Lecture 9

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.

SEQUENTIAL CIRCUITS

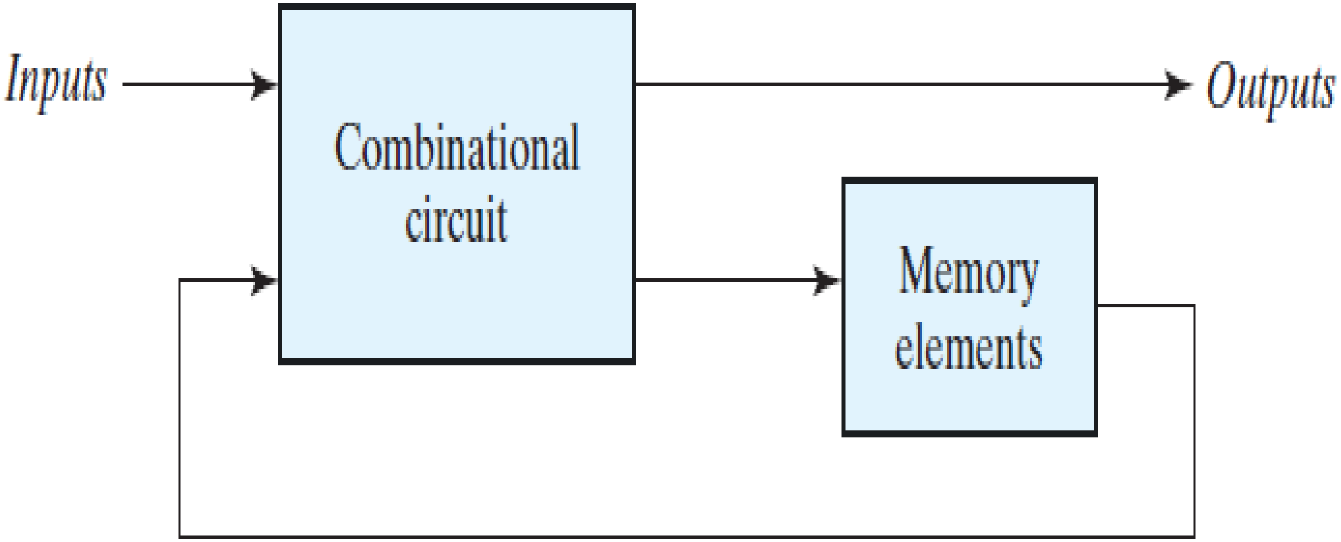


FIGURE 5.1

Block diagram of sequential circuit

SEQUENTIAL CIRCUITS

- 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 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.

SEQUENTIAL CIRCUITS

- 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.

SEQUENTIAL CIRCUITS

- 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.
- *Asynchronous* sequential circuit will not be covered in this text

SEQUENTIAL CIRCUITS

- 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.

SEQUENTIAL CIRCUITS

- In practice, the clock pulses determine *when* computational activity will occur within the circuit, and other signals 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.

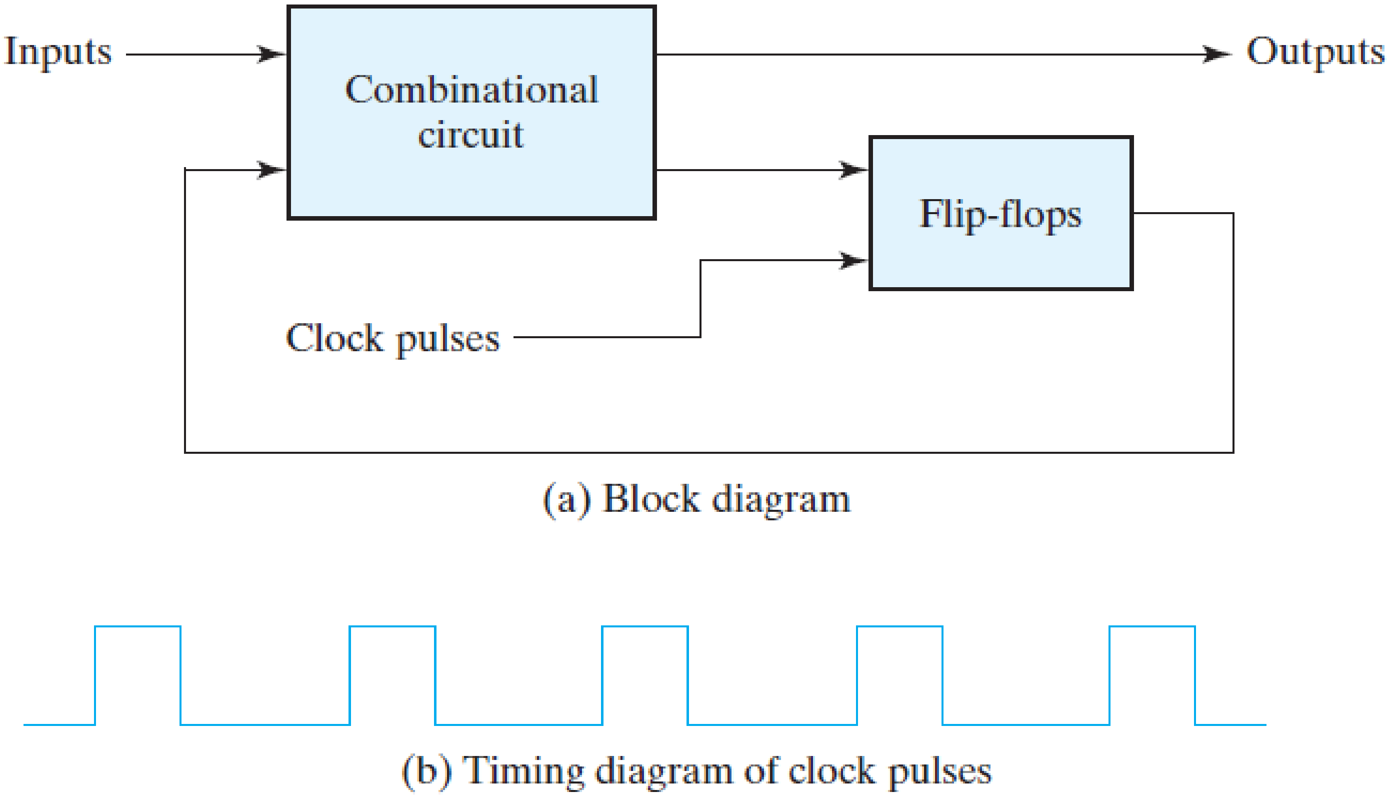
SEQUENTIAL CIRCUITS

- 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 clock pulses.
- The design of synchronous circuits is feasible because they seldom manifest instability problems.

SEQUENTIAL CIRCUITS

- The storage elements (memory) used in clocked sequential circuits are called *flipflops.*
- 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 .

SEQUENTIAL CIRCUITS



(a) Block diagram

(b) Timing diagram of clock pulses

FIGURE 5.2
Synchronous clocked sequential circuit

SEQUENTIAL CIRCUITS

- 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 when a pulse of the clock signal occurs.

SEQUENTIAL CIRCUITS

- 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.

SEQUENTIAL CIRCUITS

- 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.

