Introduction to Sequential Circuits

Topics Covered from Module 5: *SR and JK Flip-Flops, Shift Register, 3-bit Ripple Counter.*

Introduction

Digital circuits are classified into two types – Combinational circuits and Sequential circuits. Combinational circuits are the circuits in which the output depends on the present inputs only. Sequential circuits are the circuits in which the output depends on the present input as well as the previous state of the system. The sequential circuits have memory. Table 1 lists the differences between combinational and sequential circuits.

Table 1 Differences between combinational and sequential circuits

Combinational Circuits	Sequential Circuits
Output depends on the present inputs only	Output depends on present inputs as well as past outputs
Do not have the capability to store data	Capable of storing data
Do not contain memory elements	Contain memory elements
Do not require any feedback	Require feedback from output to input
Independent of clock	Generally use a clock for triggering
e.g.: Adder, Multiplexer, Decoder, etc.	e.g.: Shift register, Counter, etc.

The model of a sequential circuit is shown in Fig. 1. It consists of a combinational circuit to which memory elements are added to form a feedback path. The memory elements are devices capable of storing binary information within them. The memory elements which are used in sequential circuits are called flip-flops.

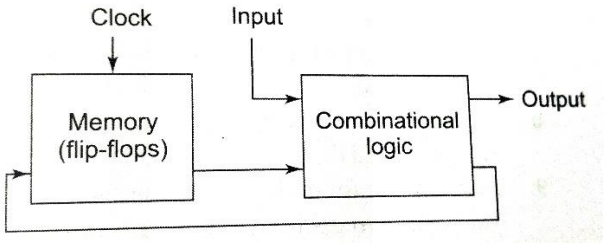


Fig. 1 Model of a sequential circuit

Flip-Flop

A flip-flop is an electronic circuit which has memory. It is a bistable digital circuit, i.e., its outputs have two stable states: logic 1 and logic 0. It is the basic element of all sequential systems.

Difference between Latches and Flip-Flops

Latches and flip-flops are the basic building blocks of the most sequential circuits. The main difference between latches and flip-flops is the method used for changing their state. Latches are controlled by an enable signal and they are level triggered, whereas, flip-flops are controlled by a clock signal and they are edge triggered.

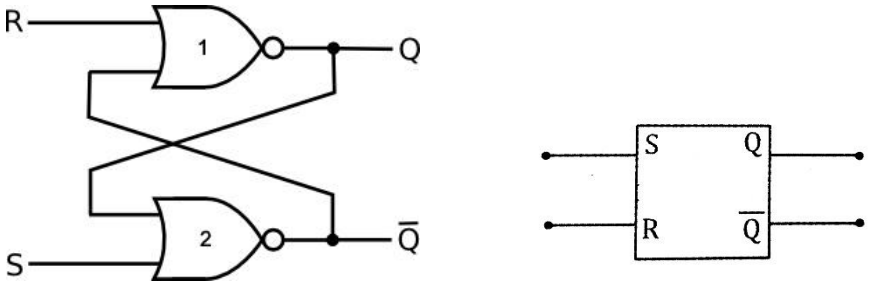
Table 2 Differences between latches and flip-flops

Latches	Flip-Flops
Latches are controlled by an *enable* signal	Flip-flops are controlled by a *clock* signal
Latches are positive or negative level-triggered, i.e., the output changes whenever the enable is 1 (for positive level-triggered) or 0 (for negative level-triggered)	Flip-flops are positive or negative edge-triggered, i.e., the output changes whenever the clock changes from 0 to 1 (for positive edge-triggered) or 1 to 0 (for negative edge-triggered)
Latches are building blocks of sequential circuits and are built from basic gates	Flip-flops are also building blocks of sequential circuits and are built from latches
A latch continuously checks its inputs and changes its output whenever the enable is HIGH	A flip-flop continuously checks its inputs and changes its output only at times determined by the clock signal
The operation of a latch is faster as they do not have to wait for clock signal	Flip-flops are comparatively slower as they have to wait for clock signal

Table 2 summarizes the differences between latches and flip-flops.

SR Flip-Flop

An SR Flip-Flop (also called RS Flip-Flop) is formed by cross connecting two NOR gates as shown in Fig. 2.



(a) SR flip-flop (b) Logic symbol

Fig. 2 SR flip-flop

Operation

The inputs are S and R. Hence there are four conditions:

i. If $S = R = 0$

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (0,1). So its output is $Q = 0$. The inputs to the gate 2 are (0,0). So its output is $\bar{Q} = 1$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,0). So its output is $Q = 1$. The inputs to the gate 2 are (1,0). So its output is $\bar{Q} = 0$.

*That is, if $S = R = 0$, there is **no change** in the output.*

ii. If $S = 0, R = 1$

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,1). So its output is $Q = 0$. The inputs to the gate 2 are (0,0). So its output is $\bar{Q} = 1$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (1,0). So its output is $Q = 0$. The inputs to the gate 2 are (0,0). So its output is $\bar{Q} = 1$.

*That is, if $S = 0, R = 1$, the output is **reset** ($Q = 0$).*

iii. If $S = 1, R = 0$

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 2 are (0,1). So its output is $\bar{Q} = 0$. The inputs to the gate 1 are (0,0). So its output is $Q = 1$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,0). So its output is $Q = 1$. The inputs to the gate 2 are (1,1). So its output is $\bar{Q} = 0$.

*That is, if $S = 1, R = 0$, the output is **set** ($Q = 1$).*

iv. If $S = R = 1$

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,1). So its output is $Q = 0$. The inputs to the gate 2 are (0,1). So its output is $\bar{Q} = 0$.

Here $Q = 0$ and $\bar{Q} = 0$, which is not possible. So the output is not valid.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (1,0). So its output is $Q = 0$. The inputs to the gate 2 are (0,1). So its output is $\bar{Q} = 0$.

