# Chapter 5

# Synchronous Sequential Logic

## 5.1 INTRODUCTION

Hand-held devices, cell phones, navigation receivers, personal computers, digital cameras, personal media players, and virtually all electronic consumer products have the ability to send, receive, store, retrieve, and process information represented in a binary format. The technology enabling and supporting these devices is critically dependent on electronic components that can store information, i.e., have memory. This chapter examines the operation and control of these devices and their use in circuits and enables you to better understand what is happening in these devices when you interact with them. The digital circuits considered thus far have been combinational—their output depends only and immediately on their inputs—they have no memory, i.e., dependence on past values of their inputs. Sequential circuits, however, act as storage elements and have memory. They can store, retain, and then retrieve information when needed at a later time. Our treatment will distinguish sequential logic from combinational logic.

## 5.2 SEQUENTIAL CIRCUITS

A block diagram of a sequential circuit is shown in Fig. 5.1. It consists of a combinational circuit to which storage elements are connected to form a feedback path. The storage elements are devices capable of storing binary information. The binary information stored in these elements at any given time defines the *state* of the sequential circuit at that time. The sequential circuit receives binary information from external inputs that, together with the present state of the storage elements, determine the binary value of the outputs. These external inputs also determine the condition for changing the state

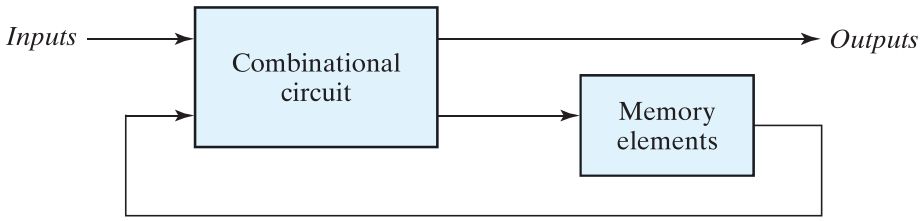
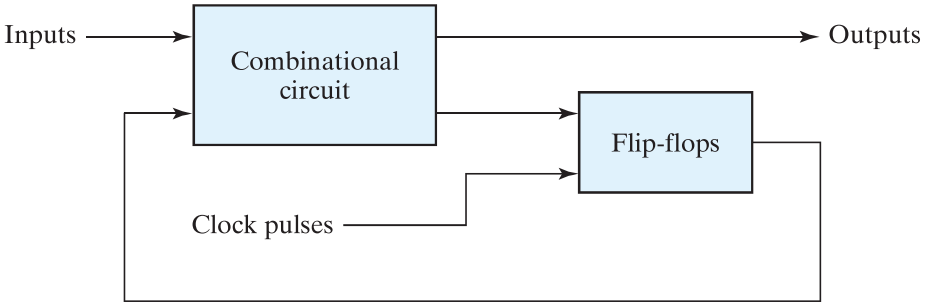



**FIGURE 5.1**
**Block diagram of sequential circuit**

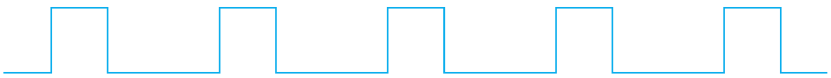
in the storage elements. The block diagram demonstrates that the outputs in a sequential circuit are a function not only of the inputs, but also of the present state of the storage elements. The next state of the storage elements is also a function of external inputs and the present state. Thus, **a sequential circuit is specified by a time sequence of inputs, outputs, and internal states**. In contrast, the outputs of combinational logic depend only on the present values of the inputs.

There are two main types of sequential circuits, and their classification is a function of the timing of their signals. A *synchronous* sequential circuit is a system whose behavior can be defined from the knowledge of its signals at discrete instants of time. The behavior of an *asynchronous* sequential circuit depends upon the input signals at any instant of time *and* the order in which the inputs change. The storage elements commonly used in asynchronous sequential circuits are time-delay devices. The storage capability of a time-delay device varies with the time it takes for the signal to propagate through the device. In practice, the internal propagation delay of logic gates is of sufficient duration to produce the needed delay, so that actual delay units may not be necessary. In gate-type asynchronous systems, the storage elements consist of logic gates whose propagation delay provides the required storage. Thus, an asynchronous sequential circuit may be regarded as a combinational circuit with feedback. Because of the feedback among logic gates, an asynchronous sequential circuit may become unstable at times. The instability problem imposes many difficulties on the designer. These circuits will not be covered in this text.

A synchronous sequential circuit employs signals that affect the storage elements at only discrete instants of time. Synchronization is achieved by a timing device called a *clock generator*, which provides a clock signal having the form of a periodic train of *clock pulses*. The clock signal is commonly denoted by the identifiers *clock* and *clk*. The clock pulses are distributed throughout the system in such a way that storage elements are affected only with the arrival of each pulse. In practice, the clock pulses determine *when* computational activity will occur within the circuit, and other signals (external inputs and otherwise) determine *what* changes will take place affecting the storage elements and the outputs. For example, a circuit that is to add and store two binary numbers would compute their sum from the values of the numbers and store the sum at the occurrence of a clock pulse. Synchronous sequential circuits that use clock pulses to control storage elements are called *clocked sequential circuits* and are the type most frequently encountered in practice. They are called *synchronous circuits* because the activity within the circuit and the resulting updating of stored values is synchronized to the occurrence of




(a) Block diagram



(b) Timing diagram of clock pulses

**FIGURE 5.2**
**Synchronous clocked sequential circuit**

clock pulses. The design of synchronous circuits is feasible because they seldom manifest instability problems and their timing is easily broken down into independent discrete steps, each of which can be considered separately.

The storage elements (memory) used in clocked sequential circuits are called *flip-flops*. A flip-flop is a binary storage device capable of storing one bit of information. In a stable state, the output of a flip-flop is either 0 or 1. A sequential circuit may use many flip-flops to store as many bits as necessary. The block diagram of a synchronous clocked sequential circuit is shown in Fig. 5.2. The *outputs* are formed by a combinational logic function of the inputs to the circuit or the values stored in the flip-flops (or both). The value that is stored in a flip-flop when the clock pulse occurs is also determined by the inputs to the circuit or the values presently stored in the flip-flop (or both). The new value is stored (i.e., the flip-flop is updated) when a pulse of the clock signal occurs. Prior to the occurrence of the clock pulse, the combinational logic forming the next value of the flip-flop must have reached a stable value. Consequently, the speed at which the combinational logic circuits operate is critical. If the clock (synchronizing) pulses arrive at a regular interval, as shown in the timing diagram in Fig. 5.2, the combinational logic must respond to a change in the state of the flip-flop in time to be updated before the next pulse arrives. Propagation delays play an important role in determining the minimum interval between clock pulses that will allow the circuit to operate correctly. A change in state of the flip-flops is initiated only by a clock pulse transition—for example, when the value of the clock signals changes from 0 to 1. When a clock pulse is not active, the feedback loop between the value stored in the flip-flop and the value formed at the input to the flip-flop is effectively broken because the flip-flop outputs cannot change even if the outputs of the combinational circuit driving their inputs change in value. Thus, the transition from one state to the next occurs only at predetermined intervals dictated by the clock pulses.

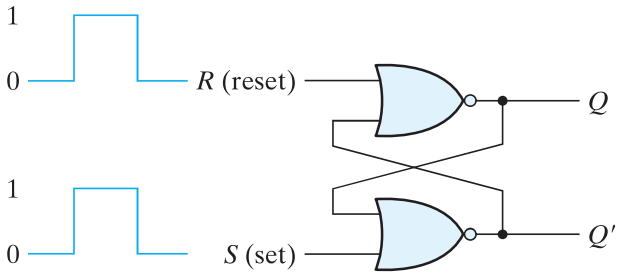
## 5.3 STORAGE ELEMENTS: LATCHES

A storage element in a digital circuit can maintain a binary state indefinitely (as long as power is delivered to the circuit), until directed by an input signal to switch states. The major differences among various types of storage elements are in the number of inputs they possess and in the manner in which the inputs affect the binary state. *Storage elements that operate with signal levels (rather than signal transitions) are referred to as latches*; *those controlled by a clock transition are flip-flops*. Latches are said to be level sensitive devices; flip-flops are edge-sensitive devices. The two types of storage elements are related because latches are the basic circuits from which all flip-flops are constructed. Although latches are useful for storing binary information and for the design of asynchronous sequential circuits, they are not practical for use as storage elements in synchronous sequential circuits. Because they are the building blocks of flip-flops, however, we will consider the fundamental storage mechanism used in latches before considering flip-flops in the next section.

### *SR* Latch

The *SR* latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates, and two inputs labeled *S* for set and *R* for reset. The *SR* latch constructed with two cross-coupled NOR gates is shown in Fig. 5.3. The latch has two useful states. When output $Q = 1$ and $Q' = 0$, the latch is said to be in the *set state*. When $Q = 0$ and $Q' = 1$, it is in the *reset state*. Outputs $Q$ and $Q'$ are normally the complement of each other. However, when both inputs are equal to 1 at the same time, a condition in which both outputs are equal to 0 (rather than be mutually complementary) occurs. If both inputs are then switched to 0 simultaneously, the device will enter an unpredictable or undefined state or a metastable state. Consequently, in practical applications, setting both inputs to 1 is forbidden.