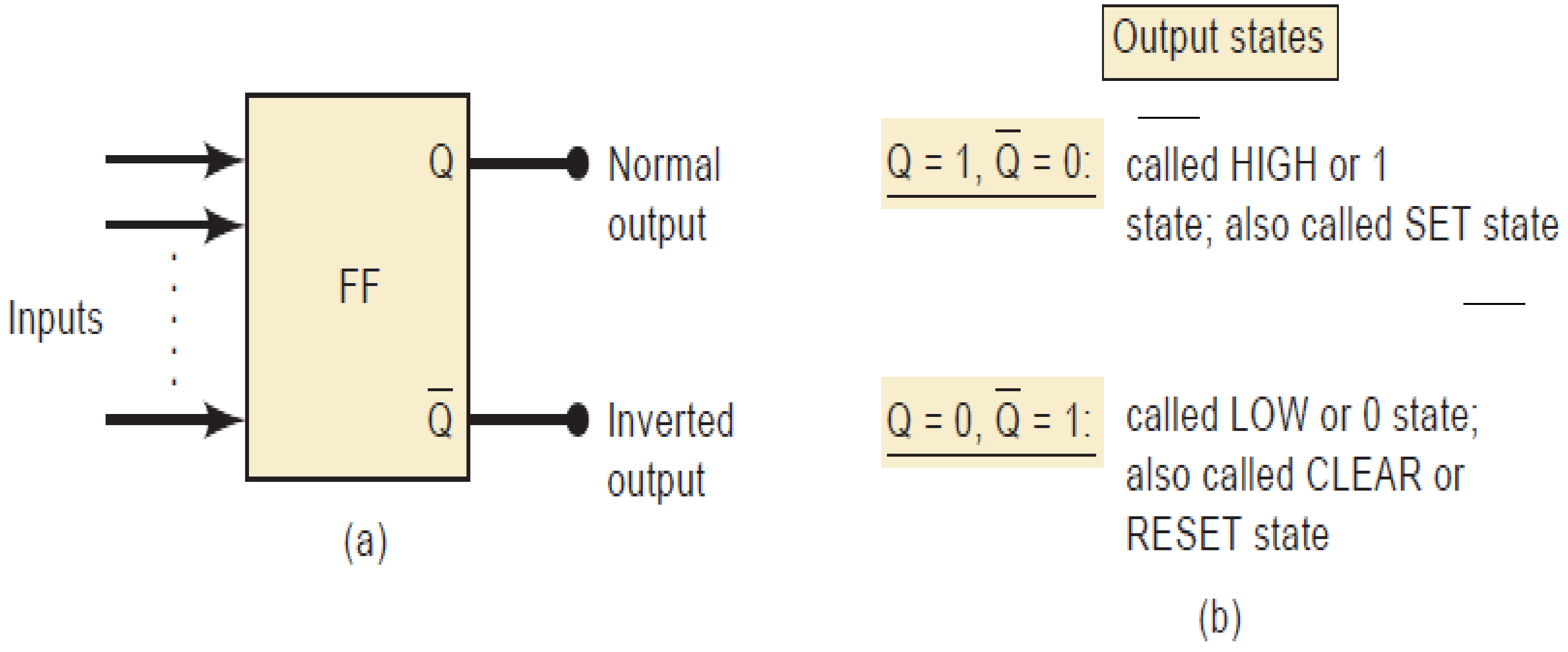
SEQUENTIAL CIRCUITS

- 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 flipflop 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.

SEQUENTIAL CIRCUITS

- Figure 5-2(a) is the general type of symbol used for a flip-flop.
- It shows two outputs, labeled *Q* and $\overline{Q}$ that are the inverse of each other.
- The *Q* output is called the *normal* FF output, and $\overline{Q}$ is the *inverted* FF output.
- Whenever we refer to the state of a FF, we are referring to the state of its normal (*Q*) output; it is understood that $\overline{Q}$ its inverted output is in the opposite state.

SEQUENTIAL CIRCUITS



Output states

$Q = 1, \overline{Q} = 0$: called HIGH or 1 state; also called SET state

$Q = 0, \overline{Q} = 1$: called LOW or 0 state; also called CLEAR or RESET state

FIGURE 5-2 General flip-flop symbol and definition of its two possible output states.

SEQUENTIAL CIRCUITS

- For example, if we say that a FF is in the HIGH (1) state, we mean that Q=1, if we say that a FF is in the LOW (0) state, we mean that Q=0.
- Of course, the Q state will always be the inverse of $\overline{Q}$.
- The two possible operating states for a FF are summarized in Figure 5-2(b).
- Note that the HIGH or 1 state(Q=1/$\overline{\text{Q}}$=0) is also referred to as the **SET** state.

SEQUENTIAL CIRCUITS

- Whenever the inputs to a FF cause it to go to the Q=1 state, we call this *setting* the FF; the FF has been set.
- In a similar way, the LOW or 0 state (Q=0/$\overline{Q}$=1) is a$\overline{\text{ls}}$o referred to as the **CLEAR** or **RESET** state.
- Whenever the inputs to a FF cause it to go to the Q=0 state, we call this *clearing* or *resetting* the FF; the FF has been cleared (reset).

SEQUENTIAL CIRCUITS

- As the symbol in Figure 5-2(a) implies, a FF can have one or more inputs.
- These inputs are used to cause the FF to switch back and forth (“flip-flop”) between its possible output states.
- We will find out that most FF inputs need only to be momentarily activated (pulsed) in order to cause a change in the FF output state, and the output will remain in that new state even after the input pulse is over.

SEQUENTIAL CIRCUITS

- This is the FF’s *memory* characteristic.
- The flip-flop is known by other names, including *latch* and *bistable multivibrator.*
- The term *latch* is used for certain types of flip-flops that we will describe.
- The term *bistable multivibrator* is the more technical name for a flip-flop, but it is too much of a mouthful to be used regularly

NAND GATE LATCH

- The most basic FF circuit can be constructed from either two NAND gates or two NOR gates.
- The NAND gate version, called a **NAND gate latch** or simply a **latch**, is shown in Figure 5-3(a).
- The two NAND gates are cross-coupled so that the output of NAND-1 is connected to one of the inputs of NAND-2, and vice versa.
- The gate outputs, labeled *Q* and Q' respectively, are the latch outputs.
- Under normal conditions, these outputs will always be the inverse of each other.

NAND GATE LATCH

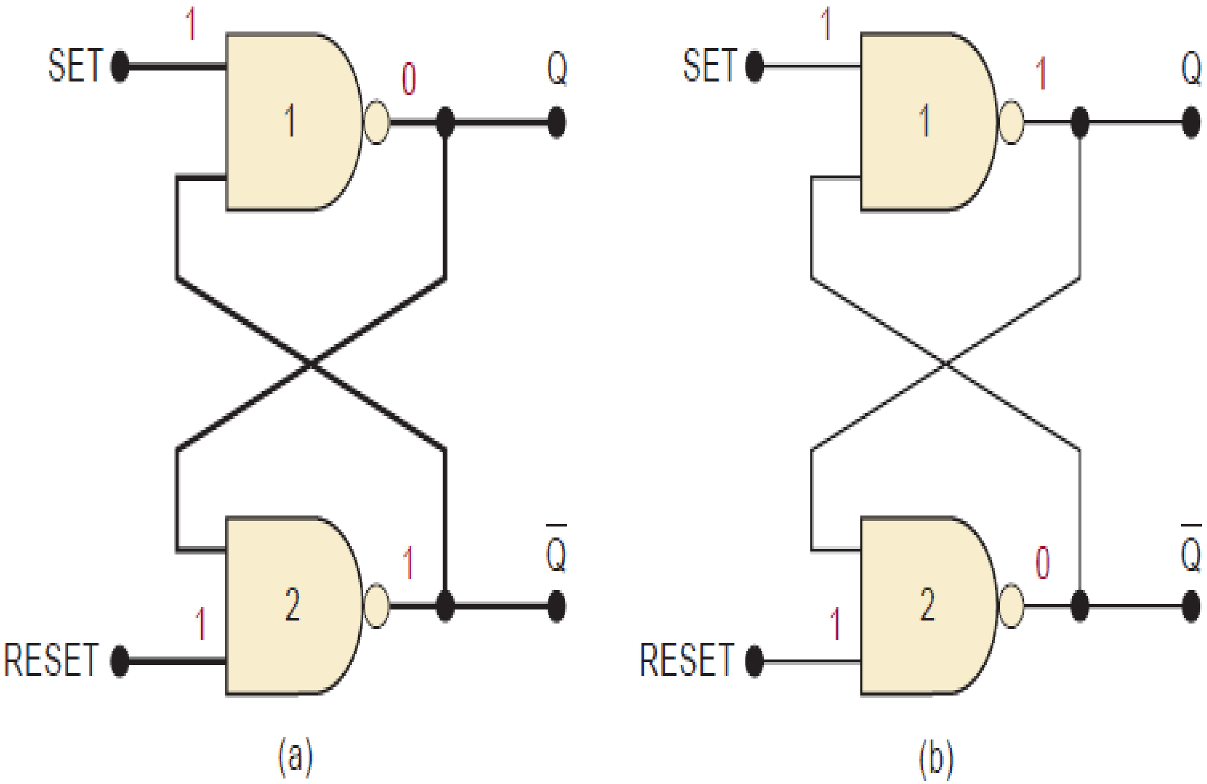
- There are two latch inputs: the SET input is the input that *sets Q* to the 1 state; the RESET input is the input that *resets Q* to the 0 state.
- The SET and RESET inputs are both normally resting in the HIGH state, and one of them will be pulsed LOW whenever we want to change the latch outputs.
- We begin our analysis by showing that there are two equally likely output states when SET=RESET=1.
- One possibility is shown in Figure 5- 3(a), where we have Q=0 and ‘Q=1.
- With Q=0 , the inputs to NAND-2 are 0 and 1, which produce ‘Q=1.

NAND GATE LATCH

- The 1 from ‘Q causes NAND-1 to have a 1 at both inputs to produce a 0 output at *Q.*
- In effect, what we have is the LOW at the NAND-1 output producing a HIGH at the NAND-2 output, which, in turn, keeps the NAND-1 output LOW.
- The second possibility is shown in Figure 5-3(b), where Q=1 and ‘Q=0.
- The HIGH from NAND-1 produces a LOW at the NAND-2 output, which, in turn, keeps the NAND-1 output HIGH. Thus, there are two possible output states when SET=RESET=1.

NAND GATE LATCH

FIGURE 5-3 A NAND latch has two possible resting states when SET = RESET = 1.



NAND GATE LATCH (Setting the Latch)

- Now let’s investigate what happens when the SET input is momentarily pulsed LOW while RESET is kept HIGH.
- Figure 5-4(a) shows what happens when Q=0 prior to the occurrence of the pulse.
- As SET is pulsed LOW at time t_0 , *Q* will go HIGH, and this HIGH will force ‘Q to go LOW so that NAND-1 now has two LOW inputs.
- Thus, when SET returns to the 1 state at t_1 , the NAND-1 output *remains* HIGH, which, in turn, keeps the NAND-2 output LOW.