Again $Q = 0$ and $\bar{Q} = 0$, which is not possible. So the output is not valid.

*That is, if $S = R = 1$, the output is not valid. Hence this is **forbidden**.*

Considering all the four cases, the truth table is written as below.

Truth Table:

S	R	Q_n	Q_{n+1}	State
0 0	0 0	0 1	0 1	No change
0 0	1 1	0 1	0 0	Reset
1 1	0 0	0 1	1 1	Set
1 1	1 1	0 1	? ?	Forbidden (Illegal)

S	R	Q_{n+1}	State
0	0	Q_n	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Forbidden (Illegal)

Q_n - Present state output

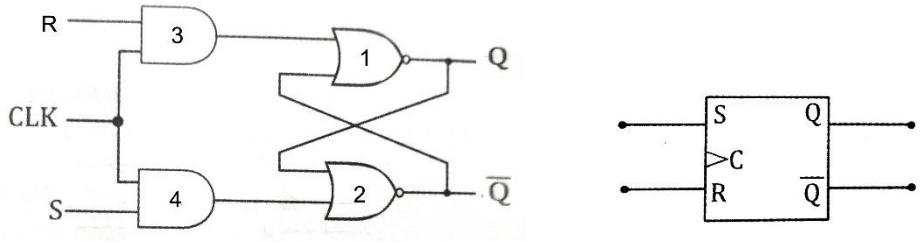
Q_{n+1} - Next state output

Clocked SR Flip-Flops

Within a digital computer, a clock is used to synchronize changes in the contents of memory elements. A clock is a signal which oscillates between 0 and 1 as shown in Fig. 3.

Fig. 3 Clock signal

In a clocked SR flip-flop, a clock (CLK) signal is fed to the AND gates 3 and 4 as shown in Fig. 4.



(a) Clocked SR flip-flop (b) Logic Symbol

Fig. 4 Clocked SR flip-flop

Operation

When the clock is HIGH (1), the outputs of gates 3 and 4 are R and S respectively. So the flip-flop operates normally.

When the clock is LOW (0), the output of the gates 3 and 4 are (0,0), which implies that the output does not change and the flip-flop is said to be disabled.

The operation can be analyzed for the following conditions:

i. If $CLK = 1$ and $S = R = 0$

In this case, the outputs of gates 3 and 4 are (0,0) respectively.

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (0,1). So its output is $Q = 0$. The inputs to the gate 2 are (0,0). So its output is $\bar{Q} = 1$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,0). So its output is $Q = 1$. The inputs to the gate 2 are (1,0). So its output is $\bar{Q} = 0$.

That is, if $CLK = 1$ and $S = R = 0$, *there is* ***no change*** *in the output.*

ii. If $CLK = 1$ and $S = 0, R = 1$

In this case, the outputs of gates 3 and 4 are (0,1) respectively.

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,1). So its output is $Q = 0$. The inputs to the gate 2 are (0,0). So its output is $\bar{Q} = 1$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (1,0). So its output is $Q = 0$. The inputs to the gate 2 are (0,0). So its output is $\bar{Q} = 1$.

That is, if $CLK = 1$ and $S = 0, R = 1$, *the output is* ***reset*** *(*$Q = 0$*).*

iii. If $CLK = 1$ and $S = 1$, R $= 0$

In this case, the outputs of gates 3 and 4 are (1,0) respectively.

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 2 are (0,1). So its output is $\bar{Q} = 0$. The inputs to the gate 1 are (0,0). So its output is $Q = 1$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,0). So its output is $Q = 1$. The inputs to the gate 2 are (1,1). So its output is $\bar{Q} = 0$.

That is, if $CLK = 1$ and $S = 1, R = 0$*, the output is* ***set*** *(*$Q = 1$*).*

iv. If $CLK = 1$ and $S = R = 1$

In this case, the outputs of gates 3 and 4 are (1,1) respectively.

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,1). So its output is $Q = 0$. The inputs to the gate 2 are (0,1). So its output is $\bar{Q} = 0$.

Here $Q = 0$ and $\bar{Q} = 0$, which is not possible. So the output is not valid.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (1,0). So its output is $Q = 0$. The inputs to the gate 2 are (0,1). So its output is $\bar{Q} = 0$.

Again $Q = 0$ and $\bar{Q} = 0$, which is not possible. So the output is not valid.

That is, if $CLK = 1$ and $S = R = 1$*, the output is not valid. Hence this is* ***forbidden****.*

v. If $CLK = 0$

In this case, the output of the gates 3 and 4 are (0,0), which implies that the output does not change and the flip-flop is said to be disabled.

Considering all the cases, the truth table is written as below.

Truth Table:

CLK	S	R	Q_n	Q_{n+1}	State
1 1	0 0	0 0	0 1	0 1	No change
1 1	0 0	1 1	0 1	0 0	Reset
1 1	1 1	0 0	0 1	1 1	Set
1 1	1 1	1 1	0 1	? ?	Forbidden (Illegal)
0 0	X X	X X	0 1	0 1	No change

CLK	S	R	Q_{n+1}	State
1	0	0	Q_n	No change
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	?	Forbidden (Illegal)
0	X	X	Q_n	No change

Q_n - Present state output

Q_{n+1} - Next state output

X – Don't care

The clocked flip-flop discussed above is a *level-triggered* flip-flop, which means that the value of the output changes depending on the level of the clock. As long as the clock is HIGH, the change in the output can happen whenever the input changes.