Under normal conditions, both inputs of the latch remain at 0 unless the state has to be changed. The application of a momentary 1 to the *S* input causes the latch to go to the set state. The *S* input must go back to 0 before any other changes take place, in order to avoid the occurrence of an undefined next state that results from the forbidden input condition. As shown in the function table of Fig. 5.3(b), two input conditions cause the circuit to be in

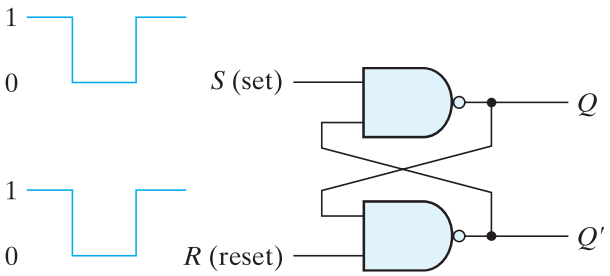



(a) Logic diagram

| $S$ | $R$ | $Q$ | $Q'$ | |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 0 | (after $S = 1, R = 0$) |
| 0 | 1 | 0 | 1 | |
| 0 | 0 | 0 | 1 | (after $S = 0, R = 1$) |
| 1 | 1 | 0 | 0 | (forbidden) |

(b) Function table

**FIGURE 5.3**
***SR* latch with NOR gates**




(a) Logic diagram

| $S$ | $R$ | $Q$ | $Q'$ | |
|---|---|---|---|---|
| 1 | 0 | 0 | 1 | |
| 1 | 1 | 0 | 1 | (after $S = 1, R = 0$) |
| 0 | 1 | 1 | 0 | |
| 1 | 1 | 1 | 0 | (after $S = 0, R = 1$) |
| 0 | 0 | 1 | 1 | (forbidden) |

(b) Function table

**FIGURE 5.4**
***SR* latch with NAND gates**

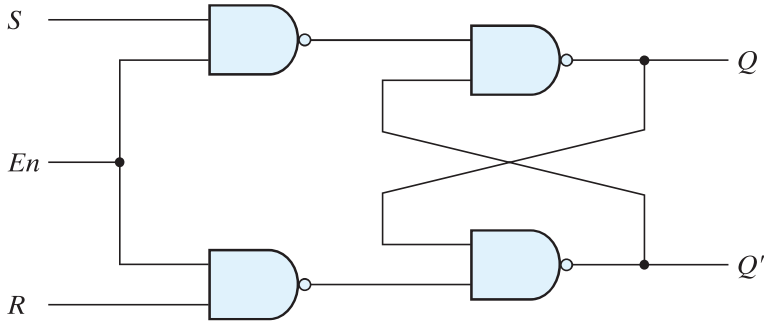
the set state. The first condition ($S = 1, R = 0$) is the action that must be taken by input $S$ to bring the circuit to the set state. Removing the active input from $S$ leaves the circuit in the same state. After both inputs return to 0, it is then possible to shift to the reset state by momentary applying a 1 to the $R$ input. The 1 can then be removed from $R$, whereupon the circuit remains in the reset state. Thus, when both inputs $S$ and $R$ are equal to 0, the latch can be in either the set or the reset state, depending on which input was most recently a 1.

If a 1 is applied to both the $S$ and $R$ inputs of the latch, both outputs go to 0. This action produces an undefined next state, because the state that results from the input transitions depends on the order in which they return to 0. It also violates the requirement that outputs be the complement of each other. In normal operation, this condition is avoided by making sure that 1's are not applied to both inputs simultaneously.

The $SR$ latch with two cross-coupled NAND gates is shown in Fig. 5.4. It operates with both inputs normally at 1, unless the state of the latch has to be changed. The application of 0 to the $S$ input causes output $Q$ to go to 1, putting the latch in the set state. When the $S$ input goes back to 1, the circuit remains in the set state. After both inputs go back to 1, we are allowed to change the state of the latch by placing a 0 in the $R$ input. This action causes the circuit to go to the reset state and stay there even after both inputs return to 1. The condition that is forbidden for the NAND latch is both inputs being equal to 0 at the same time, an input combination that should be avoided.

In comparing the NAND with the NOR latch, note that the input signals for the NAND require the complement of those values used for the NOR latch. Because the NAND latch requires a 0 signal to change its state, it is sometimes referred to as an $S'R'$ latch. The primes (or, sometimes, bars over the letters) designate the fact that the inputs must be in their complement form to activate the circuit.

The operation of the basic $SR$ latch can be modified by providing an additional input signal that determines (controls) *when* the state of the latch can be changed by determining whether $S$ and $R$ (or $S'$ and $R'$) can affect the circuit. An $SR$ latch with a control input is shown in Fig. 5.5. It consists of the basic $SR$ latch and two additional NAND gates. The control input $En$ acts as an *enable* signal for the other two inputs. **The outputs of the NAND gates stay at the logic-1 level as long as the enable signal remains at 0.** This is the quiescent condition for the $SR$ latch. When the enable input goes to 1, information from the $S$ or $R$ input is allowed to affect the latch. The set state is reached with $S = 1$, $R = 0$, and $En = 1$




(a) Logic diagram

| *En* | *S* | *R* | Next state of *Q* |
|---|---|---|---|
| 0 | X | X | No change |
| 1 | 0 | 0 | No change |
| 1 | 0 | 1 | $Q = 0$; reset state |
| 1 | 1 | 0 | $Q = 1$; set state |
| 1 | 1 | 1 | Indeterminate |

(b) Function table

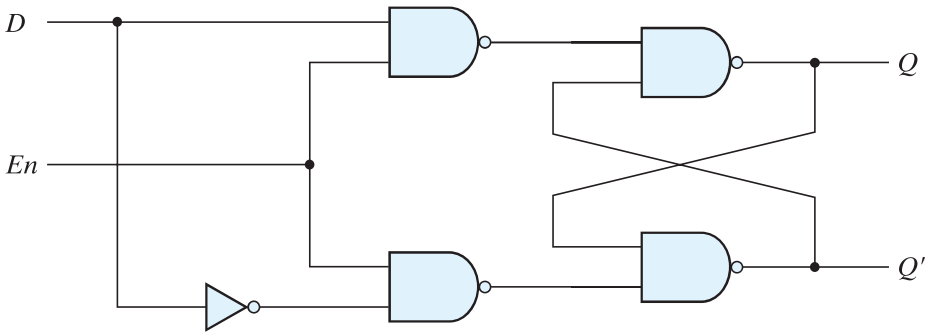
**FIGURE 5.5**
***SR* latch with control input**

(active-high enabled). To change to the reset state, the inputs must be $S = 0$, $R = 1$, and $En = 1$. In either case, when $En$ returns to 0, the circuit remains in its current state. The control input disables the circuit by applying 0 to $En$, so that the state of the output does not change regardless of the values of $S$ and $R$. Moreover, when $En = 1$ and both the $S$ and $R$ inputs are equal to 0, the state of the circuit does not change. These conditions are listed in the function table accompanying the diagram.

An indeterminate condition occurs when all three inputs are equal to 1. This condition places 0's on both inputs of the basic $SR$ latch, which puts it in the undefined state. When the enable input goes back to 0, one cannot conclusively determine the next state, because it depends on whether the $S$ or $R$ input goes to 0 first. This indeterminate condition makes this circuit difficult to manage, and it is seldom used in practice. Nevertheless, the $SR$ latch is an important circuit because other useful latches and flip-flops are constructed from it.

## *D* Latch (Transparent Latch)

One way to eliminate the undesirable condition of the indeterminate state in the $SR$ latch is to ensure that inputs $S$ and $R$ are never equal to 1 at the same time. This is done in the $D$ latch, shown in Fig. 5.6. This latch has only two inputs: $D$ (data) and




(a) Logic diagram

| *En* | *D* | Next state of *Q* |
|---|---|---|
| 0 | X | No change |
| 1 | 0 | $Q = 0$; reset state |
| 1 | 1 | $Q = 1$; set state |

(b) Function table

**FIGURE 5.6**
***D* latch**

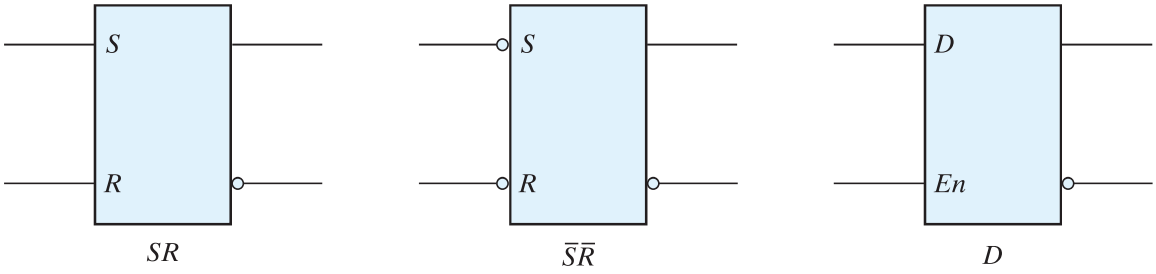



**FIGURE 5.7**
**Graphic symbols for latches**

*En* (enable). The *D* input goes directly to the *S* input, and its complement is applied to the *R* input. As long as the enable input is at 0, the cross-coupled *SR* latch has both inputs at the 1 level and the circuit cannot change state regardless of the value of *D*. The *D* input is sampled when $En = 1$. If $D = 1$, the *Q* output goes to 1, placing the circuit in the set state. If $D = 0$, output *Q* goes to 0, placing the circuit in the reset state.

The *D* latch receives that designation from its ability to hold *data* in its internal storage. It is suited for use as a temporary storage for binary information between a unit and its environment. The binary information present at the data input of the *D* latch is transferred to the *Q* output when the enable input is asserted. The output follows changes in the data input as long as the enable input is asserted. This situation provides a path from input *D* to the output, and for this reason, the circuit is often called a *transparent* latch. When the enable input signal is de-asserted, the binary information that was present at the data input at the time the transition occurred is retained (i.e., stored) at the *Q* output until the enable input is asserted again. Note that an inverter could be placed at the enable input. Then, depending on the physical circuit, the external enabling signal will be a value of 0 (active low) or 1 (active high).

The graphic symbols for the various latches are shown in Fig. 5.7. A latch is designated by a rectangular block with inputs on the left and outputs on the right. One output designates the normal output, and the other (with the bubble designation) designates the complement output. The graphic symbol for the *SR* latch has inputs *S* and *R* indicated inside the block. In the case of a NAND gate latch, bubbles are added to the inputs to indicate that setting and resetting occur with a logic-0 signal. The graphic symbol for the *D* latch has inputs *D* and *En* indicated inside the block.

## 5.4 STORAGE ELEMENTS: FLIP-FLOPS

The state of a latch or flip-flop is switched by a change in the control input. This momentary change is called a *trigger*, and the transition it causes is said to trigger the flip-flop. The *D* latch with pulses in its control input is essentially a flip-flop that is triggered every time the pulse goes to the logic-1 level. As long as the pulse input remains at this level, any changes in the data input will change the output and the state of the latch.