NAND GATE LATCH (Setting the Latch)

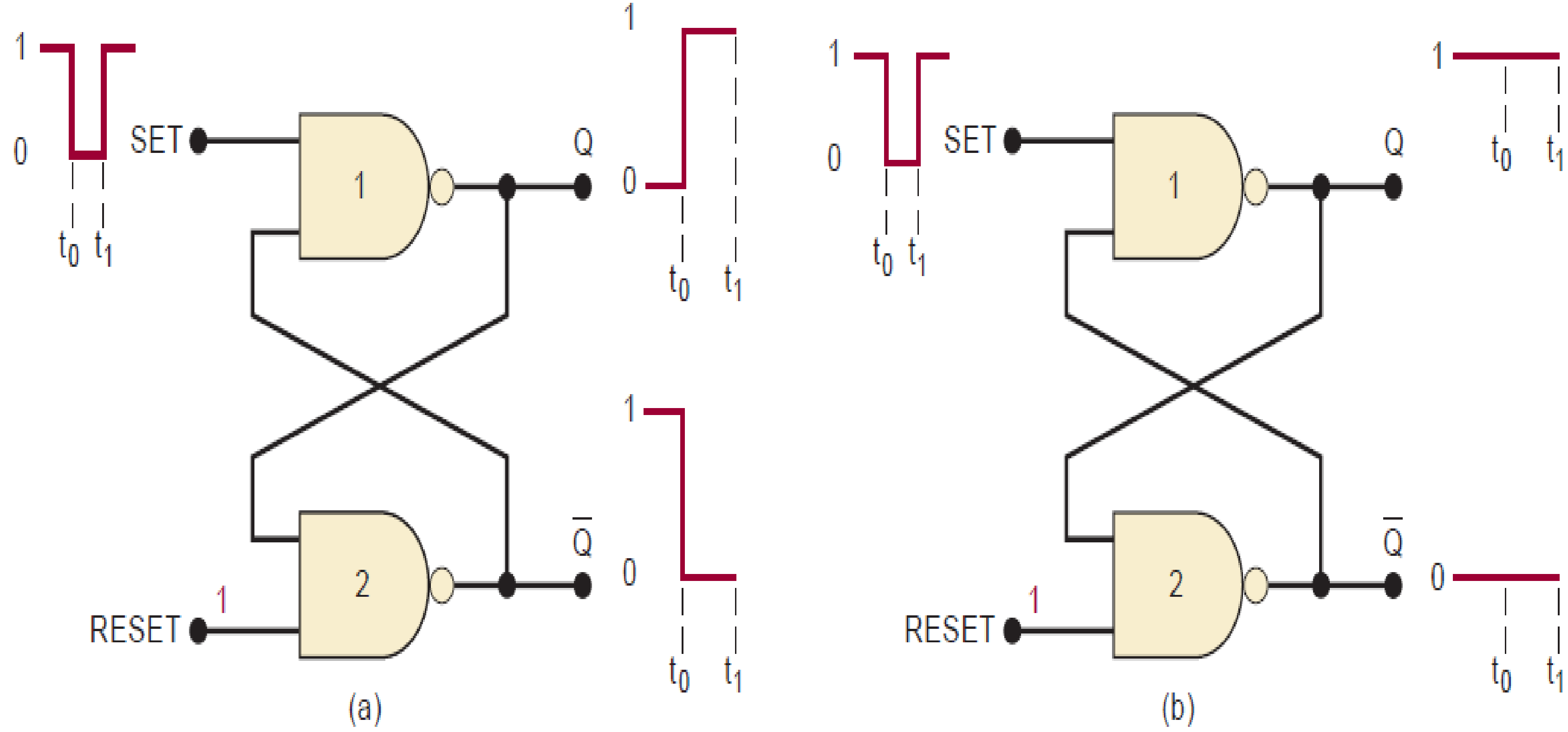


FIGURE 5-4 Pulsing the SET input to the 0 state when (a) $Q = 0$ prior to SET pulse; (b) $Q = 1$ prior to SET pulse. Note that, in both cases, Q ends up HIGH.

NAND GATE LATCH (Setting the Latch)

- Figure 5-4(b) shows what happens when Q=1 and ‘Q=0 prior to the application of the SET pulse.
- Since ‘Q=0 is already keeping the NAND-1 output HIGH, the LOW pulse at SET will not change anything.
- Thus, when SET returns HIGH, the latch outputs are still in the Q=1 and ‘Q=0 state.
- We can summarize Figure 5-4 by stating that a LOW pulse on the SET input will always cause the latch to end up in the Q=1 state.
- This operation is called *setting* the latch.

NAND GATE LATCH (Resetting the Latch)

- Now let’s consider what occurs when the RESET input is pulsed LOW while SET is kept HIGH.
- Figure 5-5(a) shows what happens when *Q* = 0 and ‘*Q* = 1 prior to the application of the pulse.
- Since Q=0 is already keeping the NAND-2 output HIGH, the LOW pulse at RESET will not have any effect.
- When RESET returns HIGH, the latch outputs are still *Q* = 0 and ‘*Q* = 1.
- Figure 5-5(b) shows the situation where Q=1 prior to the occurrence of the RESET pulse.

NAND GATE LATCH (Resetting the Latch)

- As RESET is pulsed LOW at t_0 , ‘Q will go HIGH, and this HIGH forces *Q* to go LOW so that NAND-2 now has two LOW inputs.
- Thus, when RESET returns HIGH at t_1 , the NAND-2 output *remains* HIGH, which, in turn, keeps the NAND-1 output LOW.
- Figure 5-5 can be summarized by stating that a LOW pulse on the RESET input will always cause the latch to end up in the Q=0 state.
- This operation is called *clearing* or *resetting* the latch.

NAND GATE LATCH (Resetting the Latch)