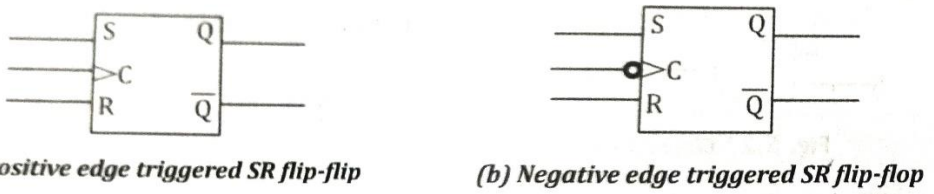
Edge-Triggered Flip-Flops

In *edge-triggered* flip-flops, the output changes only when the clock makes a transition from one level to another. The flip-flops can be positive edge-triggered or negative edge-triggered.

i. In positive edge-triggered flip-flops, the output responds to the *S* and *R* inputs only at the positive edges of the clock pulse. At other instants of time, the output does not change.

ii. In negative edge-triggered flip-flops, the output responds to the S and R inputs only at the negative edges of the clock pulse. At other instants of time, the output does not change.

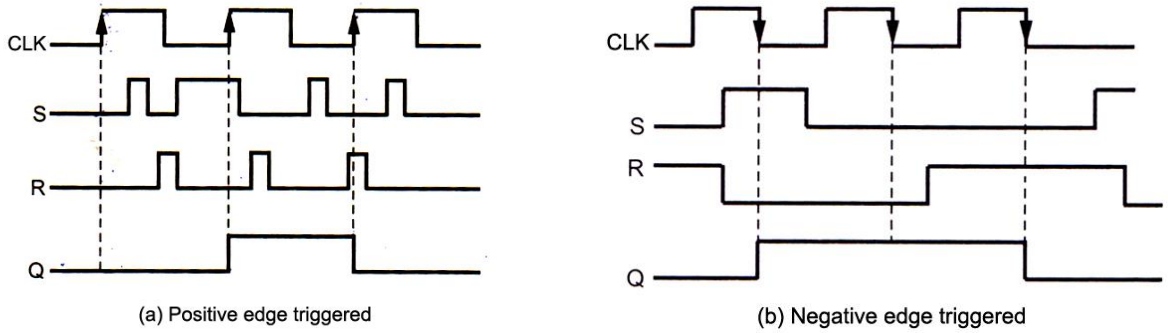
Fig. 5 shows the logic symbols of edge-triggered flip-flops.



(a) Positive edge triggered SR flip-flip *(b) Negative edge triggered SR flip-flop*

Fig. 5 Logic symbols of edge-triggered flip-flops

Fig. 6 shows the input and output waveforms for positive and negative edge-triggered SR flip-flops.



(a) Positive edge triggered (b) Negative edge triggered

Fig. 6 Input and output waveforms for positive and negative edge-triggered RS flip-flops

Pulse Generator

A positive edge-triggered flip-flop requires a narrow positive spike to trigger it. This is generated using a small circuit called pulse generator, as shown in Fig. 7

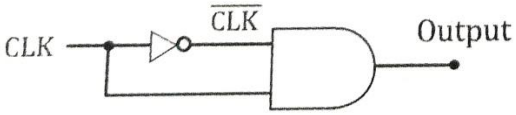


Fig. 7 Pulse generator

The inverter produces a delay of few nanoseconds. The AND gate produces an output which is a narrow positive spike that is HIGH only for a few nanoseconds, when CLK and $\overline{CLK}$ are both HIGH. The waveform of narrow positive spike at the positive edge of the clock pulse is shown in Fig. 8

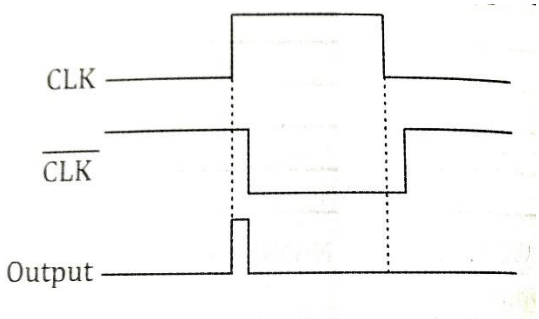
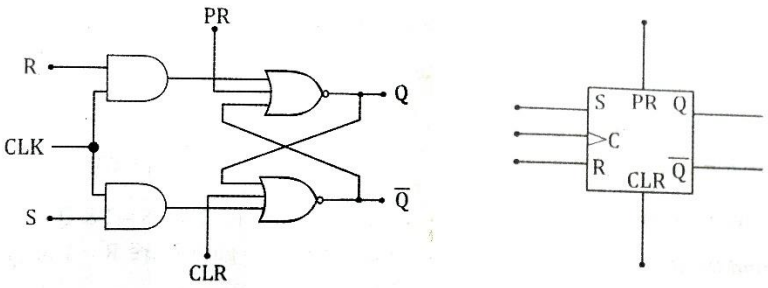


Fig. 8 Positive spike generation for positive edge-triggering

Clear and Preset

The clear (CLR) and preset (PR) inputs may be used, as shown in Fig. 9, to either clear or preset the flip-flop, independent of the S and R inputs.



(a) Logic diagram (b) Logic symbol

Fig. 9 Clocked SR flip-flop with CLR and PR

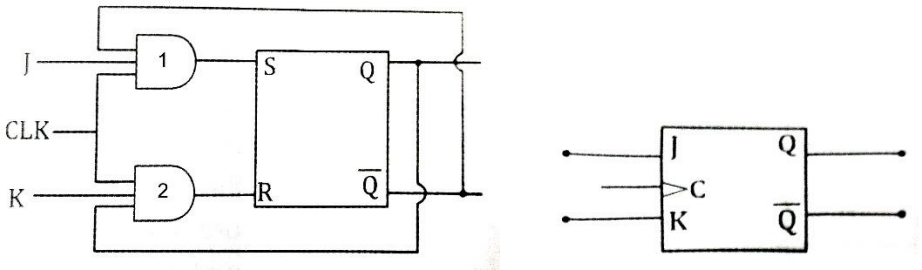
When CLR and PR are 0, the circuit behaves as a normal SR flip-flop. Making $CLR = 1$ and $PR = 0$, results in $Q = 0$ and $\bar{Q} = 1$, thereby clearing the flip-flop. Making $PR = 1$ and $CLR = 0$, results in $Q = 1$ and $\bar{Q} = 0$, thereby presetting the flip-flop.