As seen from the block diagram of Fig. 5.2, a sequential circuit has a feedback path from the outputs of the flip-flops to the input of the combinational circuit. Consequently, the inputs of the flip-flops are derived in part from the outputs of the same and other flip-flops. When latches are used for the storage elements, a serious difficulty arises. The state transitions of the latches start as soon as the clock pulse changes to the logic-1 level. The new state of a latch appears at the output while the pulse is still active. This output is connected to the inputs of the latches through the combinational circuit. If the inputs applied to the latches change while the clock pulse is still at the logic-1 level, the latches will respond to new values and a new output state may occur. The result is an unpredictable situation, since the state of the latches may keep changing for as long as the clock pulse stays at the active level. Because of this unreliable operation, the output of a latch cannot be applied directly or through combinational logic to the input of the same or another latch when all the latches are triggered by a common clock source.

Flip-flop circuits are constructed in such a way as to make them operate properly when they are part of a sequential circuit that employs a common clock. The problem with the latch is that it responds to a change in the *level* of a clock pulse. As shown in Fig. 5.8(a), a positive level response in the enable input allows changes in the output when the *D* input changes while the clock pulse stays at logic 1. The key to the proper operation of a flip-flop is to trigger it only during a signal *transition*. This can be accomplished by eliminating the feedback path that is inherent in the operation of the sequential circuit using latches. A clock pulse goes through two transitions: from 0 to 1 and the return from 1 to 0. As shown in Fig. 5.8, the positive transition is defined as the positive edge and the negative transition as the negative edge. There are two ways that a latch can be modified to form a flip-flop. One way is to employ two latches in a special configuration that isolates the output of the flip-flop and prevents it from being affected while the input to the flip-flop is changing. Another way is to produce a flip-flop that

(a) Response to positive level

(b) Positive-edge response

(c) Negative-edge response

**FIGURE 5.8**
**Clock response in latch and flip-flop**

triggers only during a signal transition (from 0 to 1 or from 1 to 0) of the synchronizing signal (clock) and is disabled during the rest of the clock pulse. We will now proceed to show the implementation of both types of flip-flops.

## Edge-Triggered *D* Flip-Flop

The construction of a *D* flip-flop with two *D* latches and an inverter is shown in Fig. 5.9. The first latch is called the master and the second the slave. The circuit samples the *D* input and changes its output *Q* only at the negative edge of the synchronizing or controlling clock (designated as *Clk*). When the clock is 0, the output of the inverter is 1. The slave latch is enabled, and its output *Q* is equal to the master output *Y*. The master latch is disabled because $Clk = 0$. When the input pulse changes to the logic-1 level, the data from the external *D* input are transferred to the master. The slave, however, is disabled as long as the clock remains at the 1 level, because its *enable* input is equal to 0. Any change in the input changes the master output at *Y*, but cannot affect the slave output. When the clock pulse returns to 0, the master is disabled and is isolated from the *D* input. At the same time, the slave is enabled and the value of *Y* is transferred to the output of the flip-flop at *Q*. Thus, *a change in the output of the flip-flop can be triggered only by and during the transition of the clock from 1* to 0.

The behavior of the master–slave flip-flop just described dictates that (1) the output may change only once, (2) a change in the output is triggered by the negative edge of the clock, and (3) the change may occur only during the clock's negative level. The value that is produced at the output of the flip-flop is the value that was *stored in the master stage immediately before the negative edge occurred*. It is also possible to design the circuit so that the flip-flop output changes on the positive edge of the clock. This happens in a flip-flop that has an additional inverter between the *Clk* terminal and the junction between the other inverter and input *En* of the master latch. Such a flip-flop is triggered with a negative pulse, so that the negative edge of the clock affects the master and the positive edge affects the slave and the output terminal.

Another construction of an edge-triggered *D* flip-flop uses three *SR* latches as shown in Fig. 5.10. Two latches respond to the external *D* (data) and *Clk* (clock) inputs. The third latch provides the outputs for the flip-flop. The *S* and *R* inputs of the output latch

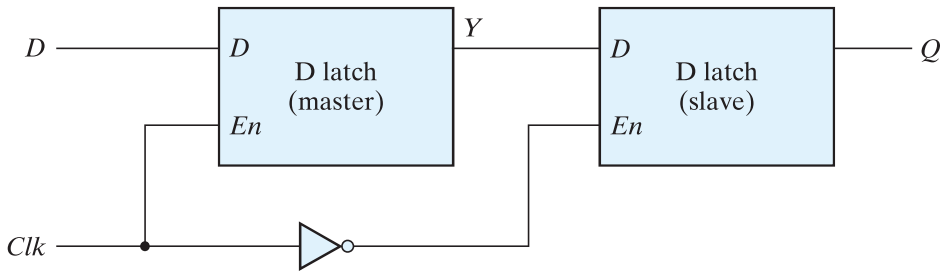



**FIGURE 5.9**
**Master–slave *D* flip-flop**

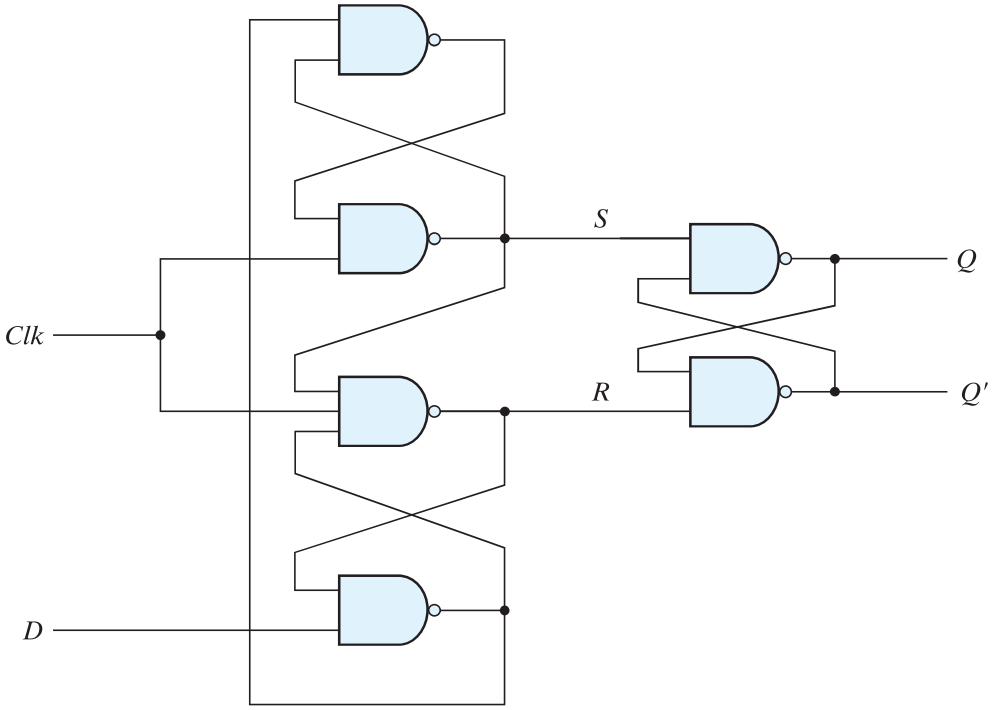
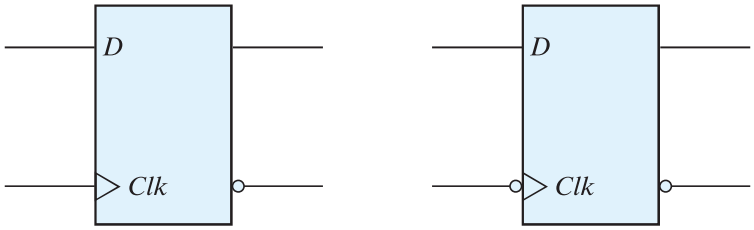



**FIGURE 5.10**
*D*-type positive-edge-triggered flip-flop

are maintained at the logic-1 level when $Clk = 0$. This causes the output to remain in its present state. Input $D$ may be equal to 0 or 1. If $D = 0$ when $Clk$ becomes 1, $R$ changes to 0. This causes the flip-flop to go to the reset state, making $Q = 0$. If there is a change in the $D$ input while $Clk = 1$, terminal $R$ remains at 0 because $Q$ is 0. Thus, the flip-flop is locked out and is unresponsive to further changes in the input. When the clock returns to 0, $R$ goes to 1, placing the output latch in the quiescent condition without changing the output. Similarly, if $D = 1$ when $Clk$ goes from 0 to 1, $S$ changes to 0. This causes the circuit to go to the set state, making $Q = 1$. Any change in $D$ while $Clk = 1$ does not affect the output.

In sum, when the input clock in the positive-edge-triggered flip-flop makes a positive transition, the value of $D$ is transferred to $Q$. A negative transition of the clock (i.e., from 1 to 0) does not affect the output, nor is the output affected by changes in $D$ when $Clk$ is in the steady logic-1 level or the logic-0 level. Hence, this type of flip-flop responds to the transition from 0 to 1 and nothing else.

The timing of the response of a flip-flop to input data and to the clock must be taken into consideration when one is using edge-triggered flip-flops. There is a minimum time called the *setup time* during which the $D$ input must be maintained at a constant value prior to the occurrence of the clock transition. Similarly, there is a minimum time called the *hold time* during which the $D$ input must not change after the application of the positive transition of the clock. The propagation delay time of the flip-flop is defined as the interval between the trigger edge and the stabilization of the output to a new state. These and other parameters are specified in manufacturers' data books for specific logic families.




(a) Positive-edge (a) Negative-edge

**FIGURE 5.11**
**Graphic symbol for edge-triggered *D* flip-flop**

The graphic symbol for the edge-triggered *D* flip-flop is shown in Fig. 5.11. It is similar to the symbol used for the *D* latch, except for the arrowhead-like symbol in front of the letter *Clk*, designating a *dynamic* input. The *dynamic indicator* (>) denotes the fact that the flip-flop responds to the edge transition of the clock. A bubble outside the block adjacent to the dynamic indicator designates a negative edge for triggering the circuit. The absence of a bubble designates a positive-edge response.

