

## SEQUENTIAL CIRCUITS

Digital electronics is classified into **combinational logic** and **sequential logic**.

Combinational logic output depends on the present inputs levels, whereas sequential logic output not only depends on the input levels, but also stored levels (previous output history).

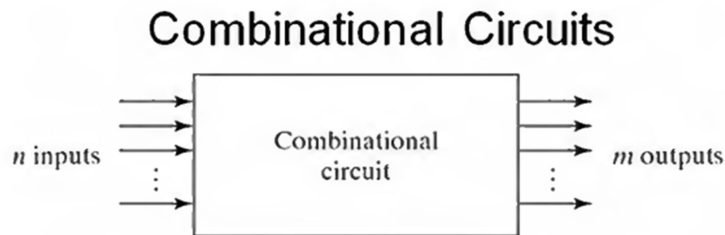
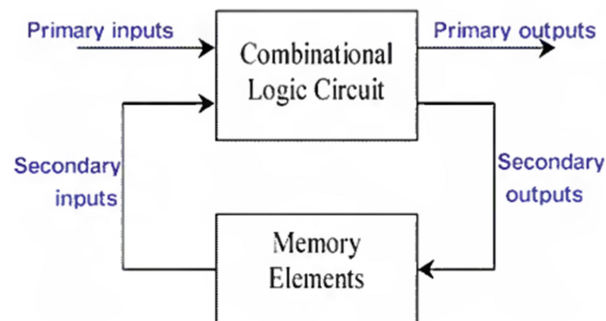


Fig. Block Diagram of Combinational Circuit

### Sequential Circuits



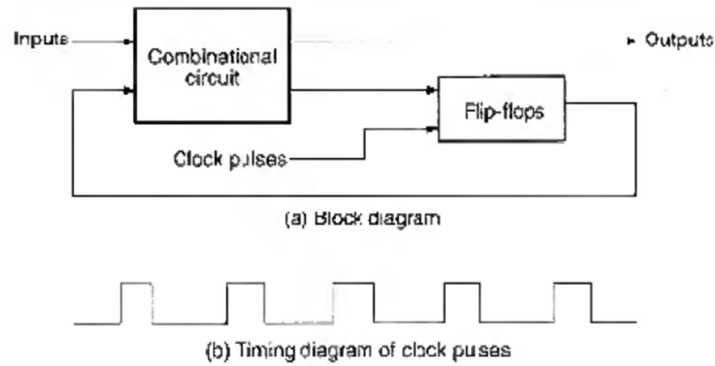
The memory elements are devices capable of storing binary info. The binary info stored in the memory elements at any given time defines the state of the sequential circuit. The input and the present state of the memory element determine the output. Memory elements next state is also a function of external inputs and present state. A sequential circuit is specified by a time sequence of inputs, outputs, and internal states.

There are two types of sequential circuits. Their classification depends on the timing of their signals:

- ❖ Synchronous sequential circuits
- ❖ Asynchronous sequential circuits

### Asynchronous sequential circuit

This is a system whose outputs depend upon the order in which its input variables change and can be affected at **any instant of time**.



## Synchronous sequential circuits

This type of system uses storage elements called flip-flops that are employed to change their binary value only **at discrete instants of time**.

Synchronous sequential circuits use logic gates and flip-flop storage devices. Sequential circuits have a clock signal as one of their inputs. All state transitions in such circuits occur only when the clock value is either 0 or 1 or happen at the rising or falling edges of the clock depending on the type of memory elements used in the circuit. Synchronization is achieved by a timing device called a clock pulse generator. Clock pulses are distributed throughout the system in such a way that the flip-flops are affected only with the arrival of the synchronization pulse. Synchronous sequential circuits that use clock pulses in the inputs are called clocked-sequential circuits.

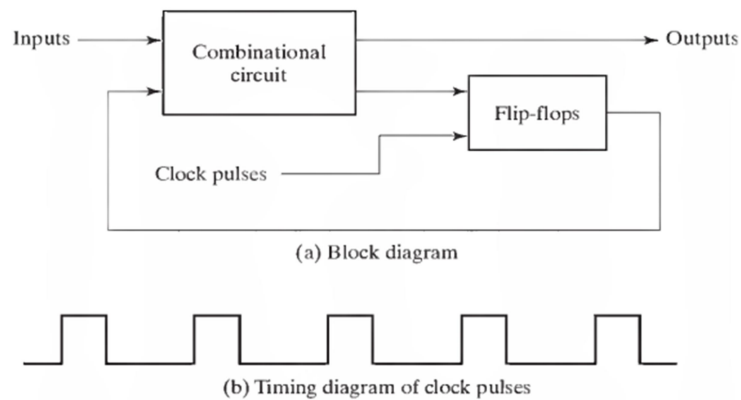
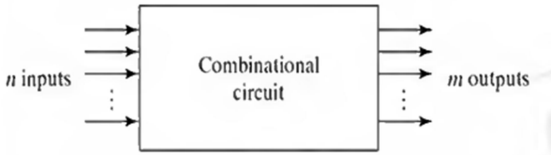
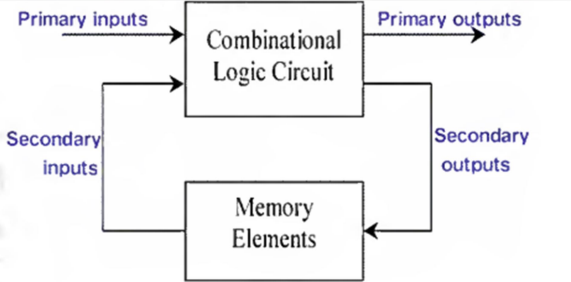


Fig. 5-2 Synchronous Clocked Sequential Circuit

| <b>Combinational Circuits</b>   | <b>Sequential Circuits</b>  |
|---|---|
| 1. The circuit whose output at any instant depends only on the input present at that instant only is known as combinational circuit.  | 1. The circuit whose output at any instant depends not only on the input present but also on the past output a is known as sequential circuit   |
| 2. This type of circuit has no memory unit.   | 2. This type of circuit has memory unit for store past output.  |
| 3. Examples of combinational circuits are half adder, full adder, magnitude comparator, multiplexer, demultiplexer e.t.c.   | 3. Examples of sequential circuits are Flip flop, register, counter e.t.c.  |
| 4. Faster in Speed  | Slower compared to Combinational Circuit  |
| <p style="text-align: center;"><b>Combinational Circuits</b></p>  <p>The diagram shows a rectangular box labeled 'Combinational circuit'. On the left side, there are three horizontal arrows pointing into the box, with the text 'n inputs' and a vertical ellipsis below them. On the right side, there are three horizontal arrows pointing out of the box, with the text 'm outputs' and a vertical ellipsis below them.</p> <p>Fig. Block Diagram of Combinational Circuit</p> |  <p>The diagram shows two rectangular boxes. The top box is labeled 'Combinational Logic Circuit'. The bottom box is labeled 'Memory Elements'. On the left side of the 'Combinational Logic Circuit' box, there are three horizontal arrows pointing into it, labeled 'Primary inputs'. On the right side of the 'Combinational Logic Circuit' box, there are three horizontal arrows pointing out of it, labeled 'Primary outputs'. On the left side of the 'Memory Elements' box, there are three horizontal arrows pointing into it, labeled 'Secondary inputs'. On the right side of the 'Memory Elements' box, there are three horizontal arrows pointing out of it, labeled 'Secondary outputs'. A feedback loop is shown where the 'Secondary outputs' of the 'Memory Elements' box point back into its 'Secondary inputs', and the 'Secondary outputs' of the 'Combinational Logic Circuit' box point into the 'Secondary inputs' of the 'Memory Elements' box.</p> |

A sequential circuit can further be categorized into **Synchronous** and **Asynchronous**.