These are called asynchronous inputs as their action does not need a clock pulse. The truth table of an SR flip-flop with CLR and PR inputs is given below.

PR	***CLR***	**Response**
0	0	Behaves as SR FF
0	1	$Q = 0$ (clear)
1	0	$Q = 1$ (preset)
1	1	Not used

JK Flip-Flop

An SR flip-flop can be converted to JK flip-flop by using two AND gates at the input. The operation of a JK flip-flop is identical to that of an SR flip-flop, except that it has no forbidden state. The circuit diagram and logic symbol of a JK flip-flop is as shown in Fig. 10.



(a) Circuit diagram (b) Logic symbol

Fig. 10 JK flip-flop

Operation:

In the presence of clock signal, the operation can be analyzed for the following conditions:

i. **If $J = K = 0$**

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,0,1). So its output is 0. The inputs to the gate 2 are (1,0,0). So its output is 0.

Now $S = R = 0$. Hence output $Q = 0$ and $\bar{Q} = 1$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,0,1). So its output is 0. The inputs to the gate 2 are (1,0,1). So its output is 0.

Now $S = R = 0$. Hence output $Q = 1$ and $\bar{Q} = 0$.

*That is, if $J = K = 0$, there is **no change** in the output.*

ii. **If $J = 0, K = 1$**

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,0,1). So its output is 0. The inputs to the gate 2 are (1,1,0). So its output is 0.

Now $S = R = 0$. Hence output $Q = 0$ and $\bar{Q} = 1$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,0,1). So its output is 0. The inputs to the gate 2 are (1,1,1). So its output is 1.

Now $S = 0, R = 1$. Hence output $Q = 0$ and $\bar{Q} = 1$.

*That is, if $J = 0, K = 1$, the output is **reset** ($Q = 0$).*

iii. **If $J = 1, \mathrm{K} = 0$**

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,1,1). So its output is 1. The inputs to the gate 2 are (1,0,0). So its output is 0.

Now $S = 1, R = 0$. Hence output $Q = 1$ and $\bar{Q} = 0$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,1,1). So its output is 0. The inputs to the gate 2 are (1,0,1). So its output is 0.

Now $S = R = 0$. Hence output $Q = 1$ and $\bar{Q} = 0$.

*That is, if $J = 1, K = 0$, the output is **set** ($Q = 1$).*

iv. **If $J = K = 1$**

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,1,1). So its output is 1. The inputs to the gate 2 are (1,1,0). So its output is 0.

Now $S = 1, R = 0$. Hence output $Q = 1$ and $\bar{Q} = 0$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,1,1). So its output is 0. The inputs to the gate 2 are (1,1,1). So its output is 1.

Now $S = 0, R = 1$. Hence output $Q = 0$ and $\bar{Q} = 1$.

*That is, if $J = K = 1$, the output changes from 0 to 1 or 1 to 0. The output is said to **toggle**.*

Considering all the cases, the truth table is written as below.

Truth Table:

J	K	Q_n	Q_{n+1}	State
0 0	0 0	0 1	0 1	No change
0 0	1 1	0 1	0 0	Reset
1 1	0 0	0 1	1 1	Set
1 1	1 1	0 1	1 0	Toggle

J	K	Q_{n+1}	State
0	0	Q_n	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q_n}$	Toggle

In JK flip-flop, when $J = K = 1$, the output toggles from 0 to 1 or 1 to 0. If the clock pulse is too long, the output will change more than once and final state of the flip-flop will be indeterminate. To avoid this problem, two flip-flops are connected in master-slave form.

Master-Slave JK Flip-Flop

A master-slave JK flip-flop is constructed from two flip-flops. One flip-flop acts as the master and the other serves as a slave as shown in Fig. 11.

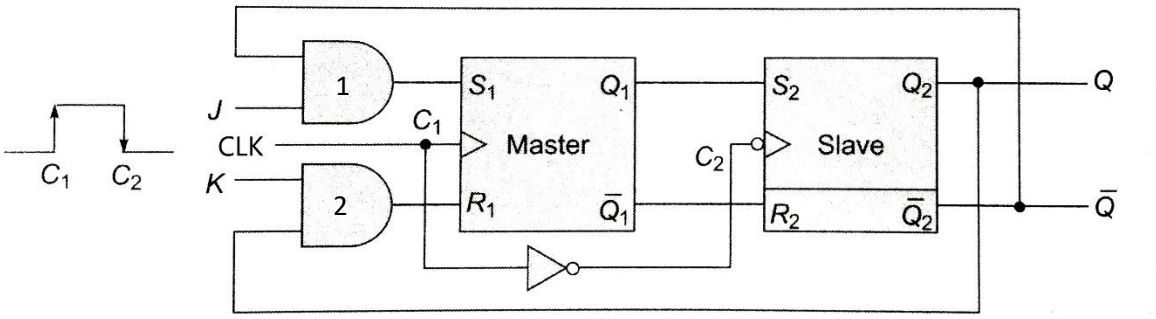


Fig. 11 Master slave JK flip-flop

Operation

S_1 and R_1 are the inputs and Q_1 is the output of the master flip-flop. S_2 and R_2 are the inputs and Q_2 or Q is the output of the slave flip-flop. CLK is the clock pulse to the circuit. $C_1 = CLK$ is the clock to master and $C_2 = \overline{CLK}$ is the clock to the slave.

