# Sequential Circuit

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

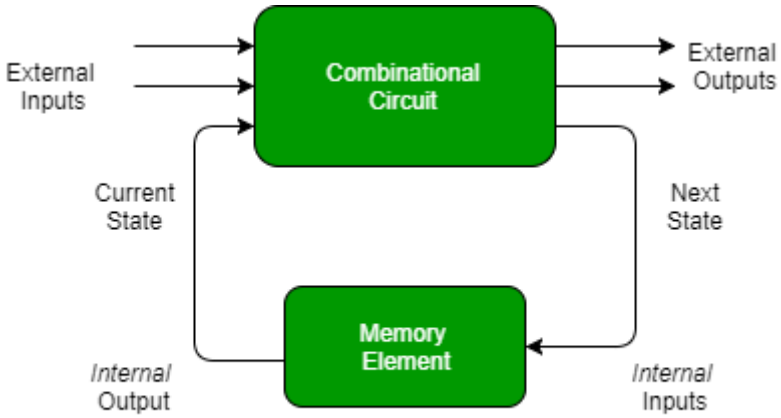



**Figure:** Sequential Circuit

**Types of Sequential Circuits –** There are two types of sequential circuit :

1. **Asynchronous sequential circuit –** These circuit **do not use a clock signal** but uses the pulses of the inputs. These circuits are **faster** than synchronous sequential circuits because there is clock pulse and change their state immediately when there is a change in the input signal. We use asynchronous sequential circuits when speed of operation is important and **independent** of internal clock pulse.

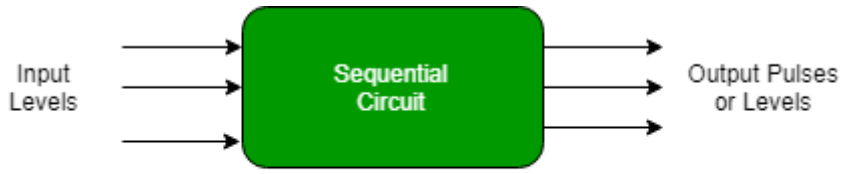



**Figure:** Asynchronous Sequential Circuit

2. **Synchronous sequential circuit** – These circuit **uses clock signal** and level inputs (or pulsed) (with restrictions on pulse width and circuit propagation). The output pulse is the same duration as the clock pulse for the clocked sequential circuits. Since they wait for the next clock pulse to arrive to perform the next operation, so these circuits are bit **slower** compared to asynchronous. Level output changes state at the start of an input pulse and remains in that until the next input or clock pulse.

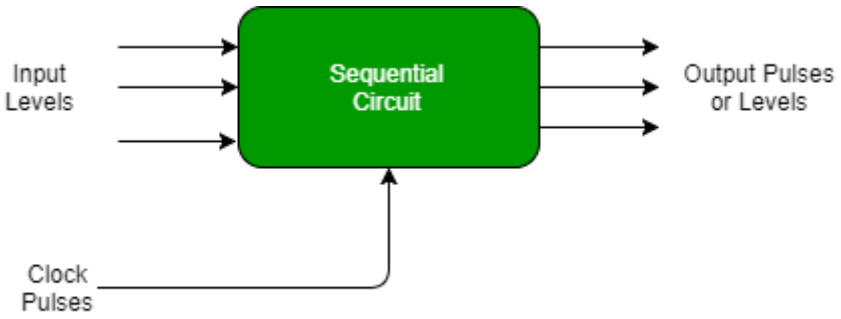
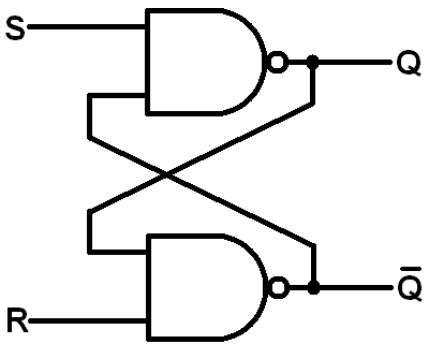



**Figure:** Synchronous Sequential Circuit

**Latches** is **the** basic elements for storing information. One **latch** can store one bit of information. . It is an electronic logic circuit that has two inputs and one output. One of the inputs is **called** the SET input; the other is **called** the RESET input. **Latch** circuits can be either active-high or active-low.There are basically four main **types** of **latches**: SR, D, JK, and T.

