DIGITAL ELECTRONICS

Contents

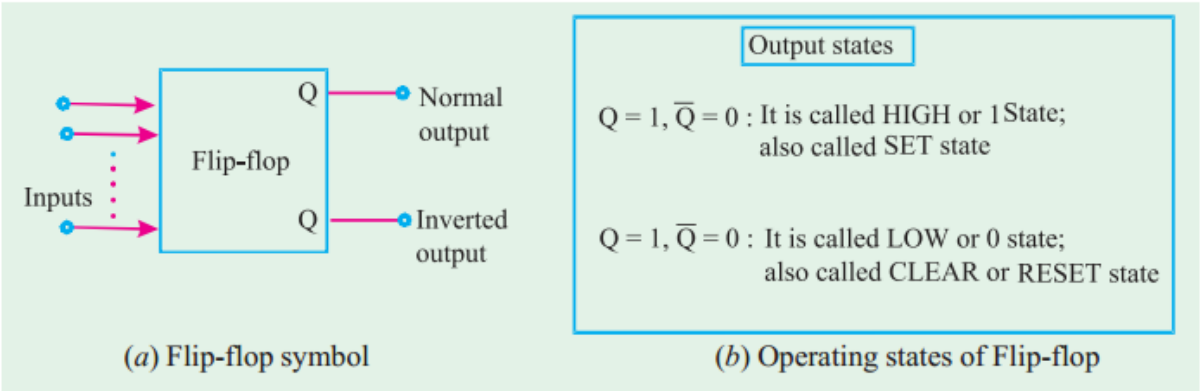
Chapter 1: Flip-flop and related devices

Introduction

The output logic levels of combinatory logic circuits at any instant of time are dependent on the logic levels present at the inputs at that time. Any prior input-logic level conditions have no effect on the present outputs. This is due to the fact that combinational logic circuit circuits have **no memory**. However, it will be interesting to know that most digital systems are made up of both combinational logic circuit and memory elements.

The most important memory element is the Flip-Flop (FF). The FF is made up of an assembly of logic gates. It may be noted that even though a logic gate, by itself, has no storage capability, but several such logic gates can be connected together in ways that permit information to be stored. Several different gate arrangements are used to produce these flip-flops. The flip-flops are used extensively as a **memory cell** in static random access memory (SRAM) of a computer.

A flip-flop is a bistable circuit. Both of its output states are stable. The circuit remains in a particular output state indefinitely until something is done to change that output status. A flip-flop is made up of logic gate and it permits information to be stored in it. Fig 1 shows a general type of symbol used for a flip-flop.



(*a*) Flip-flop symbol (*b*) Operating states of Flip-flop

As seen from this diagram, the flip-flop has two outputs labeled as Q and $\overline{Q}$ that are inverse of each other. The Q output is called the **normal** flip-flop output and $\overline{Q}$ is the **inverted** flip-flop output. It may be carefully noted that whenever we refer to the state of a flip-flop we are referring to the state of its normal Q output. For instance if we say flip-flop is in the HIGH

state, we mean that $Q=1$ on the other hand if we say flip-flop is in LOW state, we $Q=0$. It is automatically understood that the $\overline{Q}$ state will always be inverse of Q, i.e., if $Q=1$, $\overline{Q}=0$ and if $Q=0$, $\overline{Q}=1$

1- R-S flip-flop

The R-S flip-flop is the most basic of all flip-flops. The letters 'R' and 'S' here stand for RESET and SET. When the flip-flop is **SET**, its Q output goes to a '1' state, and when it is **RESET** it goes to a '0' state. The $\overline{Q}$ output is the complement of the Q output at all times.

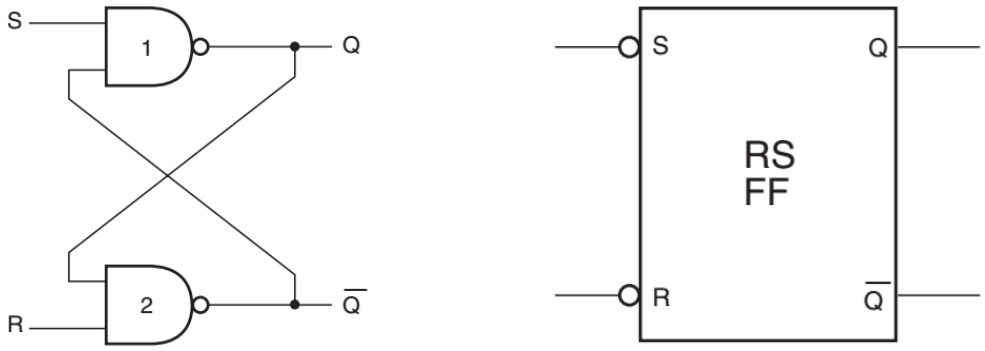
Fig 1 (b) shows the two possible operation states of R-S flip-flop. One possible state is $Q=1$, $\overline{Q}=0$ and the other $Q=0$, $\overline{Q}=1$. The state $Q=1$, $\overline{Q}=0$ is called HIGH state or 1 state. It is also called *SET* state. Whenever, the inputs to a flip-flop cause it to go to $Q=1$ state, we call this **setting** the flip-flop or the flip-flop is set.

In a similar way, the state $Q=0$, $\overline{Q}=1$ is called LOW state or 0 state. It is called RESET or CLEAR state. Whenever, the inputs to a flip-flop cause it to go to the $Q=0$ state, we call this **resetting or clearing** the flip-flop.

It may be noted from the flip-flop symbol fig (a) that a flip-flop can have one or more inputs. These inputs can be used to switch the flip-flop back and forth between its two possible output states.

1.1- R-S *Flip-Flop with Active LOW Inputs*

Figure 2(a) shows a NAND gate implementation of an R-S flip-flop with active LOW inputs. The two NAND gates are cross-coupled. That is, the output of NAND 1 is fed back to one of the inputs of NAND 2, and the output of NAND 2 is fed back to one of the inputs of NAND 1. The remaining inputs of NAND 1 and NAND 2 are the S and R inputs. The outputs of NAND 1 and NAND 2 are respectively Q and $\overline{Q}$ outputs.



Operation

Normally, the SET and CLEAR input of the flip-flop are resting in the HIGH state. Whenever we want to change the outputs, one of the two inputs will be pulsed LOW. Let us begin our study by showing that there are two equally likely output states when

SET=CLEAR=1. One possibility is shown in fig 2(a), where we have $Q=0$ and $\overline{Q}=1$. With $Q=0$, the inputs to the NAND-2 are 0 and 1, which produces $\overline{Q}=1$. The 1 from $\overline{Q}$ causes NAND-1 to have 1 at both inputs to produce a 0 output at Q thus we find that a LOW at the NAND-1 output produces a HIGH at NAND-2 output which in turn keeps the NAND-1 output LOW.

Fig. 2(b) shows the second possibility. Here $Q=1$ and $\overline{Q}=0$. As seen from this figure, the HIGH from NAND-1 produces LOW at the NAND-2 output, which in turn keep the NAND-1 output HIGH. Thus there are two possible output states when SET=CLEAR=1. However which one of these two states exist will depend on what has occurred previously at the inputs. This is discussed below in more detail where we will take up three situations: (1) setting the flip-flop, (2) clearing the flip-flop and (3) simultaneously setting and clearing the flip-flop.

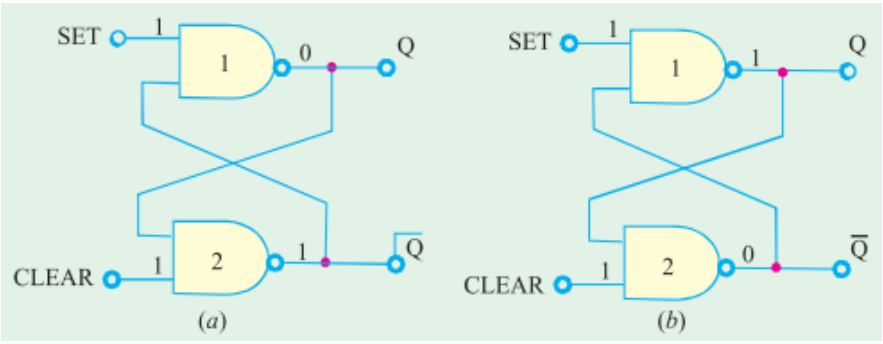


Fig. 2. Illustrating the two possible output state of NAND gate flip-flop

a) Setting the flip-flop

We have already mentioned above that if SET=CLEAR=1, the output can be in two possible states i.e. $Q=1$ and $Q=0$. Now we will study as what happens when the SET input is momentarily pulsed LOW while CLEAR is kept HIGH.

Fig. 3 (a) shows what happens when $Q=0$ prior to the occurrence of the pulse….

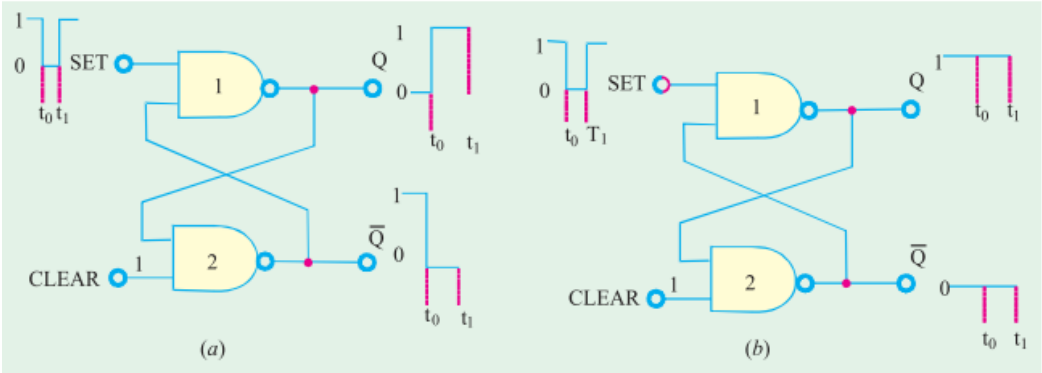


Fig. 3. Setting the NAND gate flip-flop

Fig.3. (b) shows what happens when $Q=1$ and $\overline{Q}=0$. Prior to the application of the SET pulse

b) Clearing the flip-flop

Here, we study what happens when CLEAR input is pulsed LOW while SET is kept HIGH. Fig. 4 (a) shows the situation when $Q=0$ and $\overline{Q}=1$ prior to the application of the pulse…

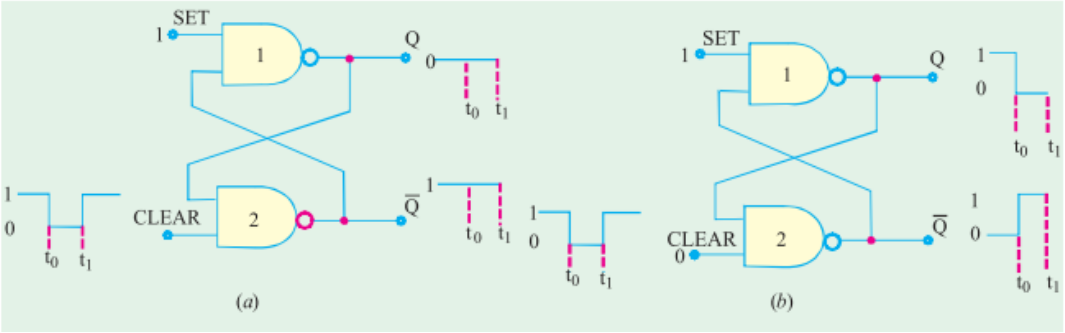


Fig. 3(b) shows the situation where $Q=1$ prior to the occurrence of the CLEAR pulse….

c) Simultaneous setting and clearing

SET and CLEAR inputs are simultaneously pulsed LOW, this will produce HIGH levels at both NAND output so that $Q=\overline{Q}=1$. This is an undesirable condition as the two outputs are supposed to be inversed of each other. Moreover, when the SET and CLEAR input return HIGH, the resulting output stat will depend on which input HIGH first. Simultaneous transition back to 1 state will produce unpredictable results. Because of these reasons the SET=CLEAR=0 condition is normally not used for the NAND flip-flop and is considered as invalid condition.

The operation of NAND gate flip-flop may be summarized in the form of a table as shown in fig. 5. Each line of the truth table is described as below:

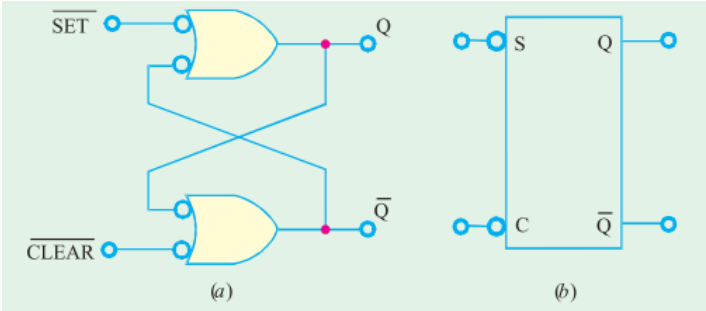
- **SET=CLEAR=1.** This condition is the normal resting state of the flip-flop. The Q and $\overline{Q}$ output will remain in the same state in which they were prior to this input condition.
- **SET=0, CLEAR=1.** This condition will always cause the output to go to $Q=1$state where it will remain even after SET returns HIGH. This is called *setting* the flip-flop.
- **SET=1, CLEAR=0.** This condition will always produce $Q=0$. The output will remain in this state even after CLEAR returns HIGH. This is called *clearing or resetting* the flip-flop.
- **SET=CLEAR=0.** This condition tries to set and clear the flip-flop simultaneously. It produces invalid results and should not be used.

Inputs		Output
SET	CLEAR	
1	1	No change
0	1	$Q = 1$
1	0	$Q = 0$
0	0	$Q = \bar{Q} = 1$ (Invalid)

Fig. 5. Truth table of NAND flip-flop

Alternative representation of NAND gate R-S flip-flop

NAND gate flip-flop is often drawn using the alternative representation for each NAND gate as shown in Fig. 6.

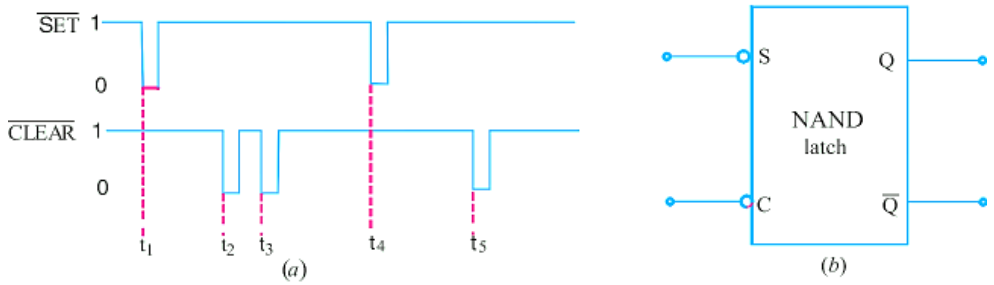


The diagram shows a block representation, the S and C label represent the SET and CLEAR inputs. While the bubbles at S and C inputs indicate the active LOW nature of these inputs.

Note. The action of clearing a flip-flop is also called **resetting**. Both these terms (clearing or resetting) is used interchangeably in the field of digital electronics. Thus a SET-CLEAR flip-flop can also be referred to as SET-RESET flip-flop (or simply S-R Latch)

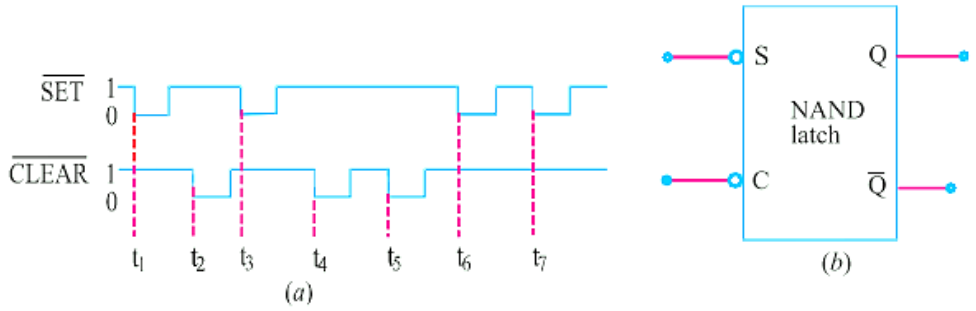
Exercise 1

The waveforms shown below are applied to the input of the NAND flip-flop as shown in the figure. Assume that initially $Q = 0$ and determine the $\bar{Q}$ waveform.



