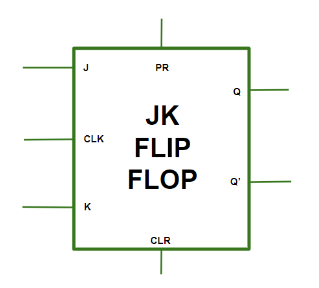
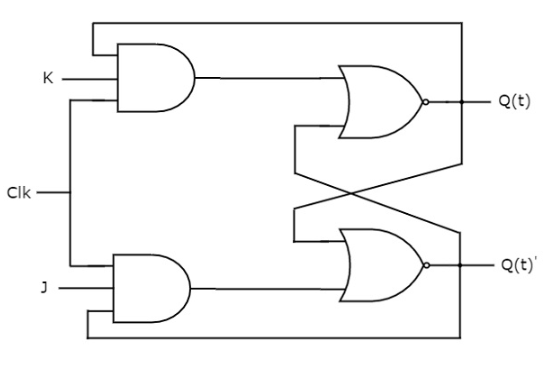
Q1.) Explain functioning of JK and SR flip flop ?

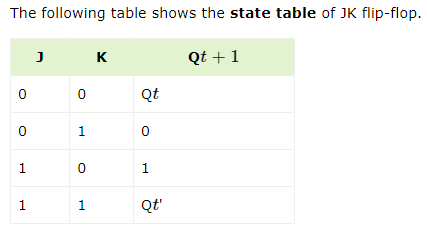
Ans.

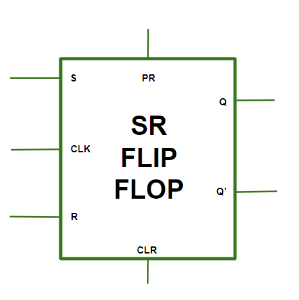
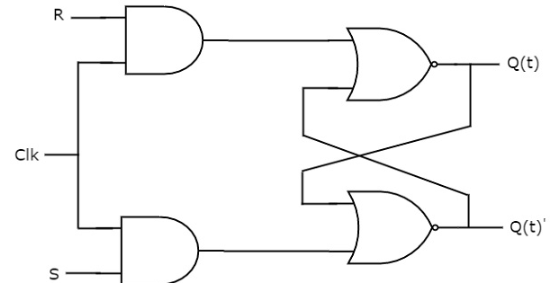
[Flip-Flop](https://www.geeksforgeeks.org/flip-flop-types-their-conversion-and-applications/) is popularly known as the basic digital memory circuit. It has two states as logic 1(High) and logic 0(low) states. A flip flop is a sequential circuit which consists of a single binary state of information or data. The digital circuit is a flip flop which has two outputs and are of opposite states. It is also known as a Bistable Multivibrator.

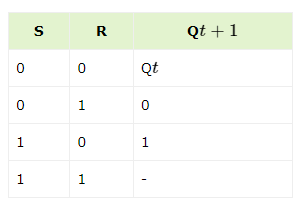
**1. JK Flip Flop :**

The JK flip flop diagram below represents the basic structure which consists of Clock (CLK), Clear (CLR), and Preset (PR).

  
  
**Race Around Condition in JK Flip-Flop –**  
When the J and K both are set to 1, the input remains high for a longer duration of time, then the output keeps on toggling. Toggle means switching in the output instantly i.e. Q = 0, Q’ = 1 will immediately change to Q = 1 and Q’ = 0 and this continuation keeps on changing. This change in output leads to Race Around Condition.  
  
  
  
  
**2. SR Flip-Flop :**  
In SR flip flop, with the help of Preset and Clear, when the power is switched ON, the state of the circuit keeps on changing, i.e. it is uncertain. It may come to Set (Q = 1) or Reset (Q’ = 0) state. In many applications, it is desired to initially Set or Reset the flip flop. This thing is accomplished by the Preset (PR) and the Clear (CLR).