Here is the difference between synchronous and asynchronous sequential circuits:

**Synchronous Sequential Circuit:** Output changes at discrete interval of time. It is a circuit based on an equal state time or a state time defined by external means such as clock. Examples of synchronous sequential circuit are Flip Flops, Synchronous Counter.

**Asynchronous Sequential Circuit:** Output can be changed at any instant of time by changing the input. It is a circuit whose state time depends solely upon the internal logic circuit delays. Example of asynchronous sequential circuit is Asynchronous Counter.

## Basic Flip Flops:

A circuit that changes from 1 to 0 or from 0 to 1 when current is applied. It is one bit storage location.

Flip flops are actually an application of logic gates. When a certain input value is given to them, they will be remembered and executed, if the logic gates are designed correctly. A higher application of flip flops is helpful in designing better electronic circuits.

The most commonly used application of flip flops is in the implementation of a feedback circuit. As a memory relies on the feedback concept, flip flops can be used to design it.

There are basically four main types of latches and flip-flops: SR, D, JK, and T. The major differences in these flip-flop types are the number of inputs they have and how they change state. For each type, there are also different variations that enhance their operations. In

### 1. RS Latch

- ❖ RS latch have two inputs, S and R. S is called set and R is called reset.
- ❖ The S input is used to produce HIGH on Q ( i.e. store binary 1 in flip-flop).
- ❖ The R input is used to produce LOW on Q (i.e. store binary 0 in flip-flop).
- ❖ Q' is Q complementary output, so it always holds the opposite value of Q.
- ❖ The output of the S-R latch depends on current as well as previous inputs or state, and its state (value stored) can change as soon as its inputs change.

There are mainly four types of flip flops that are used in electronic circuits.

1. **The basic Flip Flop or S-R Flip Flop**
2. **Delay Flip Flop [D Flip Flop]**
3. **J-K Flip Flop**
4. **T Flip Flop**

### 2. S-R Flip Flop:

The SET-RESET flip flop is not designed with the help of two NOR gates and also two NAND gates. These flip flops are also called S-R Latch.

#### S-R Flip Flop using NOR Gate

The design of such a flip flop includes two inputs, called the SET [S] and RESET [R]. There are also two outputs, Q and Q'. The diagram and truth table is shown below.



**2. When  $S=0, R=1$ , the output becomes  $Q=0, Q'=1$**

In this state When  $R=1$  it resets the output. So this state is known as the RESET state. In both the states you can see that the outputs are just compliments of each other and that the value of  $Q$  follows the value of  $S$ .

**3. When  $S=0, R=0$  the output is  $Q \& Q' = \text{Remember (memory)}$**

If both the values of  $S$  and  $R$  are switched to 0, then the circuit remembers the value of  $S$  and  $R$  in their previous state.

**4. When  $S=1, R=1$  the output  $Q=0, Q'=0$  [Invalid]**

This is an invalid state because the values of both  $Q$  and  $Q'$  are 0. They are supposed to be compliments of each other. Normally, this state must be avoided.

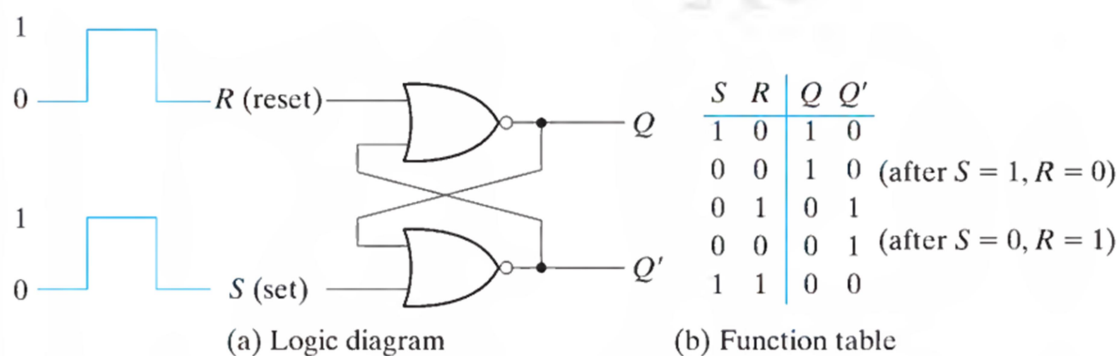


Fig. 5-3 SR Latch with NOR Gates

The operation has to be analyzed with the 4 inputs combinations together with the 2 possible previous states.

From the diagram it is evident that the flip flop has mainly four states. They are

**1. When  $S=1, R=0$  the output becomes  $Q=1, Q'=0$**

This SR flip flop function table is constructed based on the XOR gate. In XOR gate if any of the input is 1 the output becomes 1.

In this state when  $S=1$  and  $R=0$  the output  $Q$  becomes set (1). So this state is also called the SET state.

## S-R Flip Flop using NAND Gate

The above SR flip flop can be constructed using NAND gate.

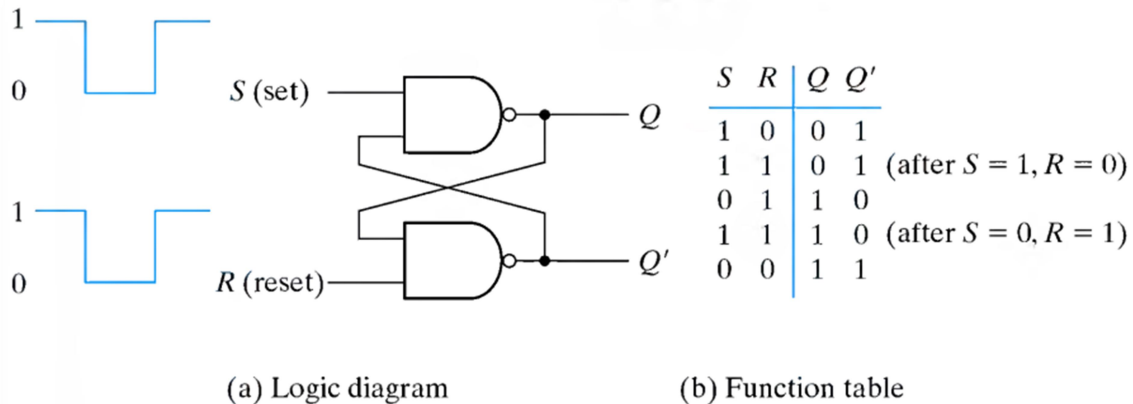


Fig. 5-4 SR Latch with NAND Gates

Like the NOR Gate S-R flip flop, this one also has four states. They are

**1.  $S=1, R=0, Q=0, Q'=1$**

This state is also called the SET state.

**2.  $S=0, R=1, Q=1, Q'=0$**

This state is known as the RESET state.

In both the states you can see that the outputs are just compliments of each other and that the value of Q follows the compliment value of S.

**3.  $S=0, R=0, Q=1, \& Q'=1$  [Invalid]**

If both the values of S and R are switched to 0 it is an invalid state because the values of both Q and Q' are 1. They are supposed to be compliments of each other. Normally, this state must be avoided.