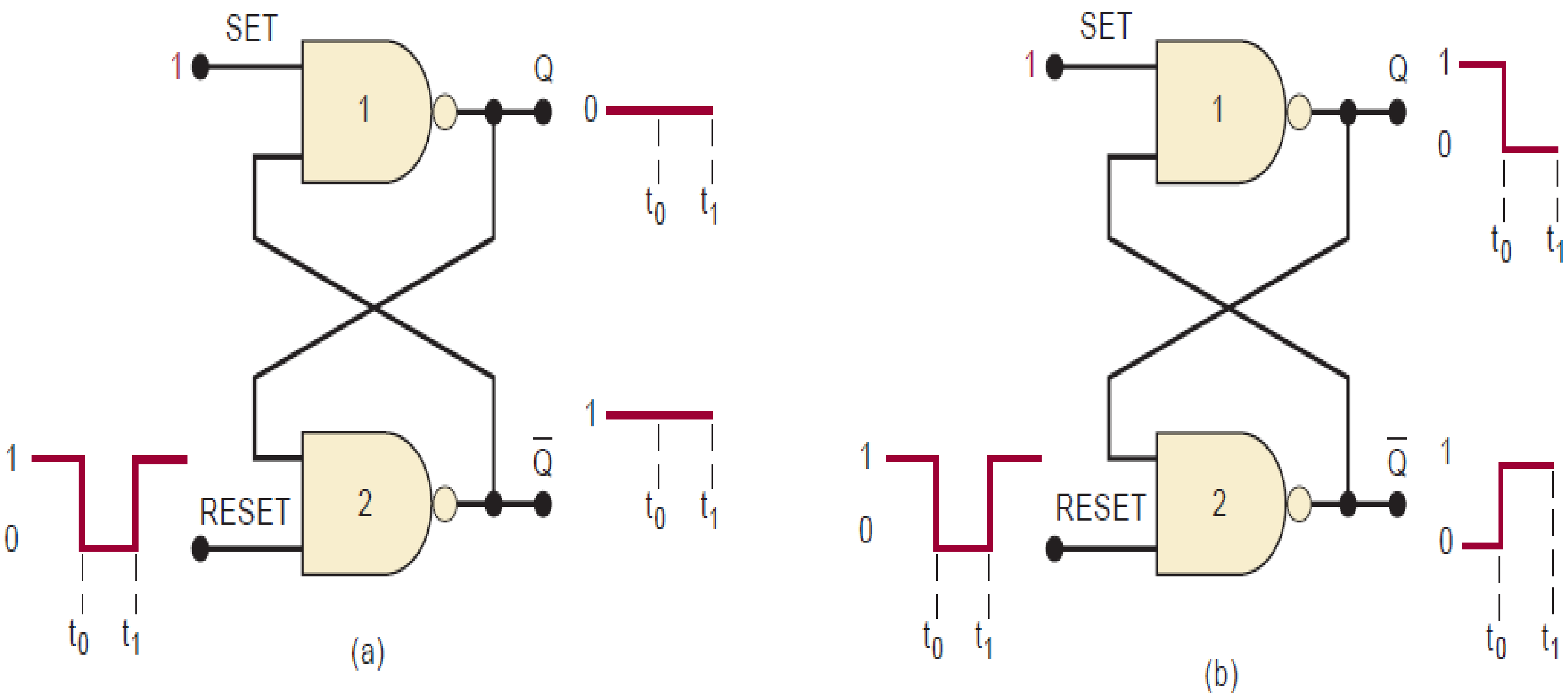


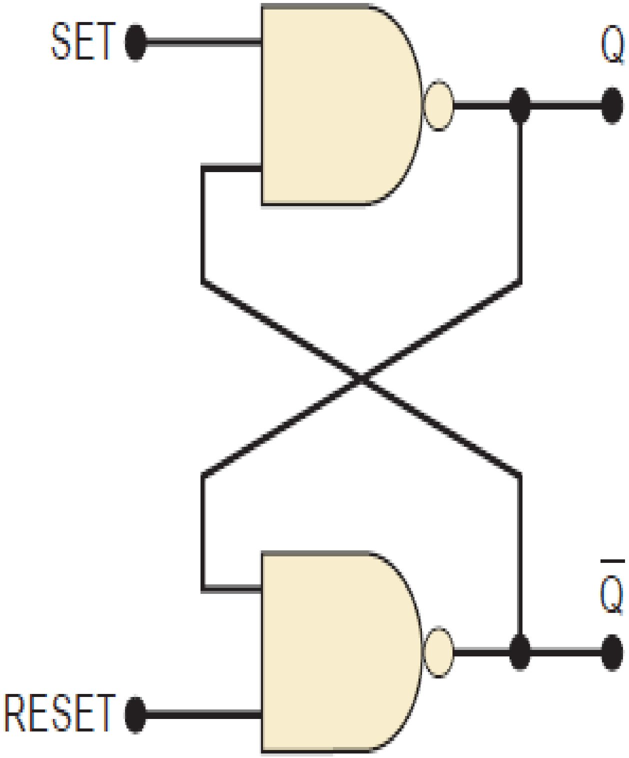
FIGURE 5-5 Pulsing the RESET input to the LOW state when (a) *Q* = 0 prior to RESET pulse; (b) *Q* = 1 prior to RESET pulse. In each case, *Q* ends up LOW.

NAND GATE LATCH (Summary of NAND Latch)

- 1. SET=RESET=1.This condition is the normal resting state, and it has no effect on the output state. The *Q* and ‘Q outputs will remain in whatever state they were in prior to this input condition.
- 2. SET=0,RESET=1 This will always cause the output to go to the Q=1 state, where it will remain even after SET returns HIGH. This is called *setting* the latch.
- 3. SET=1,RESET=0 This will always produce the state, where the output will remain even after RESET returns HIGH. This is called *clearing* or *resetting* the latch.
- 4. SET=0,RESET=0 This condition tries to set and clear the latch at the same time, and it produces Q=‘Q=1 If the inputs are returned to 1 simultaneously, the resulting state is unpredictable. This input condition should not be used.

NAND GATE LATCH

FIGURE 5-6 (a) NAND latch; (b) function table.



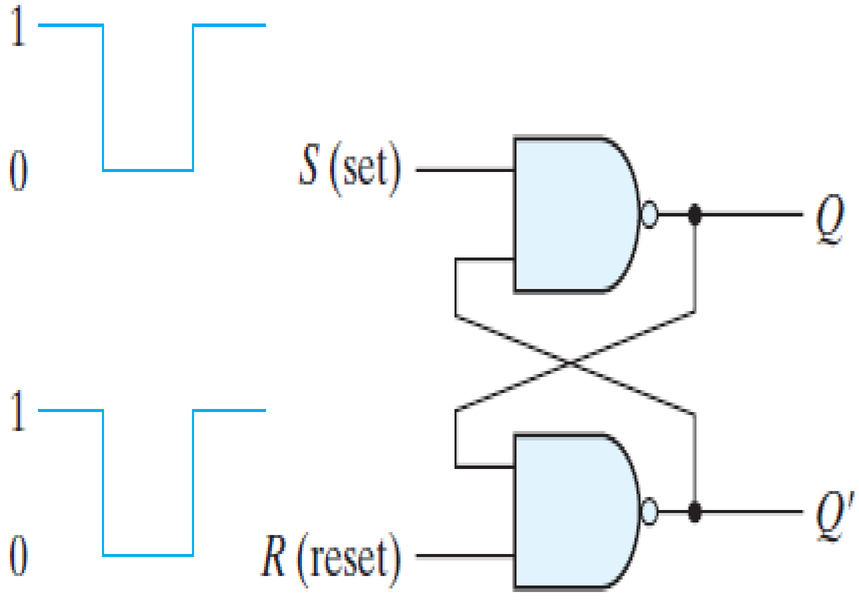
(a)

Set	Reset	Output
1	1	No change
0	1	Q = 1
1	0	Q = 0
0	0	Invalid*

*Produces $Q = \overline{Q} = 1$.

(b)

NAND GATE LATCH



(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

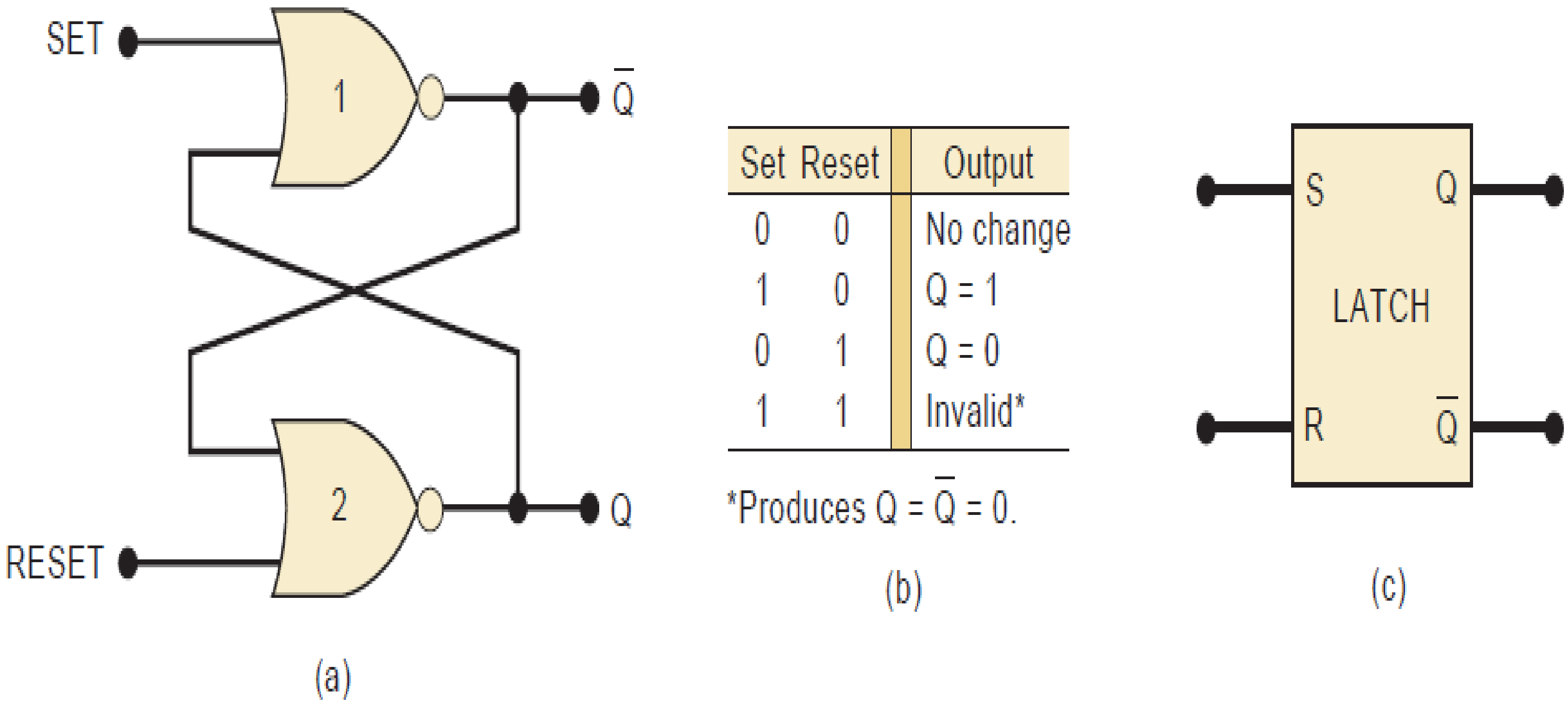
FIGURE 5.4

SR latch with NAND gates

NOR GATE LATCH

- Two cross-coupled NOR gates can be used as a **NOR gate latch**.
- The arrangement, shown in Figure 5-10(a), is similar to the NAND latch except that the *Q* and ‘*Q* outputs have reversed positions.
- The analysis of the operation of the NOR latch can be performed in exactly the same manner as for the NAND latch.
- The results are given in the function table in Figure 5-10(b) and are summarized as follows:

NOR GATE LATCH



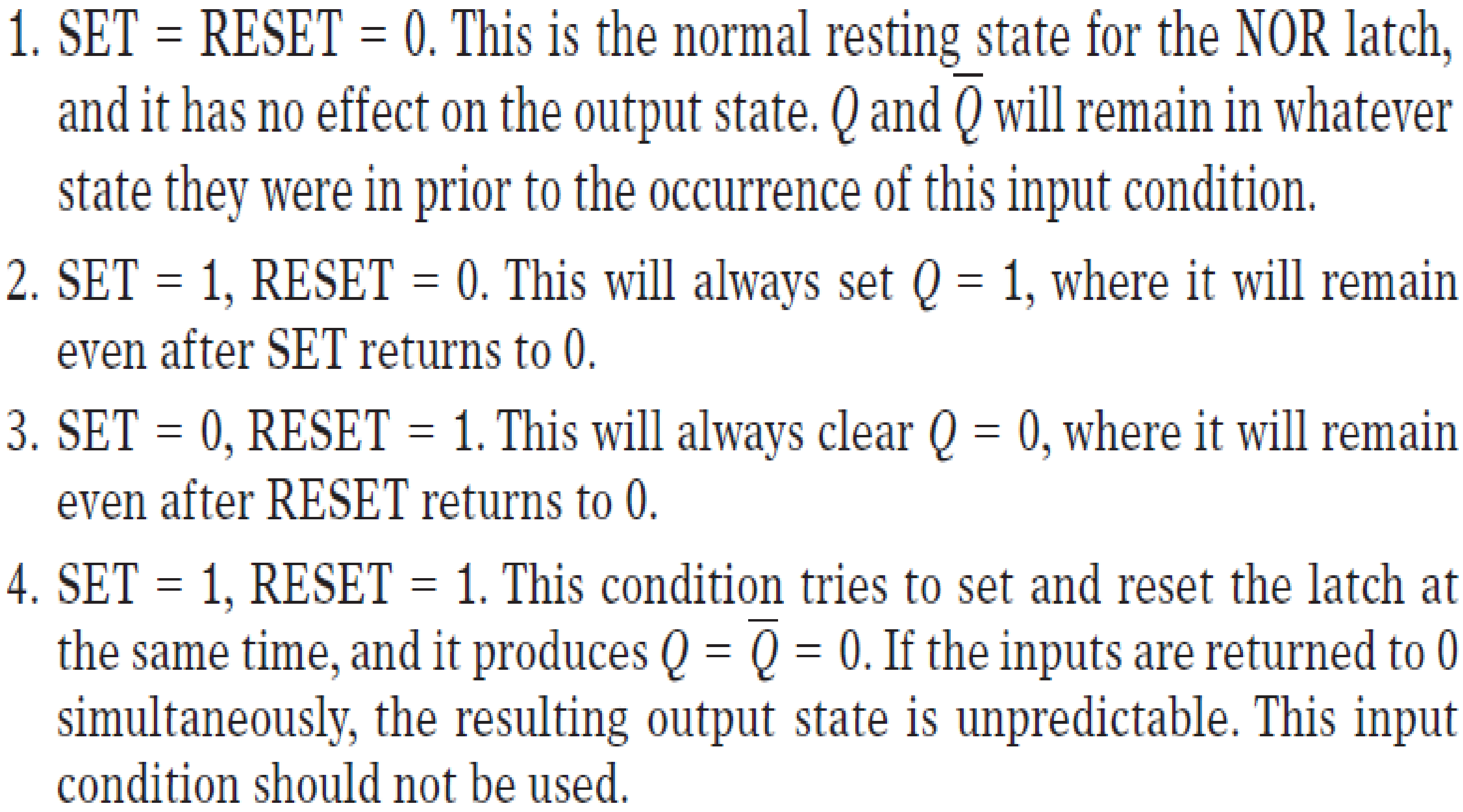
Set	Reset	Output
0	0	No change
1	0	Q = 1
0	1	Q = 0
1	1	Invalid*

*Produces Q = Q̄ = 0.

(b)

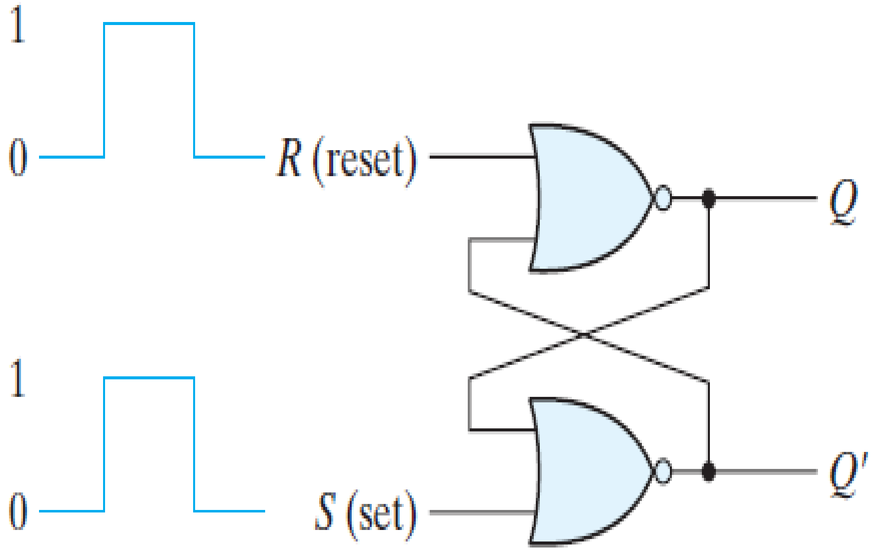
FIGURE 5-10 (a) NOR gate latch; (b) function table; (c) simplified block symbol.

NOR GATE LATCH



1. SET = RESET = 0. This is the normal resting state for the NOR latch, and it has no effect on the output state. Q and $\overline{Q}$ will remain in whatever state they were in prior to the occurrence of this input condition.
2. SET = 1, RESET = 0. This will always set $Q = 1$, where it will remain even after SET returns to 0.
3. SET = 0, RESET = 1. This will always clear $Q = 0$, where it will remain even after RESET returns to 0.
4. SET = 1, RESET = 1. This condition tries to set and reset the latch at the same time, and it produces $Q = \overline{Q} = 0$. If the inputs are returned to 0 simultaneously, the resulting output state is unpredictable. This input condition should not be used.

NOR GATE LATCH



(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

NAND GATE LATCH

- 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.**

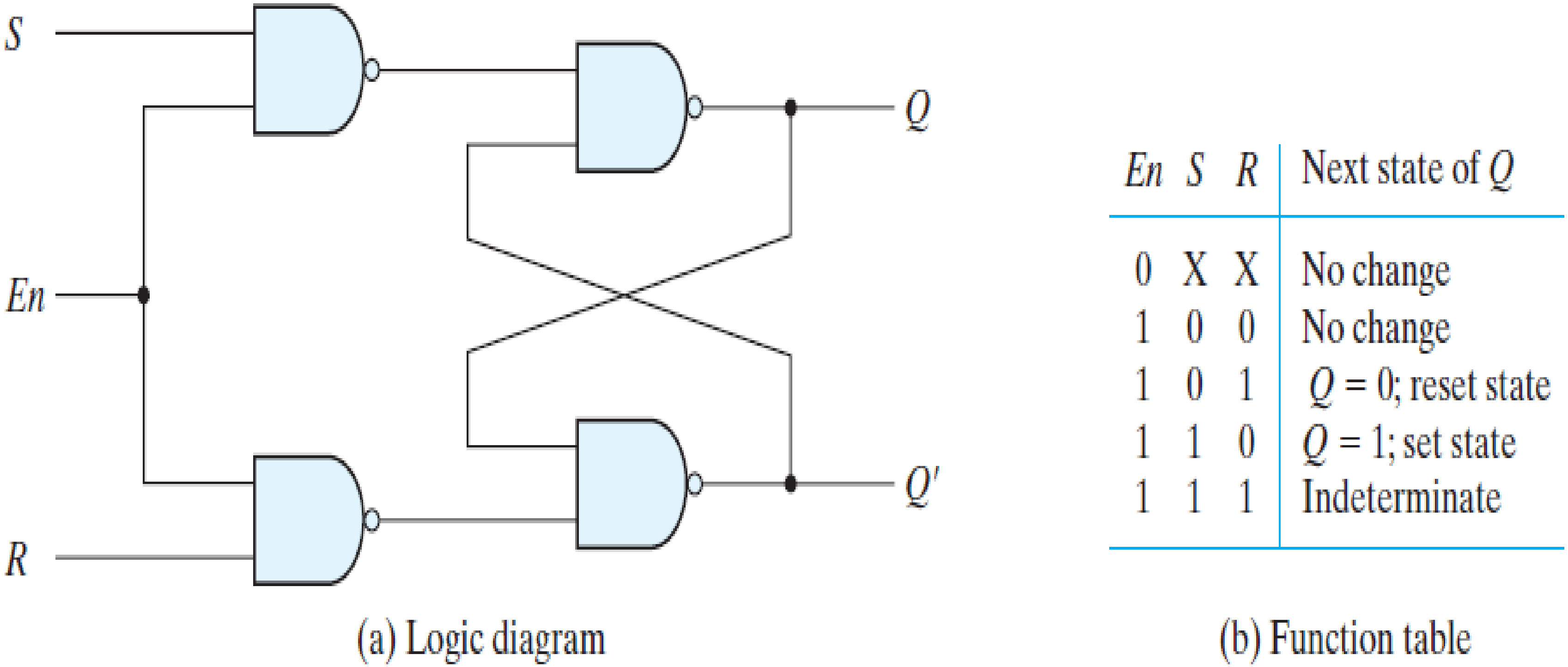
NAND GATE LATCH

- 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.
- 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* .