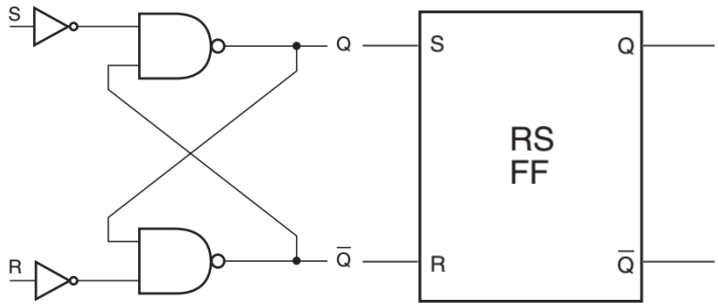
Exercise 2

If the $\overline{SET}$ and $\overline{CLEAR}$ waveforms shown in the figure below are applied to the inputs of NAND latch shown in the figure. Determine the waveform that will be observed on the output Q. Assume that initially $Q = 0$.



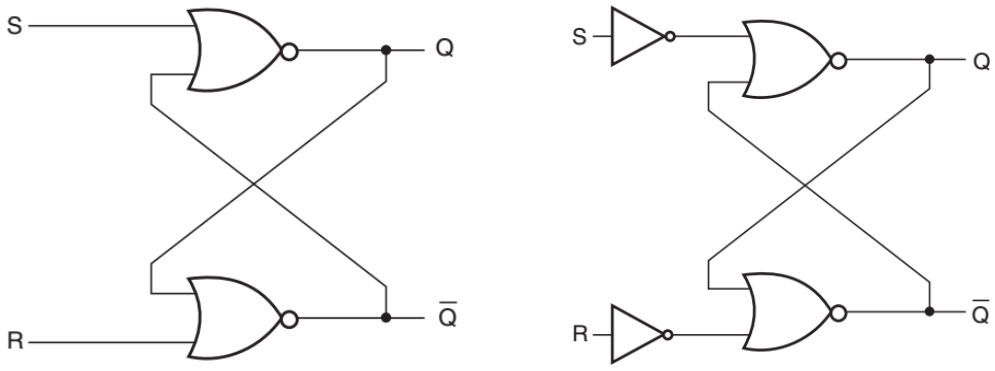
1.2- R-S Flip-Flop with Active HIGH Inputs

Fig. 7(a) shows another NAND gate implementation of the R-S flip-flop. Fig. 7(b) and (c) respectively show its circuit symbol and function table. Such a circuit would have active HIGH
inputs. The input combination R = S = 1 would be forbidden as SET and RESET inputs in an R-S
flip-flop cannot be active at the same time.



Operation Mode	S	R	Q_{n+1}
No change	0	0	Q_n
SET	1	0	1
RESET	0	1	0
Forbidden	1	1	—

The R-S flip-flops (or latches) may also be implemented with NOR gates. The NOR gate counterparts of Fig. 2(a) and Fig. 7(a) are respectively shown in Figs 8(a) and (b).

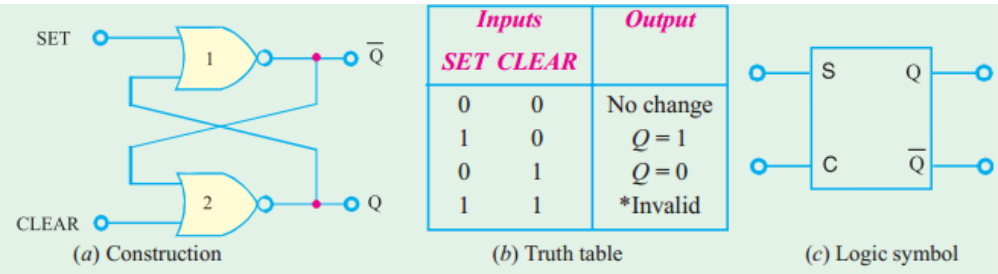


As seen in the figures, the arrangement is similar to the NAND gate latch except that the Q and $\overline{Q}$ output have reversed positions.

Operation

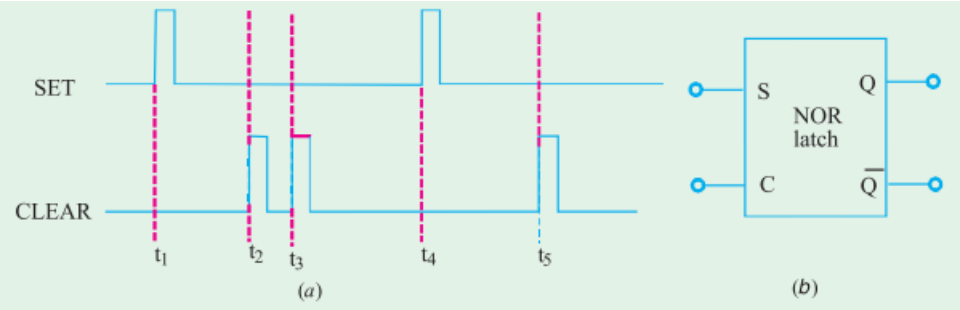
The detailed operation of the NOR latch can be understood exactly in the same manner as for the NAND flip-flop. The results are given in the truth table and are summarized as follows:

- **SET=CLEAR=0.** This condition is the normal resting state for the NOR latch. It has no effect on the output state. In other words, Q and $\overline{Q}$ will remain in the same state in which they were prior to this input condition.
- **SET=1, CLEAR=0**. This condition will always cause the output to go to $Q=1$ state where it will remain even after SET returns to 0.
- **SET=0, CLEAR=1.** This condition will always cause the output to go to $Q=0$ state wher it will remain even after CLEAR returns to 0.
- **SET=1, CLEAR=1.** This condition ries to set clear the latch simultaneously. It produces invalid result and should not be used.



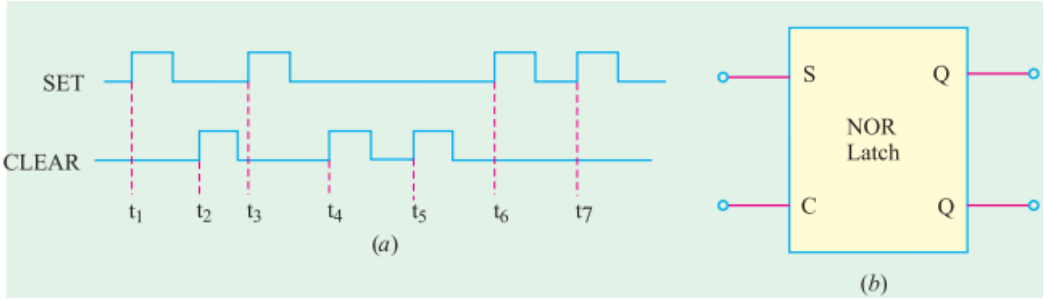
Exercise 3

The waveforms shows in the figure below are applied to the input of the NOR latch shown in the figure. Assume that initially, $Q=0$ and determine the $\overline{Q}$ waveform.



Exercise 4

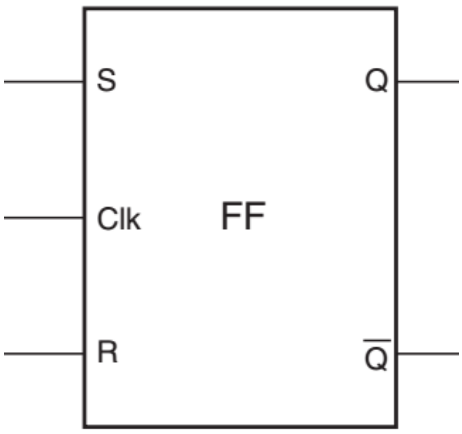
If the $\overline{SET}$ and $\overline{CLEAR}$ waveforms shown in the figure below are applied to the inputs of NOR latch shown in the figure. Determine the waveform that will be observed on the output Q. Assume that initially $Q=0$.



1.3- Clocked R-S Flip-Flop

In the case of a clocked R-S flip-flop, or for that matter any clocked flip-flop, the outputs change states as per the inputs only on the occurrence of a clock pulse. The clocked flip-flop could be a level-triggered one or an edge-triggered one. The two types are discussed in the next section.

For the time being, let us first see how the flip-flop of the previous section can be transformed into a clocked flip-flop. Figure 8(a) shows the logic implementation of a clocked flip-flop that has active HIGH inputs. The function table for the same is shown in Fig. 8(b) and is self-explanatory.



S	R	Clk	Q_{n+1}
0	0	0	Q_n
0	0	1	Q_n
0	1	0	Q_n
0	1	1	0
1	0	0	Q_n
1	0	1	1
1	1	0	Q_n
1	1	1	Invalid

Fig. 8 symbol and truth table

The two NAND gates at the input have been used to couple the R and S inputs to the flip-flop inputs under the control of the clock signal Fig. 9. When the clock signal is HIGH, the two NAND gates are enabled and the S and R inputs are passed on to flip-flop inputs with their status complemented. The outputs can now change states as per the status of R and S at the flip-flop inputs. For instance, when S = 1 and R = 0 it will be passed on as 0 and 1 respectively when the clock is HIGH. When the clock is LOW, the two NAND gates produce a '1' at their outputs, irrespective of the S and R status. This produces a logic '1' at both inputs of the flip-flop, with the result that there is no effect on the output states.

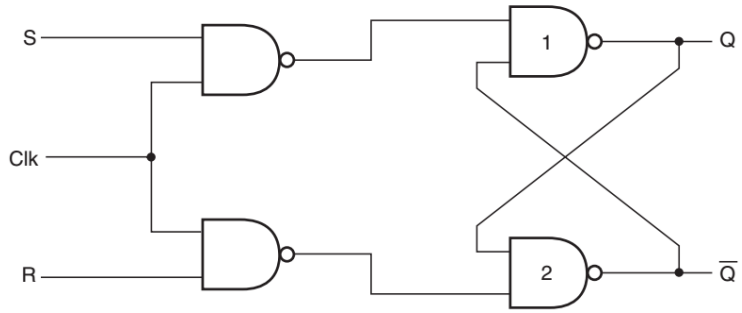
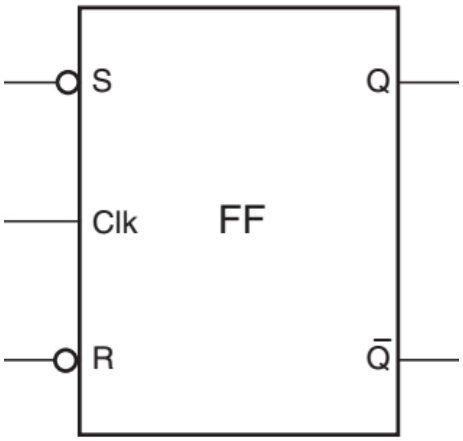


Fig. 9 logic diagram

Figure 10(a) shows the clocked R-S flip-flop with active LOW R and S inputs. The truth table of this flip-flop, as given in Fig. 11(b), is self-explanatory.



S	R	Clk	Q_{n+1}
0	0	0	Q_n
0	0	1	Invalid
0	1	0	Q_n
0	1	1	1
1	0	0	Q_n
1	0	1	0
1	1	0	Q_n
1	1	1	Q_n

Fig. 10. Symbol and truth table

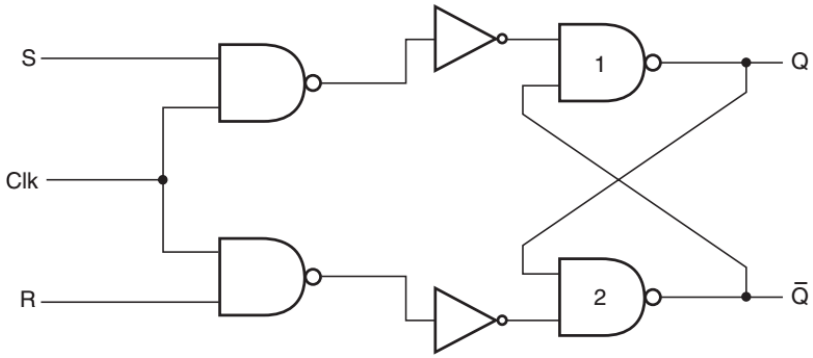


Fig. 11. Logic diagram

1.4- Level-Triggered and Edge-Triggered Flip-Flops

In a *level-triggered* flip-flop, the output responds to the data present at the inputs during the time the clock pulse level is HIGH (or LOW). That is, any changes at the input during the time the clock is active (HIGH or LOW) are reflected at the output as per its function table.

The clocked R-S flip-flop described in the preceding paragraphs is a level-triggered flip-flop that is active when the clock is HIGH.

In an *edge-triggered* flip-flop, the output responds to the data at the inputs only on LOW-to-HIGH or HIGH-to-LOW transition of the clock signal. The flip-flop in the two cases is referred to as positive edge triggered and negative edge triggered respectively. Any changes in the input during the time the clock pulse is HIGH (or LOW) do not have any effect on the output.

The clock transition are commonly referred to as **edges** and are pointed out in figure 12. It may be noted that when the clock change from 0 to 1, this called **rising edge** or positive-going transistion (PGT). On the other hand, when the clock changes from 1 to 0, this called faling edge or negative going transistion (NGT).

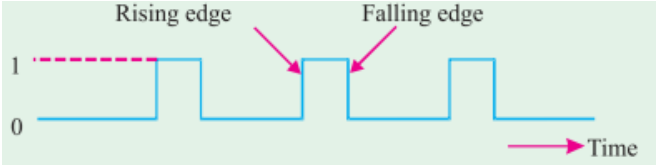
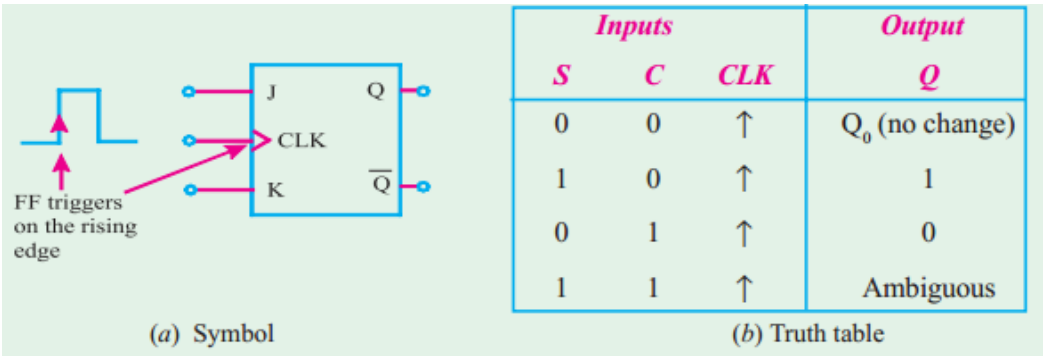


Fig. 12 Clock signal

According to clock signal, there are rising edge clocked R-S latch and the falling edge clocked R-S latch.