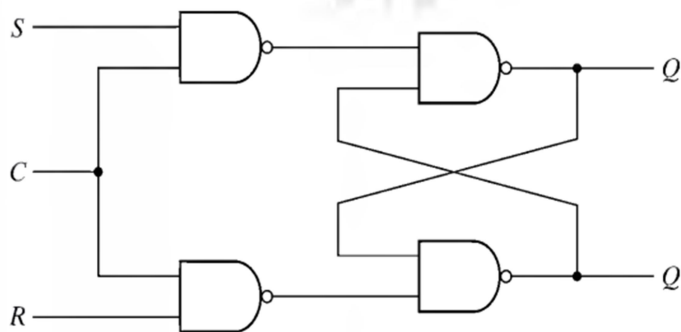
**4.  $S=1, R=1, Q \& Q'=$  Remember**

If both the values of S and R are switched to 1, then the circuit remembers the value of S and R in their previous state.

## Clocked S-R Flip Flop

- ❖ It is also called a Gated S-R flip flop.
- ❖ The problems with S-R flip flops using NOR and NAND gate is the invalid state.
- ❖ This problem can be overcome by using a bistable SR flip-flop that can change outputs when certain invalid states are met, regardless of the condition of either the Set or the Reset inputs.
- ❖ For this, a clocked S-R flip flop is designed by adding two AND neither gates to a basic NOR Gate flip flop.
- ❖ The circuit diagram and truth table is shown below.

The circuit of the S-R flip flop using NAND Gate and its truth table is shown below.



(a) Logic diagram

| C | S | R | Next state of Q    |
|---|---|---|--------------------|
| 0 | X | X | No change          |
| 1 | 0 | 0 | No change          |
| 1 | 0 | 1 | Q = 0; Reset state |
| 1 | 1 | 0 | Q = 1; set state   |
| 1 | 1 | 1 | Indeterminate      |

(b) Function table

Fig. 5-5 SR Latch with Control Input

- A clock pulse [CP] is given to the inputs of the AND Gate.
- When the value of the clock pulse is '0', the outputs of both the AND Gates remain '0'.
- As soon as a pulse is given the value of CP turns '1'.
- This makes the values at S and R to pass through the NOR Gate flip flop. But when the values of both S and R values turn '1', the HIGH value of CP causes both of them to turn to '0' for a short moment.
- As soon as the pulse is removed, the flip flop state becomes intermediate.
- Thus either of the two states may be caused, and it depends on whether the set or reset input of the flip-flop remains a '1' longer than the transition to '0' at the end of the pulse. Thus the invalid states can be eliminated.

- The excitation of the SR latch is as follows:

Excitation Table:

| Q <sub>n</sub> | S | R | Q <sub>n+1</sub> |
|----------------|---|---|------------------|
| 0              | 0 | 0 | 0                |
| 0              | 0 | 1 | 0                |
| 0              | 1 | 0 | 1                |
| 0              | 1 | 1 | Indeter          |
| 1              | 0 | 0 | 1                |
| 1              | 0 | 1 | 0                |
| 1              | 1 | 0 | 1                |
| 1              | 1 | 1 | Indeter          |

Note: Indeter = not used

K Map for Q<sub>n+1</sub>:

| SR             |   | Q <sub>n</sub> |    |    |    |
|----------------|---|----------------|----|----|----|
|                |   | 00             | 01 | 11 | 10 |
| Q <sub>n</sub> | 0 |                |    | X  | 1  |
|                | 1 | 1              |    | X  | 1  |

$$Q_{n+1} = S + \bar{R} \cdot Q_n$$

### 3. D Flip Flop

- D flip flop is actually a slight modification of the above explained clocked SR flip-flop. From the figure you can see that the D input is connected to the S input and the complement of the D input is connected to the R input.
- The D input is passed on to the flip flop when the value of CP is '1'.
- When CP is HIGH, the flip flop moves to the SET state. If it is '0', the flip flop switches to the CLEAR state.
- As long as the clock input  $C = 0$ , the SR latch has both inputs equal to 0 and it can't change its state regardless of the value of D
- When C is 1, the latch is placed in the set or reset state based on the value of D.
  - If  $D = 1$ , the Q output goes to 1.
  - If  $D = 0$ , the Q output goes to 0.

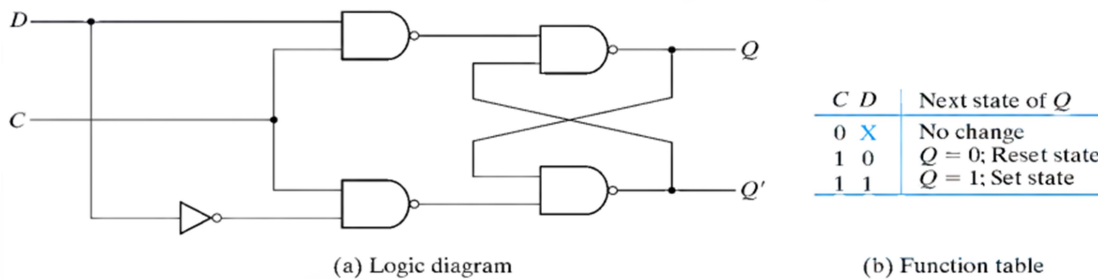
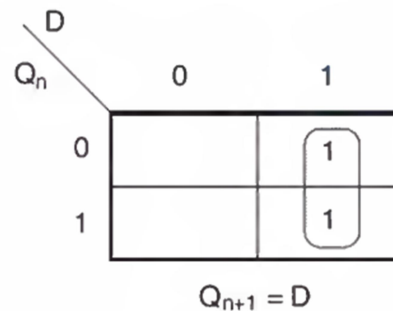


Fig. 5-6 D Latch

Excitation Table:

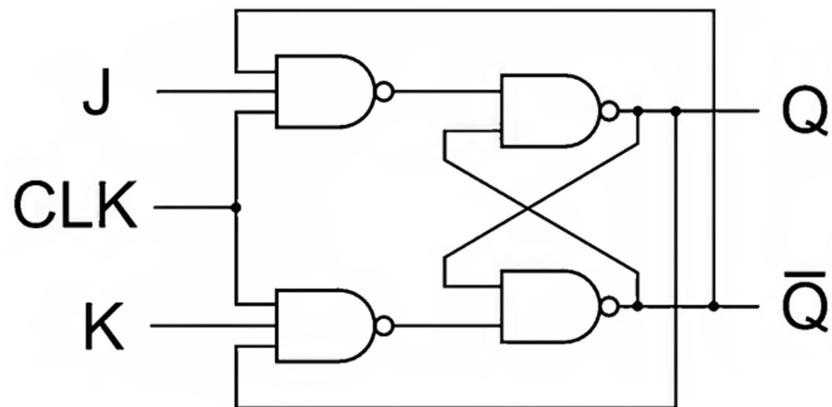
| $Q_n$ | D | $Q_{n+1}$ |
|-------|---|-----------|
| 0     | 0 | 0         |
| 0     | 1 | 1         |
| 1     | 0 | 0         |
| 1     | 1 | 1         |

K- Map for  $Q_{n+1}$ :



### 3. J-K Flip Flop

- ❖ A J-K flip flop can also be defined as a modification of the S-R flip flop. The only difference is that the intermediate state is more refined and precise than that of a S-R flip flop.



