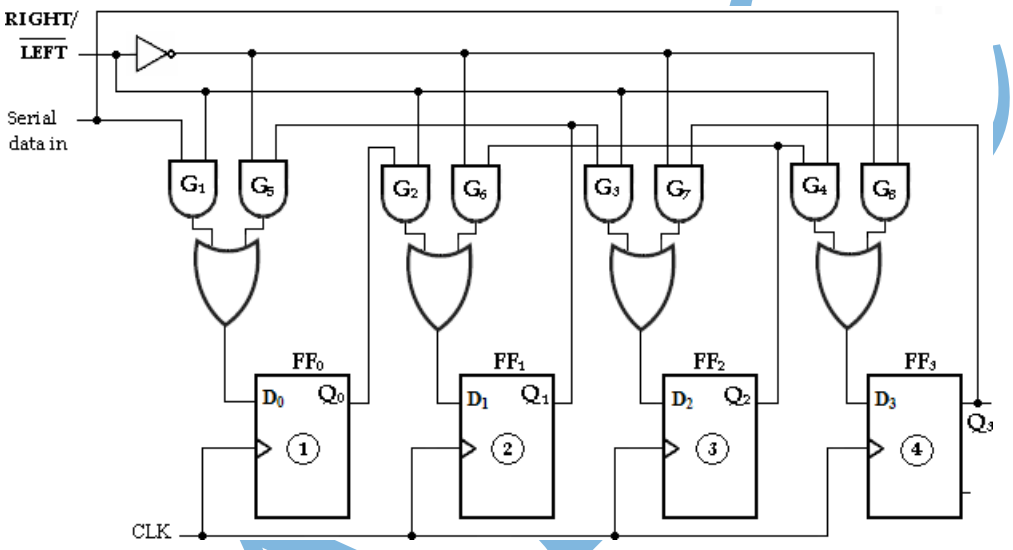
## BI-DIRECTION SHIFT REGISTERS

A bidirectional shift register is one in which the data can be shifted either left or right. It can be implemented by using gating logic that enables the transfer of a data bit from one stage to the next stage to the right or to the left depending on the level of a control line.

A 4-bit bidirectional shift register is shown below. A HIGH on the RIGHT/LEFT control input allows data bits inside the register to be shifted to the right, and a LOW enables data bits inside the register to be shifted to the left.

When the RIGHT/LEFT control input is **HIGH**, gates G1, G2, G3 and G4 are enabled, and the state of the Q output of each Flip-Flop is passed through to the D input of the following Flip-Flop. When a clock pulse occurs, the data bits are shifted one place to the right.

When the RIGHT/LEFT control input is **LOW**, gates G5, G6, G7 and G8 are enabled, and the Q output of each Flip-Flop is passed through to the D input of the preceding Flip-Flop. When a clock pulse occurs, the data bits are then shifted one place to the left.




## SYNCHRONOUS COUNTERS

Flip-Flops can be connected together to perform counting operations. Such a group of Flip- Flops is a **counter**. The number of Flip-Flops used and the way in which they are connected determine the number of states (called the modulus) and also the specific sequence of states that the counter goes through during each complete cycle.

Counters are classified into two broad categories according to the way they are clocked:

Asynchronous counters, Synchronous counters.

In asynchronous (ripple) counters, the first Flip-Flop is clocked by the external clock pulse and then each successive Flip-Flop is clocked by the output of the preceding Flip-Flop.

In synchronous counters, the clock input is connected to all of the Flip-Flops so that they are clocked simultaneously. Within each of these two categories, counters are classified primarily by the type of sequence, the number of states, or the number of Flip-Flops in the counter.

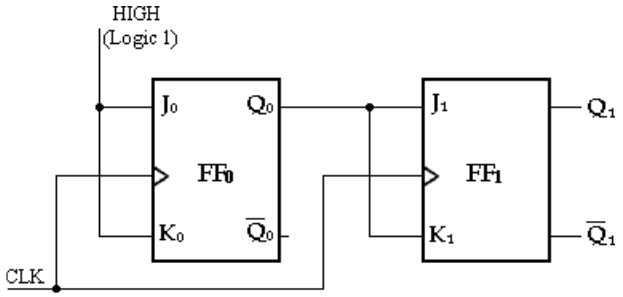
The term 'synchronous' refers to events that have a fixed time relationship with each other. In synchronous counter, the clock pulses are applied to all Flip- Flops simultaneously. Hence there is minimum propagation delay.