**Applications of Flip-Flop :**

1. Flipflops are used as a bounce elimination switch.
2. They are used as a serial to parallel and parallel to serial conversion.
3. It is used for counters.
4. It is used for frequency divider and also as a latch.

2) Difference between latch and flip flop ?

Ans :-

|  |  |
| --- | --- |
| **Latch** | **Flip-Flop** |
| The latch is transparent – because the input is directly connected to the output when enable is high. It means Latch is sensitive to pulse duration (also called soft barrier) | Flip-flop is a pair of latches (master and slave flop). Flip-flop is sensitive to pulse transition. The signal only propagates through on the rising/falling edge (also called hard barrier) |
| Less Area (less gates) | More Area (more gates) because the flip-flop contains two latches. |
| Less Power (less gates) | More Power (more gates) |
| Fast –  (The longer combinational path can be compensated by shorter path delays in the subsequent logic stages. That’s why, for higher performance, circuits designer are turning to latched-based design.) | Slow –  (The delay of a combinational logic path of a design using edge-triggered flip-flops is always less than the clock period except for those specified as false paths and multiple-cycle paths. Hence the longest path of a design limits the circuit performance.) |
| Require more tool manipulation and more hand calculations to verify that they meet the timing | Easy to check design timing using Static Timing Analysis (STA) tools |
| Cycle-borrowing to gain more setup time on the next register stage, as long as each loop completes in one cycleTo meet the timing in the design, Designers consider latches to adjust timing mismatch. | Data launches on one rising edge, so it must be set up before the next rising edge. If it arrives late, the system fails. If it arrives early, time is wasted due to hard edges in Flops |
| For ASICs with large clock skew, latches have substantial benefits for reducing the clock period | Even for the high-speed pulsed flip-flops with zero setup time, as they are not transparent, the impact of the clock skew is not reduced |
| Level-sensitive latches reduce the impact of the inaccuracy of wire load models and process variation. | Flip-flops demand the highly accurate wire load model and process |
| In [DFT](https://en.wikipedia.org/wiki/Design_for_testing), Latches are needed as a lockup state at the clock domain crossings in the scan chain to avoid unpredictable behavior | In DFT, use flops that can be scanned (controllable and observable) |
| In FPGA, level-sensitive transparent latches should be avoided in FPGAs | In FPGA, edge-sensitive flip-flops are used exclusively. Timings analysis is more appropriate with flops for FPGA tools |
| Circuit analysis is complex. You may see last minutes timing mismatch surprises at the implantation stage. | Circuit analysis is easy |
| High-speed microprocessor designs typically use master-slave latches instead of flip-flops so that logic can be added between the rising and falling clock edges.Most of these companies have written their own specialized STA tools to verify latch-based designs. | The most commonly used flop in the design world is D type flip-flop.FSM implementation mostly involves D Flip-flops due to a minimum number of logic gates and lesser cost as compared to other types of flip-flops. |
| For non-timing-critical configuration registers, latches work great, due to fewer gates and less power consumption | For non-power aware design, Flip flops are preferred over Latches |
| The latch is an asynchronous block. Therefore you must ensure that the combinational functions, which generate input signals for the latch, are race-free. Otherwise, they may generate glitches, which may be latched, causing hazards in your system. | A flip-flop, on the other hand, is edge-triggered and only changes state when a control signal goes from high to low or low to high |
| Latch-based design is noisy because any noise in the enable signal disrupts the latch output easily. | Flip-flop-based design is robust |

Flip-flop performs Synchronous operations

🡪 **Synchronous** operations mean that the state changes in the flip-flop are controlled by a clock signal. All state transitions occur in step with this clock signal, ensuring that changes happen simultaneously across the circuit, maintaining synchrony. This is crucial for maintaining timing consistency and avoiding glitches in digital circuits.

Latch performs Asynchronous operations.

* **Asynchronous** operations mean that the state transitions in a latch are not coordinated by a clock signal. Instead, the state can change immediately in response to the input signals. This allows latches to respond quickly to changes but can also make their behavior more complex to manage in large digital systems.