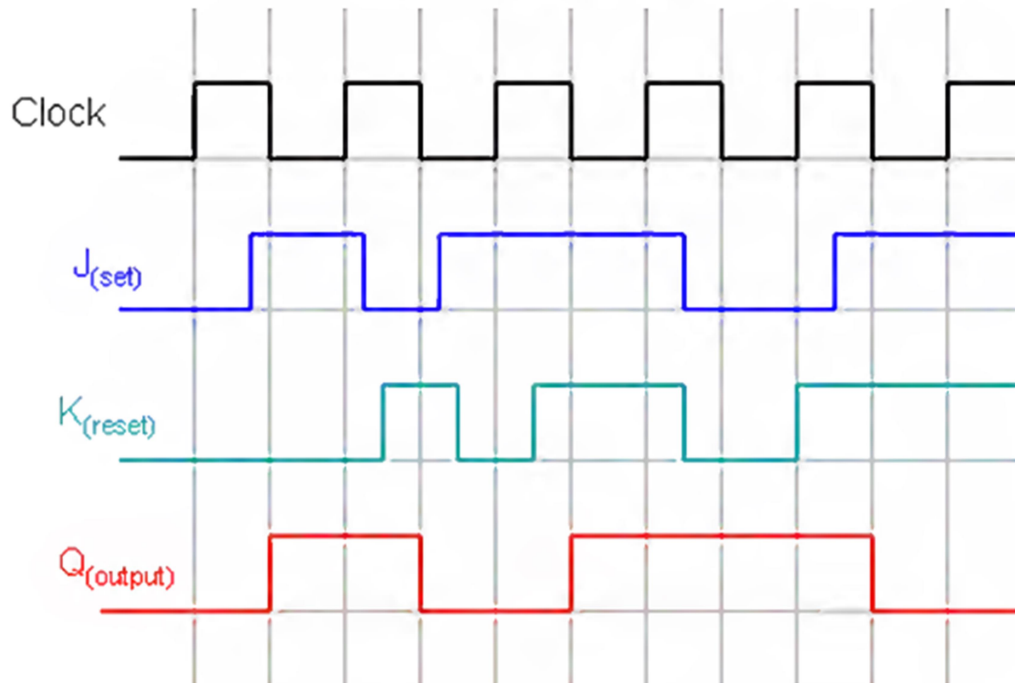
- ❖ The behavior of inputs J and K is same as the S and R inputs of the S-R flip flop. The letter J stands for SET and the letter K stands for CLEAR.
- ❖ When both the inputs J and K have a HIGH state, the flip-flop switches to the complement state. So, for a value of  $Q = 1$ , it switches to  $Q=0$  and for a value of  $Q = 0$ , it switches to  $Q=1$ .
- ❖ The circuit includes two 3-input AND gates. The output Q of the flip flop is returned back as a feedback to the input of the AND along with other inputs like K and clock pulse [CP].
- ❖ So, if the value of CP is '1', the flip flop gets a CLEAR signal and with the condition that the value of Q was earlier 1.
- ❖ Similarly output Q' of the flip flop is given as a feedback to the input of the AND along with other inputs like J and clock pulse [CP].
- ❖ So the output becomes SET when the value of CP is 1 only if the value of Q' was earlier 1.
- ❖ The output may be repeated in transitions once they have been complimented for  $J=K=1$  because of the feedback connection in the JK flip-flop.
- ❖ This can be avoided by setting a time duration lesser than the propagation delay through the flip-flop.
- ❖ The restriction on the pulse width can be eliminated with a master-slave or edge-triggered construction.



**Characteristic table:**

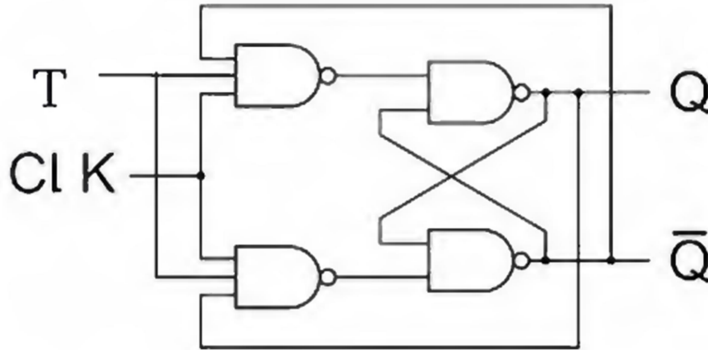
| Clk | J | K | Q <sub>n+1</sub> |
|-----|---|---|------------------|
| 0   | X | X | Memory           |
| 1   | 0 | 0 | Memory           |
| 1   | 0 | 1 | 0                |
| 1   | 1 | 0 | 1 (Set)          |
| 1   | 1 | 1 | Toggle           |

**Timing Diagram:**



#### 4. T Flip Flop

- ❖ This is a much simpler version of the J-K flip flop.
- ❖ Both the J and K inputs are connected together and thus are also called a single input J-K flip flop.
- ❖ When clock pulse is given to the flip flop, the output begins to toggle.
- ❖ Here also the restriction on the pulse width can be eliminated with a master-slave or edge-triggered construction. Take a look at the circuit and truth table below.



Excitation Table for T Flip Flop:

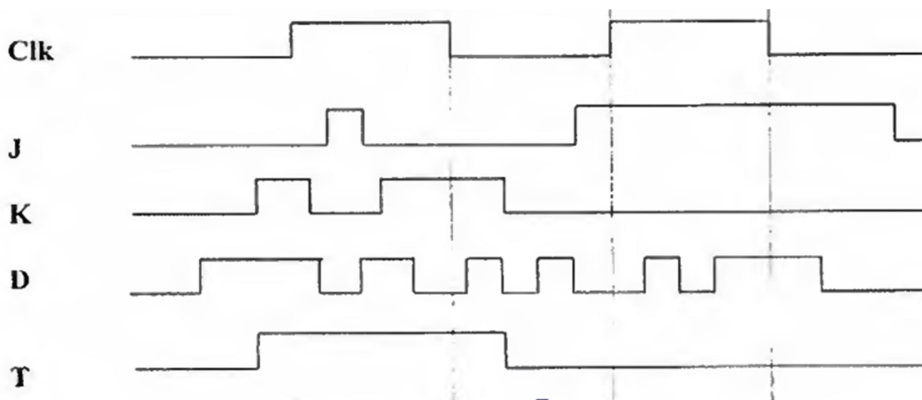
| $Q_n$ | T | $Q_{n+1}$ |
|-------|---|-----------|
| 0     | 0 | 0         |
| 0     | 1 | 1         |
| 1     | 0 | 1         |
| 1     | 1 | 0         |

K map for T Flip Flop:

|       |   | T |   |
|-------|---|---|---|
|       |   | 0 | 1 |
| $Q_n$ | 0 |   | 1 |
|       | 1 | 1 |   |

Characteristic Equation:

$$Q_{n+1} = T \cdot \overline{Q_n} + \overline{T} \cdot Q_n$$

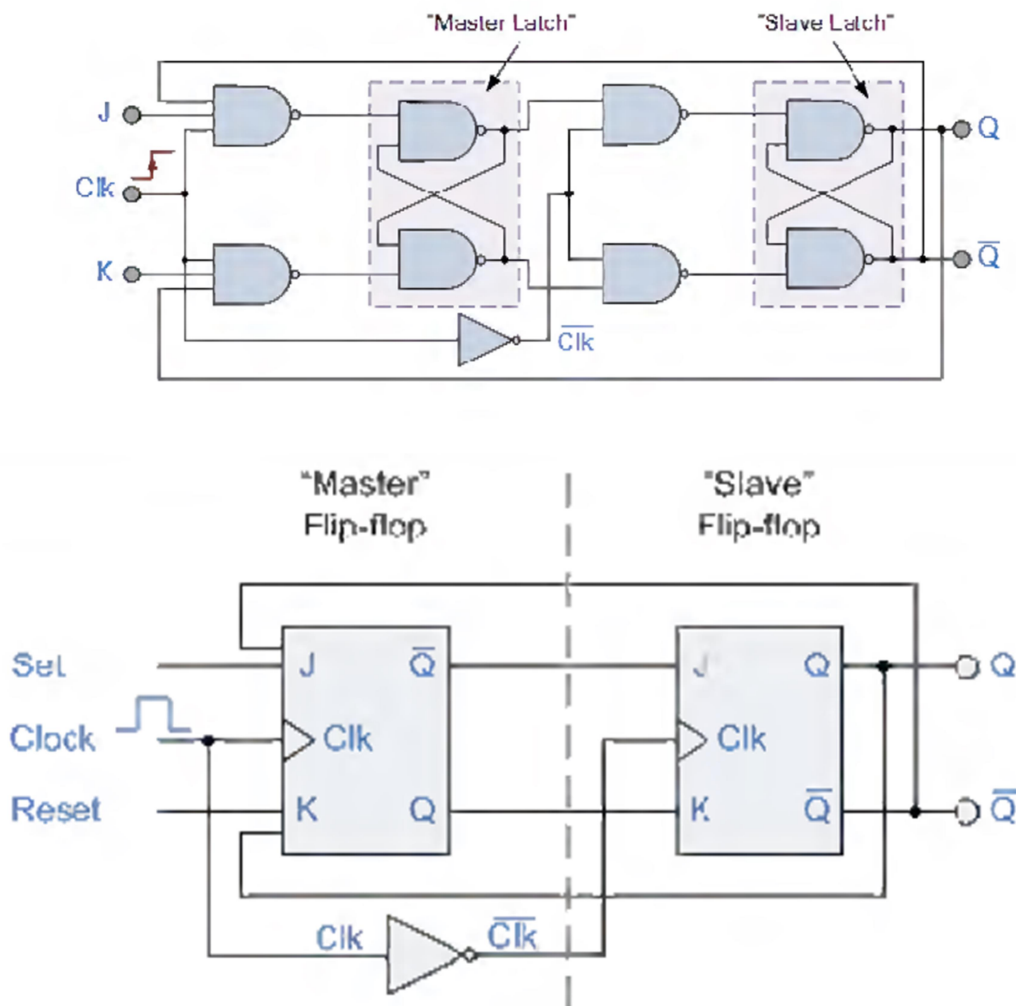


## Master-Slave Flip Flop Circuit

Before knowing more about the master-slave flip flop you have to know more on the basics of a J-K flip flop and S-R flip flop. To know more about the flip flops, click on the link below.

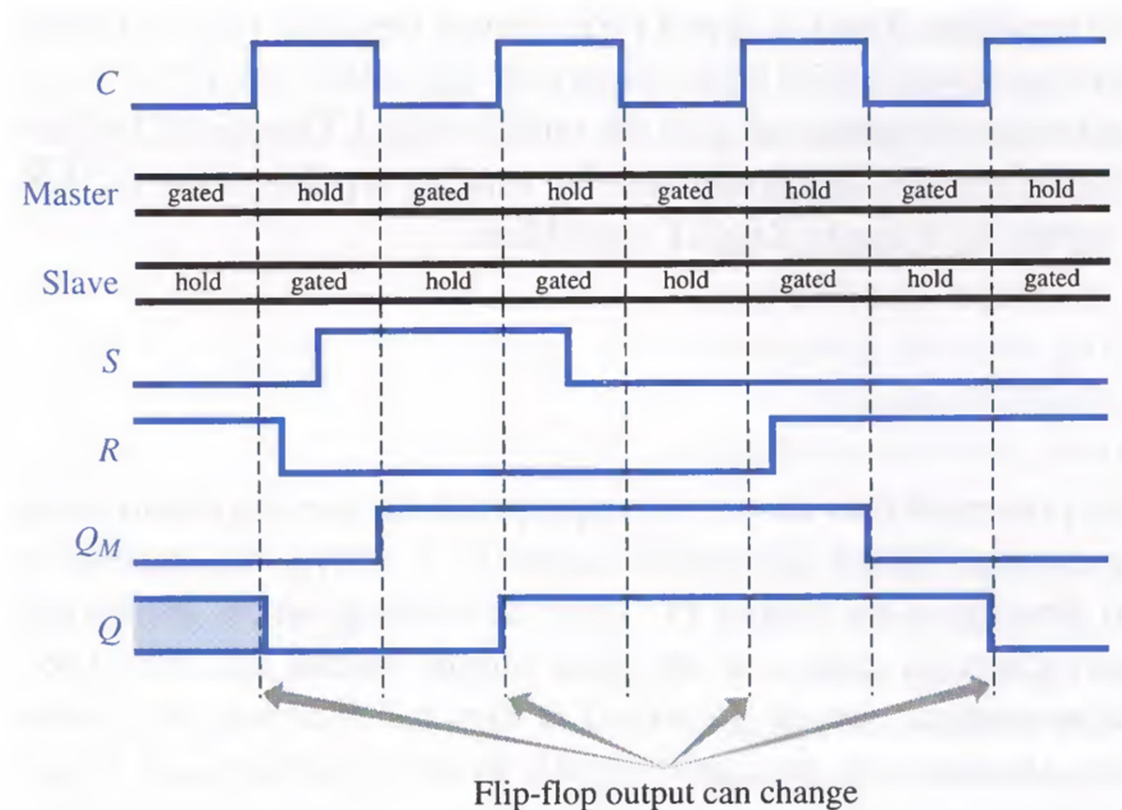
Master-slave flip flop is designed using two separate flip flops. Out of these, one acts as the master and the other as a slave. The figure of a master-slave J-K flip flop is shown below.

From the below figure you can see that both the J-K flip flops are presented in a series connection. The output of the master J-K flip flop is fed to the input of the slave J-K flip flop. The output of the slave J-K flip flop is given as a feedback to the input of the master J-K flip flop. The clock pulse [Clk] is given to the master J-K flip flop and it is sent through a NOT Gate and thus inverted before passing it to the slave J-K flip flop.



## Working

When Clock=1, the master J-K flip flop gets disabled. The Clock input of the master input will be the opposite of the slave input. So the master flip flop output will be recognized by the slave flip flop only when the Clock value becomes 0. Thus, when the clock pulse makes a transition from 1 to 0, the locked outputs of the master flip flop are fed through to the inputs of the slave flip-flop making this flip flop edge or pulse-triggered. To understand better take a look at the timing diagram illustrated below.



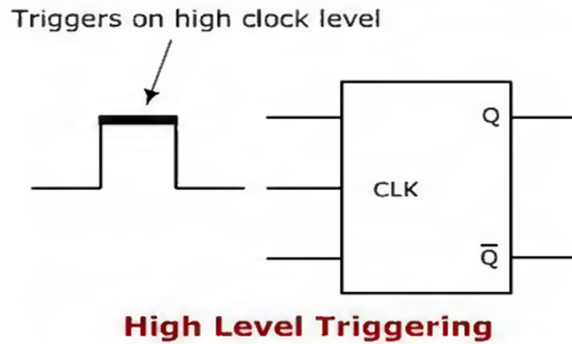
Thus, the circuit accepts the value in the input when the clock is HIGH, and passes the data to the output on the falling-edge of the clock signal. This makes the Master-Slave J-K flip flop a Synchronous device as it only passes data with the timing of the clock signal.

## Triggering of Flip Flops

The output of a flip flop can be changed by bring a small change in the input signal. This small change can be brought with the help of a clock pulse or commonly known as a trigger pulse.

