

Study of Various Flip-Flop Circuits

Objective:

To construct and study the operations of the following circuits:

- (i) RS and Clocked RS Flip-Flop
- (ii) D Flip-Flop
- (iii) JK and Master-Slave JK Flip-Flop
- (iv) T Flip-Flop

Overview:

So far you have encountered with *combinatorial logic*, i.e. circuits for which the output depends only on the inputs. In many instances it is desirable to have the next output depending on the current output. A simple example is a *counter*, where the next number to be output is determined by the current number stored. Circuits that remember their current output or state are often called *sequential logic* circuits. Clearly, sequential logic requires the ability to store the current state. In other words, *memory* is required by sequential logic circuits, which can be created with boolean gates. If you arrange the gates correctly, they will remember an input value. This simple concept is the basis of RAM (random access memory) in computers, and also makes it possible to create a wide variety of other useful circuits.

Memory relies on a concept called **feedback**. That is, the output of a gate is fed back into the input. The simplest possible feedback circuit using two inverters is shown below (Fig.1):

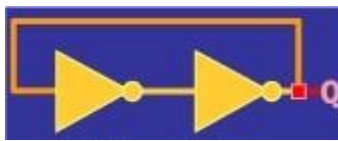


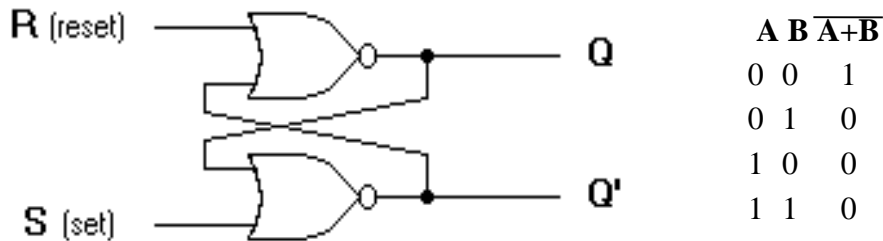
Fig.1: Simplest realization of feedback circuit

If you follow the feedback path, you can see that if Q happens to be 1 (or 0), it will always be 1 (or 0). Since it's nice to be able to control the circuits we create, this one doesn't have much use -- but it does let you see how feedback works. It turns out that in "real" sequential circuits, you can actually use this sort of simple inverter feedback approach. The memory elements in these circuits are called *flip-flops*. A flip-flop circuit has two outputs, one for the normal value and one for the complement value of the stored bit. Binary information can enter a flip-flop in a variety of ways and gives rise to different types of flip-flops.

RS Flip-Flop

RS flip-flop is the simplest possible memory element. It can be constructed from two NAND gates or two NOR gates. Let us understand the operation of the RS flip-flop using NOR gates as shown below using the truth table for 'A NOR B' gate. The inputs R and S are referred to as the Reset and Set inputs, respectively. The outputs Q and Q' are complements of each other and are referred to as the normal and complement outputs, respectively. The binary state of the flip-flop is taken to be the value of the normal output. When Q=1 and Q'=0, it is in the *set state* (or 1-state). When Q=0 and Q'=1, it is in the *reset/clear state* (or 0-state).

Circuit Diagram:



- **S=1 and R=0:** The output of the bottom NOR gate is equal to zero, Q'=0. Hence both inputs to the top NOR gate are equal to 0, thus, Q=1. Hence, the input combination S=1 and R=0 leads to the flip-flop being **set** to Q=1.
- **S=0 and R=1:** Similar to the arguments above, the outputs become Q=0 and Q'=1. We say that the flip-flop is **reset**.
- **S=0 and R=0:** Assume the flip-flop was previously in set (S=1 and R=0) condition. Now changing S to 0 results Q' still at 0 and Q=1. Similarly, when the flip-flop was previously in a reset state (S=0 and R=1), the outputs do not change. Therefore, with inputs S=0 and R=0, the flip-flop holds its state.
- **S=1 and R=1:** This condition violates the fact that both outputs are complements of each other since each of them tries to go to 0, which is not a stable configuration. It is impossible to predict which output will go to 1 and which will stay at 0. In normal operation this condition must be avoided by making sure that 1's are not applied to both inputs simultaneously, thus making it one of the main disadvantages of RS flip-flop.