From the circuit, S_2 is connected to Q_1 and R_2 is connected to $\bar{Q}_1$. So $S_2 = Q_1$ and $R_2 = \bar{Q}_1$. This means that the input to the slave is always (0,1) or (1,0).

As the clock pulse CLK goes low, $C_2 = 1$ and the slave translates the output of the master to the system output $Q = Q_2 = Q_1$.

As CLK goes high, $C_1 = 1$ and the master produces the output Q_1 as per the JK action.

Consider the different cases:

i. **If $J = 0, K = 0$**

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,0). So its output is 0. The inputs to the gate 2 are (0,0). So its output is 0.

Now $S_1 = 0, R_1 = 0$. Hence when $CLK = 1, C_1 = 1$ and the master produces output $Q_1 = 0$ and $\bar{Q}_1 = 1$.

Now $S_2 = 0, R_2 = 1$. Hence when $CLK = 0$, $C_2 = 1$ and the slave produces output $Q_2 = Q = 0$ and $\bar{Q}_2 = \bar{Q} = 1$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,0). So its output is 0. The inputs to the gate 2 are (0,1). So its output is 0.

Now $S_1 = 0, R_1 = 0$. Hence when $CLK = 1, C_1 = 1$ and the master produces output $Q_1 = 1$ and $\bar{Q}_1 = 0$.

Now $S_2 = 1, R_2 = 0$. Hence when $CLK = 0$, $C_2 = 1$ and the slave produces output $Q_2 = Q = 1$ and $\bar{Q}_2 = \bar{Q} = 0$.

*That is, if $J = K = 0$, there is **no change** in the output.*

ii. If $J = 0, K = 1$

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,0). So its output is 0. The inputs to the gate 2 are (1,0). So its output is 0.

Now $S_1 = R_1 = 0$. Hence when $CLK = 1$, $C_1 = 1$ and the master produces output $Q_1 = 0$ and $\bar{Q}_1 = 1$.

Now $S_2 = 0, R_2 = 1$. Hence when $CLK = 0$, $C_2 = 1$ and the slave produces output $Q_2 = Q = 0$ and $\bar{Q}_2 = \bar{Q} = 1$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,0). So its output is 0. The inputs to the gate 2 are (1,1). So its output is 1.

Now $S_1 = 0, R_1 = 1$. Hence when $CLK = 1, C_1 = 1$ and the master produces output $Q_1 = 0$ and $\bar{Q}_1 = 1$.

Now $S_2 = 0, R_2 = 1$. Hence when $CLK = 0$, $C_2 = 1$ and the slave produces output $Q_2 = Q = 0$ and $\bar{Q}_2 = \bar{Q} = 1$.

*That is, if $J = 0, K = 1$, the output is **reset** ($Q = 0$).*

iii. If $J = 1, K = 0$

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,1). So its output is 1. The inputs to the gate 2 are (0,0). So its output is 0.

Now $S_1 = 1, R_1 = 0$. Hence when $CLK = 1, C_1 = 1$ and the master produces output $Q_1 = 1$ and $\bar{Q}_1 = 0$.

Now $S_2 = 1, R_2 = 0$. Hence when $CLK = 0$, $C_2 = 1$ and the slave produces output $Q_2 = Q = 1$ and $\bar{Q}_2 = \bar{Q} = 0$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,1). So its output is 0. The inputs to the gate 2 are (0,1). So its output is 0.

Now $S_1 = R_1 = 0$. Hence when $CLK = 1$, $C_1 = 1$ and the master produces output $Q_1 = 1$ and $\bar{Q}_1 = 0$.

Now $S_2 = 1, R_2 = 0$. Hence when $CLK = 0$, $C_2 = 1$ and the slave produces output $Q_2 = Q = 1$ and $\bar{Q}_2 = \bar{Q} = 0$.

*That is, if $J = 1, K = 0$, the output is **set** ($Q = 1$).*

iv. If $J = K = 1$

Let the previous state output be $Q = 0$ ($\bar{Q} = 1$). The inputs to the gate 1 are (1,1). So its output is 1. The inputs to the gate 2 are (1,0). So its output is 0.

Now $S_1 = 1, R_1 = 0$. Hence when $CLK = 1$, $C_1 = 1$ and the master produces output $Q_1 = 1$ and $\bar{Q}_1 = 0$.

Now $S_2 = 1, R_2 = 0$. Hence when $CLK = 0$, $C_2 = 1$ and the slave produces output $Q_2 = Q = 1$ and $\bar{Q}_2 = \bar{Q} = 0$.

Let the previous state output be $Q = 1$ ($\bar{Q} = 0$). The inputs to the gate 1 are (0,1). So its output is 0. The inputs to the gate 2 are (1,1). So its output is 1.

Now$S_1 = 0, R_1 = 1$. Hence when $CLK = 1$, $C_1 = 1$ and the master produces output $Q_1 = 0$ and $\bar{Q}_1 = 1$.

Now $S_2 = 0, R_2 = 1$. Hence when $CLK = 0$, $C_2 = 1$ and the slave produces output $Q_2 = Q = 0$ and $\bar{Q}_2 = \bar{Q} = 1$.

*That is, if $J = K = 1$, the output changes from 0 to 1 or 1 to 0. The output is said to **toggle**.*

Considering all the cases, the truth table is written as below.

Truth Table:

J	K	Q_n	Q_{n+1}	State
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

J	K	Q_{n+1}	State
0	0	Q_n	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q_n}$	Toggle

In Master Slave JK flip-flop, when $J = K = 1$, the output toggles from 0 to 1 or 1 to 0. The master flip-flop is positive edge triggered whereas the slave flip-flop is negative edge triggered. Hence, when the master is active, the slave is inactive and vice versa. Thus, the unwanted toggling is avoided.