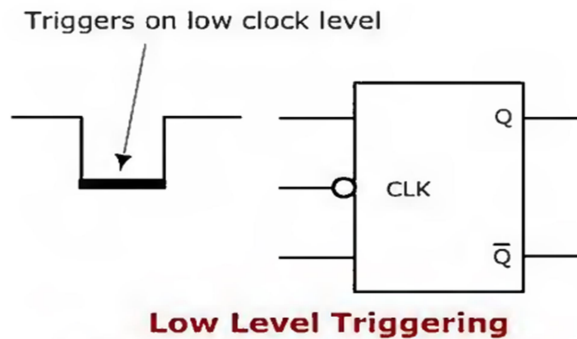
### 1. High Level Triggering

When a flip flop is required to respond at its HIGH state, a HIGH level triggering method is used. It is mainly identified from the straight lead from the clock input. Take a look at the symbolic representation shown below.



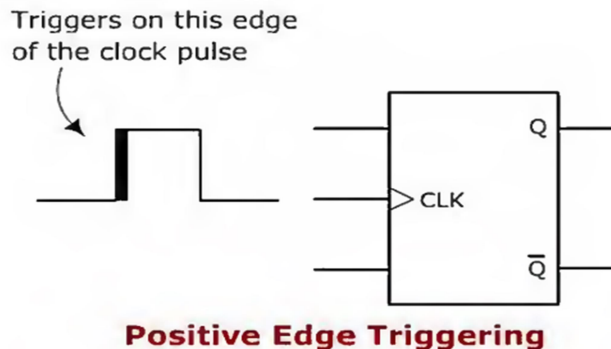
### 4. Low Level Triggering

When a flip flop is required to respond at its LOW state, a LOW level triggering method is used.. It is mainly identified from the clock input lead along with a low state indicator bubble. Take a look at the symbolic representation shown below.



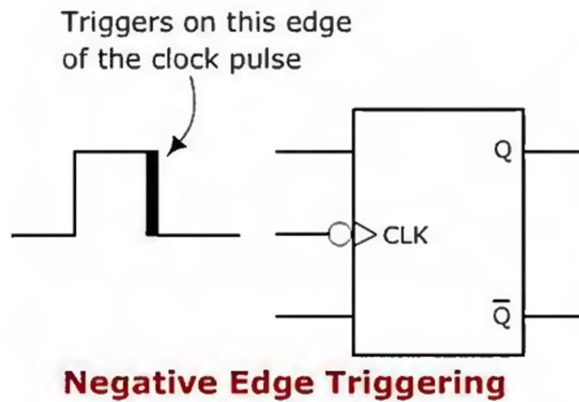
### 3. Positive Edge Triggering

When a flip flop is required to respond at a LOW to HIGH transition state, POSITIVE edge triggering method is used. It is mainly identified from the clock input lead along with a triangle. Take a look at the symbolic representation shown below.



#### 4. Negative Edge Triggering

When a flip flop is required to respond during the HIGH to LOW transition state, a NEGATIVE edge triggering method is used. It is mainly identified from the clock input lead along with a triangle. Take a look at the symbolic representation shown below.



**Flip-flops are used in numerous applications, such as**

- (1) Registers
- (2) Counters
- (3) Event Detectors
- (4) Data Synchronizers
- (5) Frequency Divider

#### **Counters:**

A digital circuit which is used for counting pulses is known *counter*. Counter is the widest application of FFs. It is a group of FFs with a CLK pulse applied. Counter is a register that goes through a prescribed series of states. Counter is a circuit which cycle through state sequence.



## Types of Counters:

The number of FFs used and the way in which they are connected determines the number of states and also the specific sequence of states that the counter goes through during each complete cycle. Counters are classified according to the way they are clocked: *Asynchronous or ripple counters* and *Synchronous counters*

| <b>Asynchronous counter</b>  | <b>Synchronous counter</b>  |
|--|---|
| <ol style="list-style-type: none"><li>1. It is also called as serial counter.</li><li>2. Simple and straight forward in operation.</li><li>3. Slower than synchronous.</li><li>4. Next FF is triggered by previous FF.</li><li>5. Problem of glitch.</li><li>6. Settling time is more.</li></ol> | <ol style="list-style-type: none"><li>1. It is also called as parallel counter.</li><li>2. Complex in operation as compared to asynchronous.</li><li>3. Faster than asynchronous.</li><li>4. All FFs are triggered simultaneously by external CLK.</li><li>5. No problem of glitch.</li><li>6. Settling time is less.</li></ol> |

### Asynchronous counter

- Also known as ripple counter. Ripple counters are the simplest type of binary counters because they require the fewest components to produce a given counting operation.
- Each FF output drives the CLK input of the next FF.
- FFs do not change states in exact synchronism with the applied clock pulses.
- There is delay between the responses of successive FFs.
- It is also often referred to as a ripple counter due to the way the FFs respond one after another in a kind of rippling effect.

## Ripple Counter/Asynchronous Counter

*A ripple counter is an asynchronous counter where only the first flip-flop is clocked by an external clock. All subsequent flip-flops are clocked by the output of the preceding flip-flop. Asynchronous counters are also called ripple-counters because of the way the clock pulse ripples its way through the flip-flops.*

The MOD of the ripple counter or asynchronous counter is  $2^n$  if  $n$  flip-flops are used. For a 4-bit counter, the range of the count is 0000 to 1111 ( $2^4-1$ ). A counter may count up or count down or count up and down depending on the input control.

### Binary Ripple Counter

A binary ripple counter consists of a series connection of complementing flip-flops (T or JK type), with the output of each flip-flop connected to the CP input of the next higher-order flip-flop.

The diagram of a 4-bit binary ripple counter is shown in Fig. All J and K inputs are equal to 1.

The small circle in the CP input indicates that the flip-flop complements during a negative-going transition or when the output to which it is connected goes from 1 to 0.

To understand the operation of the binary counter, refer to its count sequence given in Table 7 - 4. It is obvious that the lowest-order bit  $A_1$  must be complemented with each count pulse.

Every time  $A_1$  goes from 1 to 0, it complements  $A_2$ .

Every time  $A_2$  goes from 1 to 0, it complements  $A_3$ , and so on.

For example, take the transition from count 0111 to 1000.

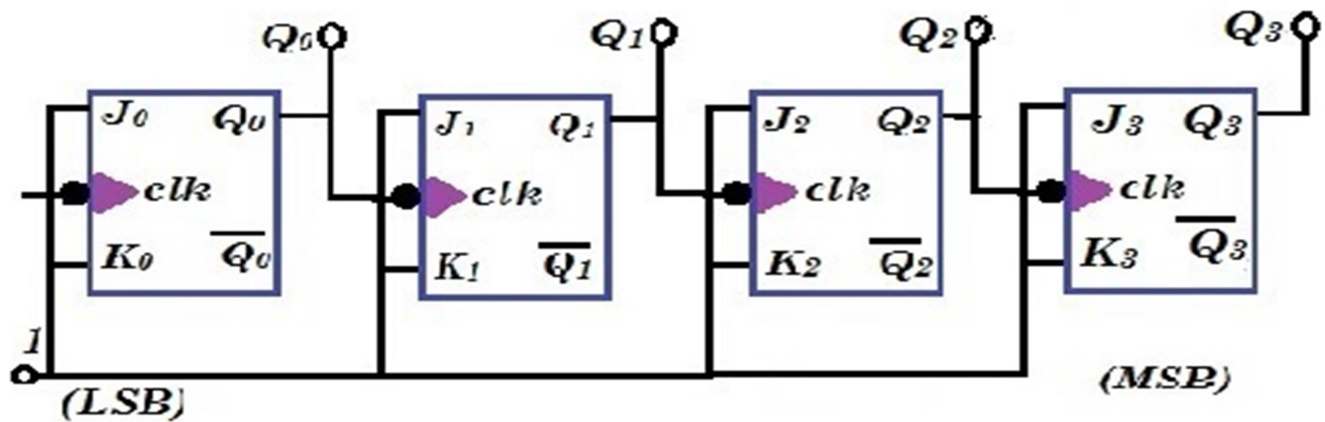
The arrows in the table emphasize the transitions in this case.