All the above conditions are summarized in the characteristic table below:

Characteristic Table:

R	S	Q	Q'	Comment
0	0	Q	Q'	Hold state
0	1	1	0	Set
1	0	0	1	Reset
1	1	?	?	Indeterminate

Debounce circuit

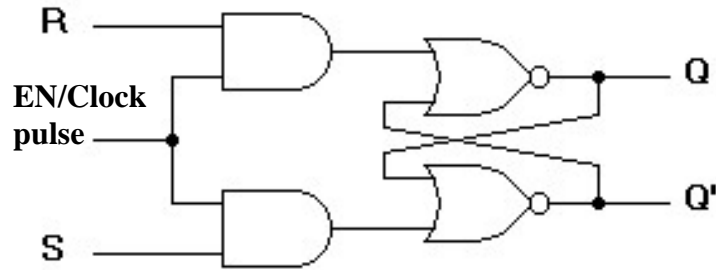
An elementary example using this flip-flop is the debounce circuit. Suppose a piece of electronics is to change state under the action of a mechanical switch. When this switch is moved from position S to R (S=0, R=1), the contacts make and break several times at R before settling to good contact. It is desirable that the electronics should respond to the first contact and then remain stable, rather than switching back and forth as the circuit makes and breaks. This is achieved by RS flip-flop which is reset to Q=0 by the first signal R=1 and remains in a fixed state until the switch is moved back to position S, when the signal S=1 sets the flip-flop to Q=1.

Gated or Clocked RS Flip-Flop

It is sometimes desirable in sequential logic circuits to have a bistable RS flip-flop that only changes state when certain conditions are met regardless of the condition of either the Set or the Reset inputs. By connecting a 2-input AND gate in series with each input terminal of the RS NOR Flip-flop a Gated RS Flip-flop can be created. This extra conditional input is called an "Enable" input and is given the prefix of "EN" as shown below. When the Enable input "EN" = 0, the outputs of the two AND gates are also at logic level 0, (AND Gate principles) regardless of the condition of the two inputs S and R, latching the two outputs Q and Q' into their last known state. When the enable input "EN" = 1, the circuit responds as a normal RS bistable flip-flop with the two AND gates becoming transparent to the Set and Reset signals. This Enable input can also be connected to a clock timing signal adding clock synchronisation to the flip-flop creating what is sometimes called a "Clocked SR Flip-flop".

So a **Gated/Clocked RS Flip-flop** operates as a standard bistable latch but the outputs are only activated when a logic "1" is applied to its EN input and deactivated by a logic "0". The property of this flip-flop is summarized in its characteristic table where Q_n is the logic state of the previous output and Q_{n+1} is that of the next output and the clock input being at logic 1 for all the R and S input combinations.

Circuit Diagram:



Characteristic Table:

Q_n	R	S	Q_{n+1}
0	0	0	0 (Hold)
0	1	0	0
0	0	1	1
0	1	1	Indeterminate
1	0	0	1 (Hold)
1	1	0	0
1	0	1	1
1	1	1	Indeterminate

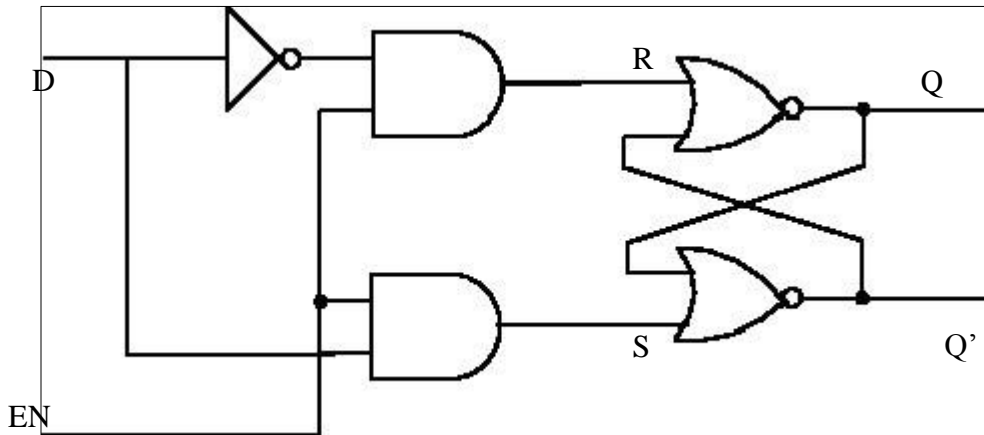
D FLIP-FLOP

An RS flip-flop is rarely used in actual sequential logic because of its undefined outputs for inputs $R = S = 1$. It can be modified to form a more useful circuit called D flip-flop, where D stands for data. The D flip-flop has only a single data input D as shown in the circuit diagram. That data input is connected to the S input of an RS flip-flop, while the inverse of D is connected to the R input. To allow the flip-flop to be in a holding state, a D-flip flop has a second input called Enable, EN. The Enable- input is AND-ed with the D-input.