| S.No | Asynchronous (ripple) counter | Synchronous counter |
|---|---|---|
| 1 | All the Flip-Flops are not clocked simultaneously. | All the Flip-Flops are clocked simultaneously. |
| 2 | The delay times of all Flip-Flops are added. Therefore there is considerable propagation delay. | There is minimum propagation delay. |
| 3 | Speed of operation is low | Speed of operation is high. |
| 4 | Logic circuit is very simple even for more number of states. | Design involves complex logic circuit as number of state increases. |
| 5 | Minimum numbers of logic devices are needed. | The number of logic devices is more than ripple counters. |
| 6 | Cheaper than synchronous counters. | Costlier than ripple counters. |

**2-Bit Synchronous Binary Counter**

In this counter the clock signal is connected in parallel to clock inputs of both the Flip-Flops (FF0 and FF1). The output of FF0 is connected to J1 and K1 inputs of the second Flip-Flop (FF1).

APEC



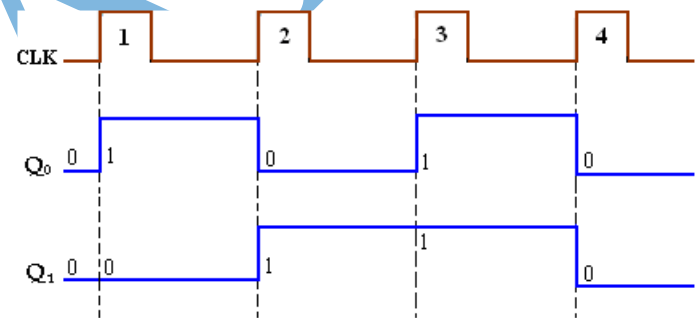

Assume that the counter is initially in the binary 0 state: i.e., both Flip-Flops are RESET. When the positive edge of the first clock pulse is applied, FF0 will toggle because J0= k0= 1, whereas FF1 output will remain 0 because J1= k1= 0. After the first clock pulse Q0=1 and Q1=0.

When the leading edge of CLK2 occurs, FF0 will toggle and Q0 will go LOW. Since FF1 has a HIGH (Q0 = 1) on its J1 and K1 inputs at the triggering edge of this clock pulse, the Flip-Flop toggles and Q1 goes HIGH. Thus, after CLK2,

Q0 = 0 and Q1 = 1.

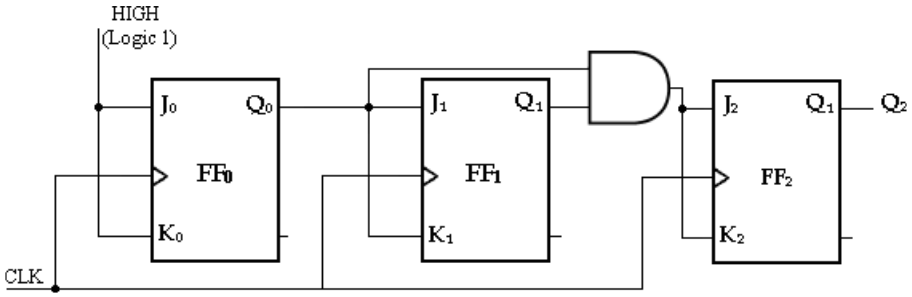
When the leading edge of CLK3 occurs, FF0 again toggles to the SET state (Q0 = 1), and FF1 remains SET (Q1 = 1) because its J1 and K1 inputs are both LOW (Q0 = 0). After this triggering edge, Q0 = 1 and Q1 = 1.

Finally, at the leading edge of CLK4, Q0 and Q1 go LOW because they both have a toggle condition on their J1 and K1 inputs. The counter has now recycled to its original state, Q0 = Q1 = 0.