**Asynchronous Flip-Flops**

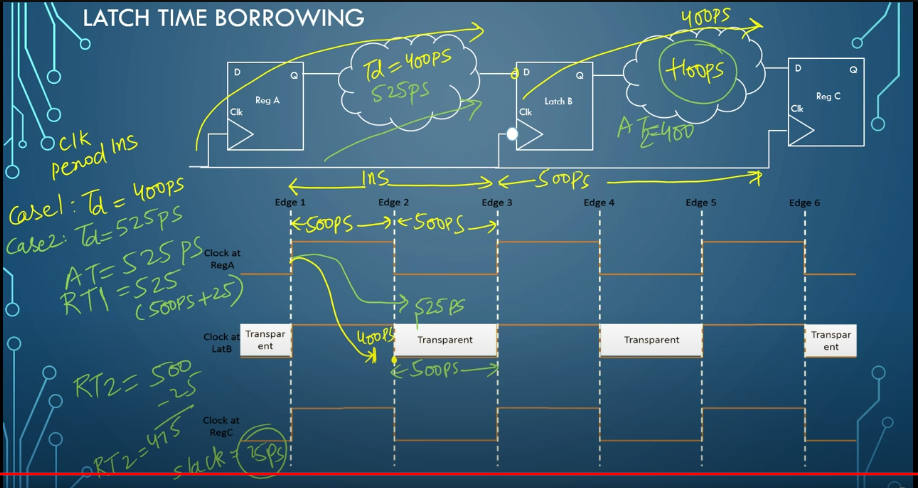
**Asynchronous flip-flops** (often referred to as **clockless flip-flops**) are a special category of flip-flops that incorporate asynchronous elements, allowing their states to be influenced not only by a clock signal but also by direct, immediate inputs. In practice, this usually means they have additional control inputs that can force the flip-flop to a particular state without waiting for the clock. These control inputs often include preset (set) and clear (reset) signals that operate asynchronously.

Difference between SR latch and SR flip flop.

 **SR Latch**: An asynchronous device that responds immediately to input signals. Suitable for simple applications and immediate responses but can be prone to timing issues.

 **SR Flip-Flop**: A synchronous device that responds to the clock signal, making it ideal for applications needing precise timing and coordination, such as registers and counters.

**Time borrowing**

****

**Q3) Why are latches faster than flip flop ?**

**Ans)**

Latches can be faster than flip-flops due to their **level-sensitive nature** and **immediate response** to input signals. Here’s a detailed explanation of the reasons:

**1. Level Sensitivity vs. Edge Sensitivity**

**Latches (Level-Sensitive)**

* **Immediate Response**: Latches are transparent when enabled, meaning they can immediately pass data through from input to output while the enable signal (or clock) is in the active state (high or low, depending on the design).
* **No Edge Delay**: There is no need to wait for a clock edge to change the state; the state can change continuously as long as the latch is enabled.

**Flip-Flops (Edge-Sensitive)**

* **Edge-Triggered**: Flip-flops change their state only at specific clock edges (rising or falling), meaning they must wait for a clock transition to capture the input data.
* **Clock Dependency**: The operation is synchronized with the clock, which can introduce delays as the data change must be aligned with the clock edge.

**2. Latency in Data Propagation**

**Latches**

* **Minimal Latency**: When enabled, latches allow data to propagate through with minimal latency, essentially as fast as the internal gate delays permit. This immediate data propagation can make latches quicker in scenarios where continuous data flow is required.

**Flip-Flops**

* **Clock Cycle Dependency**: Flip-flops introduce a half or full clock cycle delay because they capture and hold data only at clock edges. This means that any change in the input is not reflected in the output until the next clock edge.

**3. Clocking Overhead**

**Latches**

* **Lower Overhead**: Latches typically involve fewer gates and simpler logic compared to flip-flops, leading to reduced clocking overhead and potentially faster operation.

**Flip-Flops**

* **Additional Logic**: Flip-flops usually involve additional logic to handle edge triggering, such as master-slave configurations or extra gates, which can add delay compared to the simpler latch structure.