NAND GATE LATCH

- 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.

NAND GATE LATCH



(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

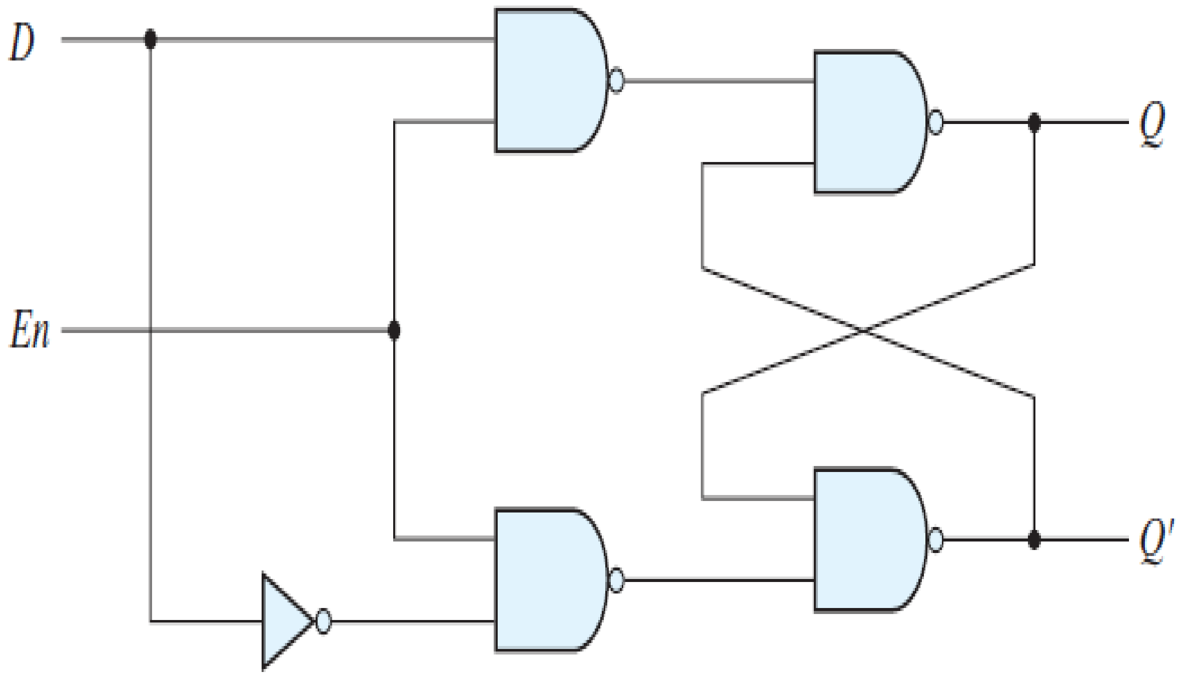
FIGURE 5.5

SR latch with control input

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 *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.

D Latch (Transparent Latch)



(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

D Latch (Transparent Latch)

- 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.

D Latch (Transparent Latch)

- 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 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 .

D Latch (Transparent Latch)

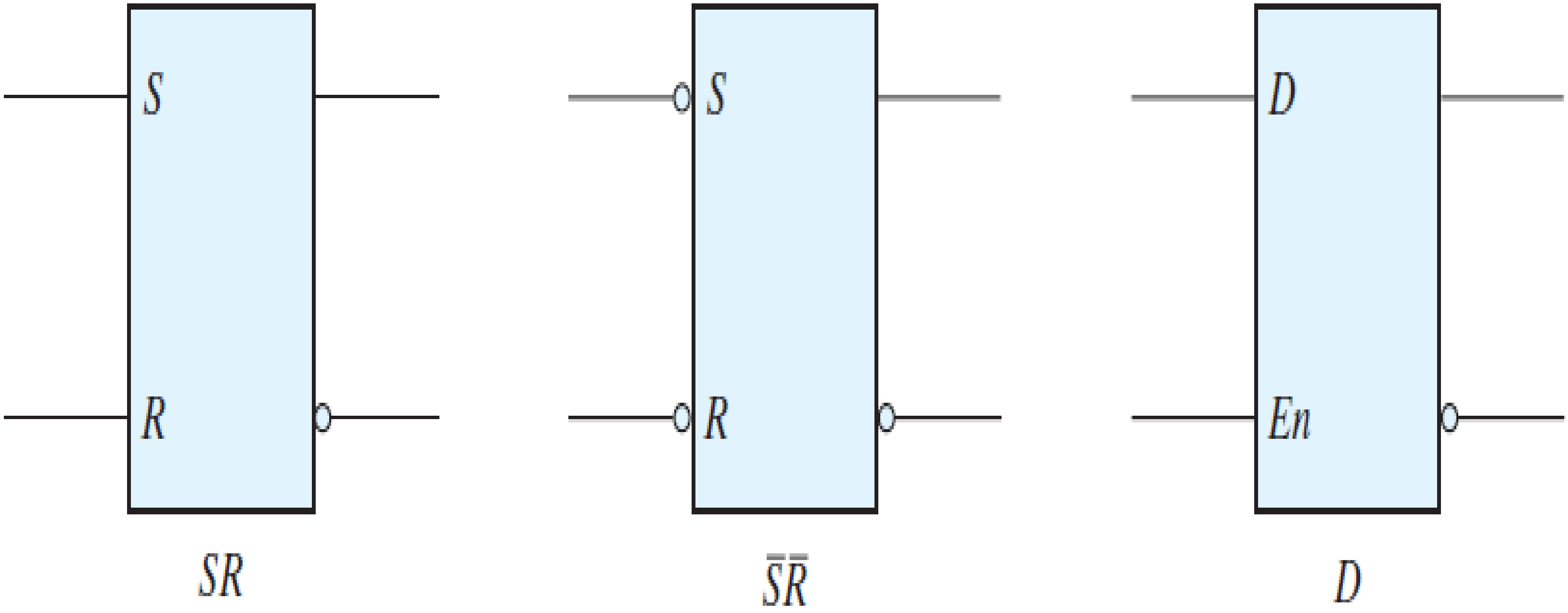


FIGURE 5.7

Graphic symbols for latches

D Latch (Transparent Latch)

- 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.

STORAGE ELEMENTS: FLIP-FLOPS

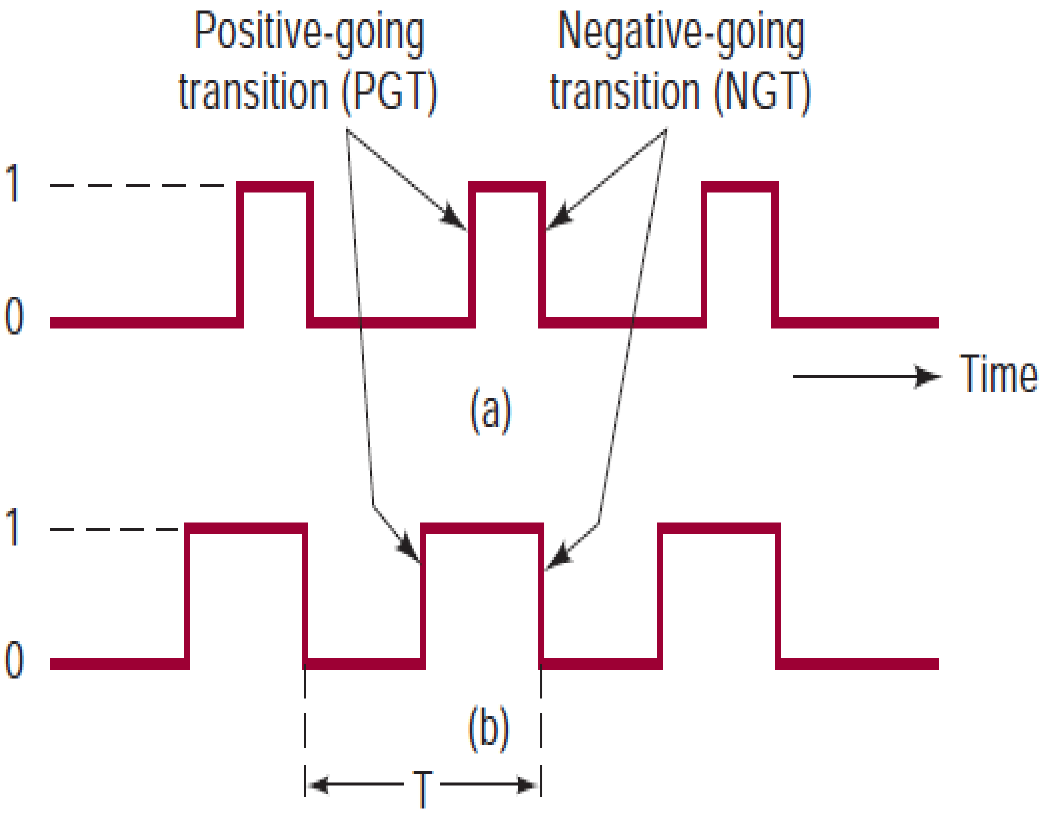
- Digital systems can operate either *asynchronously* or *synchronously.*
- In asynchronous systems, the outputs of logic circuits can change state any time one or more of the inputs change.
- An asynchronous system is generally more difficult to design and troubleshoot than a synchronous system.
- In synchronous systems, the exact times at which any output can change states are determined by a signal commonly called the **clock**.
- This clock signal is generally a rectangular pulse train or a square wave, as shown in Figure 5-16.