**3-Bit Synchronous Binary Counter**

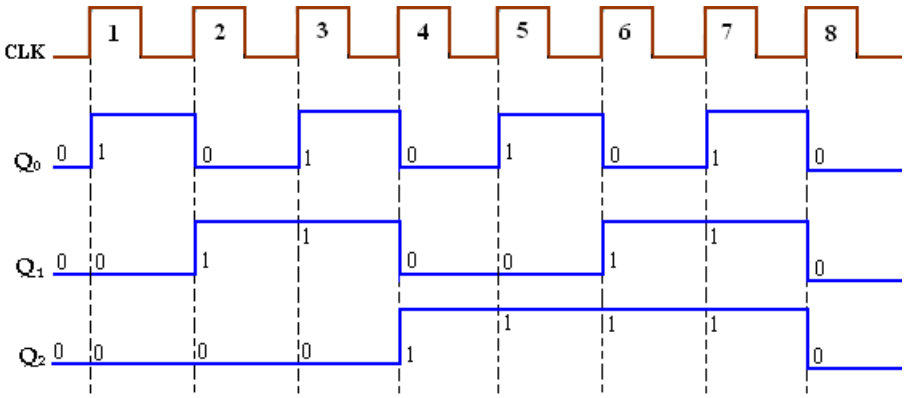
A 3 bit synchronous binary counter is constructed with three JK Flip-Flops and an AND gate. The output of FF0 (Q0) changes on each clock pulse as the counter progresses from its original state to its final state and then back to its original state. To produce this operation, FF0 must be held in the toggle mode by constant HIGH, on its J0 and K0 inputs.




The output of FF1 (Q1) goes to the opposite state following each time Q0= 1. This change occurs at CLK2, CLK4, CLK6, and CLK8. The CLK8 pulse causes the counter to recycle. To produce this operation, Q0 is connected to the J1 and K1 inputs of FF1. When Q0= 1 and a clock pulse occurs, FF1 is in the toggle mode and therefore changes state. When Q0= 0, FF1 is in the no-change mode and remains in its present state.

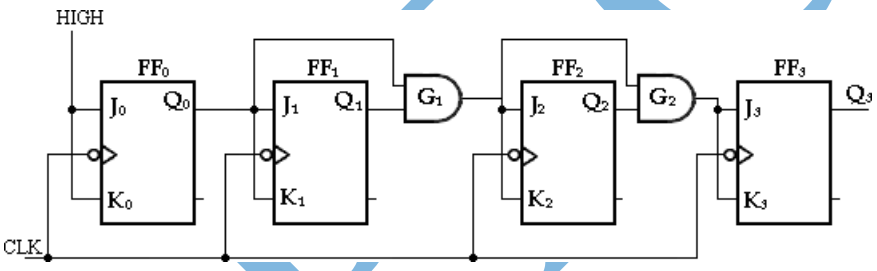
The output of FF2 (Q2) changes state both times; it is preceded by the unique condition in which both Q0 and Q1 are HIGH. This condition is detected by the AND gate and applied to the J2 and K2 inputs of FF2. Whenever both outputs Q0= Q1= 1, the output of the AND gate makes the J2= K2= 1 and FF2 toggles on the following clock pulse. Otherwise, the J2 and K2 inputs of FF2 are held LOW by the AND gate output, FF2 does not change state.

| CLOCK Pulse | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|
| Initially | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 |
| 6 | 1 | 1 | 0 |
| 7 | 1 | 1 | 1 |
| 8 (recycles) | 0 | 0 | 0 |




**4-Bit Synchronous Binary Counter**

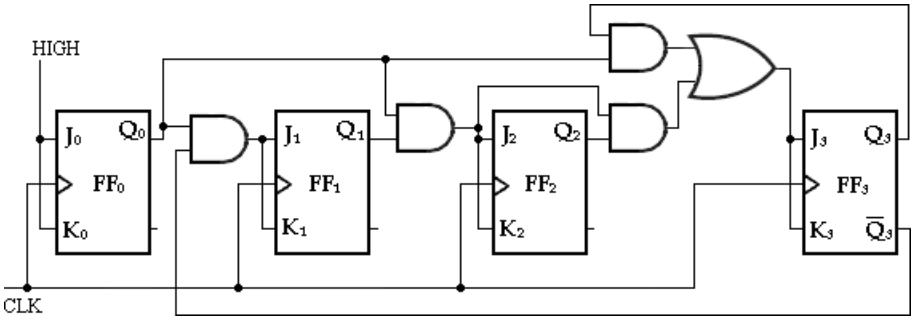
This particular counter is implemented with negative edge-triggered Flip- Flops. The reasoning behind the J and K input control for the first three Flip- Flops is the same as previously discussed for the 3-bit counter. For the fourth stage, the Flip- Flop has to change the state when Q0= Q1= Q2= 1. This condition is decoded by AND gate G2.