**4. Time Borrowing Capability**

Latches can take advantage of **time borrowing**:

* **Time Borrowing**: Latches can "borrow" time from one phase of the clock to another in a two-phase clocking system, allowing flexibility in timing that can reduce critical path delays and improve overall speed.

**5. Clock Frequency and Setup Time**

**Latches**

* **Flexible Timing**: Because latches are level-sensitive, they can adapt more flexibly to changes in the input signal timing, which can reduce the need for stringent setup and hold time constraints.
* **High-Speed Applications**: This makes them suitable for high-speed applications where clock periods are short, and the ability to pass data quickly is critical.

**Flip-Flops**

* **Fixed Timing**: Flip-flops require data to be stable before and after the clock edge (setup and hold times), which can impose stricter timing constraints and limit the maximum clock frequency.

**6. Comparison of Operation**

Here’s a comparison table that summarizes why latches can be faster than flip-flops:

| **Aspect** | **Latches (Level-Sensitive)** | **Flip-Flops (Edge-Sensitive)** |
| --- | --- | --- |
| **Response to Input** | Immediate when enabled | Delayed until next clock edge |
| **Data Propagation** | Continuous as long as enabled | Captured and held at clock edge |
| **Timing Overhead** | Lower (fewer gates, no edge triggering) | Higher (additional logic for edge triggering) |
| **Time Borrowing** | Possible, allowing flexibility | Not possible, strict edge-based operation |
| **Setup and Hold Time** | More flexible, less stringent constraints | Strict setup and hold time requirements |
| **Speed in Fast Systems** | Generally faster due to continuous data flow | Generally slower due to dependency on clock edges |

**Example Scenario: Pipelined Data Path**

In a high-speed pipelined data path:

* **Latches**: Can pass data immediately through each stage when enabled, which can reduce overall delay if stages can overlap their operation (e.g., during different phases of the clock).
* **Flip-Flops**: Each stage must wait for the clock edge to capture and pass data, potentially introducing more cumulative delay through the pipeline stages.

**Summary**

Latches can be faster than flip-flops primarily due to their level-sensitive nature, allowing immediate data propagation without waiting for clock edges. This immediate response reduces latency, clocking overhead, and can offer flexibility in timing (time borrowing), which makes them particularly useful in high-speed applications. Flip-flops, while providing synchronized and stable data storage, inherently introduce delays tied to the clock edges, making them slower in scenarios requiring rapid data flow.

This flexibility of latches is why they are often used in applications where speed is critical, while flip-flops are preferred in systems requiring precise synchronization and data integrity across clock cycles.

* **Difference Between Combinational and Sequential Circuit**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Combinational Circuit** | **Sequential Circuit** |
| Meaning and Definition | It is a type of circuit that generates an output by relying on the input it receives at that instant, and it stays independent of time. | It is a type of circuit in which the output does not only rely on the current input. It also relies on the previous ones. |
| Feedback | A Combinational Circuit requires no feedback for generating the next output. It is because its output has no dependency on the time instance. | The output of a Sequential Circuit, on the other hand, relies on both- the previous feedback and the current input. So, the output generated from the previous inputs gets transferred in the form of feedback. The circuit uses it (along with inputs) for generating the next output. |
| Performance | We require the input of only the current state for a Combinational Circuit. Thus, it performs much faster and better in comparison with the Sequential Circuit. | In the case of a Sequential Circuit, the performance is very slow and also comparatively lower. Its dependency on the previous inputs makes the process much more complex. |
| Complexity | It is very less complex in comparison. It is because it basically lacks implementation of feedback. | This type of circuit is always more complex in its nature and functionality. It is because it implements the feedback, depends on previous inputs and also on clocks. |
| Elementary Blocks | Logic gates form the building/ elementary blocks of a Combinational Circuit. | Flip-flops form the building/ elementary blocks of a Sequential Circuit. |
| Operation | One can use these types of circuits for both- Boolean as well as Arithmetic operations. | You can mainly make use of these types of circuits for storing data. |

Q4) Explain the use of Latch and Flip-Flop ?