**SR Latch:** An **SR latch** (Set/Reset) is an asynchronous device: it works independently of control signals and relies only on the state of the S and R inputs. In the image we can see that an **SR latch** can be created with two NOR gates that have a cross-feedback loop.

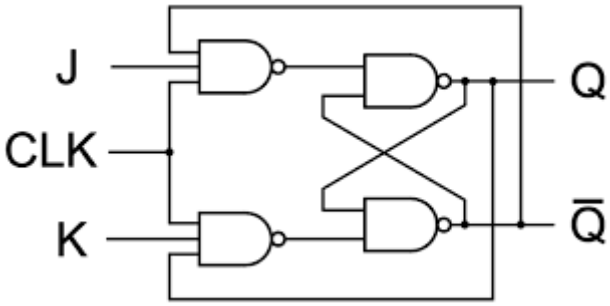
| S | R | Q(t) | Q(t+1) |
|---|---|---|---|
| 0 | 0 | 0 | 0 (No Change) |
| 0 | 0 | 1 | 1 (No Change) |
| 0 | 1 | 0 | 0 (Reset) |
| 0 | 1 | 1 | 0 (Reset) |
| 1 | 0 | 0 | 1(Set) |
| 1 | 0 | 1 | 1(Set) |
| 1 | 1 | 0 | *(Forbidden State) |
| 1 | 1 | 1 | *(Forbidden State) |




**J-K Latch- JK latch** is similar to RS **latch**. This **latch** consists of 2 inputs J and K as shown in the below figure. The ambiguous state has been eliminated here: when the inputs of **Jk**

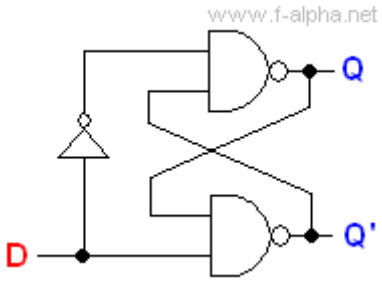
**latch** are high, then output toggles. The output feedback to inputs is the only difference we see here, which is not there in the RS **latch**.

| J | K | Q(t) | Q(t+1) |
|---|---|---|---|
| 0 | 0 | 0 | 0 (No Change) |
| 0 | 0 | 1 | 1 (No Change) |
| 0 | 1 | 0 | 0 (Reset) |
| 0 | 1 | 1 | 0 (Reset) |
| 1 | 0 | 0 | 1(Set) |
| 1 | 0 | 1 | 1(Set) |
| 1 | 1 | 0 | 1(Toggle State) |
| 1 | 1 | 1 | 0(Toggle State) |

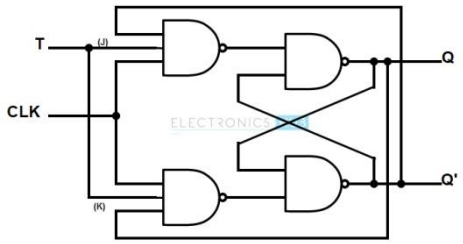



**D latch - Latch** is an electronic device that can be used to store one bit of information. The **D latch** is used to capture, or '**latch**' the logic level which is present on the Data line when the clock input is high.

| D | Q(t) | Q(t+1) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



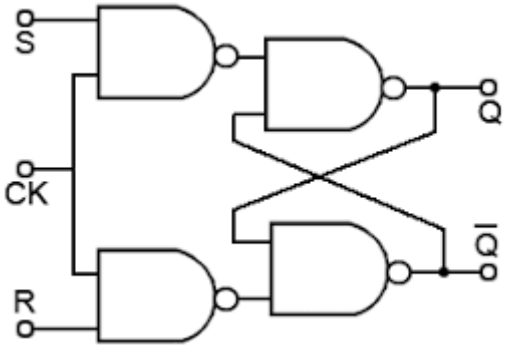
**T Latch**: This **latch** is obtained from JK by connecting both the inputs. This is also known as Toggle **latch** as output is toggled if **T**=1.