## Other Flip-Flops

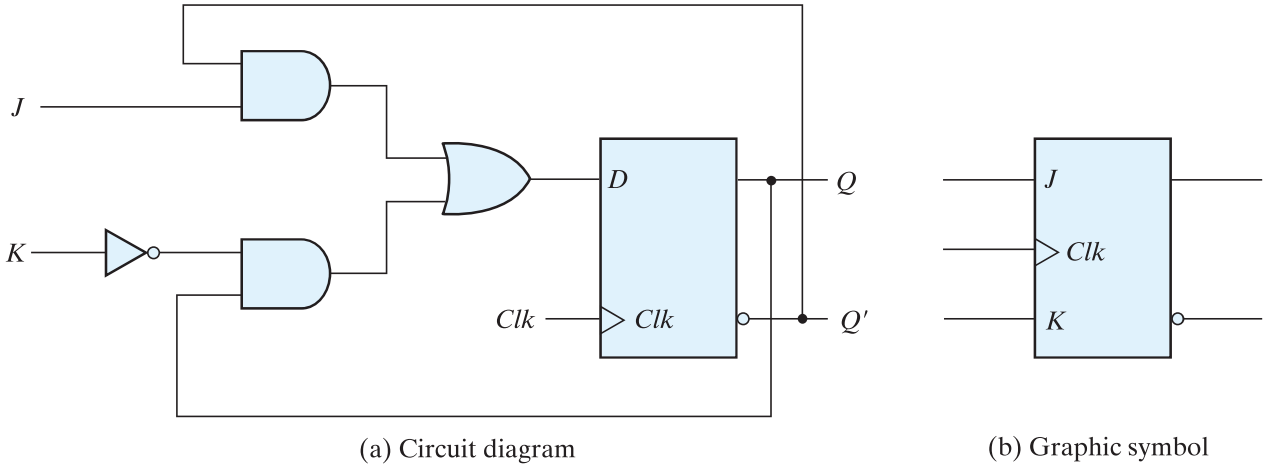
Very large-scale integration circuits contain several thousands of gates within one package. Circuits are constructed by interconnecting the various gates to provide a digital system. Each flip-flop is constructed from an interconnection of gates. The most economical and efficient flip-flop constructed in this manner is the edge-triggered *D* flip-flop, because it requires the smallest number of gates. Other types of flip-flops can be constructed by using the *D* flip-flop and external logic. Two flip-flops less widely used in the design of digital systems are the *JK* and *T* flip-flops.

There are three operations that can be performed with a flip-flop: Set it to 1, reset it to 0, or complement its output. With only a single input, the *D* flip-flop can set or reset the output, depending on the value of the *D* input immediately before the clock transition. Synchronized by a clock signal, the *JK* flip-flop has two inputs and performs all three operations. The circuit diagram of a *JK* flip-flop constructed with a *D* flip-flop and gates is shown in Fig. 5.12(a). The *J* input sets the flip-flop to 1, the *K* input resets it to 0, and when both inputs are enabled, the output is complemented. This can be verified by investigating the circuit applied to the *D* input:

$$D = JQ' + K'Q$$

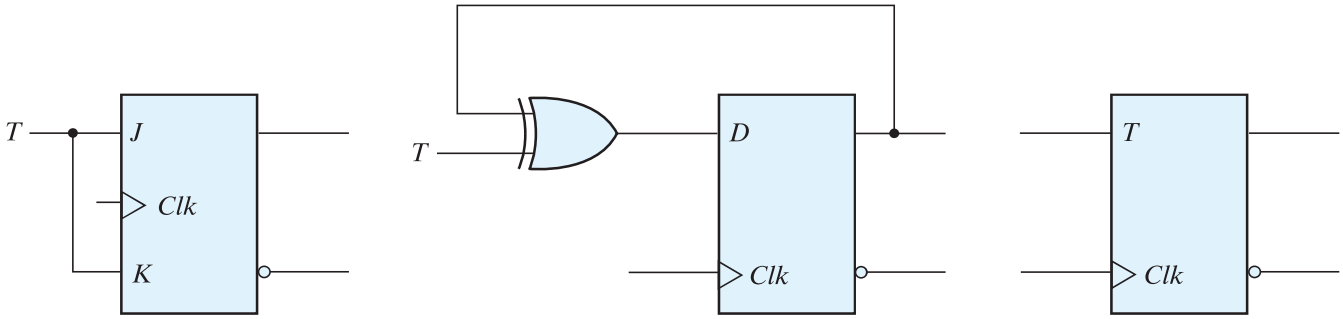
When $J = 1$ and $K = 0$, $D = Q' + Q = 1$, so the next clock edge sets the output to 1. When $J = 0$ and $K = 1$, $D = 0$, so the next clock edge resets the output to 0. When both $J = K = 1$ and $D = Q'$, the next clock edge complements the output. When both $J = K = 0$ and $D = Q$, the clock edge leaves the output unchanged. The graphic symbol for the *JK* flip-flop is shown in Fig. 5.12(b). It is similar to the graphic symbol of the *D* flip-flop, except that now the inputs are marked *J* and *K*.

The *T* (toggle) flip-flop is a complementing flip-flop and can be obtained from a *JK* flip-flop when inputs *J* and *K* are tied together. This is shown in Fig. 5.13(a). When




(a) Circuit diagram (b) Graphic symbol

**FIGURE 5.12**
***JK* flip-flop**




(a) From *JK* flip-flop (b) From *D* flip-flop (c) Graphic symbol

**FIGURE 5.13**
***T* flip-flop**

$T = 0$ ($J = K = 0$), a clock edge does not change the output. When $T = 1$ ($J = K = 1$), a clock edge complements the output. The complementing flip-flop is useful for designing binary counters.

The $T$ flip-flop can be constructed with a $D$ flip-flop and an exclusive-OR gate as shown in Fig. 5.13(b). The expression for the $D$ input is

$$D = T \oplus Q = TQ' + T'Q$$

When $T = 0$, $D = Q$ and there is no change in the output. When $T = 1$, $D = Q'$ and the output complements. The graphic symbol for this flip-flop has a $T$ symbol in the input.

## Characteristic Tables

A characteristic table defines the logical properties of a flip-flop by describing its operation in tabular form. The characteristic tables of three types of flip-flops are presented in Table 5.1. They define the next state (i.e., the state that results from a clock transition)

**Table 5.1**
*Flip-Flop Characteristic Tables*

***JK* Flip-Flop**

| *J* | *K* | *Q*(*t* + 1) | |
|---|---|---|---|
| 0 | 0 | $Q(t)$ | No change |
| 0 | 1 | 0 | Reset |
| 1 | 0 | 1 | Set |
| 1 | 1 | $Q'(t)$ | Complement |

***D* Flip-Flop**

| *D* | *Q*(*t* + 1) | |
|---|---|---|
| 0 | 0 | Reset |
| 1 | 1 | Set |

***T* Flip-Flop**

| *T* | *Q*(*t* + 1) | |
|---|---|---|
| 0 | $Q(t)$ | No change |
| 1 | $Q'(t)$ | Complement |

as a function of the inputs and the present state. $Q(t)$ refers to the present state (i.e., the state present prior to the application of a clock edge). $Q(t + 1)$ is the next state one clock period later. Note that the clock edge input is not included in the characteristic table, but is implied to occur between times $t$ and $t + 1$. Thus, $Q(t)$ denotes the state of the flip-flop immediately before the clock edge, and $Q(t + 1)$ denotes the state that results from the clock transition.

The characteristic table for the *JK* flip-flop shows that the next state is equal to the present state when inputs $J$ and $K$ are both equal to 0. This condition can be expressed as $Q(t + 1) = Q(t)$, indicating that the clock produces no change of state. When $K = 1$ and $J = 0$, the clock resets the flip-flop and $Q(t + 1) = 0$. With $J = 1$ and $K = 0$, the flip-flop sets and $Q(t + 1) = 1$. When both $J$ and $K$ are equal to 1, the next state changes to the complement of the present state, a transition that can be expressed as $Q(t + 1) = Q'(t)$.

The next state of a $D$ flip-flop is dependent only on the $D$ input and is independent of the present state. This can be expressed as $Q(t + 1) = D$. It means that the next-state value is equal to the value of $D$. Note that the $D$ flip-flop does not have a "no-change" condition. Such a condition can be accomplished either by disabling the clock or by operating the clock by having the output of the flip-flop connected into the $D$ input. Either method effectively circulates the output of the flip-flop when the state of the flip-flop must remain unchanged.

The characteristic table of the $T$ flip-flop has only two conditions: When $T = 0$, the clock edge does not change the state; when $T = 1$, the clock edge complements the state of the flip-flop.

## Characteristic Equations

The logical properties of a flip-flop, as described in the characteristic table, can be expressed algebraically with a characteristic equation. For the $D$ flip-flop, we have the characteristic equation

$$Q(t + 1) = D$$

which states that the next state of the output will be equal to the value of input $D$ in the present state. The characteristic equation for the $JK$ flip-flop can be derived from the characteristic table or from the circuit of Fig. 5.12. We obtain

$$Q(t + 1) = JQ' + K'Q$$

where $Q$ is the value of the flip-flop output prior to the application of a clock edge. The characteristic equation for the $T$ flip-flop is obtained from the circuit of Fig. 5.13:

$$Q(t + 1) = T \oplus Q = TQ' + T'Q$$

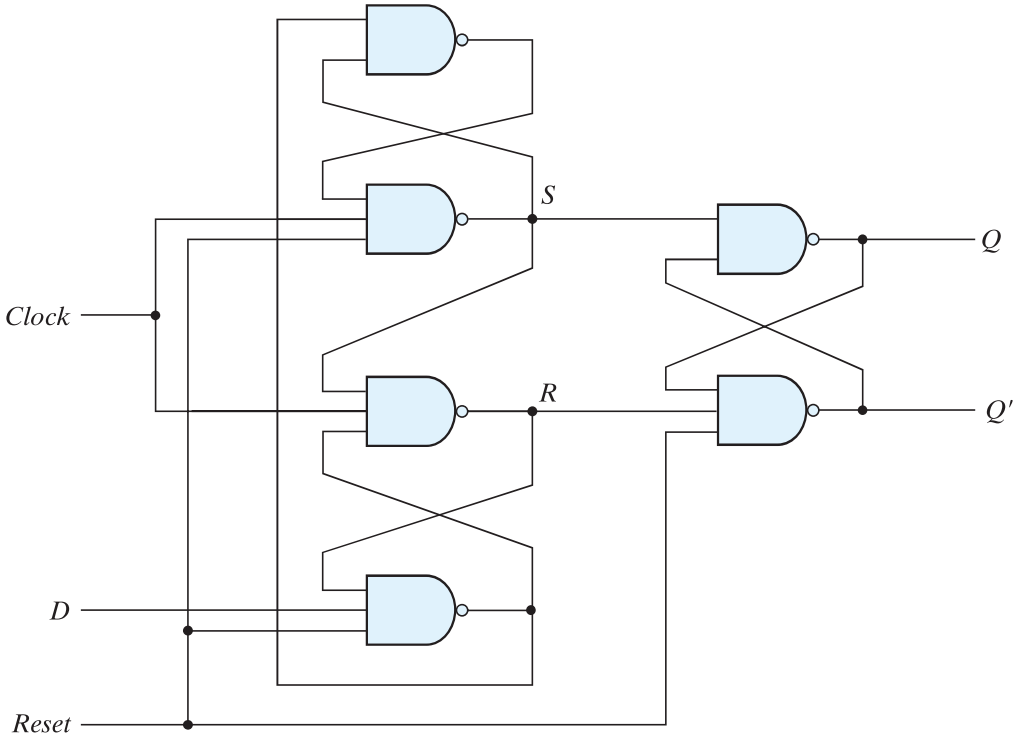
## Direct Inputs

Some flip-flops have asynchronous inputs that are used to force the flip-flop to a particular state independently of the clock. The input that sets the flip-flop to 1 is called *preset* or *direct set*. The input that clears the flip-flop to 0 is called *clear* or *direct reset*. When power is turned on in a digital system, the state of the flip-flops is unknown. The direct inputs are useful for bringing all flip-flops in the system to a known starting state prior to the clocked operation.

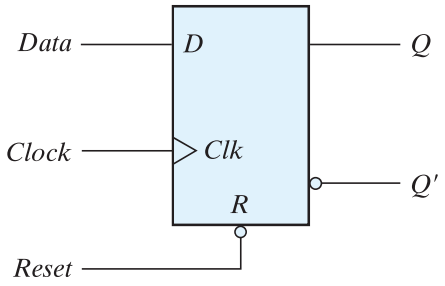
A positive-edge-triggered $D$ flip-flop with active-low asynchronous reset is shown in Fig. 5.14. The circuit diagram is the same as the one in Fig. 5.10, except for the additional reset input connections to three NAND gates. When the reset input is 0, it forces output $Q'$ to stay at 1, which, in turn, clears output $Q$ to 0, thus resetting the flip-flop. Two other connections from the reset input ensure that the $S$ input of the third $SR$ latch stays at logic 1 while the reset input is at 0, regardless of the values of $D$ and $Clk$.

The graphic symbol for the $D$ flip-flop with a direct reset has an additional input marked with $R$. The bubble along the input indicates that the reset is active at the logic-0 level. Flip-flops with a direct set use the symbol $S$ for the asynchronous set input.

The function table specifies the operation of the circuit. When $R = 0$, the output is reset to 0. This state is independent of the values of $D$ or $Clk$. Normal clock operation can proceed only after the reset input goes to logic 1. The clock at $Clk$ is shown with an upward arrow to indicate that the flip-flop triggers on the positive edge of the clock. The value in $D$ is transferred to $Q$ with every positive-edge clock signal, provided that $R = 1$.




(a) Circuit diagram




(b) Graphic symbol

| R | Clk | D | Q | Q′ |
|---|---|---|---|---|
| 0 | X | X | 0 | 1 |
| 0 | ↑ | 0 | 0 | 1 |
| 0 | ↑ | 1 | 1 | 0 |

(b) Function table

**FIGURE 5.14**
***D* flip-flop with asynchronous reset**

## 5.5 ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS

Analysis describes what a given circuit will do under certain operating conditions. The behavior of a clocked sequential circuit is determined from the inputs, the outputs, and the state of its flip-flops. The outputs and the next state are both a function of the inputs and the present state. The analysis of a sequential circuit consists of obtaining a table or a diagram for the time sequence of inputs, outputs, and internal states. It is also possible

to write Boolean expressions that describe the behavior of the sequential circuit. These expressions must include the necessary time sequence, either directly or indirectly.

A logic diagram is recognized as a clocked sequential circuit if it includes flip-flops with clock inputs. The flip-flops may be of any type, and the logic diagram may or may not include combinational logic gates. In this section, we introduce an algebraic representation for specifying the next-state condition in terms of the present state and inputs. A state table and state diagram are then presented to describe the behavior of the sequential circuit. Another algebraic representation is introduced for specifying the logic diagram of sequential circuits. Examples are used to illustrate the various procedures.

## State Equations

The behavior of a clocked sequential circuit can be described algebraically by means of state equations. A *state equation* (also called a *transition equation*) specifies the next state as a function of the present state and inputs. Consider the sequential circuit shown in Fig. 5.15. We will later show that it acts as a 0-detector by asserting its output when a

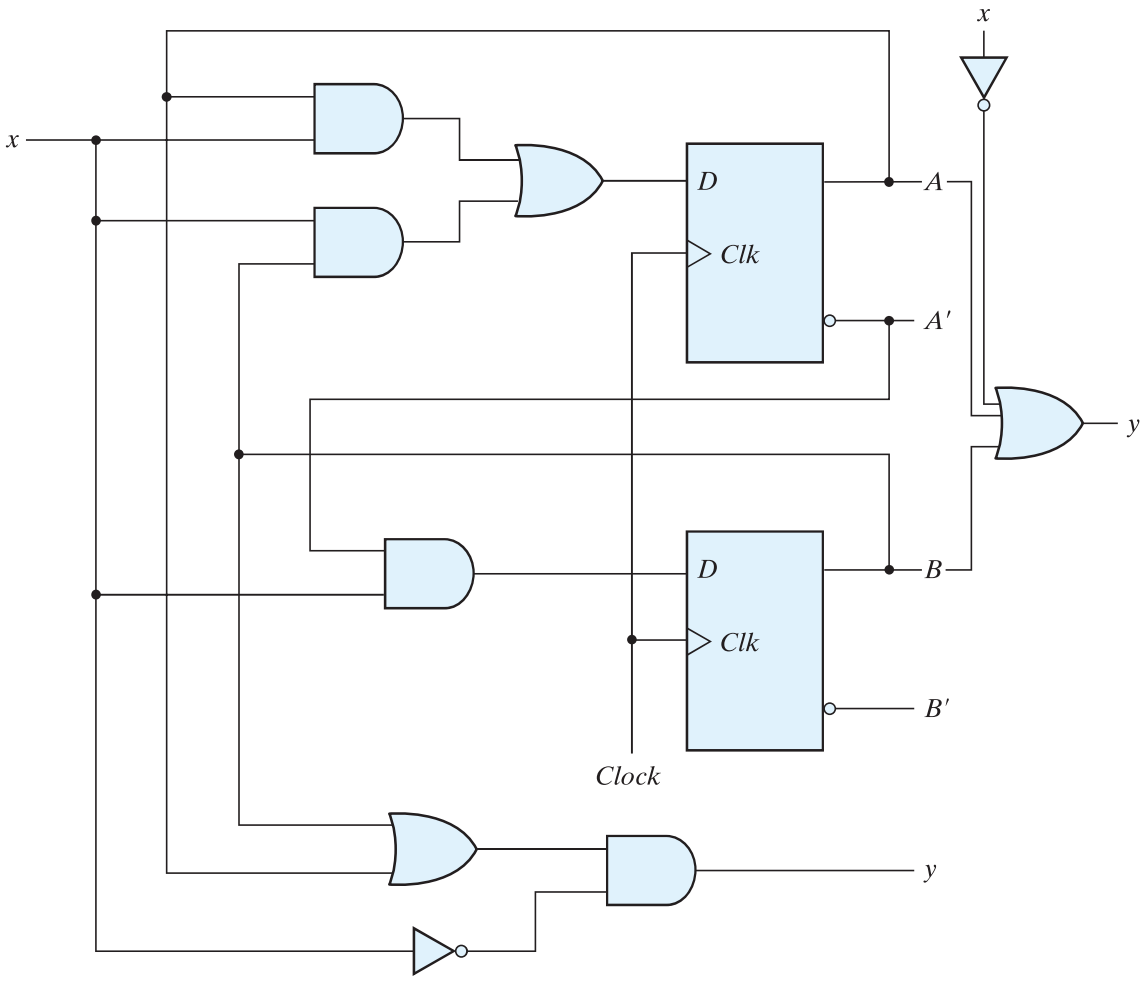



**FIGURE 5.15**
**Example of sequential circuit**

0 is detected in a stream of 1s. It consists of two $D$ flip-flops $A$ and $B$, an input $x$ and an output $y$. Since the $D$ input of a flip-flop determines the value of the next state (i.e., the state reached after the clock transition), it is possible to write a set of state equations for the circuit:

$$A(t + 1) = A(t)x(t) + B(t)x(t)$$
$$B(t + 1) = A'(t)x(t)$$

A state equation is an algebraic expression that specifies the condition for a flip-flop state transition. The left side of the equation, with $(t + 1)$, denotes the next state of the flip-flop one clock edge later. The right side of the equation is a Boolean expression that specifies the present state and input conditions that make the next state equal to 1. Since all the variables in the Boolean expressions are a function of the present state, we can omit the designation $(t)$ after each variable for convenience and can express the state equations in the more compact form

$$A(t + 1) - Ax + Bx$$
$$B(t + 1) - A'x$$

The Boolean expressions for the state equations can be derived directly from the gates that form the combinational circuit part of the sequential circuit, since the $D$ values of the combinational circuit determine the next state. Similarly, the present-state value of the output can be expressed algebraically as

$$y(t) = [A(t) + B(t)]x'(t)$$

By removing the symbol $(t)$ for the present state, we obtain the output Boolean equation:

$$y = (A + B)x'$$

## State Table

The time sequence of inputs, outputs, and flip-flop states can be enumerated in a *state table* (sometimes called a *transition table*). The state table for the circuit of Fig. 5.15 is shown in Table 5.2. The table consists of four sections labeled *present state*, *input*, *next state*, and *output*. The present-state section shows the states of flip-flops $A$ and $B$ at any given time $t$. The input section gives a value of $x$ for each possible present state. The next-state section shows the states of the flip-flops one clock cycle later, at time $t + 1$. The output section gives the value of $y$ at time $t$ for each present state and input condition.