Ans) Use of latch

A **latch** is a fundamental building block in digital electronics, used extensively in various applications to store and control data. Here are some practical uses and applications of latches:

**1. Data Storage**

* **Registers:** Latches form the basis of registers, which are used to store data temporarily in processors, microcontrollers, and other digital systems.
* **Memory Elements:** In RAM and other memory devices, latches are used to store bits of data.

**2. Data Synchronization**

* **Metastability Resolution:** Latches help in synchronizing asynchronous signals to a clocked system, reducing the chance of metastability.
* **Pipeline Stages:** In pipelined processors, latches are used to hold data between pipeline stages, ensuring that operations can be executed in an orderly manner.

**Use of flip flop**

**Flip-flops** are fundamental elements in digital electronics used for data storage, synchronization, and state retention. They are bistable devices, meaning they have two stable states and can store a single bit of data. Here are practical applications and uses of flip-flops:

**1. Data Storage**

**Registers:**

* **General Purpose Registers:** Store temporary data in CPUs.
* **Configuration Registers:** Hold configuration settings for various digital systems.

**Memory Devices:**

* **SRAM:** Flip-flops are used in the design of Static Random-Access Memory (SRAM) cells.
* **Buffering:** Temporary storage in memory buffers for data synchronization and transfer.

**2. State Machines**

**Finite State Machines (FSMs):**

* **Control Units:** Flip-flops hold the current state in FSMs used for control logic in CPUs, communication protocols, and other digital systems.
* **Sequential Logic:** Create the required sequence of operations in automated systems.

**Sequence Generators:**

* **Pattern Generation:** Used to generate specific sequences for test patterns or repetitive tasks.

**3. Counters and Dividers**

**Binary Counters:**

* **Up/Down Counters:** Counting operations in digital systems, such as timers or event counters.
* **Frequency Division:** Frequency dividers that generate lower-frequency signals from a higher-frequency clock.

**Digital Clocks:**

* **Timekeeping:** Counting seconds, minutes, and hours in digital clocks and timers.

**4. Shift Registers**

**Serial-to-Parallel Conversion:**

* **Data Transmission:** Convert serial data into parallel data for interfacing with parallel buses.
* **Parallel-to-Serial Conversion:** Convert parallel data into serial form for transmission over serial communication links.

**Data Storage:**

* **Shift Operations:** Temporary storage and shifting of data in digital circuits.

**5. Timing and Synchronization**

**Edge Detection:**

* **Pulse Detection:** Detect edges of signals (rising or falling) in digital circuits.

**Synchronization:**

* **Metastability Handling:** Synchronize asynchronous signals to a clocked system, minimizing metastability risks.

**6. Control Systems**

**Flip-Flop Based Control:**

* **Debouncing:** Debounce switches and mechanical contacts to produce clean signals.
* **Control Signals:** Generate and hold control signals in complex digital systems.

**7. Memory and Storage Applications**

**Flip-Flop Chains:**

* **Delay Elements:** Create delays in digital signals, useful in timing adjustments.
* **Pipeline Registers:** Store intermediate results in pipelined processors.

**Static RAM (SRAM):**

* **Memory Cells:** Basic storage elements in SRAM use flip-flops to store data.

**8. Digital Communication**

**Synchronization:**

* **Clock Recovery:** Recover clock signals from data streams in communication systems.
* **Data Recovery:** Ensure correct alignment and timing of data bits.

**9. Specialized Flip-Flop Applications**

**T Flip-Flop:**

* **Toggle Operations:** Used in applications requiring toggling action, such as in frequency division.

**JK Flip-Flop:**

* **Versatile Storage:** Used for various applications including counters, shift registers, and FSMs.

**D Flip-Flop:**

* **Data Latching:** Commonly used in synchronous circuits for latching data on clock edges.