**Flip-Flop –** A Flip-Flop (FF) can also store one bit of information like latch but the main difference between these two is, the flipflop has a clock pulse in the input.

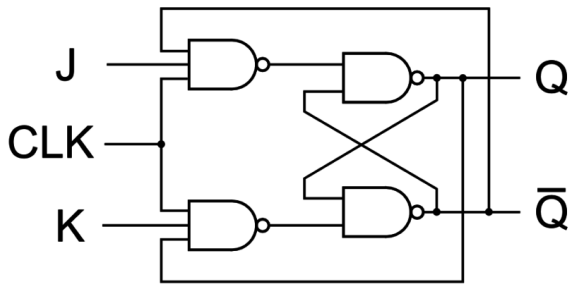
**S-R Flip-Flop**

| CLK | S | R | Q(t) | Q(t+1) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 (No Change) |
| 0 | 0 | 0 | 1 | 1(No Change) |
| 0 | 0 | 1 | 0 | 0(No Change) |
| 0 | 0 | 1 | 1 | 1(No Change) |
| 0 | 1 | 0 | 0 | 0(No Change) |
| 0 | 1 | 0 | 1 | 1(No Change) |
| 0 | 1 | 1 | 0 | 0(No Change) |
| 0 | 1 | 1 | 1 | 1(No Change) |
| 1 | 0 | 0 | 0 | 0(No Change) |
| 1 | 0 | 0 | 1 | 1(No Change) |
| 1 | 0 | 1 | 0 | 0 (Reset) |
| 1 | 0 | 1 | 1 | 0(Reset) |
| 1 | 1 | 0 | 0 | 1 (Set) |
| 1 | 1 | 0 | 1 | 1(Set) |
| 1 | 1 | 1 | 0 | *(Forbidden) |
| 1 | 1 | 1 | 1 | *(Forbidden) |




**JK Flip-Flop**

| CLK | J | K | Q(t) | Q(t+1) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 (No Change) |
| 0 | 0 | 0 | 1 | 1(No Change) |
| 0 | 0 | 1 | 0 | 0(No Change) |
| 0 | 0 | 1 | 1 | 1(No Change) |
| 0 | 1 | 0 | 0 | 0(No Change) |
| 0 | 1 | 0 | 1 | 1(No Change) |
| 0 | 1 | 1 | 0 | 0(No Change) |
| 0 | 1 | 1 | 1 | 1(No Change) |
| 1 | 0 | 0 | 0 | 0(No Change) |
| 1 | 0 | 0 | 1 | 1(No Change) |
| 1 | 0 | 1 | 0 | 0 (Reset) |
| 1 | 0 | 1 | 1 | 0(Reset) |
| 1 | 1 | 0 | 0 | 1 (Set) |
| 1 | 1 | 0 | 1 | 1(Set) |
| 1 | 1 | 1 | 0 | 1(Toggle) |
| 1 | 1 | 1 | 1 | 0(Toggle) |

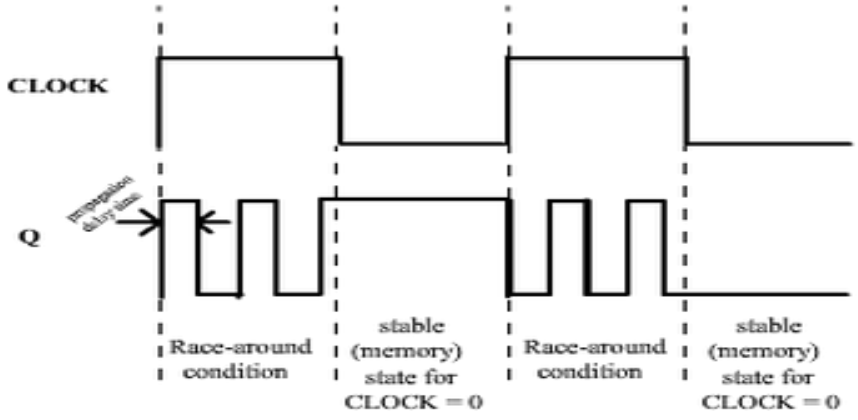


## What is RACE AROUND CONDITION of JK Flip-Flop ?

**Race around condition** arises in **J-K flip flop** when both J=K=1 & it can be **overcome** by using master slave **JK flip flops**. When the S and R inputs of an SR **flip flop** is at logical 1 and then the input is changed to any other **condition**, then the output becomes unpredictable and this is called the **race** around **condition**. The set input causes the output of 0 (top output) and 1 (bottom output). It is an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly.