Rising edge clocked R-S latch

The symbol and the truth tables are given in figure 13



Inputs			*Output*
S	*C*	*CLK*	*Q*
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	Ambiguous

Figure 13. Symbol and truth table of rising clocked R-S latch

The input and output waveforms of rising edge clocked R-S latch is shown in figure 14

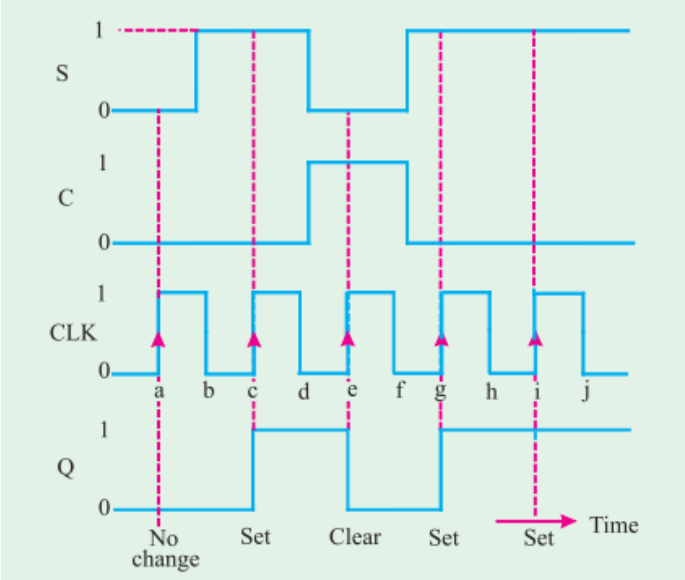
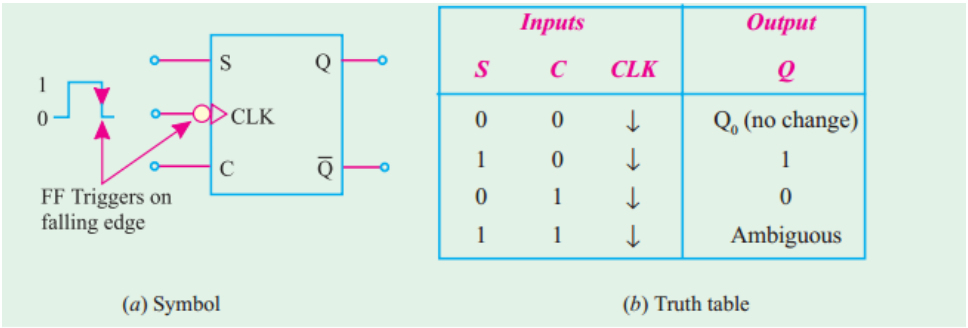


Fig. 10. Waveforms of inputs and output

Falling edge clocked R-S latch

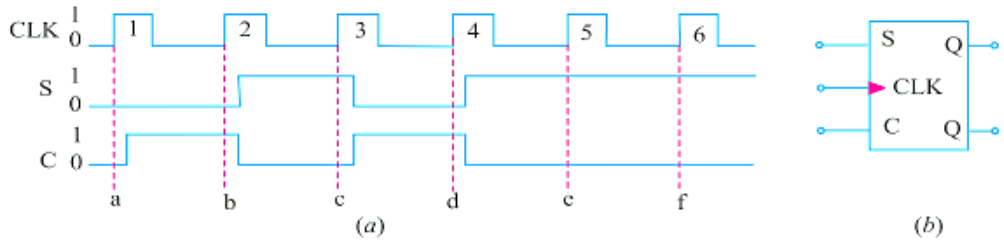


Inputs			*Output*
S	*C*	*CLK*	*Q*
0	0	↓	Q_0 (no change)
1	0	↓	1
0	1	↓	0
1	1	↓	Ambiguous

Figure 15. Symbol and truth table of falling clocked R-S latch

Exercise 5

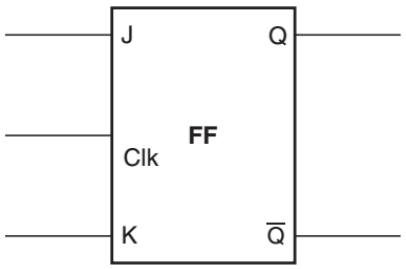
The figure below shows the SET and RESET waveforms applied at the inputs of the clocked R-S latch shown in the same figure. Sketch the waveforms at Q and $\overline{Q}$ output of the flip-flop. Assume that initially $Q = 0$ and $\overline{Q} = 1$



2- *J-K* Flip-Flop

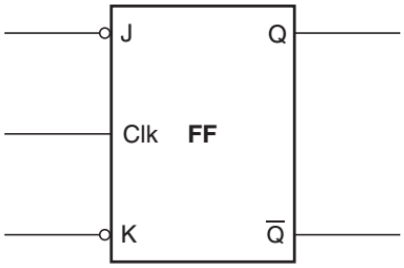
A J-K flip-flop behaves in the same fashion as an R-S flip-flop except for one of the entries in the function table. In the case of an R-S flip-flop, the input combination S = R = 1 (in the case of a flip-flop with active HIGH inputs) and the input combination S = R = 0 (in the case of a

flip-flop with active LOW inputs) are prohibited. In the case of a J-K flip-flop with active HIGH inputs, the output of the flip-flop toggles, that is, it goes to the other state, for J = K = 1 . The output toggles for J = K = 0 in the case of the flip-flop having active LOW inputs. Thus, a J-K flip-flop overcomes the problem of a forbidden input combination of the R-S flip-flop. Figures 16(a) and (b) respectively show the circuit symbol of level-triggered J-K flip-flops with active HIGH and active LOW inputs, along with their function tables. Figure 17 shows the realization of a J-K flip-flop with an R-S flip-flop.



Operation Mode	J	K	Clk	Q_{n+1}
SET	1	0	1	1
RESET	0	1	1	0
NO CHANGE	0	0	1	Q_n
TOGGLE	1	1	1	$\overline{Q}_n$

(a)



Operation Mode	J	K	Clk	Q_{n+1}
SET	0	1	1	1
RESET	1	0	1	0
NO CHANGE	1	1	1	Q_n
TOGGLE	0	0	1	$\overline{Q}_n$

(b)

Figure 16 (a) J-K flip-flop active HIGH inputs and (b) J-K flip-flop active LOW inputs.

The characteristic tables for a J-K flip-flop with active HIGH J and K inputs and a J-K flip-flop with active LOW J and K inputs are respectively shown in Figs 10.28(a) and (b)

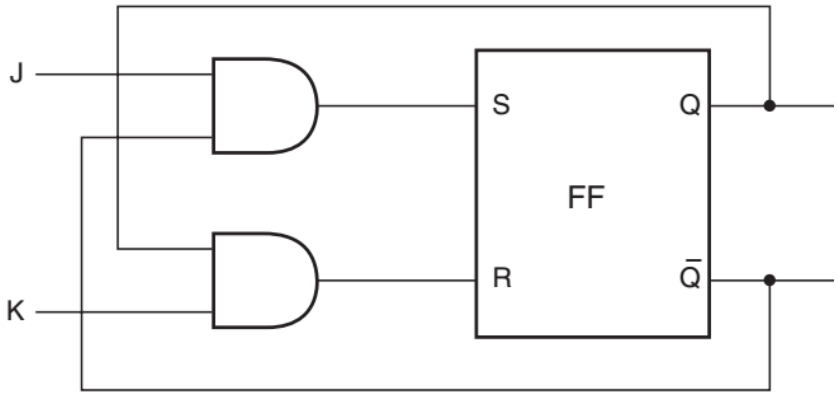


Figure 17 Realization of a J-K flip-flop using an R-S flip-flop

Q_n	J	K	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(a)

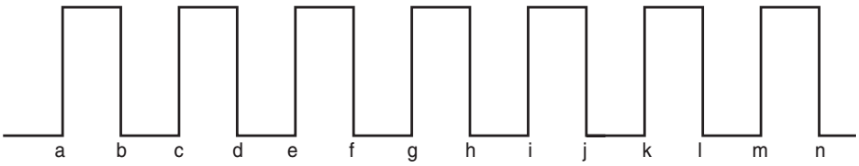
Q_n	J	K	Q_{n+1}
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

(b)

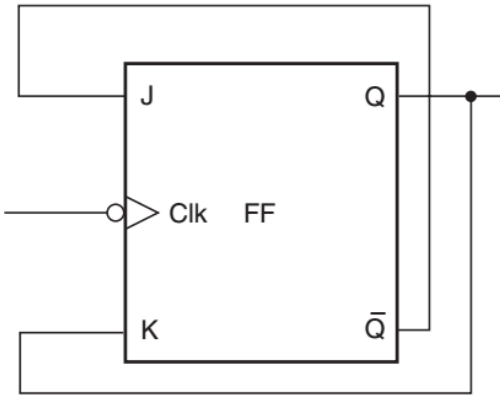
Figure 18 characteristic tables for (a) a J-K flip-flop with active HIGH J and K inputs and (b) a J-K flip-flop with active LOW J and K inputs

Exercice 6

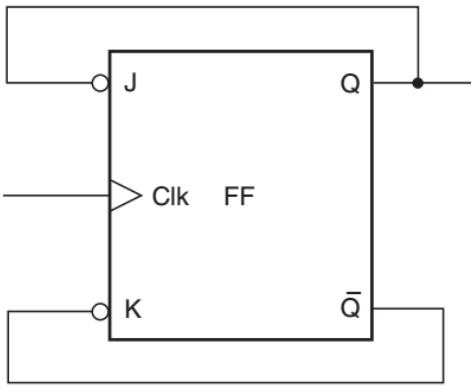
The 100 kHz square waveform of Fig. (a) is applied to the clock input of the flip-flops shown in Figs. (b) and (c). If the Q output is initially '0', draw the Q output waveform in the two cases. Also, determine the frequency of the Q output in the two cases.



(a)



(b)

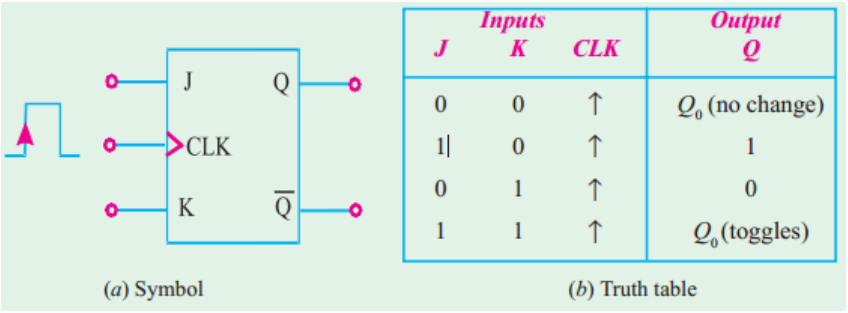


(c)

According to clock signal, there are rising edge triggered J-K latch and the falling edge clocked J-K latch.

Rising edge triggered* J-K *latch

The symbol and truth tables of the rising edge triggered J-K latch are shown in fig. 19.



Inputs J	Inputs K	Inputs CLK	Output Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	Q_0 (toggles)

Fig.19. *Rising edge triggered J-K flip-flop*

The operation of the rising edge triggered J-K flip-flop can be understood using the following diagram

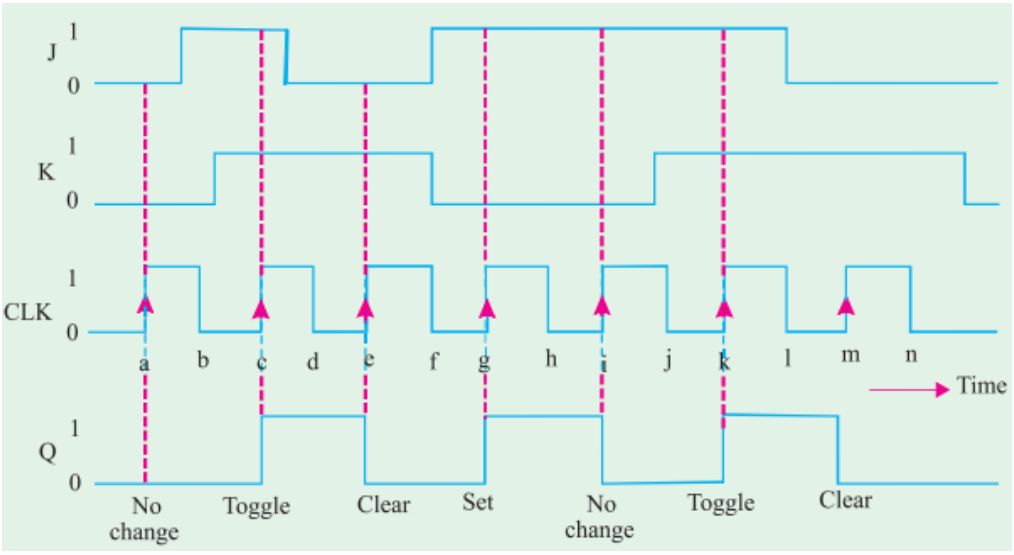
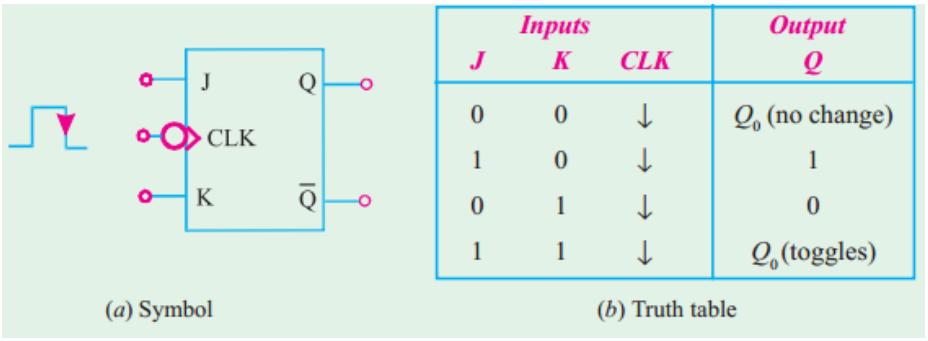


Fig.20. Waveforms of the rising edge triggered J-K flip-flop

It may be noted from the waveforms that the flip-flop is not affected by the falling edge of the clock pulses. Also the J and K input levels have no effect except the occurrence of the rising edge of the clock signal. The J and K inputs by themselves cannot cause the flip-flop to change states.

Falling edge triggered* J-K *latch

The symbol and truth tables of the falling edge triggered J-K latch are shown in fig. 21.



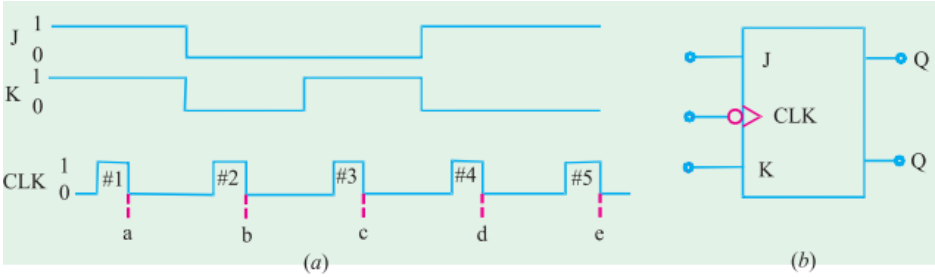
Inputs J	Inputs K	Inputs CLK	Output Q
0	0	↓	Q_0 (no change)
1	0	↓	1
0	1	↓	0
1	1	↓	Q_0 (toggles)

Fig.20. falling edge triggered J-K flip-flop

The small circle on the CLK input indicates that the flip-flop will trigger when the CLK input goes from 1 to 0. This flip-flop operates in the same ways as the rising edge of the flip-flop except that the output can change states only on the falling edge of CLK signal.

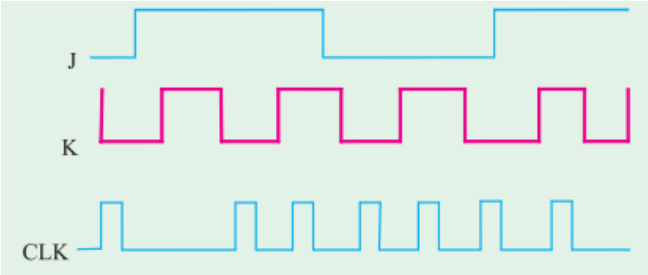
Exercise 7

Fig (a) shows the waveforms applied at J,K and CLK inputs of the clocked J-K flip-flop shown in fig (b). Sketch the Q output waveform. Assume $Q = 0$ initially.