### Conclusion

Flip-flops are versatile components that enable a wide range of functionalities in digital systems. They are essential for data storage, state retention, timing synchronization, and control in numerous digital and communication applications. Their ability to maintain state and synchronize data makes them critical in the design of modern digital electronics.

Q) Why is gated sr flip flop called asynchronous latch ?

Ans. The **gated SR flip-flop** is often called an **asynchronous latch** because of its ability to respond to inputs without being tied to a clock signal in the way synchronous latches are. Here’s a detailed explanation of why it’s considered asynchronous:

**Understanding Gated SR Flip-Flop**

A **gated SR (Set-Reset) flip-flop** is a type of bistable multivibrator that can hold a state and change it based on input signals. It has a **gate** or **enable** input in addition to the usual Set (S) and Reset (R) inputs, allowing it to be more controlled compared to a simple SR latch.

**Why It’s Called an Asynchronous Latch**

1. **Immediate Response to Inputs:**
   * Unlike synchronous latches or flip-flops that change states only on specific clock edges, a gated SR flip-flop responds to changes in its inputs (S, R) immediately when the enable signal is active. This means it does not wait for a clock pulse to change its state, which is characteristic of asynchronous behavior.
2. **Enable Signal as a Control:**
   * The enable signal controls when the SR flip-flop can respond to its Set and Reset inputs. While the enable signal provides some control, the actual state change happens as soon as S or R changes, independent of a global clock.
3. **Behavior Independent of Clock:**
   * Asynchronous circuits operate based on the timing of the input signals rather than relying on a clock signal. Since the gated SR flip-flop changes states immediately with S and R when enabled, its operation is independent of any clock signal.
4. **Simpler Timing Constraints:**
   * In asynchronous circuits, timing constraints are simpler because they do not involve clock edge synchronization. The gated SR flip-flop only requires that the inputs are stable while the enable signal is active.

**Key Characteristics of Asynchronous Latches (Gated SR Flip-Flops):**

1. **Immediate State Change:**
   * The state of the flip-flop changes as soon as the inputs (S or R) change, provided the enable is active.
2. **No Clock Signal Required:**
   * Unlike synchronous latches that require a clock pulse to trigger state changes, the gated SR flip-flop operates without a clock.
3. **Enable Control:**
   * The state changes are gated by an enable signal. When the enable is high, the flip-flop responds to S and R inputs.
4. **Simple Construction:**
   * It consists of basic logic gates and does not require complex clocking circuitry, making it simpler than synchronous designs.

**Applications of Gated SR Flip-Flop:**

* **Control Systems:** Useful in systems where immediate response to input changes is needed, such as control circuits in simple state machines.
* **Debouncing:** Helps in switch debouncing where the immediate response to input changes can be used to stabilize mechanical switches.
* **Level Triggered Devices:** Can be used in situations where level triggering is preferable over edge triggering.

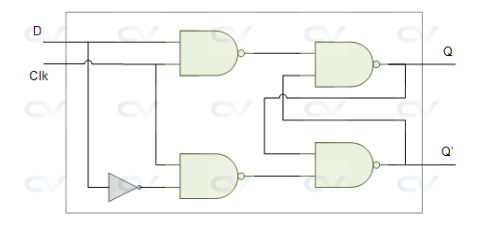
### ****Differences Between Asynchronous Latches and Synchronous Latches:****

| **Feature** | **Asynchronous (Gated SR) Latch** | **Synchronous Latch** |
| --- | --- | --- |
| **Timing** | Immediate response to inputs | Changes on clock edge |
| **Clock Requirement** | No clock signal needed | Requires a clock signal |
| **Complexity** | Simpler circuitry | More complex circuitry |
| **Usage** | Control systems, debouncing | Data storage, registers |

### ****Conclusion****

The term **asynchronous latch** for a gated SR flip-flop highlights its immediate, input-driven response that doesn’t depend on a clock pulse. This characteristic allows it to act quickly based on input signals when enabled, distinguishing it from synchronous counterparts that operate in step with clock signals.

Q) D-FF using nand gate.



Q) D-FF using 2\*1 mux