The derivation of a state table requires listing all possible binary combinations of present states and inputs. In this case, we have eight binary combinations from 000 to 111. The next-state values are then determined from the logic diagram or from the state equations. The next state of flip-flop $A$ must satisfy the state equation

$$A(t + 1) = Ax + Bx$$

**Table 5.2**
*State Table for the Circuit of Fig. 5.15*

| Present State | | Input | Next State | | Output |
|---|---|---|---|---|---|
| *A* | *B* | *x* | *A* | *B* | *y* |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

The next-state section in the state table under column $A$ has three 1's where the present state of $A$ and input $x$ are both equal to 1 or the present state of $B$ and input $x$ are both equal to 1. Similarly, the next state of flip-flop $B$ is derived from the state equation

$$B(t + 1) = A'x$$

and is equal to 1 when the present state of $A$ is 0 and input $x$ is equal to 1. The output column is derived from the output equation

$$y = Ax' + Bx'$$

The state table of a sequential circuit with $D$-type flip-flops is obtained by the same procedure outlined in the previous example. In general, a sequential circuit with $m$ flip-flops and $n$ inputs needs $2^{m+n}$ rows in the state table. The binary numbers from 0 through $2^{m+n} - 1$ are listed under the present-state and input columns. The next-state section has $m$ columns, one for each flip-flop. The binary values for the next state are derived directly from the state equations. The output section has as many columns as there are output variables. Its binary value is derived from the circuit or from the Boolean function in the same manner as in a truth table.

It is sometimes convenient to express the state table in a slightly different form having only three sections: present state, next state, and output. The input conditions are enumerated under the next-state and output sections. The state table of Table 5.2 is repeated in Table 5.3 in this second form. For each present state, there are two possible next states and outputs, depending on the value of the input. One form may be preferable to the other, depending on the application.

## State Diagram

The information available in a state table can be represented graphically in the form of a state diagram. In this type of diagram, a state is represented by a circle, and the (clock-triggered) transitions between states are indicated by directed lines connecting

**Table 5.3**
*Second Form of the State Table*

| Present State | | Next State | | | | Output | |
|---|---|---|---|---|---|---|---|
| | | $x = 0$ | | $x = 1$ | | $x = 0$ | $x = 1$ |
| $A$ | $B$ | $A$ | $B$ | $A$ | $B$ | $y$ | $y$ |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

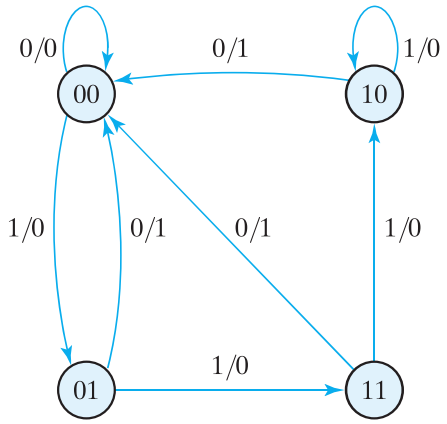



**FIGURE 5.16**
State diagram of the circuit of Fig. 5.15

the circles. The state diagram of the sequential circuit of Fig. 5.15 is shown in Fig. 5.16. The state diagram provides the same information as the state table and is obtained directly from Table 5.2 or Table 5.3. The binary number inside each circle identifies the state of the flip-flops. The directed lines are labeled with two binary numbers separated by a slash. The input value during the present state is labeled first, and the number after the slash gives the output during the present state with the given input. (It is important to remember that the bit value listed for the output along the directed line occurs during the present state and with the indicated input, and has nothing to do with the transition to the next state.) For example, the directed line from state 00 to 01 is labeled 1/0, meaning that when the sequential circuit is in the present state 00 and the input is 1, the output is 0. After the next clock cycle, the circuit goes to the next state, 01. If the input changes to 0, then the output becomes 1, but if the input remains at 1, the output stays at 0. This information is obtained from the state diagram along the two directed lines emanating from the circle with state 01. A directed line connecting a circle with itself indicates that no change of state occurs.

The steps presented in this example are summarized below:

Circuit diagram → Equations – State table → State diagram

This sequence of steps begins with a structural representation of the circuit and proceeds to an abstract representation of its behavior. An HDL model can be in the form of a gate-level description or in the form of a behavioral description. It is important to note that a gate-level approach requires that the designer understands how to select and connect gates and flip-flops to form a circuit having a particular behavior. That understanding comes with experience. On the other hand, an approach based on behavioral modeling does not require the designer to know how to invent a schematic—the designer needs only to know how to describe behavior using the constructs of the HDL, because the circuit is produced automatically by a synthesis tool. Therefore, one does not have to accumulate years of experience in order to become a productive designer of digital circuits; nor does one have to acquire an extensive background in electrical engineering.

There is no difference between a state table and a state diagram, except in the manner of representation. The state table is easier to derive from a given logic diagram and the state equation. The state diagram follows directly from the state table. *The state diagram gives a pictorial view of state transitions and is the form more suitable for human interpretation of the circuit's operation*. For example, the state diagram of Fig. 5.16 clearly shows that, starting from state 00, the output is 0 as long as the input stays at 1. The first 0 input after a string of 1's gives an output of 1 and transfers the circuit back to the initial state, 00. The machine represented by this state diagram acts to detect a zero in the bit stream of data. It corresponds to the behavior of the circuit in Fig. 5.15. Other circuits that detect a zero in a stream of data may have a simpler circuit diagram and state diagram.

## Flip-Flop Input Equations

The logic diagram of a sequential circuit consists of flip-flops and gates. The interconnections among the gates form a combinational circuit and may be specified algebraically with Boolean expressions. The knowledge of the type of flip-flops and a list of the Boolean expressions of the combinational circuit provide the information needed to draw the logic diagram of the sequential circuit. The part of the combinational circuit that generates external outputs is described algebraically by a set of Boolean functions called *output equations*. The part of the circuit that generates the inputs to flip-flops is described algebraically by a set of Boolean functions called flip-flop *input equations* (or, sometimes, *excitation equations*). We will adopt the convention of using the flip-flop input symbol to denote the input equation variable and a subscript to designate the name of the flip-flop output. For example, the following input equation specifies an OR gate with inputs $x$ and $y$ connected to the $D$ input of a flip-flop whose output is labeled with the symbol $Q$:

$$D_Q = x + y$$

The sequential circuit of Fig. 5.15 consists of two $D$ flip-flops $A$ and $B$, an input $x$, and an output $y$. The logic diagram of the circuit can be expressed algebraically with two flip-flop input equations and an output equation:

$$\begin{aligned} D_A &= Ax + Bx \\ D_B &= A'x \\ y &= (A + B)x' \end{aligned}$$

The three equations provide the necessary information for drawing the logic diagram of the sequential circuit. The symbol $D_A$ specifies a $D$ flip-flop labeled $A$. $D_B$ specifies a second $D$ flip-flop labeled $B$. The Boolean expressions associated with these two variables and the expression for output $y$ specify the combinational circuit part of the sequential circuit.

The flip-flop input equations constitute a convenient algebraic form for specifying the logic diagram of a sequential circuit. They imply the type of flip-flop from the letter symbol, and they fully specify the combinational circuit that drives the flip-flops. Note that the expression for the input equation for a $D$ flip-flop is identical to the expression for the corresponding state equation. This is because of the characteristic equation that equates the next state to the value of the $D$ input: $Q(t + 1) = D_Q$.

## Analysis with *D* Flip-Flops

We will summarize the procedure for analyzing a clocked sequential circuit with $D$ flip-flops by means of a simple example. The circuit we want to analyze is described by the input equation

$$D_A = A \oplus x \oplus y$$

The $D_A$ symbol implies a $D$ flip-flop with output $A$. The $x$ and $y$ variables are the inputs to the circuit. No output equations are given, which implies that the output comes from the output of the flip-flop. The logic diagram is obtained from the input equation and is drawn in Fig. 5.17(a).

The state table has one column for the present state of flip-flop $A$, two columns for the two inputs, and one column for the next state of $A$. The binary numbers under $Axy$ are listed from 000 through 111 as shown in Fig. 5.17(b). The next-state values are obtained from the state equation

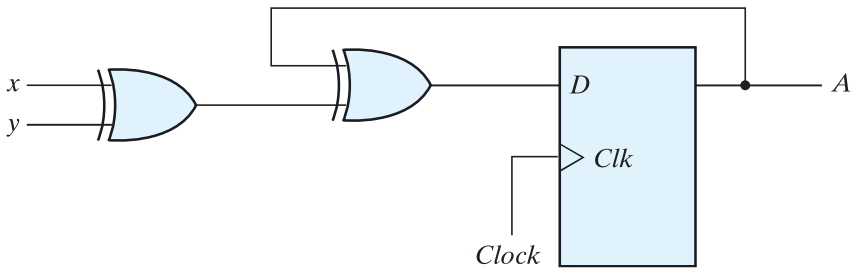
$$A(t + 1) = A \oplus x \oplus y$$

The expression specifies an odd function and is equal to 1 when only one variable is 1 or when all three variables are 1. This is indicated in the column for the next state of $A$.

The circuit has one flip-flop and two states. The state diagram consists of two circles, one for each state as shown in Fig. 5.17(c). The present state and the output can be either 0 or 1, as indicated by the number inside the circles. A slash on the directed lines is not needed, because there is no output from a combinational circuit. The two inputs can have four possible combinations for each state. Two input combinations during each state transition are separated by a comma to simplify the notation.