Exercise 8

What will be the output waveform Q of a J-K flip-flop if the following waveforms are applied at the input? Assume the flip-flop triggers at the falling edge of clock pulse.



3- Flip-Flop Timing Parameters

Certain timing parameters would be listed in the specification sheet of a flip-flop. Some of these parameters, as we will see in the paragraphs to follow, are specific to the logic family to which the flip-flop belongs. There are some parameters that have different values for different flip-flops belonging to the same broad logic family. It is therefore important that one considers these timing parameters before using a certain flip-flop in a given application. Some of the important ones are set-up and hold times, propagation delay, clock pulse HIGH and LOW times, asynchronous input active pulse width, clock transition time and maximum clock frequency.

3.1- Set-Up and Hold Times

The *set-up time* is the minimum time period for which the synchronous inputs (for example, R, S, J, K and D) and asynchronous inputs (for example, PRESET and CLEAR) must be stable prior to the active clock transition for the flip-flop output to respond reliably at the

clock transition. It is usually denoted by t_s (min) and is usually defined separately for synchronous and asynchronous inputs.

The *hold time* t_H (min) is the minimum time period for which the synchronous inputs (R, S, J, K, D) must remain stable in the desired logic state after the active clock transition for the flip-flop to respond reliably.

To sum up, for a flip-flop to respond properly and reliably at the active clock transition, the synchronous inputs must be stable in their intended logic states and the asynchronous inputs must be stable in their inactive states for at least a time period equal to the specified minimum set-up times prior to the clock transition, and the synchronous inputs must be stable for a time period equal to at least the specified minimum hold time after the clock transition.

3.2- Propagation Delay

There is always a time delay, known as the *propagation delay*, from the time instant the signal is applied to the time the output makes the intended change. The flip-flop data sheet usually specifies propagation delays for both HIGH-to-LOW (t_{pHL}) and for LOW-to-HIGH (t_{pLH}) output transitions. The propagation delay is measured between 50 % points on input and output waveforms and is usually specified for all types of input including synchronous and asynchronous inputs.

3.3- Clock Pulse HIGH and LOW Times

The clock pulse HIGH time t_W (H) and clock pulse LOW time, t_W (L) are respectively the minimum time durations for which the clock signal should remain HIGH and LOW. Failure to meet these requirements can lead to unreliable triggering.

3.4- Asynchronous Input Active Pulse Width

This is the minimum time duration for which the asynchronous input (PRESET or CLEAR) must be kept in its active state, usually LOW, for the output to respond properly.

3.5- Clock Transition Times

The manufacturers specify the maximum transition times (rise time and fall time) for the output to respond properly. If these specified figures are exceeded, the flip-flop may respond erratically or even may not respond at all. This parameter is logic family specific and is not specified for individual devices.

3.6- Maximum Clock Frequency

This is the highest frequency that can be applied to the clock input. If this figure is exceeded, there is no guarantee that the device will work reliably and properly. This figure may vary slightly from device to device of even the same type number. The manufacturer usually specifies a safe value. If this specified value is not exceeded, the manufacturer guarantees that the device will trigger reliably.

Review questions

1. What is a flip-flop? Show the logic implementation of an R-S flip-flop having active HIGH R and S inputs. Draw its truth table and mark the invalid entry.

2. With the help of the logic diagram, describe the operation of a clocked R-S flip-flop with active LOW R and S inputs. Draw the truth table of this flip-flop if it were negatively edge triggered.

3. What is a clocked J-K flip-flop? What improvement does it have over a clocked R-S flip-flop?

4. Differentiate between:

(a) synchronous and asynchronous inputs;
(b) level-triggered and edge-triggered flip-flops;
(c) active LOW and active HIGH inputs.

5. Briefly describe the following flip-flop timing parameters:

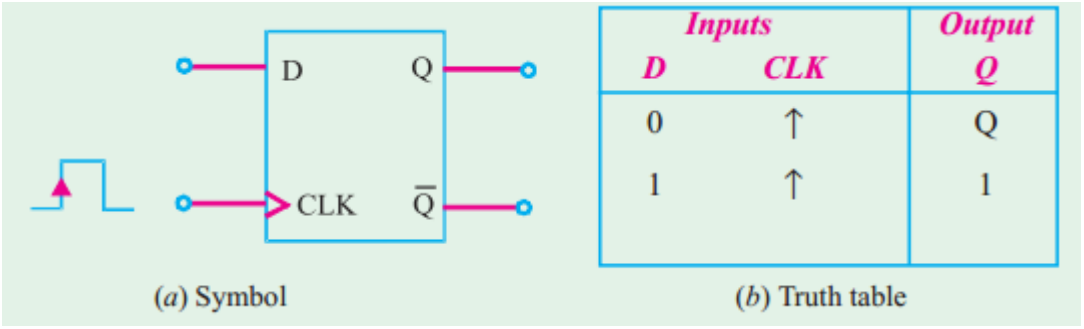
(a) set-up time and hold time;
(b) propagation delay;
(c) maximum clock frequency.

D Flip-Flop

A D flip-flop, also called a *delay flip-flop*, can be used to provide temporary storage of one bit of information.

Rising edge flip-flop

Figure 21 shows the circuit symbol and truth table for a clocked D flip-flop that triggers on the rising edge of the clock pulse. Notice that this flip-flop has only one synchronous control input D which stands for data.



Inputs		Output
D	*CLK*	*Q*
0	↑	Q
1	↑	1

Figure 21. Symbol and truth table of D flip-flop

The operation of the clocked D flip-flop is very simple. The output Q will go to the same that is presen on the D input when the rising edge occurs at the CLK. In orther words, th level present at D will be stored in the flip flop at the instant the rising edge occurs.

In order to understand th operation of flip-flop in more details, let consider the waveforms at D and CLK input shown in fig. 22. Assume tha Q is initially 0.

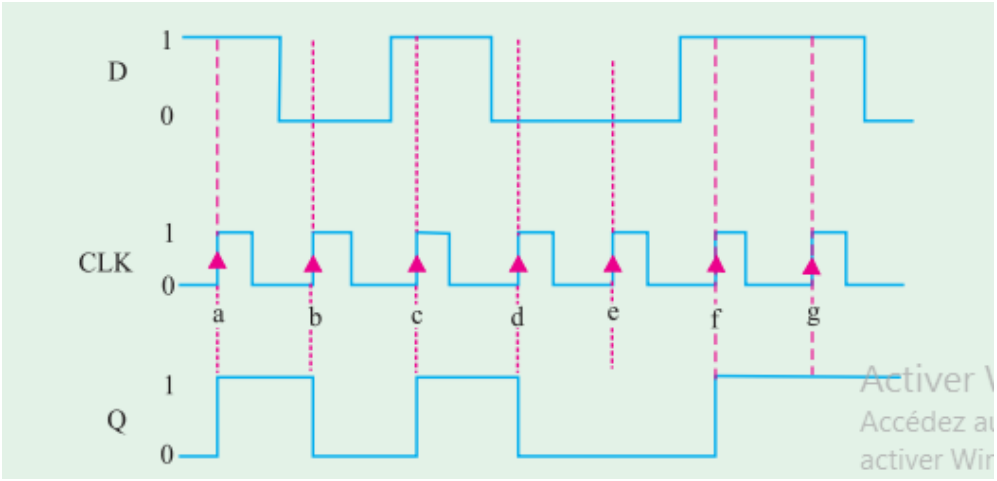
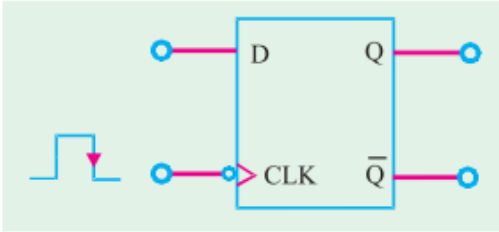


Fig. 22. Input and output waveforms to illustrate the operation of clocked D flip-flop

Falling edge flip-flop

A falling edge triggered D flip-flop operates in the same way as the rising edge triggered D flip-flop. However, the difference is that Q will take on the value of D when a falling edge occurs at the CLK. The symbol of the D flip-flop that triggered on the falling edge is shown in figure 23.



Implementation of D flip-flop can be obtained easily by adding a single INVERTER to an edge triggered J-K flip-flop as shown in figure 24. A similar approach can be used to convert a R-S flip-flop to a D flip-flop.

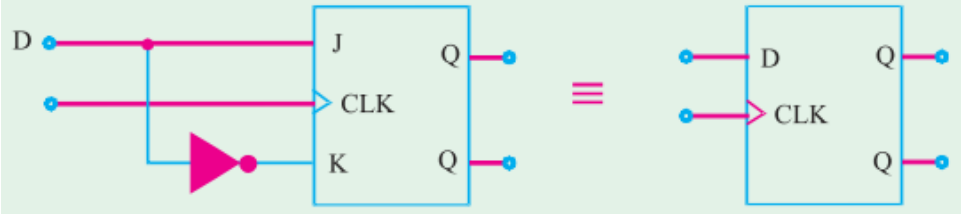


Figure 24. implementation of a D flip-flop from a J-K flip-flop

In the D flip-flop, the output is delay from the input by one clock period. This is an advantage as a D flip-flop is used sometimes to delay a binary waveform so that the binary information appears at the output a certain amount of time after it appears at the D input.

It is also possible to delay the input by two clock periods. This can be achieved by connecting Q to the D input of the second flip-flop and connect the clock signal to the second flip-flop. The output of the second flip-flop will be delayed by 2 clock periods from the input data.

Exercise 9 fig (a) shows a D flip-flop with the input and clock waveforms applied at their respective input. Determine th Q waveform.

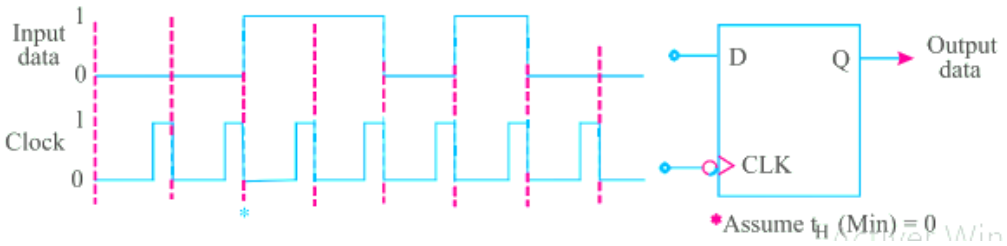


Fig. (a)

Exercise 10 an edge-triggered D flip-flop can be made to operate in the toggle mode by connecting it as shown in fig (b). Assume Q=0 initially and determine the Q waveform.

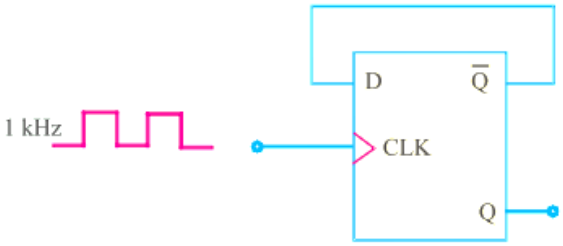


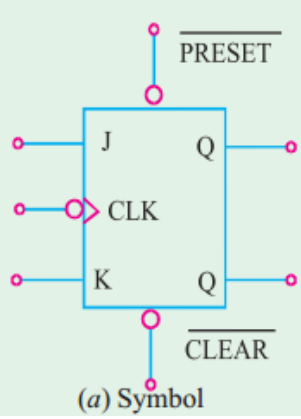
Fig. (b)

Clocked J-K flip-flop with asynchronous inputs

Strictly speaking for the clocked flip-flop, the R, S, J, K and D inputs are called ***control inputs*** or also known as ***synchronous inputs*** because the effect of these inputs is synchronized with the CLK input; therefore they need triggering signal.

Most clocked flip-flops have one-or-more **asynchronous inputs**. These inputs operate independently of the synchronous inputs and clock input. The asynchronous inputs of a flip-flop can be used to set its output to 1 state or clear to 0 state at any time regardless of the condition at the other inputs. In other word the asynchronous inputs can be used to **override** all the other inputs in the order to place the flip-flop output in one state or either. Because of this reason, the asynchronous inputs are also called **override inputs.**

Figure 25 shows a clocked J-K flip-flop with two asynchronous inputs namely $\overline{PRESET}$ and $\overline{CLEAR}$ then the truth table. Both these inputs are active LOW, indicated by the bubbles on flip-flop symbol.



(*a*) Symbol

Inputs		*Output / Flip-flop response*
$\overline{PRESET}$	$\overline{CLEAR}$	
1	1	Clocked operation*
0	1	Q = 1 (regardless of CLK)
1	0	Q = 0 (regardless of CLK)
0	0	Not used

(*b*) Truth table

Fig. 25. Symbol and truth table of J-K with asynchronous inputs

$\overline{PRESET} = \overline{CLEAR} = 1$. This condition indicates that both the asynchronous inputs are inactive and the flip-flop is free to respond to the J, K and CLK inputs. In other words, the flip-flop operates as a normal clocked flip-flop.

$\overline{PRESET} = 0, \overline{CLEAR} = 1$. This condition indicates that the asynchronous input $\overline{PRESET}$ is activated. Because of this, the output, Q is immediately set to 1, no matter what conditions are presenT at the J, K and CLK inputs. It may be carefully note that the CLK input cannot affect the flip-flop while $\overline{PRESET} = 0$

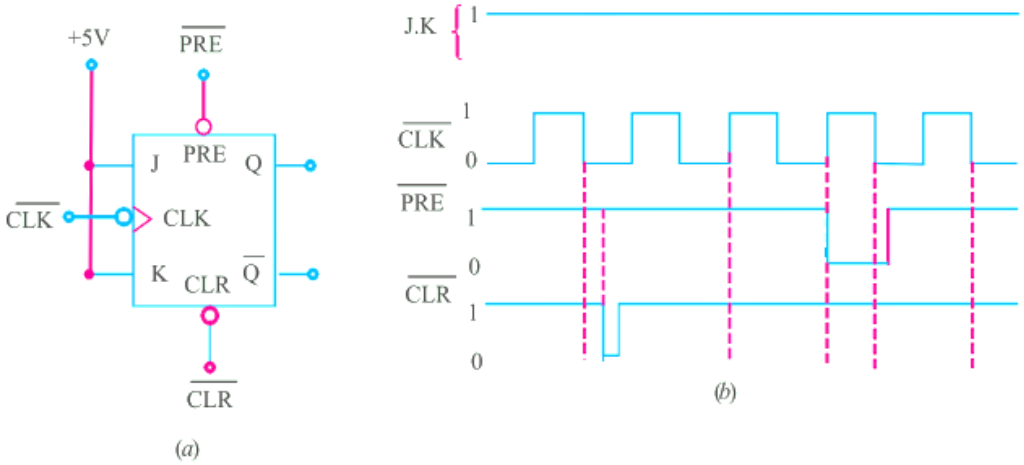
$\overline{PRESET} = 1, \overline{CLEAR} = 0$. This condition indicates that the asynchronous input $\overline{CLEAR}$ is activated. Because of this, the output, Q is immediately set to 0, no matter what conditions are present at the J, K and CLK inputs. It may be carefully note that the CLK input cannot affect the flip-flop while $\overline{CLEAR} = 0$.