Therefore, when Q0= Q1= Q2= 1, Flip-Flop FF3 toggles and for all other times it is in a no-change condition. Points where the AND gate outputs are HIGH are indicated by the shaded areas.

**4-Bit Synchronous Decade Counter: (BCD Counter):**

BCD decade counter has a sequence from 0000 to 1001 (9). After 1001 state it must recycle back to 0000 state. This counter requires four Flip-Flops and AND/OR logic as shown below.



First, notice that FF0 (Q0) toggles on each clock pulse, so the logic equation for its J0 and K0 inputs is

**J0= K0= 1**

This equation is implemented by connecting J0 and K0 to a constant HIGH level.

 Next, notice from table, that FF1 (Q1) changes on the next clock pulse each time Q0 = 1 and Q3 = 0, so the logic equation for the J1 and K1 inputs is

**J1= K1= Q0Q3'**

 Flip-Flop 2 (Q2) changes on the next clock pulse each time both Q0 = Q1 = 1. This requires an input logic equation as follows:

**J2= K2= Q0Q1**

This equation is implemented by ANDing Q0 and Q1 and connecting the gate output to the J2 and K2 inputs of FF2.

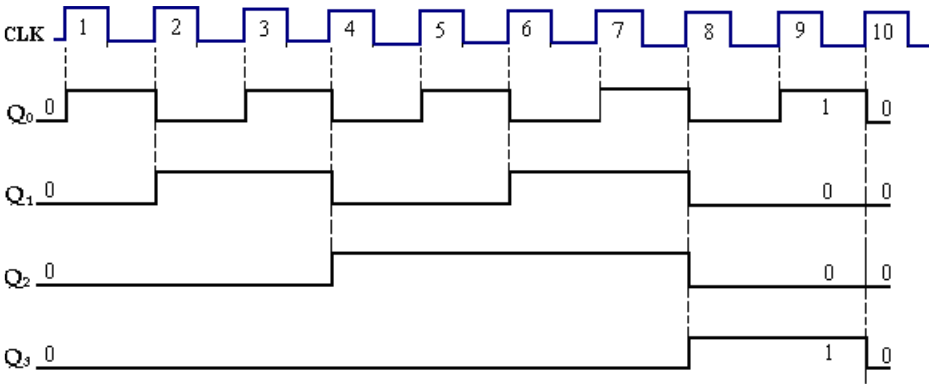
 Finally, FF3 (Q3) changes to

the opposite state on the next clock pulse each time Q0 = 1, Q1 = 1, and Q2 = 1 (state 7), or when Q0 = 1 and Q1 = 1 (state 9). The equation for this is as follows:

**J3= K3= Q0Q1Q2+ Q0Q3**

This function is implemented with the AND/OR logic connected to the J3 and K3 inputs of FF3.

| CLOCK Pulse | $Q_3$ | $Q_2$ | $Q_1$ | $Q_0$ |
|---|---|---|---|---|
| Initially | 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(recycles) | 0 | 0 | 0 | 0 |




## SYNCHRONOUS UP/DOWN COUNTER

An up/down counter is a bidirectional counter, capable of progressing in either direction through a certain sequence. A 3-bit binary counter that advances upward through its sequence (0, 1, 2, 3, 4, 5, 6, 7) and then can be reversed so that it goes through the sequence in the opposite direction (7, 6, 5, 4, 3, 2, 1,0) is an illustration of up/down sequential operation.

The complete up/down sequence for a 3-bit binary counter is shown in table below. The arrows indicate the state-to-state movement of the counter for both its UP and its DOWN modes of operation. An examination of Q0 for both the up and down sequences shows that FF0 toggles on each clock pulse. Thus, the J0 and K0 inputs of FF0 are,

J0= K0= 1

| CLOCK PULSE | UP | $Q_2$ | $Q_1$ | $Q_0$ | DOWN |
|---|---|---|---|---|---|
| 0 | | 0 | 0 | 0 | |
| 1 | | 0 | 0 | 1 | |
| 2 | | 0 | 1 | 0 | |
| 3 | | 0 | 1 | 1 | |
| 4 | | 1 | 0 | 0 | |
| 5 | | 1 | 0 | 1 | |
| 6 | | 1 | 1 | 0 | |
| 7 | | 1 | 1 | 1 | |