D Flip-Flop

A D flip-flop is obtained from a JK flip-flop by connecting J to K through a NOT gate as shown in Fig. 12.

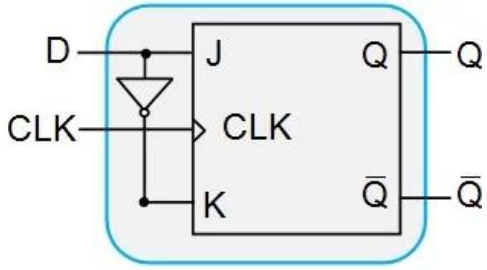


Fig. 12 D flip-flop

When $D = 0$, $(J, K) = (0,1)$ and hence $Q = 0$. When $D = 1$, $(J, K) = (1,0)$ and hence $Q = 1$. The truth table is written as below.

Truth Table:

D	Q_n	Q_{n+1}	State
0 0	0 1	0 0	Reset
1 1	0 1	1 1	Set

D	Q_{n+1}	State
0	0	Reset
1	1	Set

T Flip-Flop

A T flip-flop is obtained from a JK flip-flop by connecting J to K as shown in Fig. 13.

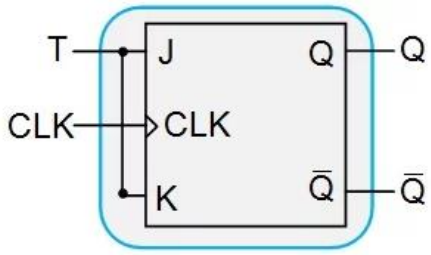


Fig. 13 T flip-flop

When $T = 0$, $(J, K) = (0,0)$ and hence Q retains its previous state. When $T = 1$, $(J, K) = (1,1)$ and hence Q toggles. The truth table is written as below.

Truth Table:

T	Q_n	Q_{n+1}	State
0 0	0 1	0 1	No change
1 1	0 1	1 0	Toggle

T	Q_{n+1}	State
0	Q_n	No change
1	$\overline{Q_n}$	Toggle

Shift Register

A flip-flop can store one bit of data. When more bits of data are to be stored, a number of flip-flops are used. A *register* is a set of flip-flops used to store binary data. Registers are used in CPU (Central Processing Unit) to store data.

A register capable of shifting its stored binary information either to the right or left is called a *shift register*. It consists of a set of flip-flops connected in cascade with the output of one flip-flop connected to the input of the next one.

A 4-bit shift register employing D flip-flops is shown in Fig. 13. The bubble at the clock input indicates that each flip-flop is trailing (negative) edge triggered.

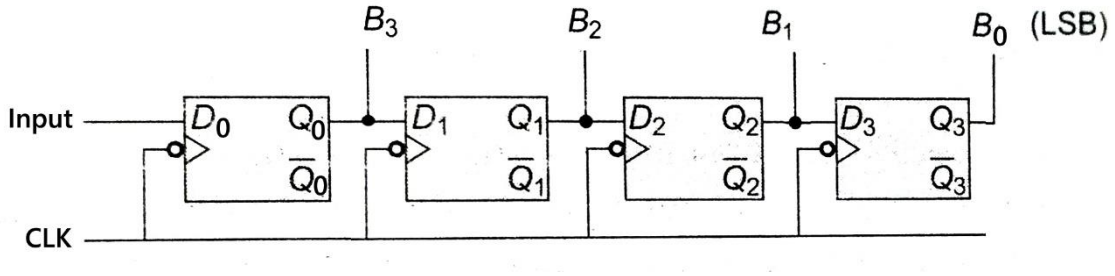


Fig. 13 4-bit shift register

Let the input sequence be A_0, A_1, A_2, A_3 and the initial state of the register be $Q_0 = Q_1 = Q_2 = Q_3 = 0$. Before the first pulse arrives, the four data bits are $D_0 = A_0$, $D_1 = Q_0 = 0$, $D_2 = Q_1 = 0$ and $D_3 = Q_2 = 0$. After the first pulse arrives, the data in all the flip-flops shift to output Qs and the state of the register becomes A_0000. After the second pulse, the state of the register becomes A_1A_000 and finally at the end of the fourth pulse, the state becomes $A_3A_2A_1A_0$. This register state can be read simultaneously at $B_3B_2B_1B_0$. The register in this mode is called *serial-in, parallel-out* shift register.

By applying four more clock pulses, the output can be read serially at B_0. The register is then called *serial-in, serial-out* shift register.

The sequence of the states of the shift register is as shown in the table below.

Truth Table:

CLK	B_3	B_2	B_1	B_0
Initial	0	0	0	0
↓	A_0	0	0	0
↓	A_1	A_0	0	0
↓	A_2	A_1	A_0	0
↓	A_3	A_2	A_1	A_0
↓	X	A_3	A_2	A_1
↓	X	X	A_3	A_2
↓	X	X	X	A_3

→ *serial-in, parallel-out* simultaneously at all outputs

→ *serial-in, serial-out* at B_0

Counter

A counter is a sequential circuit that counts the number of input pulses. A counter that counts in terms of binary is called a ***binary counter***. The count output of an n-bit binary counter

is 2^n states and it can count from 0 to $2^n - 1$. The number of states of a counter is referred to as its modulus (m). For an n-bit counter, $m \leq 2^n$.

Depending on the manner in which flip-flops are triggered to count, the counter are classified into two types:

- Asynchronous counters
- Synchronous counters

In asynchronous counters, flip-flops are clocked sequentially whereas in synchronous counters, the flip-flops are clocked simultaneously. The differences between asynchronous and synchronous counters are listed in Table 3.