$\overline{PRESET} = \overline{CLEAR} = 1$. This condition should not be used because it can produce an ambiguous response

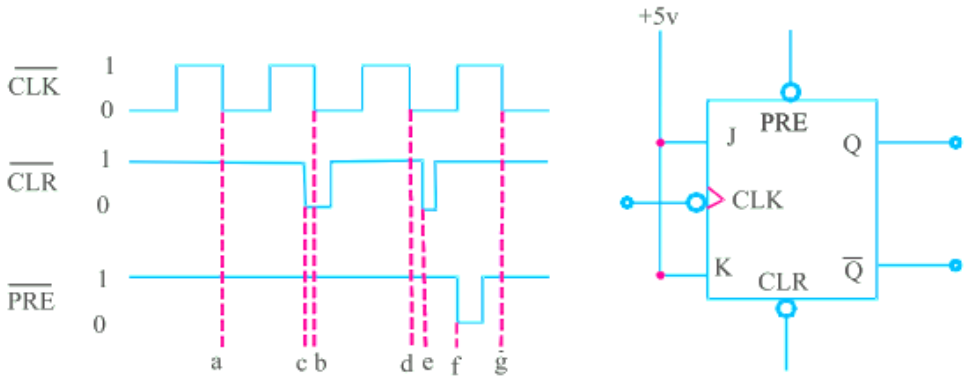
Many clocked flip-flop that are commercially available as ICs, will have both $\overline{PRESET}$ and $\overline{CLEAR}$. However, some ICs will have only $\overline{CLEAR}$ input. Some other ICs will have asynchronous inputs that are active-High. For these ICs, the flip-flop symbol would not have bubble on the asynchronous inputs.

ICs 7476 and 74LS76 are popular J-K flip-flop with asynchronous inputs preset and clear, designated by $\overline{S}_D$ and $\overline{R}_D$, respectively. The IC 7476 is the rising edge while IC 74LS76 is the falling edge device.

Exercise 11. Figure (a) shows the logic symbol for a J-K flip-flop that responds to the falling edge on its clock pulse and has active-LOW asynchronous inputs. The J and K inputs are tied HIGH. Determine the Q-output in response to the waveform shown in figure (b). Assume that initially Q = 0.



Exercise 12. Determine the Q-output for the J-K flip-flop shown in figure below, assume that Q = 0 initially and remember that the asynchronous inputs override all other inputs.



Chapter 2 and 3. Counters and Registers

Counters and *registers* have similar architecture, as both counters and registers comprise a cascaded arrangement of more than one flip-flop with or without combinational logic devices. Both constitute very important building blocks of sequential logic and different types of counter and register available in integrated circuit (IC) form are used in a wide range of digital systems. While counters are mainly used in counting applications, where they either measure the time interval between two unknown time instants or measure the frequency of a

given signal, registers are primarily used for the temporary storage of data present at the output of a digital circuit before they are fed to another digital circuit. We are all familiar with the role of different types of register used inside a microprocessor, and also their use in microprocessor-based applications. Because of the very nature of operation of registers, they form the basis of a very important class of counters called *shift counters*. In this chapter, we will discuss different types of counter and register as regards their operational basics, design methodology and application-relevant aspects.

I. COUNTERS

1. Ripple (Asynchronous) Counter

A *ripple counter* is a cascaded arrangement of flip-flops where the output of one flip-flop drives the clock input of the following flip-flop. The number of flip-flops in the cascaded arrangement depends upon the number of different logic states that it goes through before it repeats the sequence, a parameter known as the modulus of the counter.
In a ripple counter, also called an ***asynchronous counter*** **or a** ***serial counter***, the clock input is applied only to the first flip-flop, also called the input flip-flop, in the cascaded arrangement. The clock input to any subsequent flip-flop comes from the output of its immediately preceding flip-flop. For instance, the output of the first flip-flop acts as the clock input to the second flip-flop, the output of the second flip-flop feeds the clock input of the third flip-flop and so on. In general, in an arrangement of *n* flip-flops, the clock input to the nth flip-flop comes from the output of the (n−1)th flip-flop for n > 1. Figure 11.1 shows the generalized block schematic arrangement of an n-bit binary ripple counter.

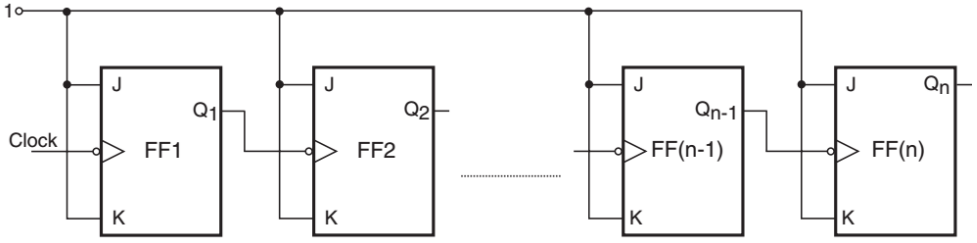


Figure 1 Generalized block schematic of n-bit binary ripple counter.

In general, the nth flip-flop will change state only after a delay equal to n times the propagation delay of one flip-flop. The term ‘ripple counter’ comes from the mode in which the clock information ripples through the counter. It is also called an ‘asynchronous counter’ as different flip-flops comprising the counter do not change state in synchronization with the input clock.

In a counter like this, after the occurrence of each clock input pulse, the counter has to wait for a time period equal to the sum of propagation delays of all flip-flops before the next clock pulse can be applied. The propagation delay of each flip-flop, of course, will depend upon the logic family to which it belongs.

1.1. Propagation Delay in Ripple Counters

The effective propagation delay in a ripple counter is equal to the sum of propagation delays due to different flip-flops. In larger bit counters an increased propagation delay puts a limit on the maximum frequency used as clock input to the counter.

The clock signal time period must be equal to or greater than the total propagation delay. The maximum clock frequency therefore corresponds to a time period that equals the total propagation delay. If t_{pd} is the propagation delay in each flip-flop, then, in a counter with N flip-flops having a modulus of less than or equal to 2^N, the maximum usable clock frequency is given by $f_{max} = 1/(N \times tpd)$.

Often, two propagation delay times are specified in the case of flip-flops, one for LOW-to-HIGH transition (t_{pLH}) and the other for HIGH-to-LOW transition (t_{pHL}) at the output. In such a case, the larger of the two should be considered for computing the maximum clock frequency.

As an example, in the case of a ripple counter IC belonging to the low-power Schottky TTL (LSTTL) family, the propagation delay per flip-flop typically is of the order of 25 ns. This implies that a four-bitripple counter from this logic family cannot be clocked faster than 10 MHz. The upper limit on the clock frequency further decreases with increase in the number of bits to be handled by the counter.

1.2. Modulus of a Counter

The *modulus* (MOD number) of a counter is the number of different logic states it goes through before it comes back to the initial state to repeat the count sequence. An *n*-bit counter that counts through all its natural states and does not skip any of the states has a modulus of 2^n.

Example: modulus 2, 4, 8, 16 and so on, respectively.

These can be modified with the help of additional combinational logic to get a modulus of less than 2^n.

To determine the number of flip-flops required to build a counter having a given modulus, identify the smallest integer *m* that is either equal to or greater than the desired modulus and is also equal to an integral power of 2. For instance, if the desired modulus is 10, which is the case in a decade counter, the smallest integer greater than or equal to 10 and which is also an integral power of 2 is *16*. The number of flip-flops in this case would be 4, as $16 = 2^4$. On the same lines, the number of flip-flops required to construct counters with MOD numbers of 3, 6, 14, 28 and 63 would be 2, 3, 4, 5 and 6 respectively. In general, the arrangement of a minimum number of N flip-flops can be used to construct any counter with a modulus given by the equation

$$\left(2^{N-1}+1\right) \leq \text{modulus} \leq 2^N$$

1.3. Binary Ripple Counter – Operational Basics

The operation of a binary ripple counter can be best explained with the help of a typical counter of this type. Figure 2(a) shows a four-bit ripple counter implemented with negative edge-triggered *J-K* flip-flops wired as toggle flip-flops. The outputs of the four flip-flops are designated as Q0 (LSB flip-flop), Q1, Q2 and Q3 (MSB flip-flop). Figure 2(b) shows the waveforms appearing at Q0, Q1, Q2 and Q3 outputs as the clock signal goes through successive cycles of trigger pulses.

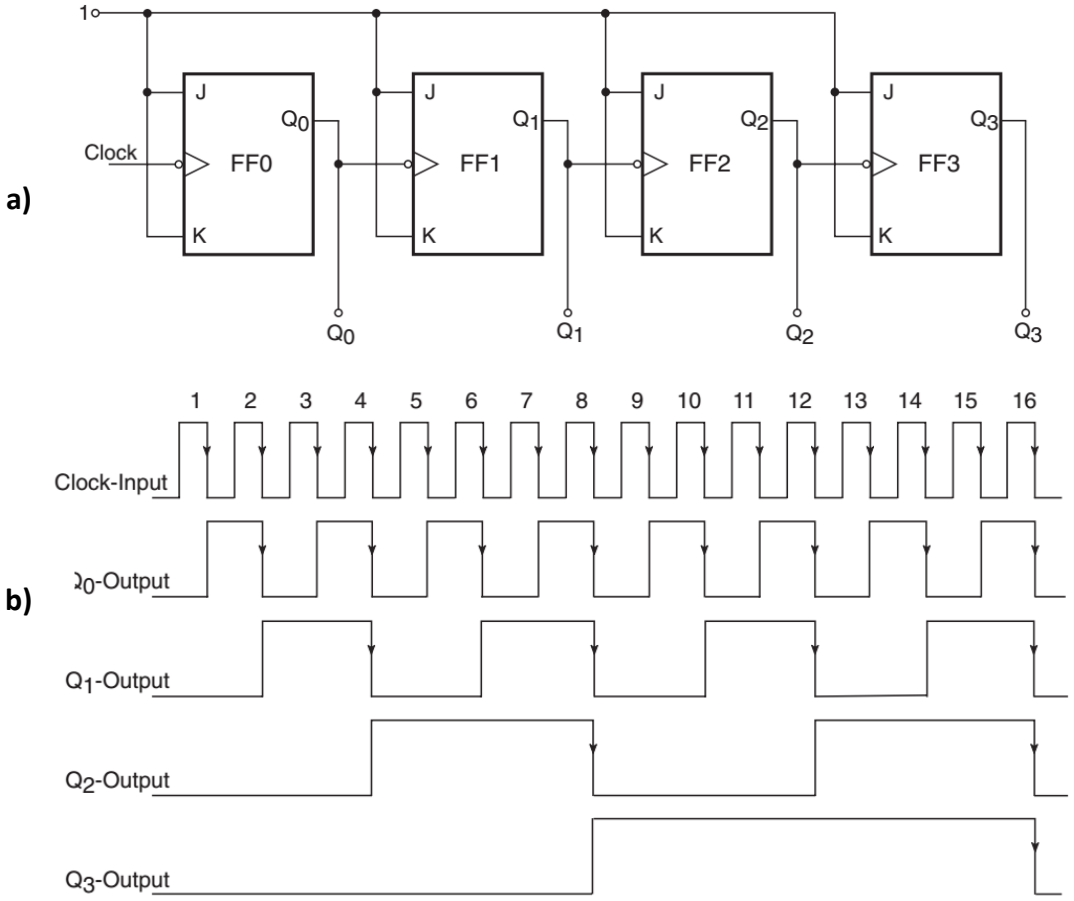


Figure 2

Operation

The counter functions as follows.

Let us assume that all the flip-flops are initially cleared to the '0' state. On HIGH-to-LOW transition of the first clock pulse, Q0 goes from '0' to '1' owing to the toggling action. As the flip-flops used are negative edge-triggered ones, the '0' to '1' transition of Q0 does not trigger flip-flop FF1. FF1, along with FF2 and FF3, remains in its '0' state. So, on the occurrence of the first negative-going clock transition, Q0 = 1, Q1 = 0, Q2 = 0 and Q3 = 0.

On the HIGH-to-LOW transition of the second clock pulse, Q0 toggles again. That is, it goes from '1' to '0'. This '1' to '0' transition at the Q0 output triggers FF1, the output Q1 of which goes from '0'to '1'. The Q2 and Q3 outputs remain unaffected. Therefore, immediately after the occurrence of the second HIGH-to-LOW transition of the clock signal, Q0 = 0, Q1 = 1, Q2 = 0 and Q3 = 0. On similar lines, we can explain the logic status of Q0, Q1, Q2 and Q3

outputs immediately after subsequent clock transitions. The logic status of outputs for the first 16 relevant (HIGH-to-LOW in the present case) clock signal transitions is summarized in Table 1.

Table 1 Output logic states for different clock signal transitions for a four-bit binary ripple counter.

Clock signal transition number	Q_0	Q_1	Q_2	Q_3
After first clock transition	1	0	0	0
After second clock transition	0	1	0	0
After third clock transition	1	1	0	0
After fourth clock transition	0	0	1	0
After fifth clock transition	1	0	1	0
After sixth clock transition	0	1	1	0
After seventh clock transition	1	1	1	0
After eighth clock transition	0	0	0	1
After ninth clock transition	1	0	0	1
After tenth clock transition	0	1	0	1
After eleventh clock transition	1	1	0	1
After twelfth clock transition	0	0	1	1
After thirteenth clock transition	1	0	1	1
After fourteenth clock transition	0	1	1	1
After fifteenth clock transition	1	1	1	1
After sixteenth clock transition	0	0	0	0

Thus, we see that the counter goes through 16 distinct states from 0000 to 1111 and then, on the occurrence of the desired transition of the sixteenth clock pulse, it resets to the original state of 0000 from where it had started. In general, if we had N flip-flops, we could count up to 2^N pulses before the counter resets to the initial state. We can also see from the Q0, Q1, Q2 and Q3 waveforms, as shown in Fig. 2(b), that the frequencies of the Q0, Q1, Q2 and Q3 waveforms are *f/2*,*f/4*, *f/8* and *f/16* respectively. Here, *f* is the frequency of the clock input. This implies that a counter of this type can be used as a divide-by-2^N circuit, where *N* is the number of flip-flops in the counter chain. In fact, such a counter provides frequency-divided outputs of $f/2^N$, $f/2^{N-1}$, $f/2^{N-2}$, $f/2^{N-3}$,… , *f/2* at the outputs of the N^{th}, $(N - 1)^{th}$, $(N - 2)^{th}$, $(N - 3)^{th}$,… , first flip-flops. In the case of a four-bit counter of the type shown in Fig. 2(a), outputs are available at *f/2* from the Q0 output, at *f/4* from the Q1 output, at *f/8* from the Q2 output and at *f/16* from the Q3 output. It may be noted that frequency division is one of the major applications of counters.

Exercise 1. It is desired to design a binary ripple counter of the type shown in Fig. 11.1 that is capable of counting the number of items passing on a conveyor belt. Each time an item passes a given point, a pulse is generated that can be used as a clock input. If the maximum number of items to be counted is 6000, determine the number of flip-flops required.

1.4. Binary Ripple Counters with a Modulus of Less than 2^N

An N-flip-flop binary ripple counter can be modified, to have any other modulus less than 2^N with the help of simple externally connected combinational logic.

We will illustrate this simple concept with the help of an example. Consider the four-flip-flop binary ripple counter arrangement of Fig. 3(a). The counter keeps counting as long as the asynchronous CLEAR inputs of the different flip-flops are inactive. That is, the NAND gate output is HIGH. This is the case until the counter reaches 0110. With the seventh clock pulse it tends to go to 0111, which makes all NAND gate inputs HIGH, forcing its output to LOW.

This HIGH-to-LOW transition at the NAND gate output clears all flip-flop outputs to the logic '0' state, thus disallowing the counter to settle at 0111. From the eighth clock pulse onwards, the counter repeats the sequence. The counter thus always counts from 0000 to 0110 and resets back to 0000. The remaining nine states, which include 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110 and 1111, are skipped, with the result that we get an MOD-7 counter.

Examination of timing waveforms shown in Fig. 3(b) also reveals that the frequency of the Q2 output is one-seventh of the input clock frequency.

Steps to be followed to design any binary ripple counter that starts from 0000 and has a modulus of X are summarized as follows:

- Determine the minimum number of flip-flops N so that $2N \geq X$ Connect these flip-flops as a binary ripple counter. If $2N = X$, do not go to steps 2 and 3.
- Identify the flip-flops that will be in the logic HIGH state at the count whose decimal equivalent is X. Choose a NAND gate with the number of inputs equal to the number of flip-flops that would be in the logic HIGH state. As an example, if the objective were to design an MOD-12 counter, then, in the corresponding count, that is, 1100, two flip-flops would be in the logic HIGH state. The desired NAND gate would therefore be a two-input gate.
- Connect the Q outputs of the identified flip-flops to the inputs of the NAND gate and the NAND gate output to asynchronous clear inputs of all flip-flops.

