5 Introduction to Sequential Logic

In digital circuits, **sequential logic** allows systems to store and use previous information, responding to past states as well as current inputs. This differs from **combinational logic**, which operates purely based on current inputs.

Key Concepts

- **Sequential Circuits**: Depend on both present inputs and past outputs, using feedback for memory.
- **Flip-Flops**: Basic memory elements in sequential circuits, each storing one bit of information.
- **Applications**: Registers, counters, memory units, and finite state machines (FSMs), are all built from flip-flops, making them foundational in digital systems.

6 SR (Set-Reset) Flip-Flop

The **SR flip-flop** stores a single bit of data with two main inputs, **Set (S)** and **Reset (R)**, and two outputs, Q and $\overline{Q}$, which are complementary.

6.1 Circuit and Operation

The SR flip-flop is often constructed with two cross-coupled NAND gates as shown in Figure 1.

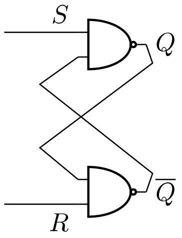


Figure 1: SR Flip-Flop Circuit

6.2 Truth Table

The operation of the SR flip-flop can be described by the truth table:

S	R	Q	$\overline{Q}$
0	0	Hold	Hold
0	1	0	1
1	0	1	0
1	1	Invalid	Invalid

Key Points:

- **Hold State**: When $S = 0$ and $R = 0$, the output holds the previous state.
- **Set State**: $S = 1$, $R = 0$ sets $Q = 1$.
- **Reset State**: $S = 0$, $R = 1$ sets $Q = 0$.
- **Invalid State**: $S = 1$ and $R = 1$ create an invalid condition, to be avoided.

Example

- **Try this**: If $Q = 0$ initially, and $S = 1$, $R = 0$, what is the new value of Q?
- **Solution**: $Q = 1$ since the set input is activated.

7 JK Flip-Flop

The **JK flip-flop** improves on the SR flip-flop by resolving the invalid state. It has two inputs, **J** and **K**, similar to Set and Reset, but the flip-flop toggles its output when both are high.

7.1 Truth Table

The JK flip-flop operates according to the following truth table:

J	K	Q_{next}
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Explanation:

- $J = 0$, $K = 0$: The output Q retains its state.
- $J = 0$, $K = 1$: Resets the output to 0.
- $J = 1$, $K = 0$: Sets the output to 1.
- $J = 1$, $K = 1$: Toggles Q to the opposite of its current state.

Example

- **Try this**: For an initial state of $Q = 0$, with inputs $J = 1$ and $K = 1$, what will Q be after one clock pulse?
- **Solution**: $Q = 1$, as the JK flip-flop toggles.

8 D (Data) Flip-Flop

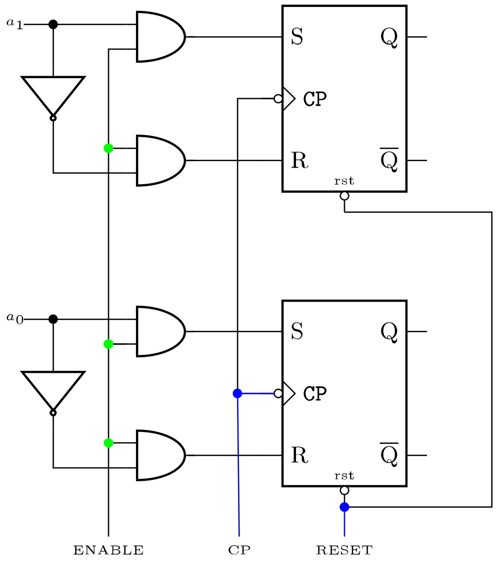
The **D flip-flop** is designed for a single data input D, with a clock input CLK. The output Q simply follows D at each clock edge.

$$Q_{\text{next}} = D$$

Example

- **Try this**: Assume a D flip-flop with $D = 1$ at the next clock edge. If Q starts at 0, what will be the value of Q after the clock pulse?
- **Solution**: $Q = 1$, since the D flip-flop directly reflects D at the clock edge.

9 Edge-Triggered Flip-Flops



Edge-triggered flip-flops change state only on a clock edge—rising (0 to 1) or falling (1 to 0). This allows precise timing control essential for synchronized operations.

9.1 Positive and Negative Edge Triggering

- **Positive-edge Triggered**: Activates on the rising clock edge.
- **Negative-edge Triggered**: Activates on the falling clock edge.

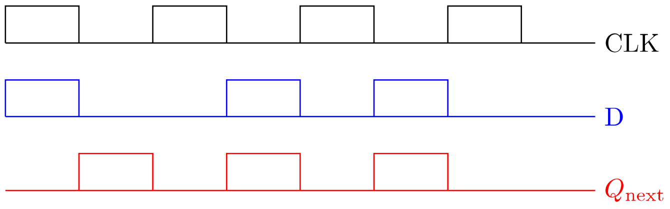


Figure 2: Positive-edge Triggered D Flip-Flop Timing Diagram

Example

- **Try this**: In a positive-edge triggered D flip-flop, if D switches from 0 to 1 at the rising clock edge, what will Q be?
- **Solution**: $Q = 1$, matching D at the clock edge.

10 Applications of Flip-Flops

Flip-flops are essential in various digital applications:

- **Registers**: Collections of flip-flops store multiple bits.
- **Counters**: Use flip-flops to count pulses and create frequency dividers.
- **Memory Elements**: Form basic building blocks of memory storage.