Table 3 Differences between asynchronous and synchronous counters

Asynchronous Counters	Synchronous Counters
The flip-flops are clocked sequentially	The flip-flops are clocked simultaneously
The output of one flip-flop drives the clock of the next flip-flop	All the flip-flops are driven by the same clock
Slower in operation	Faster in operation

Asynchronous Counters (Ripple Counters)

Asynchronous counters are also called ripple counters. Here, the output of one flip-flop is connected to the clock of the next flip-flop.

3-bit Ripple Counter

A 3-bit ripple counter can be implemented using three T flip-flops as shown in Fig. 14.

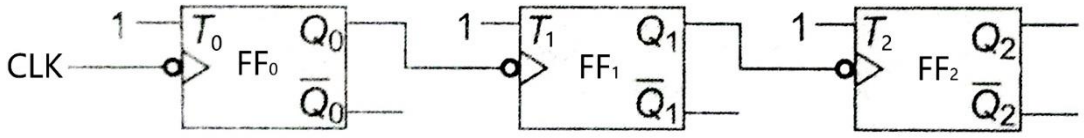


Fig. 14 3-bit ripple counter

The flip-flops are negative edge triggered. All the T inputs are kept HIGH (1) for toggling state. The CLK is given to the first T flip-flop and the other two receive clock pulses from the outputs of the preceding flip-flops. The clock pulse ripples through the flip-flops and hence the name ripple counter.

The counter is initially reset to $Q_0Q_1Q_2 = 000$. When the first clock pulse is applied, at the negative edge of the clock, FF_0 toggles and hence Q_0 goes from LOW to HIGH. This becomes a positive edge at the input of FF_1 and hence it is not affected. The state of the counter after first clock pulse will be $Q_0Q_1Q_2 = 100$.

When the second clock pulse is applied, at the negative edge of the clock, FF_0 toggles and hence Q_0 goes from HIGH to LOW. This becomes a negative edge at the input of FF_1 and hence it toggles from LOW to HIGH. This becomes a positive edge at the input of FF_2 and hence it is not affected. The state of the counter after first clock pulse will be $Q_0Q_1Q_2 = 010$.

This process continues till the state of the counter becomes $Q_0Q_1Q_2 = 111$ at the end of seven clock pulses. At the eighth clock pulse, the counter resets to 000. Thus, the counter attains eight states from 000 to 111 and hence it is called a ***mod-8 counter***.

The truth table of a 3-bit ripple counter is as given below.

Truth Table:

CLK	Q_2	Q_1	Q_0	Input Pulse Count
Initial	0	0	0	0
↓	0	0	1	1
↓	0	1	0	2
↓	0	1	1	3
↓	1	0	0	4
↓	1	0	1	5
↓	1	1	0	6
↓	1	1	1	7
↓	0	0	0	8

The timing diagram of a 3-bit ripple counter is as shown in Fig. 15.

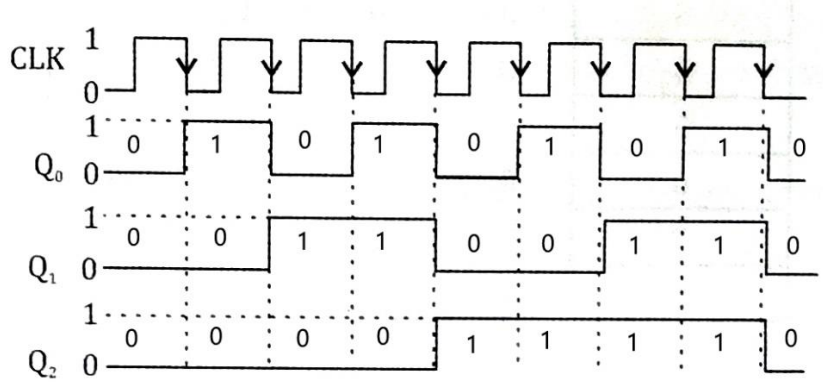


Fig. 15 Timing diagram of 3-bit ripple counter

Modulo-5 Counter

A mod-5 counter can be implemented using the circuit of a 3-bit counter as shown in Fig. 16.

As $2^3 = 8 > 5$, 3 T flip-flops are used. In a mod-5 counter, when the count reaches 5 (101 in binary), the counter resets to 000. This can be achieved by using a NAND gate. The outputs corresponding to 1's in 5 (101), i.e., Q_0 and Q_2 are connected to the inputs of the NAND gate and the output of the NAND gate is connected to the CLR inputs of all the flip-flops. Thus, when the count reaches 5, the counter resets to 000.

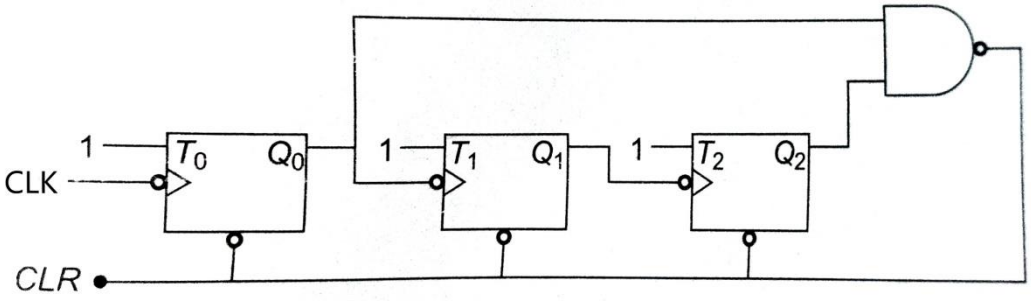


Fig. 16 Modulo-5 counter

Synchronous Counters

In synchronous counters, the clock pulse is applied to all the flip-flops simultaneously. Hence, they are faster than asynchronous counters.