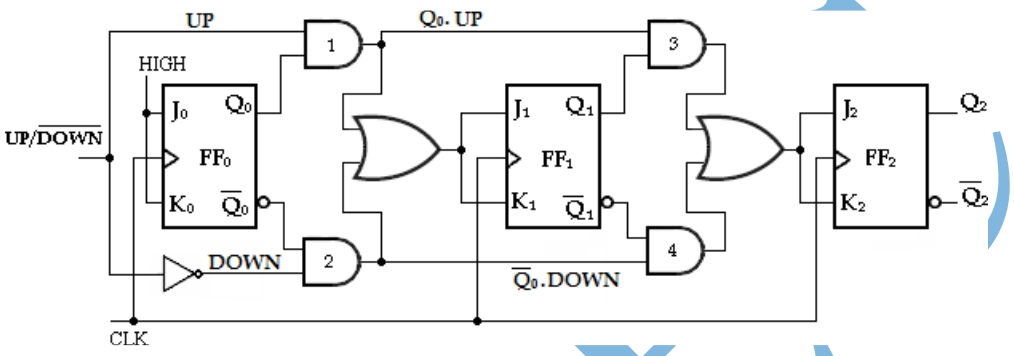
To form a synchronous UP/DOWN counter, the control input (UP/DOWN) is used to allow either the normal output or the inverted output of one Flip-Flop to

the J and K inputs of the next Flip-Flop. When UP/DOWN= 1, the MOD 8 counter will count from 000 to 111 and UP/DOWN= 0, it will count from 111 to 000.

When UP/DOWN= 1, it will enable AND gates 1 and 3 and disable AND gates 2 and 4. This allows the Q0 and Q1 outputs through the AND gates to the J and K inputs of the following Flip-Flops, so the counter counts up as pulses are applied. When UP/DOWN= 0, the reverse action takes place.

**J1= K1= (Q0.UP)+ (Q0'.DOWN)**

**J2= K2= (Q0. Q1.UP)+ (Q0'.Q1'.DOWN)**




## MODULUS-N-COUNTERS

The counter with 'n' Flip-Flops has maximum MOD number 2n. Find the number of Flip-Flops (n) required for the desired MOD number (N) using the equation,

**2n ≥ N**

(i) For example, a 3 bit binary counter is a **MOD 8 counter**. The basic counter can be modified to produce MOD numbers less than 2n by allowing the counter to skin those are normally part of counting sequence.

n= 3

N= 8

2n = 23= 8= N

**(ii) MOD 5 Counter:**

2n= N

2n= 5

22= 4 less than N.

23= 8 > N(5)

Therefore, 3 Flip-Flops are required.

**(iii) MOD 10 Counter:**

2n= N= 10

23= 8 less than N.

24= 16 > N(10).

To construct any MOD-N counter, the following methods can be used.

1. Find the number of Flip-Flops (n) required for the desired MOD number (N) using the equation,

**2n ≥ N.**

2. Connect all the Flip-Flops as a required counter.

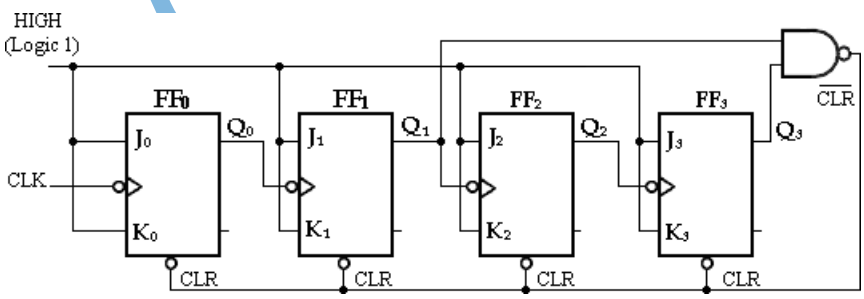
3. Find the binary number for N.

4. Connect all Flip-Flop outputs for which Q= 1 when the count is N, as inputs to NAND gate.

5. Connect the NAND gate output to the CLR input of each Flip-Flop. When the counter reaches Nth state, the output of the NAND gate goes LOW, resetting all Flip-Flops to 0. Therefore the counter counts from 0 through N-1.