## Analysis with *JK* Flip-Flops

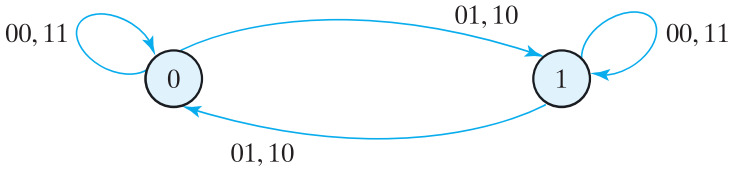
A state table consists of four sections: present state, inputs, next state, and outputs. The first two are obtained by listing all binary combinations. The output section is determined from the output equations. The next-state values are evaluated from the state equations. For a $D$-type flip-flop, the state equation is the same as the input equation. When a flip-flop other than the $D$ type is used, such as $JK$ or $T$, it is necessary to refer




(a) Circuit diagram

| Present state | Inputs | | Next state |
|---|---|---|---|
| $A$ | $x$ | $y$ | $A$ |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

(b) State table




(c) State diagram

**FIGURE 5.17**
**Sequential circuit with *D* flip-flop**

to the corresponding characteristic table or characteristic equation to obtain the next-state values. We will illustrate the procedure first by using the characteristic table and again by using the characteristic equation.

The next-state values of a sequential circuit that uses *JK*- or *T*-type flip-flops can be derived as follows:

1. Determine the flip-flop input equations in terms of the present state and input variables.
2. List the binary values of each input equation.
3. Use the corresponding flip-flop characteristic table to determine the next-state values in the state table.

As an example, consider the sequential circuit with two *JK* flip-flops $A$ and $B$ and one input $x$, as shown in Fig. 5.18. The circuit has no outputs; therefore, the state table does not need an output column. (The outputs of the flip-flops may be considered as the outputs in this case.) The circuit can be specified by the flip-flop input equations

$$J_A = B \quad K_A = Bx'$$
$$J_B = x' \quad K_B = A'x + Ax' = A \oplus x$$

The state table of the sequential circuit is shown in Table 5.4. The present-state and input columns list the eight binary combinations. The binary values listed under the

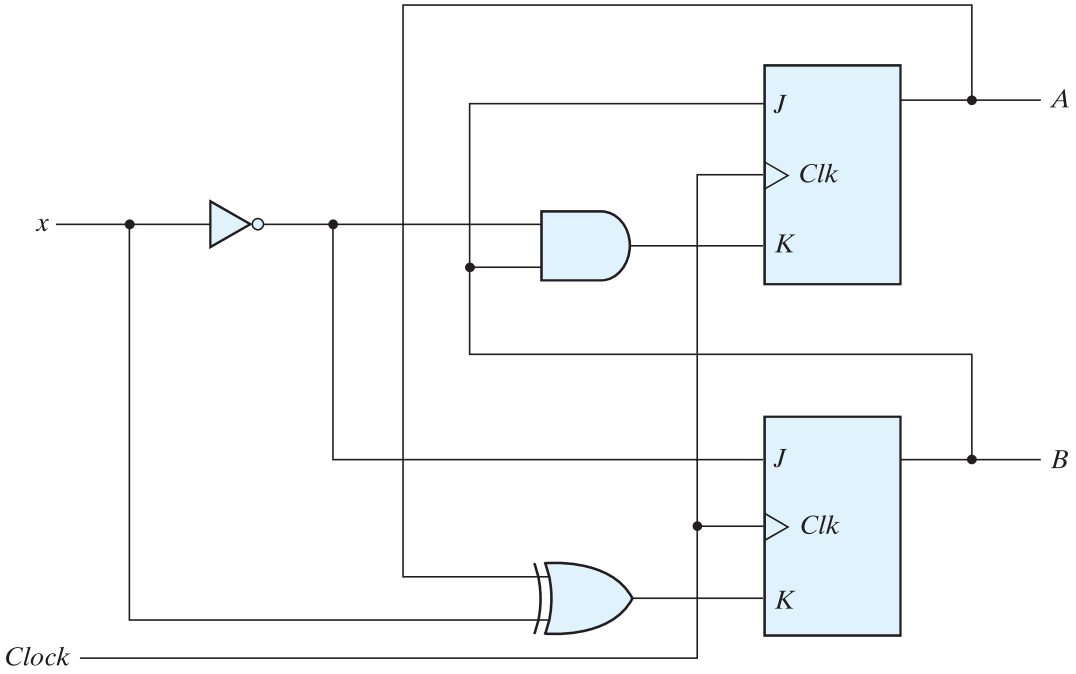



**FIGURE 5.18**
Sequential circuit with *JK* flip-flop

columns labeled *flip-flop inputs* are not part of the state table, but they are needed for the purpose of evaluating the next state as specified in step 2 of the procedure. These binary values are obtained directly from the four input equations in a manner similar to that for obtaining a truth table from a Boolean expression. The next state of each flip-flop is evaluated from the corresponding $J$ and $K$ inputs and the characteristic table of the $JK$ flip-flop listed in Table 5.1. There are four cases to consider. When $J = 1$ and

**Table 5.4**
*State Table for Sequential Circuit with JK Flip-Flops*

| Present State | | Input | Next State | | Flip-Flop Inputs | | | |
|---|---|---|---|---|---|---|---|---|
| *A* | *B* | *x* | *A* | *B* | $J_A$ | $K_A$ | $J_B$ | $K_B$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

$K = 0$, the next state is 1. When $J = 0$ and $K = 1$, the next state is 0. When $J = K = 0$, there is no change of state and the next-state value is the same as that of the present state. When $J = K = 1$, the next-state bit is the complement of the present-state bit. Examples of the last two cases occur in the table when the present state $AB$ is 10 and input $x$ is 0. $JA$ and $KA$ are both equal to 0 and the present state of $A$ is 1. Therefore, the next state of $A$ remains the same and is equal to 1. In the same row of the table, $JB$ and $KB$ are both equal to 1. Since the present state of $B$ is 0, the next state of $B$ is complemented and changes to 1.

The next-state values can also be obtained by evaluating the state equations from the characteristic equation. This is done by using the following procedure:

1. Determine the flip-flop input equations in terms of the present state and input variables.
2. Substitute the input equations into the flip-flop characteristic equation to obtain the state equations.
3. Use the corresponding state equations to determine the next-state values in the state table.

The input equations for the two $JK$ flip-flops of Fig. 5.18 were listed a couple of paragraphs ago. The characteristic equations for the flip-flops are obtained by substituting $A$ or $B$ for the name of the flip-flop, instead of $Q$:

$$A(t + 1) = JA' + K'A$$
$$B(t + 1) = JB' + K'B$$

Substituting the values of $J_A$ and $K_A$ from the input equations, we obtain the state equation for $A$:

$$A(t + 1) = BA' + (Bx')'A = A'B + AB' + Ax$$

The state equation provides the bit values for the column headed "Next State" for $A$ in the state table. Similarly, the state equation for flip-flop $B$ can be derived from the characteristic equation by substituting the values of $J_B$ and $K_B$:

$$B(t + 1) = x'B' + (A \oplus x)'B = B'x' + ABx + A'Bx'$$

The state equation provides the bit values for the column headed "Next State" for $B$ in the state table. Note that the columns in Table 5.4 headed "Flip-Flop Inputs" are not needed when state equations are used.

The state diagram of the sequential circuit is shown in Fig. 5.19. Note that since the circuit has no outputs, the directed lines out of the circles are marked with one binary number only, to designate the value of input $x$.

## Analysis with *T* Flip-Flops

The analysis of a sequential circuit with $T$ flip-flops follows the same procedure outlined for $JK$ flip-flops. The next-state values in the state table can be obtained by using either

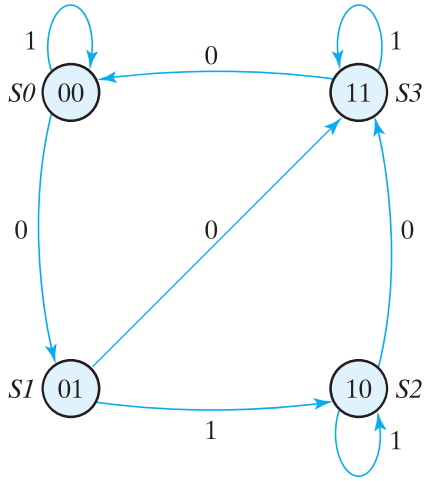



**FIGURE 5.19**
**State diagram of the circuit of Fig. 5.18**

the characteristic table listed in Table 5.1 or the characteristic equation

$$Q(t + 1) = T \oplus Q = T'Q + TQ'$$

Now consider the sequential circuit shown in Fig. 5.20. It has two flip-flops $A$ and $B$, one input $x$, and one output $y$ and can be described algebraically by two input equations and an output equation:

$$T_A = Bx$$
$$T_B = x$$
$$y = AB$$

The state table for the circuit is listed in Table 5.5. The values for $y$ are obtained from the output equation. The values for the next state can be derived from the state equations by substituting $T_A$ and $T_B$ in the characteristic equations, yielding