### Steps to avoid racing condition in JK Flip flop:

1. If the Clock On or High time is less than the propagation delay of the flip flop then racing can be avoided. This is done by using edge triggering rather than level triggering.
2. If the flip flop is made to toggle over one clock period then racing can be avoided.



## RACE AROUND CONDITION

- The race-around condition (Problem) occurs when both the inputs of JK-Flip-flop are 1.
- If the width of the clock pulse $t_P$ is too long, the state of the flip-flop will keep on changing from 0 to 1, 1 to 0 , 0 to 1 and so on, and at the end of the clock pulse, its state will be uncertain.
- This phenomenon is called the *rece around condition*.
- So in order to study race around condition, we have to clear some terms related to clock pulse.
- So let's start with clock pulse.

### MasterSlave FlipFlop

- The **master slave JK flip flop** is a combination of a clocked **JK** latch and a clocked SR latch. The clocked **JK** latch acts as the **master** and the clocked SR latch acts as the **slave**. **Master** is positive level triggered and due to the presence of an inverter in the clock line, the **slave** is negative level edge triggered. n computer networking, **master/slave** is a model for a communication protocol in which one device or process (known as the **master**) controls one or more other devices or processes (known as slaves). Once the **master/slave** relationship is established, the direction of control is always from the **master** to the **slave**(s). The master slave JK flip flop is a combination of a clocked JK latch and a clocked SR latch. The clocked JK latch acts as the master and the clocked SR latch acts as the slave.

- Master is positive level triggered and due to the presence of an inverter in the clock line, the slave is negative level edge triggered. Hence when clock=1, the master is active and slave is inactive. Vice versa happens when clock=0.

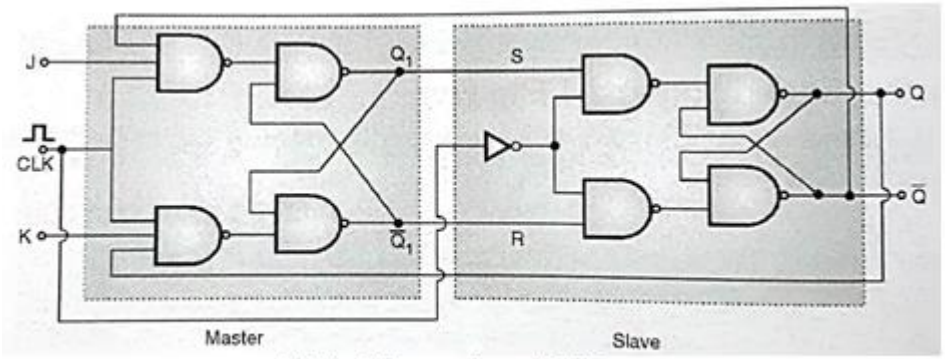


Fig6. Master slave JK FF

- The following is truth table of master slave flip flop.

| Case | Inputs | | | Outputs | | Remark |
|---|---|---|---|---|---|---|
| | CLK | J | K | $Q_{n+1}$ | $\bar{Q}_{n+1}$ | |
| I | × | 0 | 0 | $Q_n$ | $\bar{Q}_n$ | No change |
| II | ⎍(1) | 0 | 0 | $Q_n$ | $\bar{Q}_n$ | No change |
| III | ⎍(1) | 0 | 1 | 0 | 1 | Reset |
| IV | ⎍(1) | 1 | 0 | 1 | 0 | Set |
| V | ⎍(1) | 1 | 1 | $\bar{Q}_n$ | $Q_n$ | Toggle |

Fig7. Truth table of Master slave JK FF

- Operation:

**Case I:** When clock is not given, both master and slave are inactive and there will be no change in outputs.

**Case II:** For clock=1, master is active, slave inactive. As J=K=0, output of master ie Q and Q' will not change. As soon as clock goes to 0, slave becomes active, and master inactive. But since input to slave S and R is same, output of slave will also remain same.