Modulo-16 Synchronous Counter

A mod-16 synchronous counter can be implemented using four T flip-flops. The truth table of a mod-16 counter is as given below.

Truth Table:

CLK	Q_3	Q_2	Q_1	Q_0	T_3	T_2	T_1	T_0
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	1
2	0	0	1	0	0	0	0	1
3	0	0	1	1	0	1	1	1
4	0	1	0	0	0	0	0	1
5	0	1	0	1	0	0	1	1
6	0	1	1	0	0	0	0	1
7	0	1	1	1	1	1	1	1
8	1	0	0	0	0	0	0	1
9	1	0	0	1	0	0	1	1
10	1	0	1	0	0	0	0	1
11	1	0	1	1	0	1	1	1
12	1	1	0	0	0	0	0	1
13	1	1	0	1	0	0	1	1
14	1	1	1	0	0	0	0	1
15	1	1	1	1	1	1	1	1
16	0	0	0	0				

From the truth table, the following observations are made:

Q_0 toggles continuously. Therefore $T_0 = 1$.

Q_1 toggles when $Q_0 = 1$. Therefore $T_1 = Q_0$.

Q_2 toggles when $Q_0 = Q_1 = 1$. Therefore $T_2 = Q_1 Q_0$.

Q_3 toggles when $Q_0 = Q_1 = Q_2 = 1$. Therefore $T_3 = Q_2 Q_1 Q_0$.

The corresponding values of T are indicated in the same table above.

To meet these requirements, the circuit diagram of a mod-16 (4-bit) synchronous counter is as shown in Fig. 17.

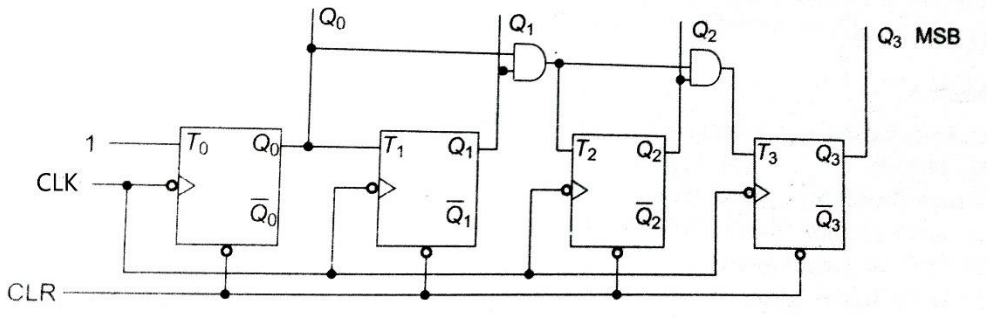


Fig. 17 Module-16 synchronous counter

Questions

1. Explain the differences between combinational and sequential circuits.
2. What is a flip-flop? Distinguish between a latch and a flip-flop.
   (Jul '19 – 2M, Jan '19 – 4M, Jul '18 – 5M, Jan '18 – 4M, Jul '17 – 4M, Jan '17 – 2M, Jul '16, Jan '16, Jul '15 – 4M, MQP '14 – 4M)
3. What is a flip-flop? List out the applications of flip-flop. ***(Sep '20 – 4M, Jan '19 – 4M)***
4. With the help of a logic diagram and truth table, explain the operation of an SR flip-flop.
   (Jan '19 – 6M, Jan '18 – 6M, Jul '17 – 8M, Jul '16 – 5M, Jan '16 – 5M, Jul '15 – 6M, MQP '15 – 6M)
5. With the help of a logic diagram and truth table, explain the working of a clocked SR flip-flop.
   (Sep '20 – 6M, Jan '20 – 6M, Jul '19 – 8M, Jan '19 – 7M, MQP '18 – 6M, Jul '18 – 6M, Jan '18 – 8M, Jul '17 – 8M, Jan '17 – 8M, Jul '16 – 5M, Jan '16 – 8M, Jan '15 – 6M, MQP '15 - 4M, MQP '14 – 5M)
6. Explain the working of a clocked SR flip-flop with a suitable circuit, symbol, truth table and input-output waveforms considering positive edge triggered SR flip-flop.
   (Jul '18 – 6M)
7. With a neat circuit diagram and truth table, explain the working of a JK flip-flop.
   (MQP '18 – 6M)
8. What is a flip-flop? Explain the operation of Master Slave JK flip-flop.
   (Jan '20 – 8M, Jul '19 – 6M, MQP '18 – 5M)
9. What are the differences between level-triggered and edge-triggered flip-flops?

10. What is a shift register? Explain the working of a 4-bit SISO shift register. ***(MQP '18 – 8M)***
11. Draw and explain 4-bit shift register. ***(Feb '21 – 6M)***
12. What is a counter? What are the differences between synchronous and asynchronous counters?
13. What is a counter? With a neat timing and block diagram, explain three-bit (Mod-8) asynchronous (ripple) counter operation. ***(Jan '19 – 8M, MQP '18 – 7M)***
14. With a block diagram, explain the working of a 3-bit ripple(asynchronous) counter. ***(Jan '20 – 6M, Jul '19 – 6M)***
15. With a neat block diagram, explain the operation of four-bit (Mod-16) asynchronous (ripple) counter.
16. With a neat block diagram, explain the operation of Mod-8 synchronous counter.
17. With a neat block diagram, explain the operation of Mod-16 synchronous counter.

References

1. D. P. Kothari, I. J. Nagrath, ***"Basic Electronics"***, McGraw Hill Education (India) Private Limited, Second Edition, 2014.
2. John M. Yarbrough, ***"Digital Logic Applications and Design"***, Thomson Learning, 2001.
3. Donald D. Givone, ***"Digital Principles and Design"***, McGraw Hill, 2002.