$A_1$  is complemented with the count pulse. S

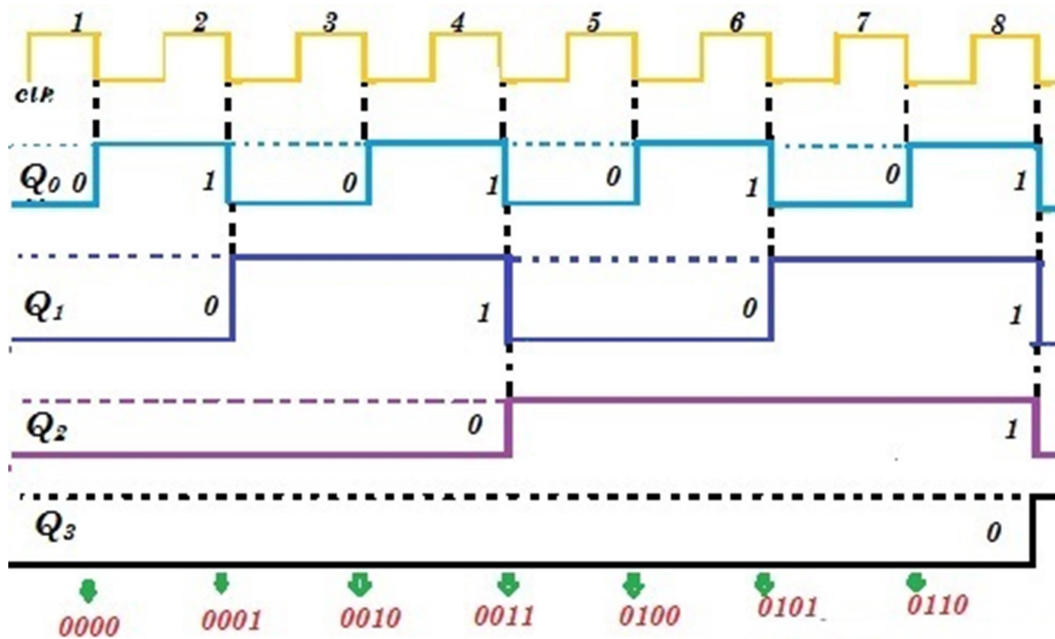
ince  $A_1$  goes from 1 to 0, it triggers  $A_2$ , and complements it.

As a result,  $A_3$ , goes from 1 to 0, which in turn complements  $A_3$ .

$A_3$  now goes from 1 to 0, which complements  $A_4$



4 bit Ripple counter Timing diagram



**Summary:**

- **Asynchronous Counters** can be made from Toggle or D-type flip-flops.
- They are called asynchronous counters because the clock input of the flip-flops are not all driven by the same clock signal.
- Each output in the chain depends on a change in state from the previous flip-flops output.
- Asynchronous counters are sometimes called ripple counters because the data appears to "ripple" from the output of one flip-flop to the input of the next.
- They can be implemented using "divide-by-n" circuits.
- Truncated counters can produce any modulus number count.

**Disadvantages of Asynchronous Counters:**

- An extra "re-synchronizing" output flip-flop may be required.
- To count a truncated sequence not equal to  $2^n$ , extra feedback logic is required.
- Counting a large number of bits, propagation delay by successive stages may become

undesirably large.

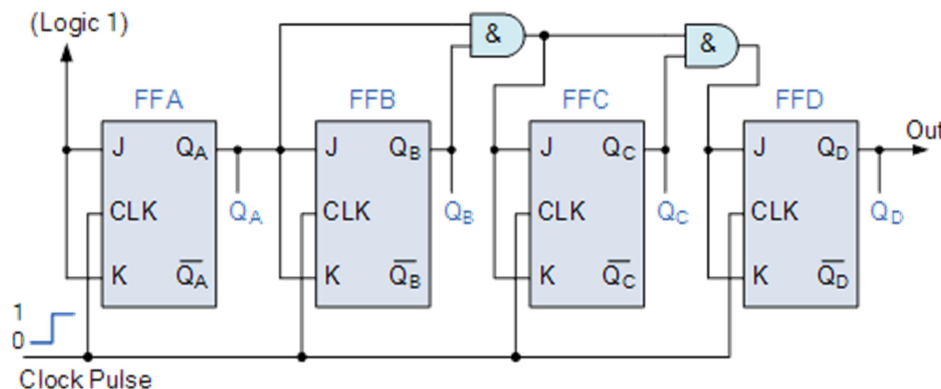
- This delay gives them the nickname of "Propagation Counters".
- Counting errors at high clocking frequencies.
- Synchronous Counters are faster using the same clock signal for all flip-flops.

### Synchronous Counter

In the previous Asynchronous binary counter tutorial, we discussed that the output of one counter stage is connected directly to the clock input of the next counter stage and so on along the chain, and as a result the asynchronous counter suffers from what is known as "Propagation Delay" in which the timing signal is delayed a fraction through each flip-flop.

However, with the **Synchronous Counter**, the external clock signal is connected to the clock input of EVERY individual flip-flop within the counter so that all of the flip-flops are clocked together simultaneously (in parallel) at the same time giving a fixed time relationship. In other words, changes in the output occur in "synchronization" with the clock signal. This results in all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay.

#### Binary 4-bit Synchronous Up Counter

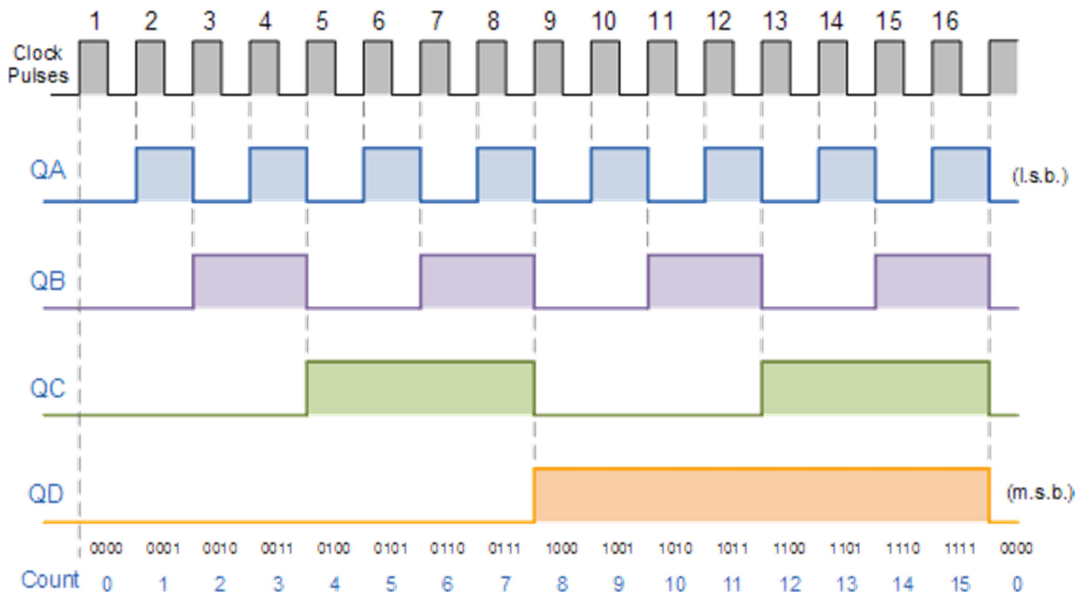


It can be seen that the external clock pulses (pulses to be counted) are fed directly to each **J-K flip-flop** in the counter chain and that both the J and K inputs are all tied together in toggle mode, but only in the first flip-flop, flip-flop A (LSB) are they connected HIGH, logic "1" allowing the flip-flop to toggle on every clock pulse. Then the synchronous counter follows a predetermined sequence of states in response to the common clock signal, advancing one state for each pulse.