STORAGE ELEMENTS: FLIP-FLOPS

- The clock signal is distributed to all parts of the system, and most (if not all) of the system outputs can change state only when the clock makes a transition.
- The transitions (also called *edges*) are pointed out in Figure 5-16.
- When the clock changes from a 0 to a 1, this is called the **positive-going transition (PGT)**; when the clock goes from 1 to 0, this is the **negative-going transition (NGT)**.
- Most digital systems are principally synchronous because synchronous circuits are easier to design and troubleshoot because the circuit outputs can change only at specific instants of time. In other words, almost everything is synchronized to the clock-signal transitions.

STORAGE ELEMENTS: FLIP-FLOPS

FIGURE 5-16 Clock signals.


Positive-going transition (PGT)
Negative-going transition (NGT)
1
0
Time
(a)
1
0
(b)
T

STORAGE ELEMENTS: FLIP-FLOPS

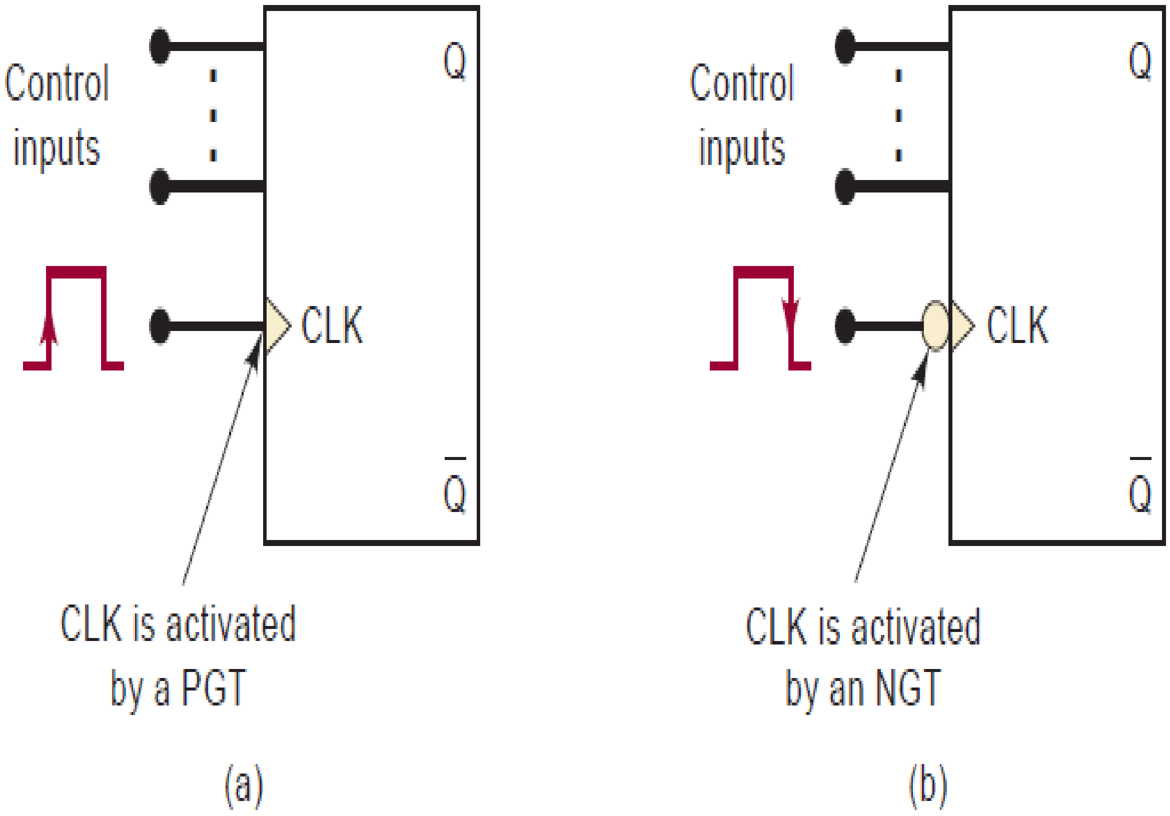
- Clocked FFs have a clock input that is typically labeled *CLK, CK,* or *CP.*
- We will normally use *CLK,* as shown in Figure 5-17. In most clocked FFs, the *CLK* input is **edge-triggered**, which means that it is activated by a signal transition; this is indicated by the presence of a small triangle on the *CLK* input.
- This contrasts with the latches, which are level-triggered.
- Figure 5-17(a) is a FF with a small triangle on its *CLK* input to indicate that this input is activated *only* when a positive-going transition (PGT) occurs; no other part of the input pulse will have an effect on the *CLK* input. I

STORAGE ELEMENTS: FLIP-FLOPS

- n Figure 5-17(b), the FF symbol has a bubble as well as a triangle on its *CLK* input.
- This signifies that the *CLK* input is activated *only* when a negative-going transition occurs; no other part of the input pulse will have an effect on the *CLK* input.
- The control inputs will have no effect on *Q* until the active clock transition occurs.
- In other words, their effect is synchronized with the signal applied to *CLK.*
- For this reason they are called **synchronous control inputs**.

STORAGE ELEMENTS: FLIP-FLOPS

FIGURE 5-17 Clocked FFs have a clock input (*CLK*) that is active on either (a) the PGT or (b) the NGT. The control inputs determine the effect of the active clock transition.