For example, MOD-10 counter reaches state 10 (1010). i.e., Q3Q2Q1Q0= 1 0 1 0. The outputs Q3 and Q1 are connected to the NAND gate and the output of the NAND gate goes LOW and resetting all Flip-Flops to zero. Therefore MOD-10 counter counts from 0000 to 1001. And then recycles to the zero value.

The MOD-10 counter circuit is shown below.



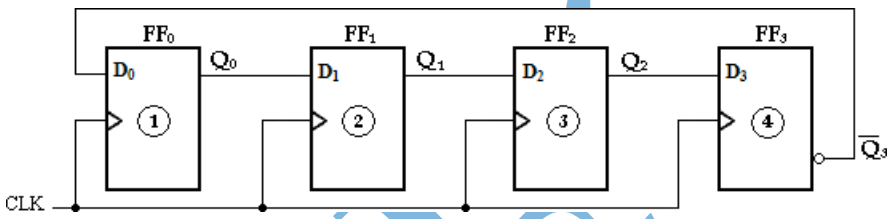

## SHIFT REGISTER COUNTERS

A shift register counter is basically a shift register with the serial output connected back to the serial input to produce special sequences. Two of the most common types of shift register counters are:

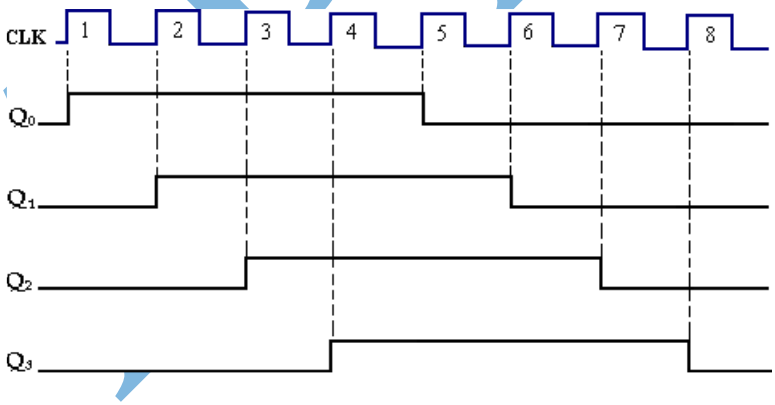
Johnson counter (Shift Counter), Ring counter,

**Johnson counter (Shift Counter):**

In a Johnson counter the complement of the output of the last Flip-Flop is connected back to the D input of the first Flip-Flop. This feedback arrangement produces a characteristic sequence of states as shown in table below. The 4-bit sequence has a total of eight states, and that the 5-bit sequence has a total of ten states. In general, a Johnson counter will produce a modulus of **2n**, where 'n' is the number of stages in the counter.

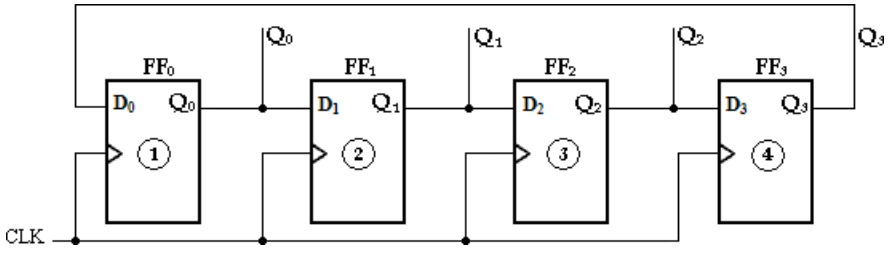



The Q output of each stage is connected to the D input of the next stage (assuming that D Flip-Flops are used). The complement output of the last stage is connected back to the D input of the first stage.

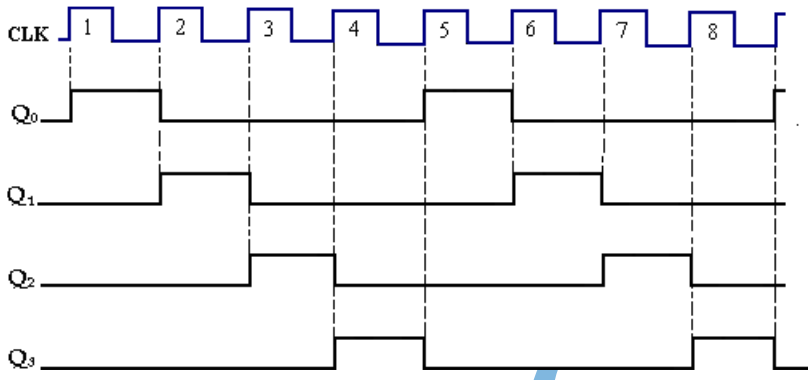



**Ring Counters:**

The ring counter utilizes one Flip-Flop for each state in its sequence. It has the advantage that decoding gates are not required. In the case of a l0-bit ring counter, there is a unique output for each decimal digit.