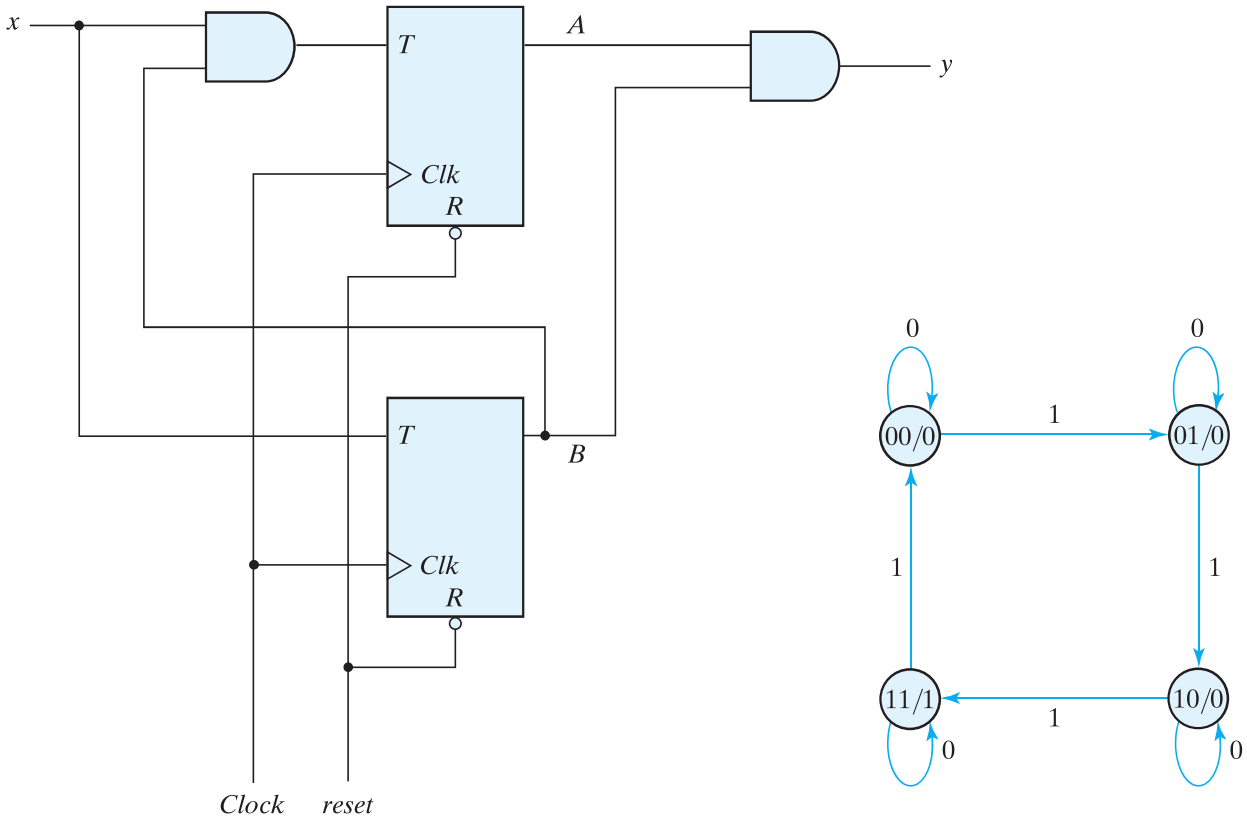
$$A(t + 1) = (Bx)'A + (Bx)A' = AB' + Ax' + A'Bx$$
$$B(t + 1) = x \oplus B$$

The next-state values for $A$ and $B$ in the state table are obtained from the expressions of the two state equations.

The state diagram of the circuit is shown in Fig. 5.20(b). As long as input $x$ is equal to 1, the circuit behaves as a binary counter with a sequence of states 00, 01, 10, 11, and back to 00. When $x = 0$, the circuit remains in the same state. Output $y$ is equal to 1 when the present state is 11. Here, the output depends on the present state only and is independent of the input. The two values inside each circle and separated by a slash are for the present state and output.

## Mealy and Moore Models of Finite State Machines

The most general model of a sequential circuit has inputs, outputs, and internal states. It is customary to distinguish between two models of sequential circuits: the Mealy model and the Moore model. Both are shown in Fig. 5.21. They differ only in the way the output




(a) Circuit diagram (b) State diagram

**FIGURE 5.20**
**Sequential circuit with *T* flip-flops (Binary Counter)**

**Table 5.5**
*State Table for Sequential Circuit with T Flip-Flops*

| Present State | | Input | Next State | | Output |
|---|---|---|---|---|---|
| *A* | *B* | *x* | *A* | *B* | *y* |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

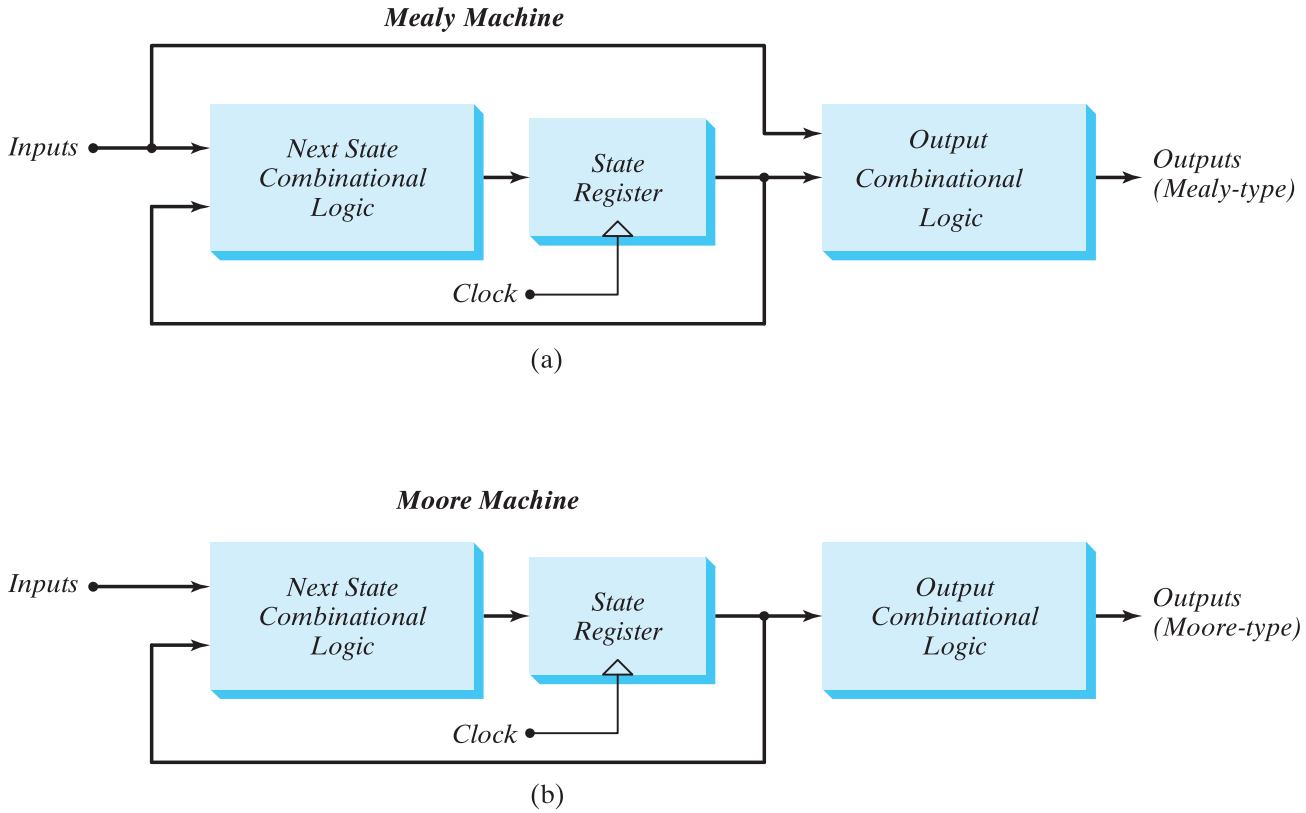



**FIGURE 5.21**
**Block diagrams of Mealy and Moore state machines**

is generated. In the Mealy model, the output is a function of both the present state and the input. In the Moore model, the output is a function of only the present state. A circuit may have both types of outputs. The two models of a sequential circuit are commonly referred to as a finite state machine, abbreviated FSM. The Mealy model of a sequential circuit is referred to as a Mealy FSM or Mealy machine. The Moore model is referred to as a Moore FSM or Moore machine.

The circuit presented previously in Fig. 5.15 is an example of a Mealy machine. Output $y$ is a function of both input $x$ and the present state of $A$ and $B$. The corresponding state diagram in Fig. 5.16 shows both the input and output values, separated by a slash along the directed lines between the states.

An example of a Moore model is given in Fig. 5.18. Here, the output is a function of the present state only. The corresponding state diagram in Fig. 5.19 has only inputs marked along the directed lines. The outputs are the flip-flop states marked inside the circles. Another example of a Moore model is the sequential circuit of Fig. 5.20. The output depends only on flip-flop values, and that makes it a function of the present state only. The input value in the state diagram is labeled along the directed line, but the output value is indicated inside the circle together with the present state.

**In a Moore model, the outputs of the sequential circuit are synchronized with the clock, because they depend only on flip-flop outputs that are synchronized with the clock.** In a Mealy model, the outputs may change if the inputs change during the clock

cycle. Moreover, the outputs may have momentary false values because of the delay encountered from the time that the inputs change and the time that the flip-flop outputs change. In order to synchronize a Mealy-type circuit, the inputs of the sequential circuit must be synchronized with the clock and the outputs must be sampled immediately before the clock edge. The inputs are changed at the inactive edge of the clock to ensure that the inputs to the flip-flops stabilize before the active edge of the clock occurs. Thus, **the output of the Mealy machine is the value that is present immediately before the active edge of the clock.**

## 5.6 SYNTHESIZABLE HDL MODELS OF SEQUENTIAL CIRCUITS

The Verilog HDL was introduced in Section 3.9. Combinational circuits were described in Section 4.12, and behavioral modeling with Verilog was introduced in that section as well. Behavioral models are abstract representations of the functionality of digital hardware. That is, they describe how a circuit behaves, but don't specify the internal details of the circuit. Historically, the abstraction has been described by truth tables, state tables, and state diagrams. An HDL describes the functionality differently, by language constructs that represent the operations of registers in a machine. This representation has "added value," i.e., it is important for you to know how to use, because it can be simulated to produce waveforms demonstrating the behavior of the machine.

### Behavioral Modeling

There are two kinds of abstract behaviors in the Verilog HDL. Behavior declared by the keyword **initial** is called *single-pass behavior* and specifies a single statement or a block statement (i.e., a list of statements enclosed by either a **begin** . . . **end** or a **fork** . . . **join** keyword pair). A single-pass behavior expires after the associated statement executes. In practice, designers use single-pass behavior primarily to prescribe stimulus signals in a test bench—never to model the behavior of a circuit—because synthesis tools do not accept descriptions that use the **initial** statement. The **always** keyword declares a *cyclic behavior.* Both types of behaviors begin executing when the simulator launches at time $t = 0$. The **initial** behavior expires after its statement executes; the **always** behavior executes and reexecutes indefinitely, until the simulation is stopped. A module may contain an arbitrary number of **initial** or **always** behavioral statements. They execute concurrently with respect to each other, starting at time 0, and may interact through common variables. Here's a word description of how an **always** statement works for a simple model of a $D$ flip-flop: Whenever the rising edge of the clock occurs, if the reset input is asserted, the output $q$ gets 0; otherwise the output $Q$ gets the value of the input $D$. The execution of statements triggered by the clock is repeated until the simulation ends. We'll see shortly how to write this description in Verilog.