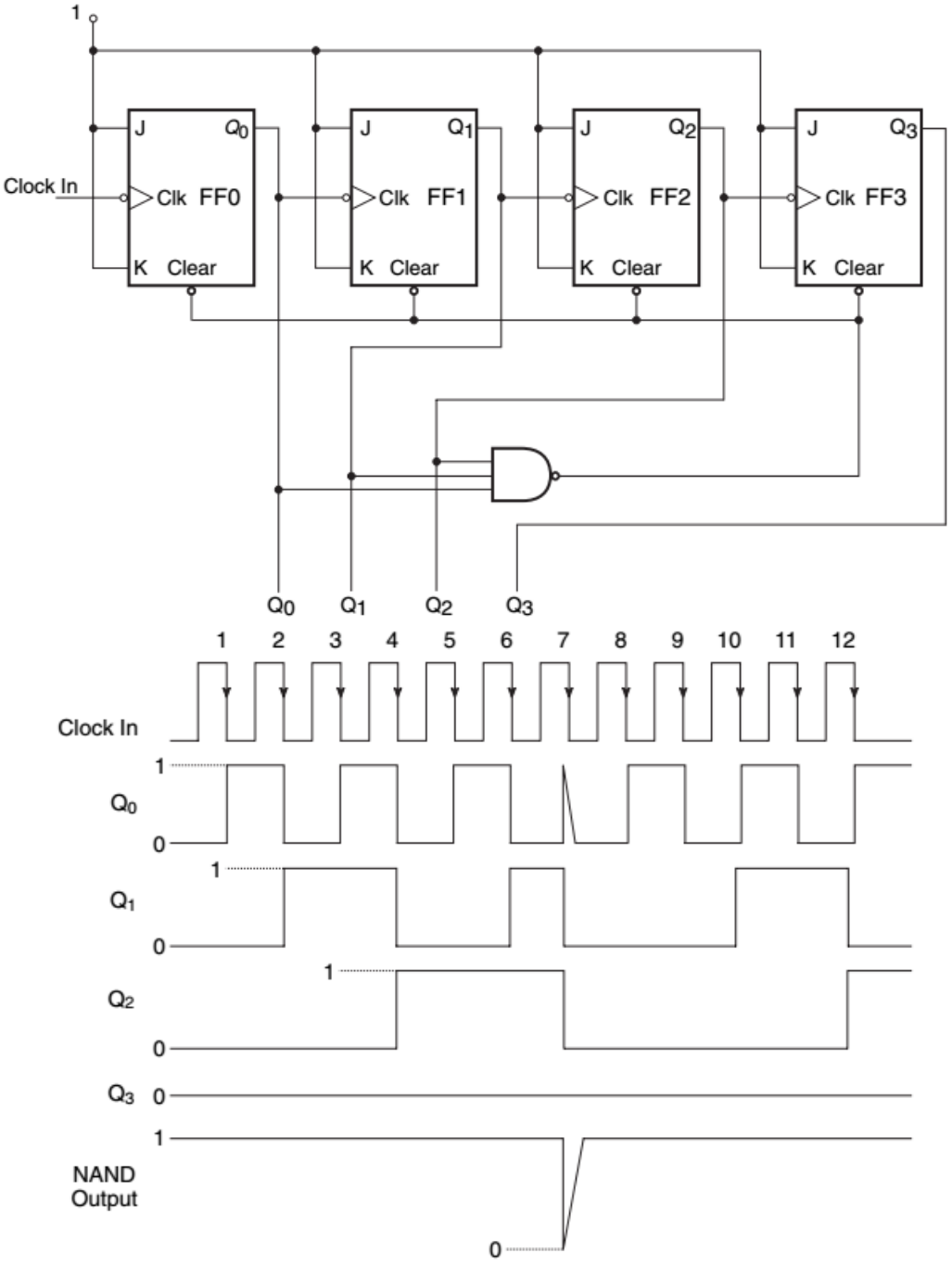
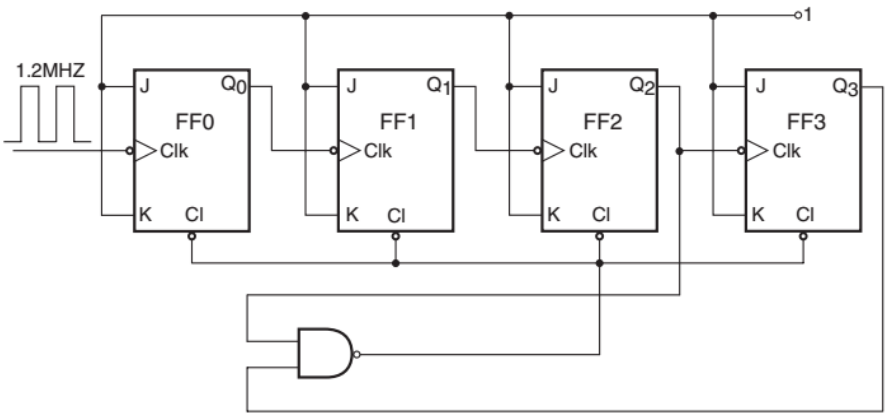


Fig. 3

Exercise 2. Refer to the binary ripple counter of the following figure. Determine the modulus of the counter and also the frequency of the flip-flop Q3 output.



Exercise 3. Design a binary ripple counter that counts 000 and 111 and skips the remaining six states, that is, 001, 010, 011, 100, 101 and 110. Use presentable, clearable negative edge-triggered J-K flip-flops with active LOW PRESET and CLEAR inputs. Also, draw the timing waveforms and determine the frequency of different flip-flop outputs for a given clock frequency, fc.

2. Synchronous Counter

In a synchronous counter, also known as a parallel counter, all the flip-flops in the counter change state at the same time in synchronism with the input clock signal. The clock signal in this case is simultaneously applied to the clock inputs of all the flip-flops. The delay involved in this case is equal to the propagation delay of one flip-flop only, irrespective of the number of flip-flops used to construct the counter. In other words, the delay is independent of the size of the counter.

Since the different flip-flops in a synchronous counter are clocked at the same time, there needs to be additional logic circuitry to ensure that the various flip-flops toggle at the right time. For instance, if we look at the count sequence of a four-bit binary counter shown in Table 2, we find that flip-flop FF0 toggles with every clock pulse, flip-flop FF1 toggles only when the output of FF0 is in the ‘1’ state, flip-flop FF2 toggles only with those clock pulses when the outputs of FF0 and FF1 are both in the logic ‘1’ state and flip-flop FF3 toggles only with those clock pulses when Q0 Q1 and Q2 are all in the logic ‘1’ state. Such logic can be easily implemented with AND gates. Figure 4(a) shows the schematic arrangement of a four-bit synchronous counter. The timing waveforms are shown in Fig. 4(b). The diagram is self-explanatory. As an example, ICs 74162 and 74163 are four-bit synchronous counters, with the former being a decade counter and the latter a binary counter.

Table 2 Count sequence of a four-bit binary counter

Count	Q_3	Q_2	Q_1	Q_0	Count	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	8	1	0	0	0
1	0	0	0	1	9	1	0	0	1
2	0	0	1	0	10	1	0	1	0
3	0	0	1	1	11	1	0	1	1
4	0	1	0	0	12	1	1	0	0
5	0	1	0	1	13	1	1	0	1
6	0	1	1	0	14	1	1	1	0
7	0	1	1	1	15	1	1	1	1

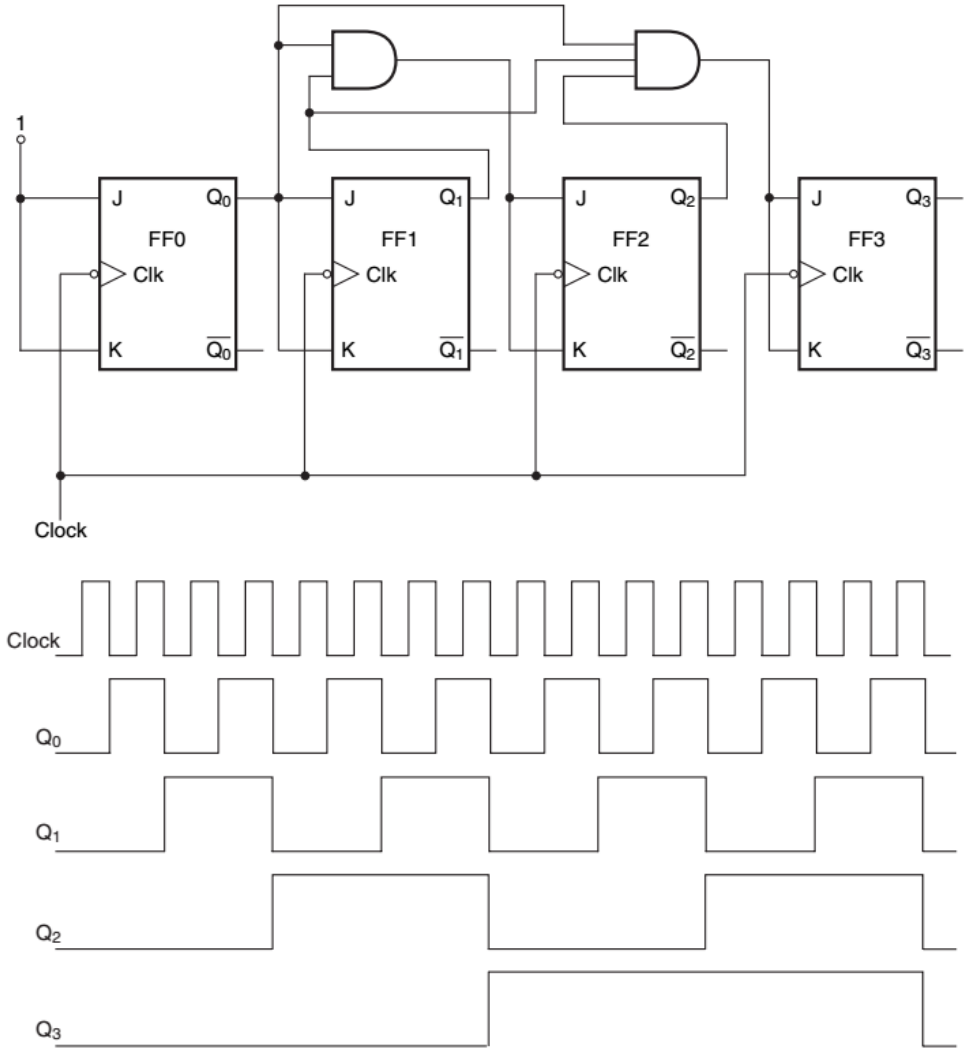


Fig. 4

A synchronous counter that counts in the reverse or downward sequence can be constructed in a similar manner by using complementary outputs of the flip-flops to drive the J and K inputs of the following flip-flops. Refer to the reverse or downward count sequence as given in Table 3. As is evident from the table, FF0 toggles with every clock pulse, FF1 toggles only when Q0 is logic '0', FF2 toggles only when both Q0 and Q1 are in the logic '0' state and FF3 toggles only when Q0, Q1 and Q2 are in the logic '0' state.

Referring to the four-bit synchronous UP counter of Fig. 4(a), if the J and K inputs of flip-flop FF1 are fed from the $\overline{Q}_0$ output instead of the Q_0 output, the inputs to the two-input AND gate are $\overline{Q}_0$ and $\overline{Q}_1$ instead of Q_0 and Q_1, and the inputs to the three-input AND gate are $\overline{Q}_0$, $\overline{Q}_1$ and $\overline{Q}_2$ instead of Q_0, Q_1 and Q_2, we get a counter that counts in reverse order. In that case it becomes a four-bit synchronous DOWN counter.

Table 3 Reverse or downward count sequence synchronous counter.

Count	Q_3	Q_2	Q_1	Q_0	Count	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	8	1	0	0	0
1	1	1	1	1	9	0	1	1	1
2	1	1	1	0	10	0	1	1	0
3	1	1	0	1	11	0	1	0	1
4	1	1	0	0	12	0	1	0	0
5	1	0	1	1	13	0	0	1	1
6	1	0	1	0	14	0	0	1	0
7	1	0	0	1	15	0	0	0	1

3. UP/DOWN Counters

Some counter ICs have separate clock inputs for UP and DOWN counts, while others have a single clock input and an UP/DOWN control pin. The logic status of this control pin decides the counting mode. As an example, ICs 74190 and 74191 are four-bit UP/DOWN counters in the TTL family with a single clock input and an UP/DOWN control pin. While IC 74190 is a BCD decade counter, IC 74191 is a binary counter. Also, ICs 74192 and 74193 are four-bit UP/DOWN counters in the TTL family, with separate clock input terminals for UP and DOWN counts. While IC 74192 is a BCD decade counter, IC 74193 is a binary counter.

Figure 5 shows a three-bit binary UP/DOWN counter. This is only one possible logic arrangement. As we can see, the counter counts upwards when UP control is logic '1' and DOWN

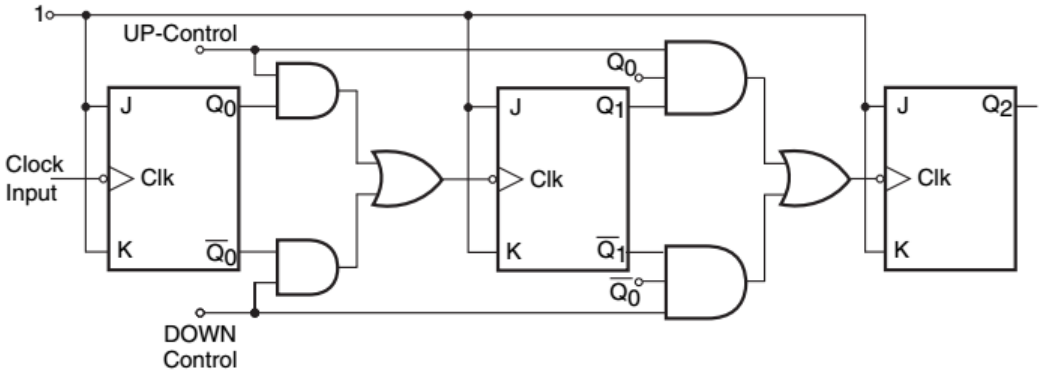


Figure 5 Four-bit UP/DOWN counter

4. Decade and BCD Counters

A *decade counter* is one that goes through 10 unique output combinations and then resets as the clock proceeds further. Since it is an MOD-10 counter, it can be constructed with a

minimum of four flip-flops. A four-bit counter would have 16 states. By skipping any of the six states by using some kind of feedback or some kind of additional logic, we can convert a normal four-bit binary counter into a decade counter. A decade counter does not necessarily count from 0000 to 1001. It could even count as 0000, 0001, 0010, 0101, 0110, 1001, 1010, 1100, 1101, 1111, 0000, … In this count sequence, we have skipped 0011, 0100, 0111, 1000, 1011 and 1110.

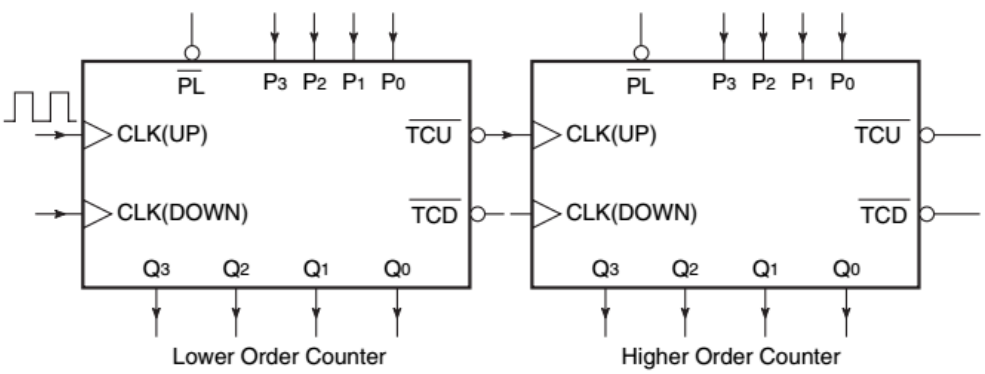
A *BCD counter* is a special case of a decade counter in which the counter counts from 0000 to 1001 and then resets. The output weights of flip-flops in these counters are in accordance with 8421-code. For instance, at the end of the seventh clock pulse, the counter output will be 0111, which is the binary equivalent of decimal 7. In other words, different counter states in this counter are binary equivalents of the decimal numbers 0 to 9. These are different from other decade counters that provide the same count by using some kind of forced feedback to skip six of the natural binary counts.