The J and K inputs of flip-flop B are connected to the output "Q" of flip-flop A, but the J and K inputs of flip-flops C and D are driven from AND gates which are also supplied with signals from the input and output of the previous stage. If we enable each J- K flip-flop to toggle based on whether or not all preceding flip-flop outputs (Q) are "HIGH" we can obtain the same counting sequence as with the asynchronous circuit but without the ripple effect, since each flip-flop in this

circuit will be clocked at exactly the same time. As there is no propagation delay in synchronous counters because all the counter stages are triggered in parallel the maximum operating frequency of this type of counter is much higher than that of a similar asynchronous counter.

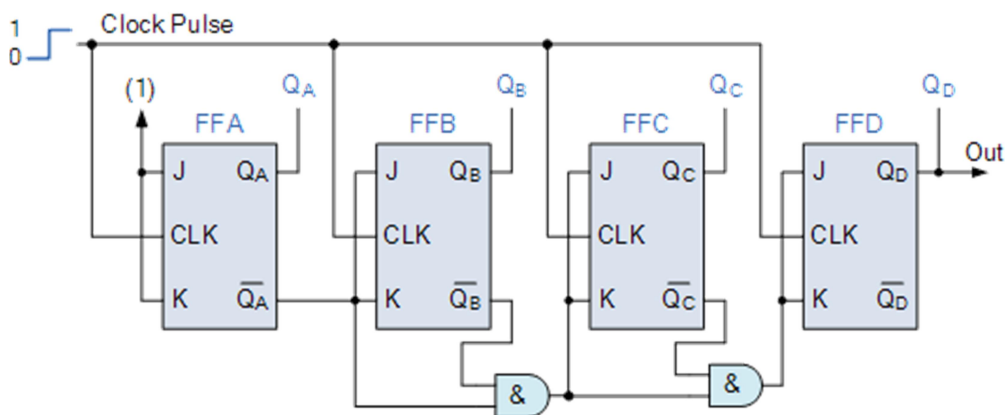
### 4-bit Synchronous Counter Waveform Timing Diagram



Because this 4-bit synchronous counter counts sequentially on every clock pulse the resulting outputs count upwards from 0 ( "0000" ) to 15 ( "1111" ). Therefore, this type of counter is also known as a **4-bit Synchronous Up Counter**.

As synchronous counters are formed by connecting flip-flops together and any number of flip-flops can be connected or "cascaded" together to form a "divide-by-n" binary counter, the modulo's or "MOD" number still applies as it does for asynchronous counters so a Decade counter or BCD counter with counts from 0 to  $2^n-1$  can be built along with truncated sequences.

### Binary 4-bit Synchronous Down Counter



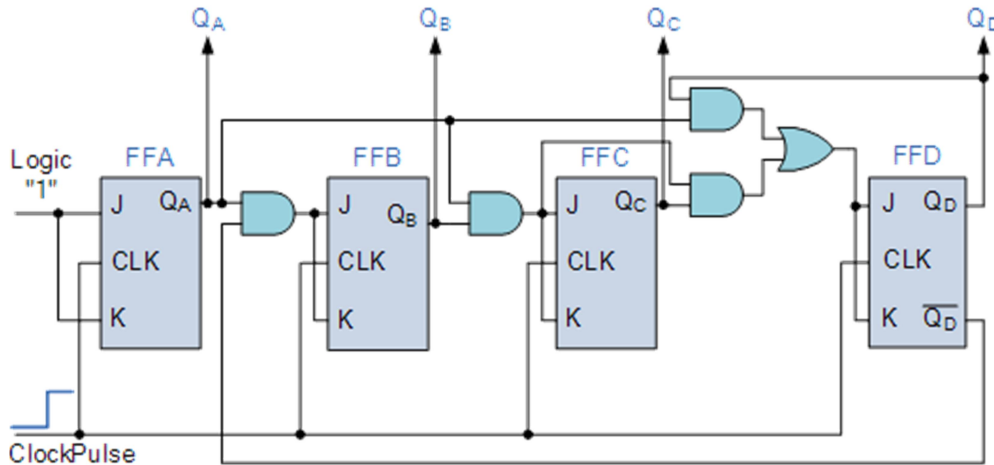
As synchronous counters are formed by connecting flip-flops together and any number

of flip-flops can be connected or “cascaded” together to form a “divide-by-n” binary counter, the modulo’s or “MOD” number still applies as it does for asynchronous counters so a Decade counter or BCD counter with counts from 0 to  $2^n-1$  can be built along with truncated sequences. All we need to increase the MOD count of an up or down synchronous counter is an additional flip-flop and AND gate across it.

### Decade 4-bit Synchronous Counter

A 4-bit decade synchronous counter can also be built using synchronous binary counters to produce a count sequence from 0 to 9. A standard binary counter can be converted to a decade (decimal 10) counter with the aid of some additional logic to implement the desired state sequence. After reaching the count of “1001”, the counter recycles back to “0000”. We now have a decade or Modulo-10 counter.

### Decade 4-bit Synchronous Counter



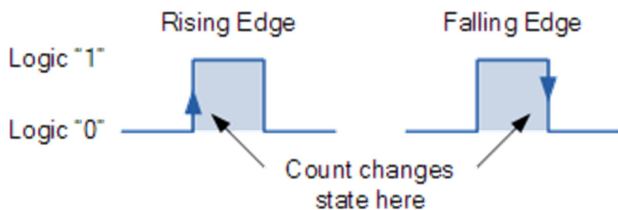
The additional AND gates detect when the counting sequence reaches “1001”, (Binary 10) and causes flip-flop FF3 to toggle on the next clock pulse. Flip-flop FF0 toggles on every clock pulse. Thus, the count is reset and starts over again at “0000” producing a synchronous decade counter.

We could quite easily re-arrange the additional AND gates in the above counter circuit to produce other count numbers such as a Mod-12 counter which counts 12 states from “0000” to “1011” (0 to 11) and then repeats making them suitable for clocks, etc.

### Triggering The Counter

Synchronous Counters use edge-triggered flip-flops that change states on either the “positive-edge” (rising edge) or the “negative-edge” (falling edge) of the clock pulse on the control input resulting in one single count when the clock input changes state.

Generally, synchronous counters count on the rising-edge which is the low to high transition of the clock signal and asynchronous ripple counters count on the falling-edge which is the high to low transition of the clock signal.



It may seem unusual that ripple counters use the falling-edge of the clock cycle to change state, but this makes it easier to link counters together because the most significant bit (MSB) of one counter can drive the clock input of the next.



This works because the next bit must change state when the previous bit changes from high to low – the point at which a carry must occur to the next bit. Synchronous counters usually have a carry-out and a carry-in pin for linking counters together without introducing any propagation delays.