The output Q0 sets D1 input, Q1 sets D2, Q2 sets D3 and Q3 is fed back to D0. Because of these conditions, bits are shifted left one position per positive clock edge and fed back to the input. All the Flip-Flops are clocked together. When CLR goes low then back to high, the output is 0000.




The first positive clock edge shifts MSB to LSB position and other bits to one position left so that the output becomes Q= 0010. This process continues on second and third clock edge so that successive outputs are 0100 and 1000. The fourth positive clock edge starts the cycle all over again and the output is 0001. Thus the stored 1 bit follows a circular path (i.e., the stored 1 bits move left through all Flip- Flops and the final Flip-Flop sends it back to the first Flip-Flop). This action has given the name of ring counter.

## DESIGN OF COUNTERS

The procedure for design of counters as follows:

1. Specify the counter sequence and draw a state diagram.

2. Derive a next-state table from the state diagram.

3. Make the state assignment and develop a transition table showing the flip- flop inputs required.

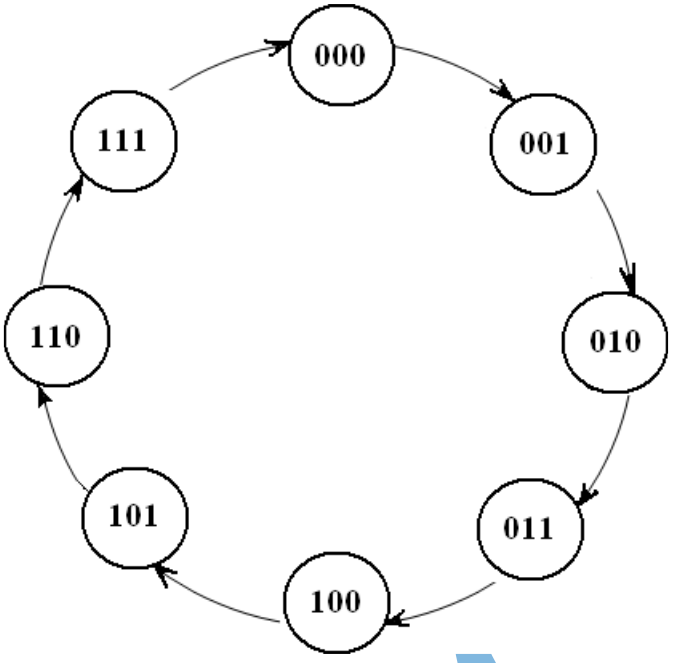
4. Draw the K-maps for each input of each Flip-Flop.

5. Derive the logic expression for each Flip-Flop input from the K-maps.

6. Implement the expressions with combinational logic and combine with the Flip-Flops to form the counter.

**Examples:**

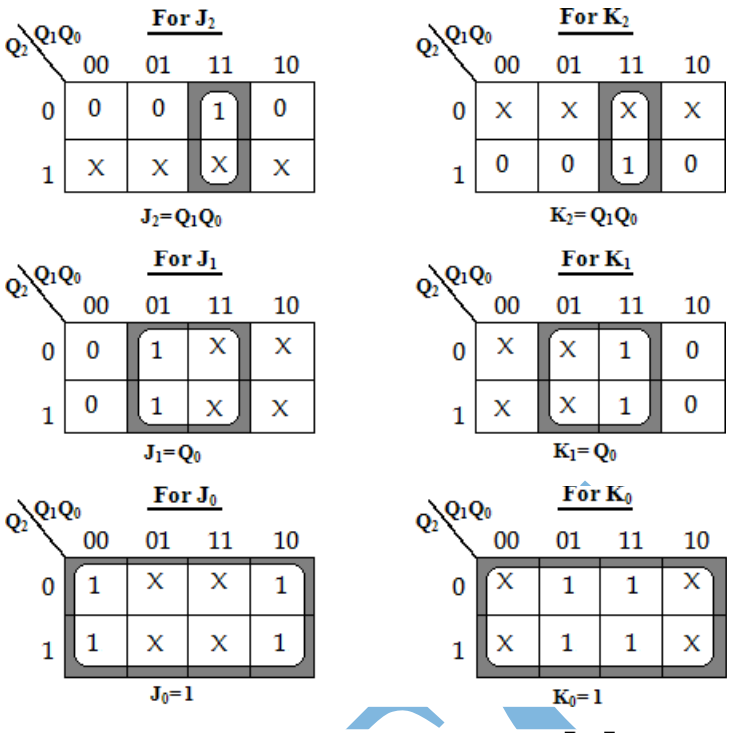
**1.** Using JK Flip-Flops, design a synchronous counter which counts in the sequence, **000, 001, 010, 011, 100, 101, 110, 111, 000.**