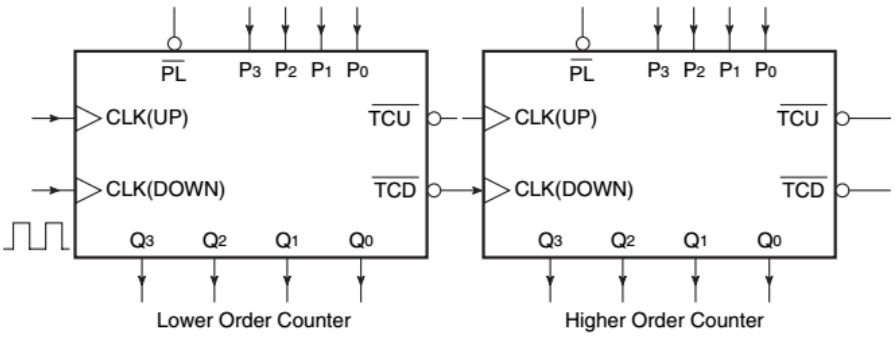
5. Cascading Counters

A cascade arrangement allows us to build counters with a higher modulus than is possible with a single stage. The terminal count outputs allow more than one counter to be connected in a cascade arrangement. In the following paragraphs, we will examine some such cascade arrangements in the case of binary and BCD counters.

5.1. Cascading Binary Counters

In order to construct a multistage UP counter, all counter stages are connected in the count UP mode. The clock is applied to the clock input of a lowest-order counter, the terminal count UP (TCU), also called the carry-out (Co), of this counter is applied to the clock input of the next higher counter stage and the process continues. If it is desired to build a multistage DOWN counter, all counters are wired as DOWN counters, the clock is applied to the clock input of the lowest-order counter and the terminal count DOWN (TCD), also called the borrow-out (Bo, of the lowest-order counter is applied to the clock input of the next higher counter stage. The process continues in the same fashion, with the TCD output of the second stage feeding the clock input of the third stage and so on. The modulus of the multistage counter arrangement equals the product of the moduli of individual stages. Figures 6(a) and (b) respectively show two-stage arrangements of four-bit synchronous UP and DOWN counters respectively.



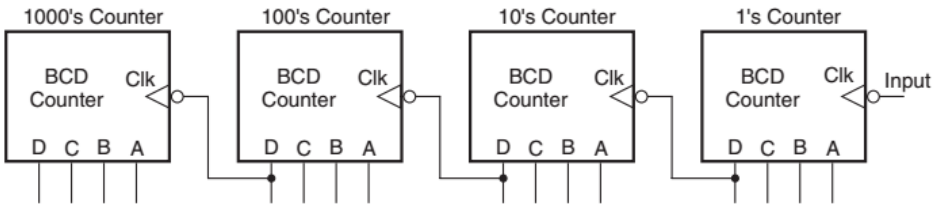


Figures 6

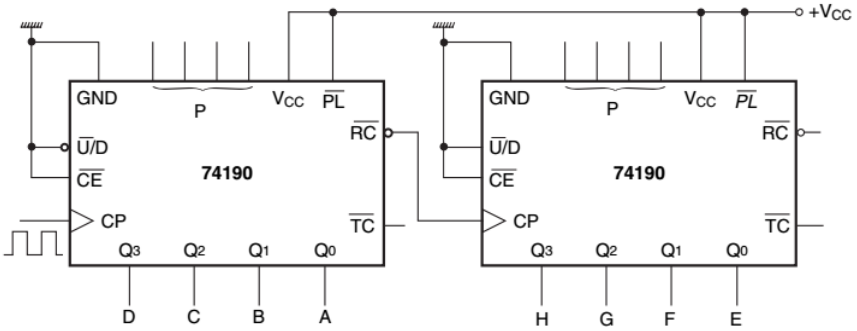
Cascading BCD Counters

BCD counters are used when the application involves the counting of pulses and the result of counting is to be displayed in decimal. A single-stage BCD counter counts from 0000 (decimal equivalent '0') to 1001 (decimal equivalent '9') and thus is capable of counting up to a maximum of nine pulses. The output in a BCD counter is in binary coded decimal (BCD) form. The BCD output needs to be decoded appropriately before it can be displayed.

Figure 7 shows a cascade arrangement of four BCD counter stages. The arrangement works as follows. Initially, all four counters are in the all 0s state. The counter representing the decimal digit of 1's place is clocked by the pulsed signal that needs to be counted. The successive flip-flops are clocked by the MSB of the immediately previous counter stage. The first nine pulses take 1's place counter to 1001. The tenth pulse resets it to 0000, and '1' to '0' transition at the MSB of 1's place counter clocks 10's place counter. 10's place counter gets clocked on every tenth input clock pulse. On the hundredth clock pulse, the MSB of 10's counter makes a '1' to '0' transition which clocks 100's place counter. This counter gets clocked on every successive hundredth input clock pulse. On the thousandth input clock pulse, the MSB of 100's counter makes 'l' to '0' transition for the first time and clocks 1000's place counter. This counter is clocked thereafter on every successive thousandth input clock pulse. With this background, we can always tell the output state of the cascade arrangement. For example, immediately after the 7364th input clock pulse, the state of 1000's, 100's, 10's and 1's BCD counters would respectively be 0111, 0011, 0110 and 0100.



Exercise 4. Figure (a) shows a cascade arrangement of two 74190s. Both the UP/DOWN counters are wired as UP counters. What will be the logic status of outputs designated as A, B, C, D, E, F, G and H after the 34th clock pulse?



II. REGISTERS

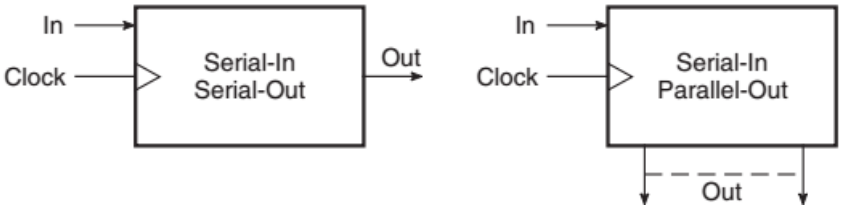
1. Shift Register

A shift register is a digital device used for storage and transfer of data. The data to be stored could be the data appearing at the output of an encoding matrix before they are fed to the main digital system for processing or they might be the data present at the output of a microprocessor before they are fed to the driver circuitry of the output devices. The shift register thus forms an important link between the main digital system and the input/output channels. The shift registers can also be configured to construct some special types of counter that can be used to perform a number of arithmetic operations such as subtraction, multiplication, division, complementation, etc. The basic building block in all shift registers is the flipflop, mainly a D-type flip-flop. Although in many of the commercial shift register ICs their internal circuit diagram might indicate the use of R-S flip-flops, a careful examination will reveal that these R-S flip-flops have been wired as D flip-flops only.

The storage capacity of a shift register equals the total number of bits of digital data it can store, which in turn depends upon the number of flip-flops used to construct the shift register. Since each flip-flop can store one bit of data, the storage capacity of the shift register equals the number of flip-flops used. As an example, the internal architecture of an eight-bit shift register will have a cascade arrangement of eight flip-flops.

Based on the method used to load data onto and read data from shift registers, they are classified as serial-in serial-out (SISO) shift registers, serial-in parallel-out (SIPO) shift registers, parallel-in serial-out (PISO) shift registers and parallel-in parallel-out (PIPO) shift registers.

Figure 8 shows a circuit representation of the above-mentioned four types of shift register.



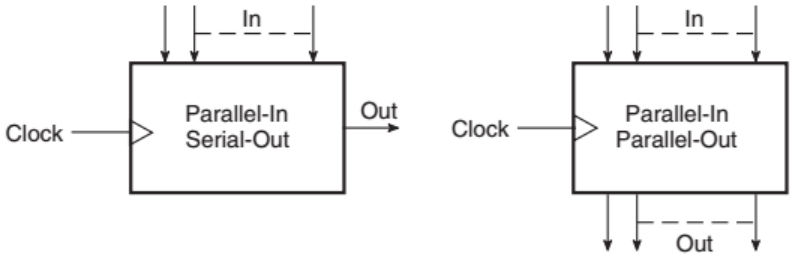


Figure 8 Circuit representation of shift registers

Serial-In Serial-Out Shift Register

Figure 9 shows the basic four-bit serial-in serial-out shift register implemented using D flip-flops. The circuit functions as follows. A reset applied to the CLEAR input of all the flip-flops resets their Q outputs to 0s. Refer to the timing waveforms of Fig. 10. The waveforms shown include the clock pulse train, the waveform representing the data to be loaded onto the shift register and the Q outputs of different flip-flops. The flip-flops shown respond to the LOW-to-HIGH transition of the clock pulses as indicated by their logic symbols. During the first clock transition, the QA output goes from logic '0' to logic '1'.

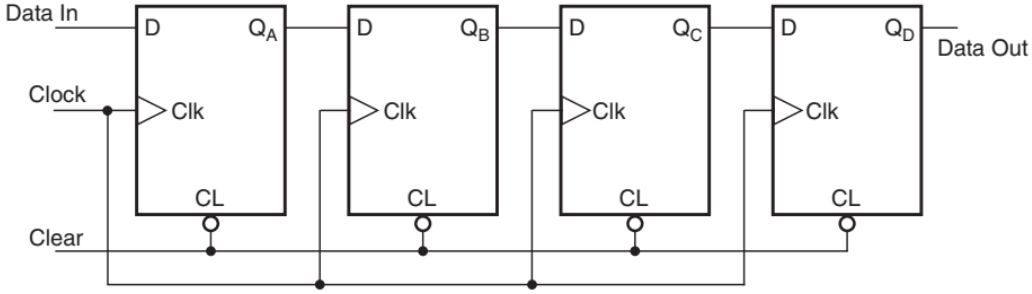
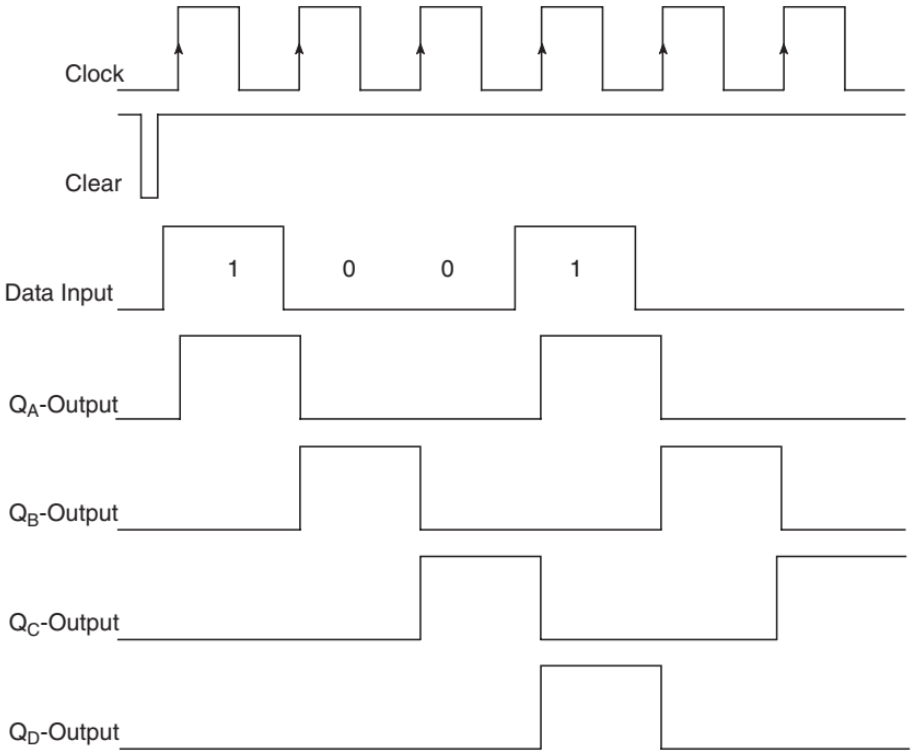


Figure 9 Serial-in, serial-out shift register

The outputs of the other three flip-flops remain in the logic '0' state as their D inputs were in the logic '0' state at the time of clock transition. During the second clock transition, the QA output goes from logic '1' to logic '0' and the QB output goes from logic '0' to logic '1', again in accordance with the logic status of the D inputs at the time of relevant clock transition.

Thus, we have seen that a logic '1' that was present at the data input prior to the occurrence of the first clock transition has reached the QB output at the end of two clock transitions. This bit will reach the QD output at the end of four clock transitions. In general, in a four-bit shift register of the type shown in Fig. 9, a data bit present at the data input terminal at the time of the nth clock transition reaches the QD output at the end of the (n + 4)th clock transition. During the fifth and subsequent clock transitions, data bits continue to shift to the right, and at the end of the eighth clock transition the shift register is again reset to all 0s. Thus, in a four-bit serial-in serial-out shift register, it takes four clock cycles to load the data bits and another four cycles to read the data bits out of the register.



The contents of the register for the first eight clock cycles are summarized in Table 4. We can see that the register is loaded with the four-bit data in four clock cycles, and also that the stored four-bit data are read out in the subsequent four clock cycles. IC 7491 is a popular eight-bit serial-in serial-out shift register.

Serial-In Parallel-Out Shift Register

A serial-in parallel-out shift register is architecturally identical to a serial-in serial-out shift register except that in the case of the former all flip-flop outputs are also brought out on the IC terminals. Figure 11 shows the logic diagram of a typical serial-in parallel-out shift register.

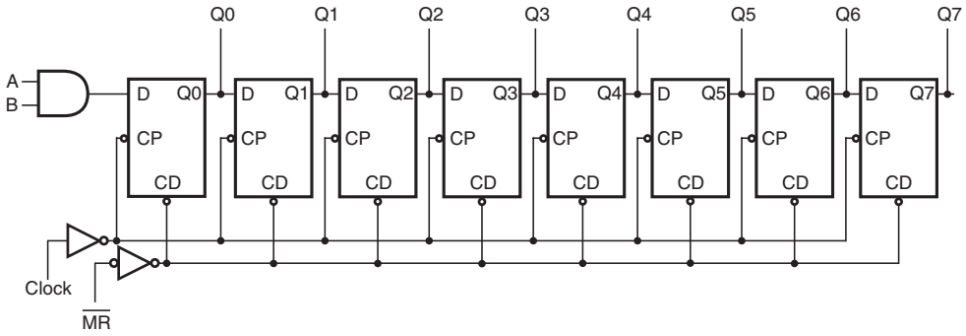


Figure 10 Logic diagram of IC 74164.

Parallel-In Serial-Out Shift Register

We will explain the operation of a parallel-in serial-out shift register with the help of the logic diagram of a practical device available in IC form. Figure 12 shows the logic diagram of one

such shift register. The logic diagram is that of IC 74166, which is an eight-bit parallel/serial-in, serial-out shift register belonging to the TTL family of devices.

The parallel-in or serial-in modes are controlled by a SHIFT/LOAD input. When the SHIFT/LOAD input is held in the logic HIGH state, the serial data input AND gates are enabled and the circuit behaves like a serial-in serial-out shift register. When the SHIFT/LOAD input is held in the logic LOW state, parallel data input AND gates are enabled and data are loaded in parallel, in synchronism with the next clock pulse.