- When **EN=0**, irrespective of D-input, the $R = S = 0$ and the **state is held**.
- When **EN= 1**, the S input of the RS flip-flop equals the D input and R is the inverse of D. Hence, output **Q follows D**, when $EN = 1$.
- When **EN returns to 0**, the most recent input **D is 'remembered'**.

The circuit operation is summarized in the characteristic table for **EN=1**.

Circuit Diagram:



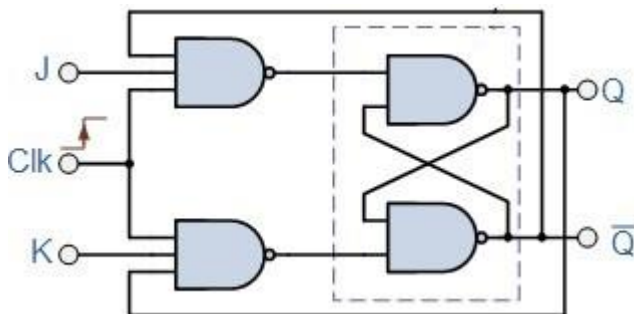
Characteristic Table:

Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

JK FLIP-FLOP:

The JK flip flop (JK means Jack Kilby, a Texas instrument engineer, who invented it) is the most versatile flip-flop, and the most commonly used flip flop. Like the RS flip-flop, it has two data inputs, J and K, and an EN/clock pulse input (CP). Note that in the following circuit diagram NAND gates are used instead of NOR gates. It has no undefined states, however. The fundamental difference of this device is the feedback paths to the AND gates of the input, i.e. Q is AND-ed with K and CP and Q' with J and CP.

Circuit Diagram:



Characteristic Table:

Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1(Toggle, \bar{Q}_n)
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0(Toggle, \bar{Q}_n)

The JK flip-flop has the following characteristics:

- If one input (J or K) is at logic 0, and the other is at logic 1, then the output is set or reset (by J and K respectively), just like the RS flip-flop.
- If both inputs are 0, then it remains in the same state as it was before the clock pulse occurred; again like the RS flip flop. CP has no effect on the output.
- If both inputs are high, however the flip-flop changes state whenever a clock pulse occurs; i.e., the clock pulse toggles the flip-flop again and again until the CP goes back to 0 as shown in the shaded rows of the characteristic table above. Since this condition is undesirable, it should be eliminated by an improvised form of this flip-flop as discussed in the next section.

Circuit components/Equipments:

1. Resistors (1K Ω , 5 Nos)
2. ICs [NOR-7402, AND(2-input)-7408, NAND(3-input)-7410, NAND-7400, NOT-7404]
3. A Surface mount dip switch
4. D.C. Power supply (5V)
5. Red/Green LEDs (4 Nos)
6. Connecting wires
7. Breadboard

Circuit Diagrams:

Already provided with text.

Procedure:

1. Assemble the circuits one after another on your breadboard as per the circuit diagrams. Circuit diagrams given here do not show connections to power supply and LEDs assuming that you are already familiar with it from your previous lab experience.
2. Connect the ICs properly to power supply (pin 14) and ground (pin 7) following the schematics for ICs given above.
3. Using dip switch and resistors, facilitate all possible combinations of inputs from the power supply. Use the switch also to facilitate pulse input to the circuit.
4. Turn on power to your experimental circuit.
5. For each input combination, note the logic state of the normal and complementary outputs as indicated by the LEDs (ON = 1; OFF = 0), and record the results in a table.
6. Compare your results with the characteristic tables.
7. When you are done, turn off the power to your experimental circuit.

Observations:

Table For RS FF: _____

Table For Gated RS FF: _____

Table For D FF: _____

Table For JK FF: _____

Discussions:

Precautions:

1. Watch out for loose connections.
 2. While changing the input condition keep the dip switch well pressed.
-