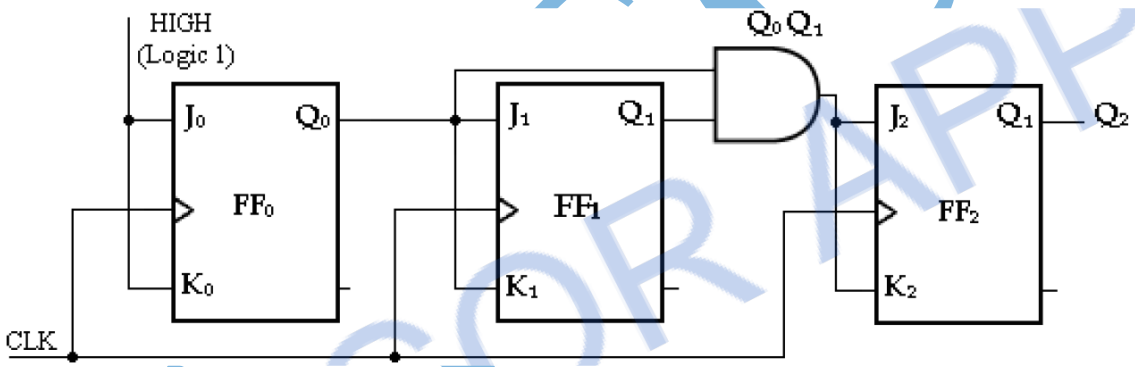



| Present State | Next State | Inputs | |
|---|---|---|---|
| **Qn** | **Qn+1** | **J** | **K** |
| 0 | 0 | 0 | x |
| 0 | 1 | 1 | x |
| 1 | 0 | x | 1 |
| 1 | 1 | x | 0 |

| Present State | | | Next State | | | Flip-Flop Inputs | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Q2** | **Q1** | **Q0** | **q2** | **q1** | **q0** | **J2** | **K2** | **J1** | **K1** | **J0** | **K0** |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | x | 0 | x | 1 | x |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | x | 1 | x | x | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | x | x | 0 | 1 | x |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | x | x | 1 | x | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | x | 0 | 0 | x | 1 | x |
| 1 | 0 | 1 | 1 | 1 | 0 | x | 0 | 1 | x | x | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | x | 0 | x | 0 | 1 | x |
| 1 | 1 | 1 | 0 | 0 | 0 | x | 1 | x | 1 | x | 1 |

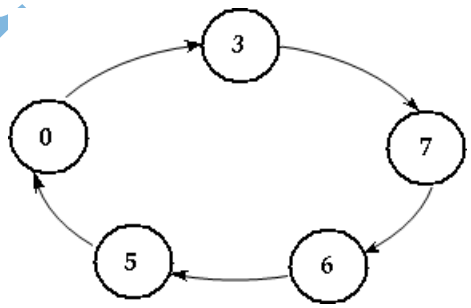
APEC




## LOCKOUT CONDITION

In a counter if the next state of some unused state is again a used state and if by chance the counter happens to find itself in the unused states and never arrived at a used state then the counter is said to be in the lockout condition




The circuit that goes in lockout condition is called ***bushless*** circuit. To make sure that the counter will come to the initial state from any unused state, the additional logic circuit is necessary. To ensure that the lockout does not occur, the counter

should be designed by forcing the next state to be the initial state from the unused states.

APEC