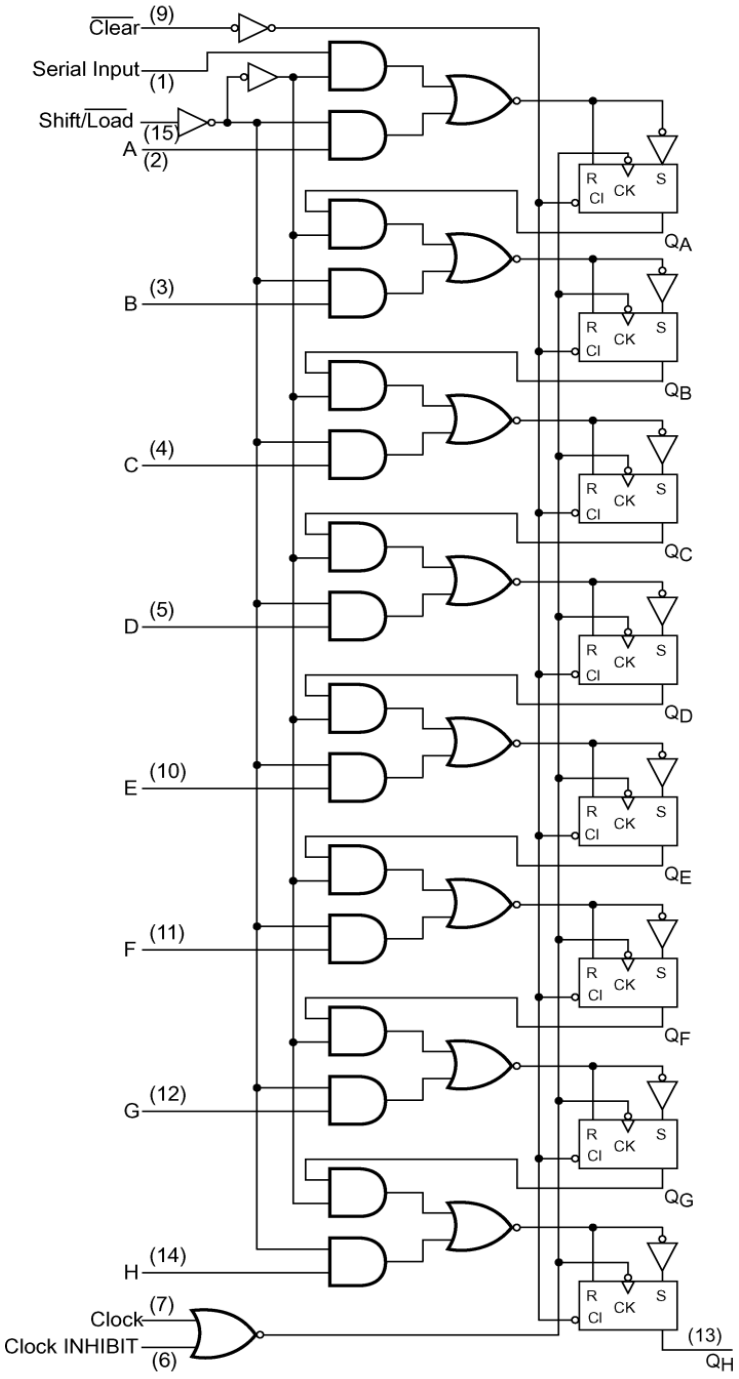


Figure 12 Logic diagram of 74166.

Parallel-In Parallel-Out Shift Register

The hardware of a parallel-in parallel-out shift register is similar to that of a parallel-in serial-out shift register. If in a parallel-in serial-out shift register the outputs of different flip-flops are brought out, it becomes a parallel-in parallel-out shift register. In fact, the logic diagram of a parallel-in parallel-out shift register is similar to that of a parallel-in serial-out shift register. As an example, IC 74199 is an eight-bit parallel-in parallel-out shift register. Figure 13 shows its logic diagram.

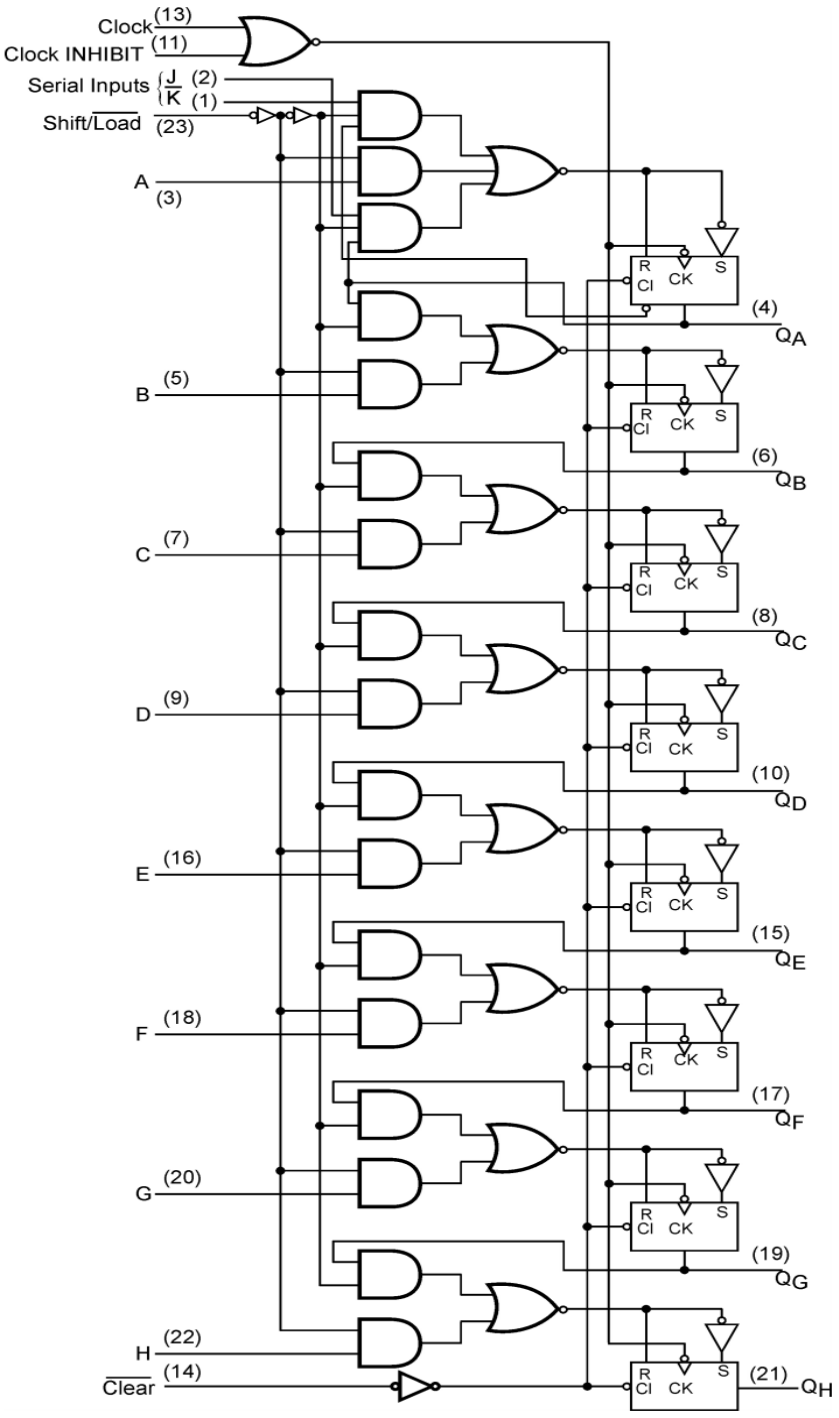


Figure 13 Logic diagram of IC 74199.

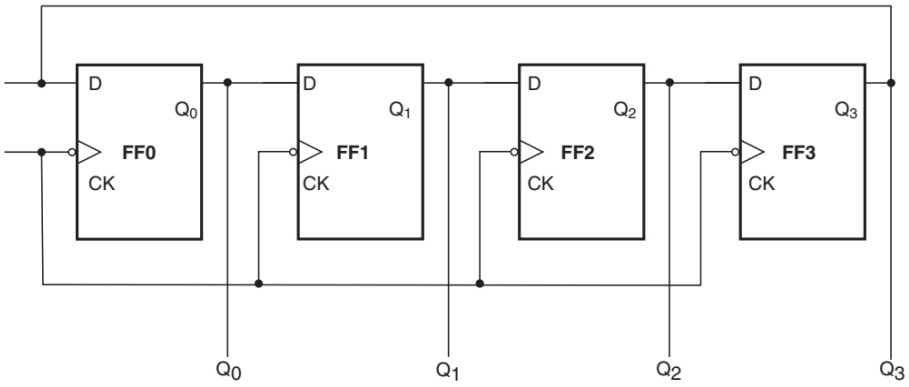
Shift Register Counters

We have seen that both counters and shift registers are some kinds of cascade arrangement of flip-flops. A shift register, unlike a counter, has no specified sequence of states. However, if the serial output of the shift register is fed back to the serial input, we do get a circuit that exhibits a specified sequence of states. The resulting circuits are known as *shift register counters*. Depending upon the nature of the feedback, we have two types of shift register counter, namely the *ring counter* and the *shift counter*, also called the *Johnson counter*. These are briefly described in the following paragraphs.

Ring Counter

A *ring counter* is obtained from a shift register by directly feeding back the true output of the output flip-flop to the data input terminal of the input flip-flop. If D flip-flops are being used to construct the shift register, the ring counter, also called a circulating register, can be constructed by feeding back the Q output of the output flip-flop back to the D input of the input flip-flop. If *J-K* flip-flops are being used, the Q and Q outputs of the output flip-flop are respectively fed back to the J and K inputs of the input flip-flop.

Figure 14 shows the logic diagram of a four-bit ring counter. Let us assume that flip-flop FF0 is initially set to the logic '1' state and all other flip-flops are reset to the logic '0' state. The counter output is therefore 1000. With the first clock pulse, this '1' gets shifted to the second flip-flop output and the counter output becomes 0100. Similarly, with the second and third clock pulses, the counter output will become 0010 and 0001. With the fourth clock pulse, the counter output will again become 1000. The count cycle repeats in the subsequent clock pulses. Circulating registers of this type find wide application in the control section of microprocessor-based systems where one event should follow the other.



Shift Counter

A shift counter on the other hand is constructed by having an inverse feedback in a shift register. For instance, if we connect the Q output of the output flip-flop back to the K input of the input flip-flop and the Q output of the output flip-flop to the J input of the input flip-flop in a serial shift register, the result is a shift counter, also called a Johnson counter. If the shift register employs D flip-flops, the Q output of the output flip-flop is fed back to the D input of the input flip-flop. If R-S flip-flops are used, the Q output goes to the R input and the Q

output is connected to the S input. Figure 15 shows the logic diagram of a basic four-bit shift counter.

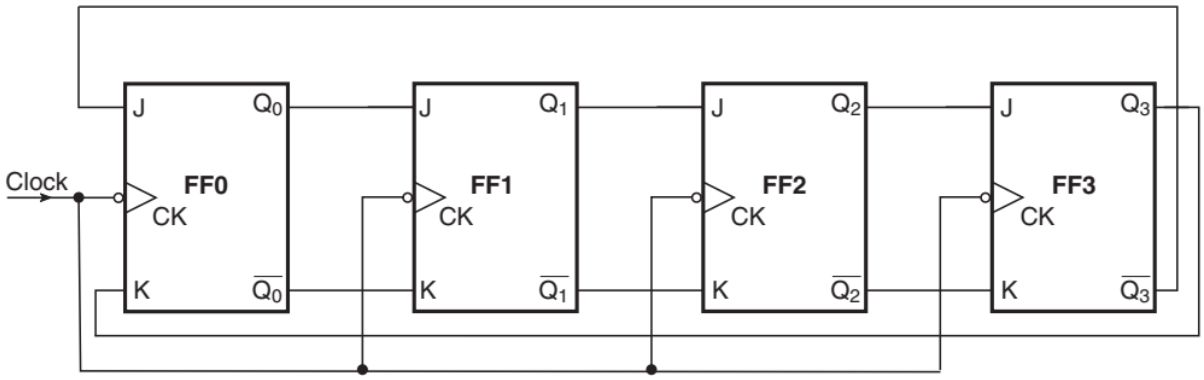


Figure 15 Four-bit shift counter.

Let us assume that the counter is initially reset to all 0s. With the first clock cycle, the outputs will become 1000. With the second, third and fourth clock cycles, the outputs will respectively be 1100, 1110 and 1111. The fifth clock cycle will change the counter output to 0111. The sixth, seventh and eighth clock pulses successively change the outputs to 0011, 0001 and 0000. Thus, one count cycle is completed in eight cycles. Figure 16 shows the timing waveforms. Different output waveforms are identical except for the fact that they are shifted from the immediately preceding one by one clock cycle. Also, the time period of each of these waveforms is 8 times the period of the clock waveform. That is, this shift counter behaves as a divide-by-8 circuit.

In general, a shift counter comprising n flip-flops acts as a divide-by-2n circuit. Shift counters can be used very conveniently to construct counters having a modulus other than the integral power of 2.

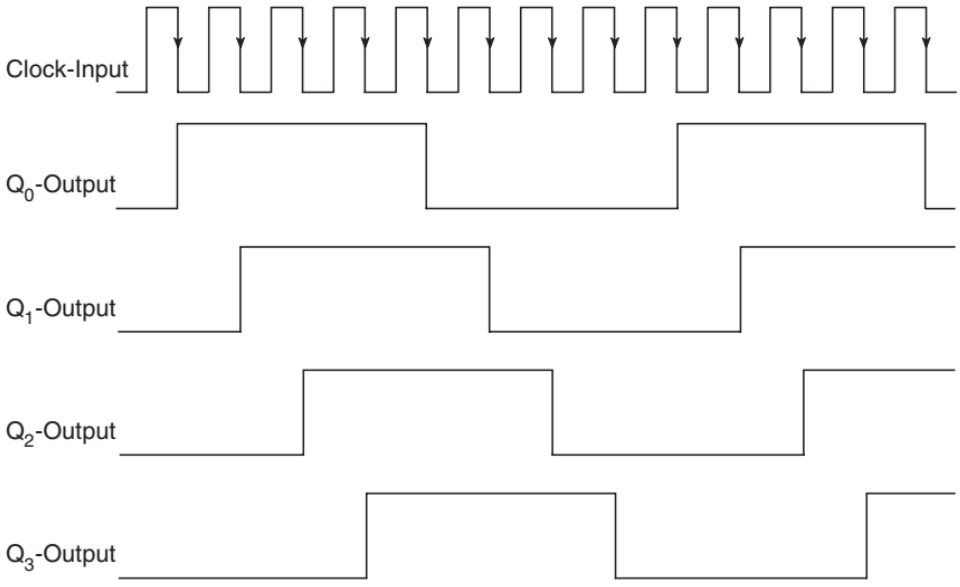


Figure 16 Timing waveforms of the shift counter.

Exercise 4 Determine the number of flip-flops required to construct (a) a MOD-10 ring counter and (b) a MOD-10Johnson counter. Also, write the count sequence in the two cases.

Questions

1. Differentiate between:

(a) asynchronous and synchronous counters;
(b) UP, DOWN and UP/DOWN counters;
(c) presettable and clearable counters;
(d) BCD and decade counters.

2. Indicate the difference between the counting sequences of:

(a) a four-bit binary UP counter and a four-bit binary DOWN counter;
(b) a four-bit ring counter and a four-bit Johnson counter.

3. Briefly describe:

(a) how the architecture of an asynchronous UP counter differs from that of a DOWN counter;
(b) how the architecture of a ring counter differs from that of a shift counter.

4. Indicate the type of shift register:

(a) into which a complete binary number can be loaded in one operation and then shifted out one-bit at a time;
(b) into which data can be entered only one bit at a time but have all data bits available as outputs;
(c) in which we have access to only the leftmost or rightmost flip-flop.

5. Give at least one IC type number for:

(a) a four-bit binary ripple counter;
(b) a four-bit synchronous counter;
(c) an eight-bit serial-in serial-out shift register;