**Case III:** For clock=1, master is active and slave is inactive. When J=0 and K=1, outputs of master will be Q=0, Q'=1, which will be inputs to slave. When clock=0, slave becomes active and takes inputs 0,1 to give output Q=0, Q'=1. This output will not change if clock is again made 1and then 0. Hence we get a stable output from master and slave.

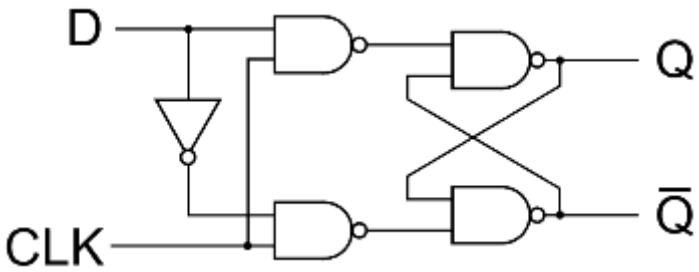
**Case IV:** For clock=1, master is active and slave is inactive. When J=1 and K=0, outputs of master will be Q=1, Q'=0, which will be inputs to slave. When clock=0, slave becomes active and takes inputs 1,0 to give output Q=1, Q'=0. This output will not change if clock is again made 1 and then 0. Hence we get a stable output from master and slave.

**Case V:** When clock =1, J=K=1, master output will toggle. So S and R will invert. But slave remains inactive all this time since clock is 1. As soon as clock becomes 0, slave becomes

active and master becomes inactive. So slave will also toggle. These changed outputs are returned through feedback to the master, but master does not respond to them because clock is now 0 and master is inactive. Thus, in one clock period, master and slave both toggle only once, avoiding race condition caused by multiple toggling.

**D Flip-Flop**

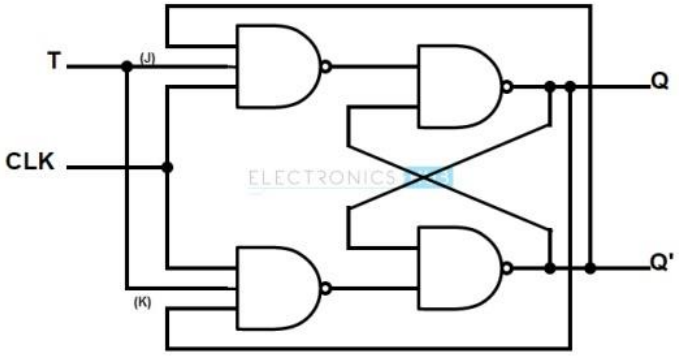
| CLK | D | Q(t) | Q(t+1) | REMARK |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | No Change |
| 0 | 0 | 1 | 1 | No Change |
| 0 | 1 | 0 | 0 | No Change |
| 0 | 1 | 1 | 1 | No Change |
| 1 | 0 | 0 | 0 | Reset |
| 1 | 0 | 1 | 0 | Reset |
| 1 | 1 | 0 | 1 | Set |
| 1 | 1 | 1 | 1 | Set |




**T Flip-Flop**

| CLK | T | Q(t) | Q(t+1) | Remarks |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | No Change |
| 0 | 0 | 1 | 1 | No Change |
| 0 | 1 | 0 | 0 | No Change |
| 0 | 1 | 1 | 1 | No Change |
| 1 | 0 | 0 | 0 | No Change |

| | | | | |
|---|---|---|---|---|
| **1** | **0** | **1** | **1** | **No Change** |
| **1** | **1** | **0** | **1** | **Toggle** |
| **1** | **1** | **1** | **0** | **Toggle** |




******** ***THERE ARE SOME FLIPFLOP CONVERSIONS WHICH ARE ALSO INCLUDED IN YOUR SYLLABUS. I WILL TEACH YOU THESE AFTER REOPENING YOUR COLLEGE.***