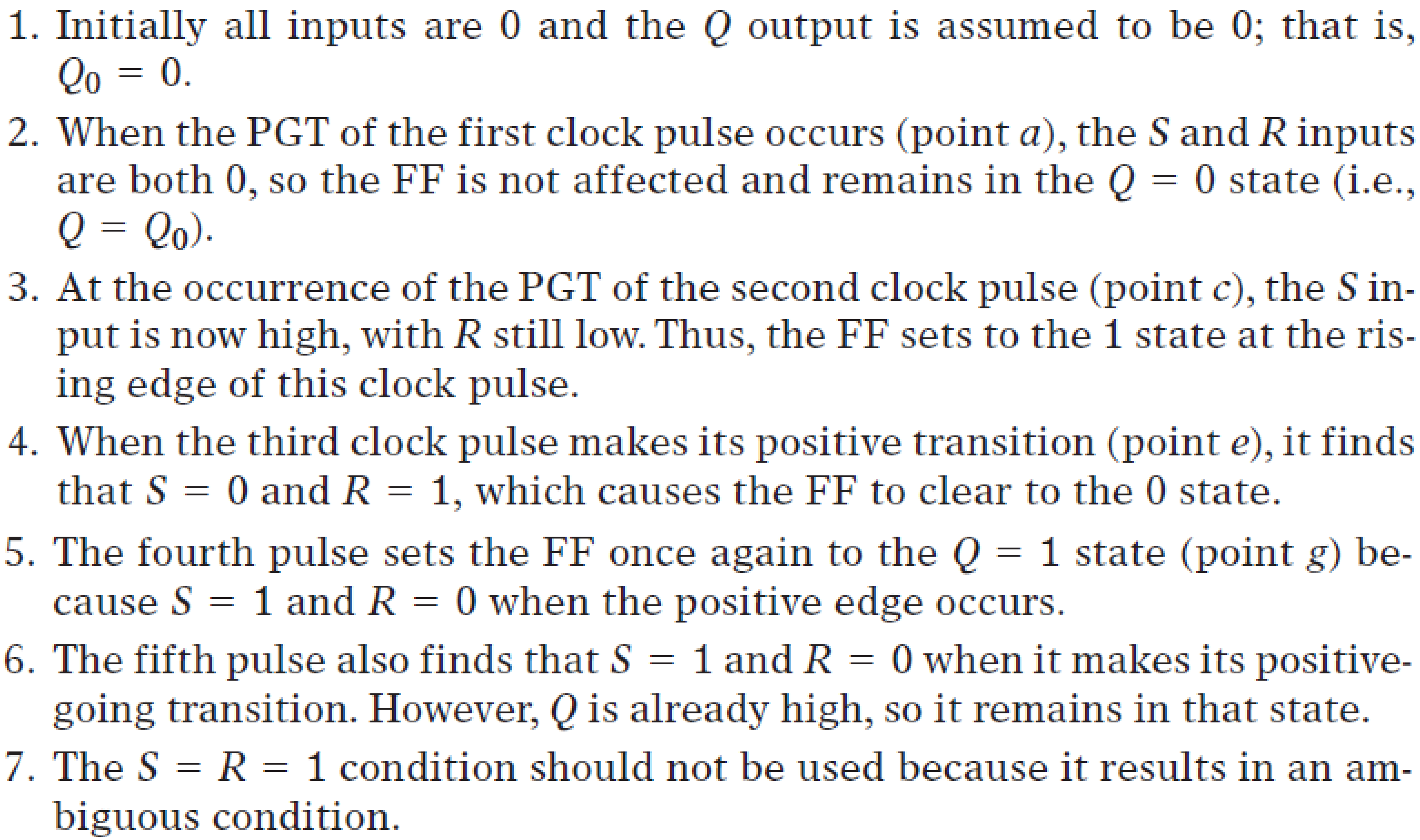
CLOCKED S-R FLIP-FLOP

- Figure 5-19(a) shows the logic symbol for a **clocked S-R flip-flop** that is triggered by the positive-going edge of the clock signal.
- This means that the FF can change states *only* when a signal applied to its clock input makes a transition from 0 to 1.
- The *S* and *R* inputs control the state of the FF in the same manner as described earlier for the NOR gate latch, but the FF does not respond to these inputs until the occurrence of the PGT of the clock signal.
- The function table in Figure 5-19(b) shows how the FF output will respond to the PGT at the *CLK* input for the various combinations of *S* and *R* inputs.

CLOCKED S-R FLIP-FLOP

- The up arrow indicates that a PGT is required at *CLK*; the label indicates the level at Q_0 prior to the PGT.
- This nomenclature is often used by IC manufacturers in their IC data manuals.
- The waveforms in Figure 5-19(c) illustrate the operation of the clocked S-R flip-flop.
- we can analyze these waveforms as follows:

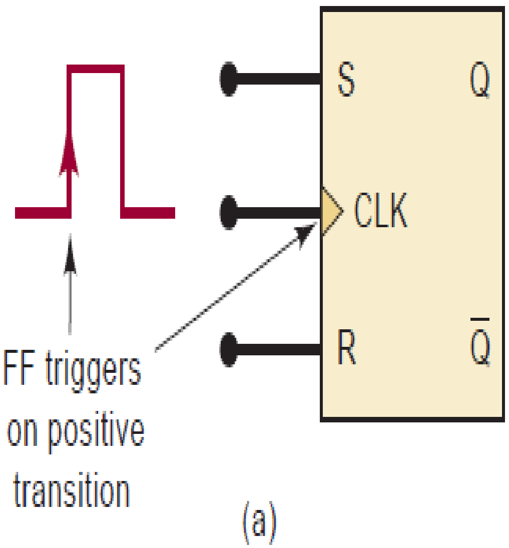
CLOCKED S-R FLIP-FLOP



1. Initially all inputs are 0 and the Q output is assumed to be 0; that is, $Q_0 = 0$.
2. When the PGT of the first clock pulse occurs (point a), the S and R inputs are both 0, so the FF is not affected and remains in the $Q = 0$ state (i.e., $Q = Q_0$).
3. At the occurrence of the PGT of the second clock pulse (point c), the S input is now high, with R still low. Thus, the FF sets to the 1 state at the rising edge of this clock pulse.
4. When the third clock pulse makes its positive transition (point e), it finds that $S = 0$ and $R = 1$, which causes the FF to clear to the 0 state.
5. The fourth pulse sets the FF once again to the $Q = 1$ state (point g) because $S = 1$ and $R = 0$ when the positive edge occurs.
6. The fifth pulse also finds that $S = 1$ and $R = 0$ when it makes its positive-going transition. However, Q is already high, so it remains in that state.
7. The $S = R = 1$ condition should not be used because it results in an ambiguous condition.

CLOCKED S-R FLIP-FLOP

FIGURE 5-19 (a) Clocked S-R flip-flop that responds only to the positive-going edge of a clock pulse; (b) function table; (c) typical waveforms.



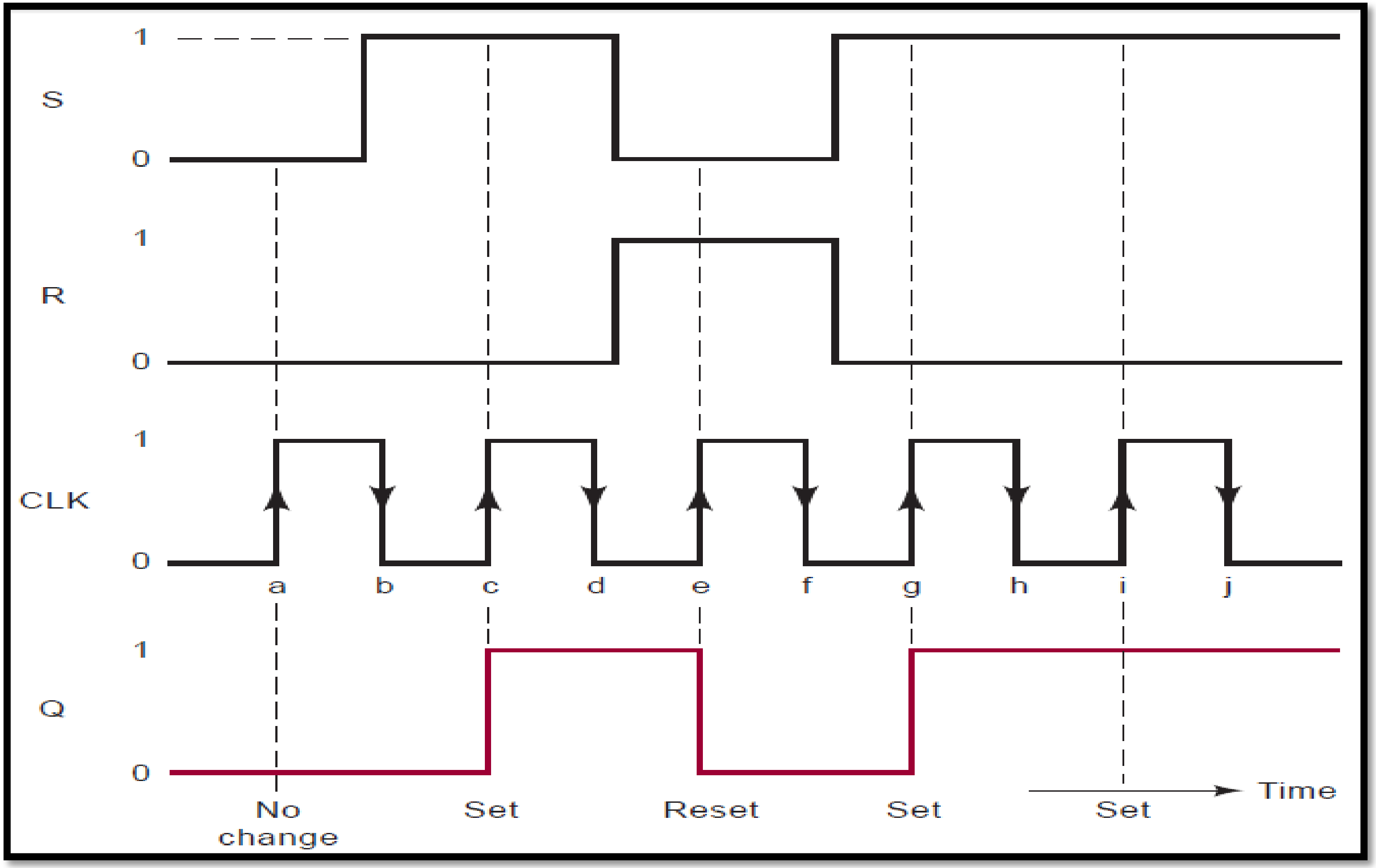
Inputs			Output
S	R	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	Ambiguous

Q_0 is output level prior to ↑ of CLK.

↓ of CLK produces no change in Q.

(b)

CLOCKED S-R FLIP-FLOP



CLOCKED S-R FLIP-FLOP

- Figure 5-20 shows the symbol and the function table for a clocked S-R flip-flop that triggers on the *negative*-going transition at its *CLK* input.
- The small circle and triangle on the *CLK* input indicates that this FF will trigger only when the *CLK* input goes from 1 to 0.
- This FF operates in the same manner as the positive-edge FF except that the output can change states only on the falling edge of the clock pulses (points *b*, *d*, *f*, *h*, and *j* in Figure 5-19).
- Both positive-edge and negative-edge triggering FFs are used in digital systems.

CLOCKED S-R FLIP-FLOP

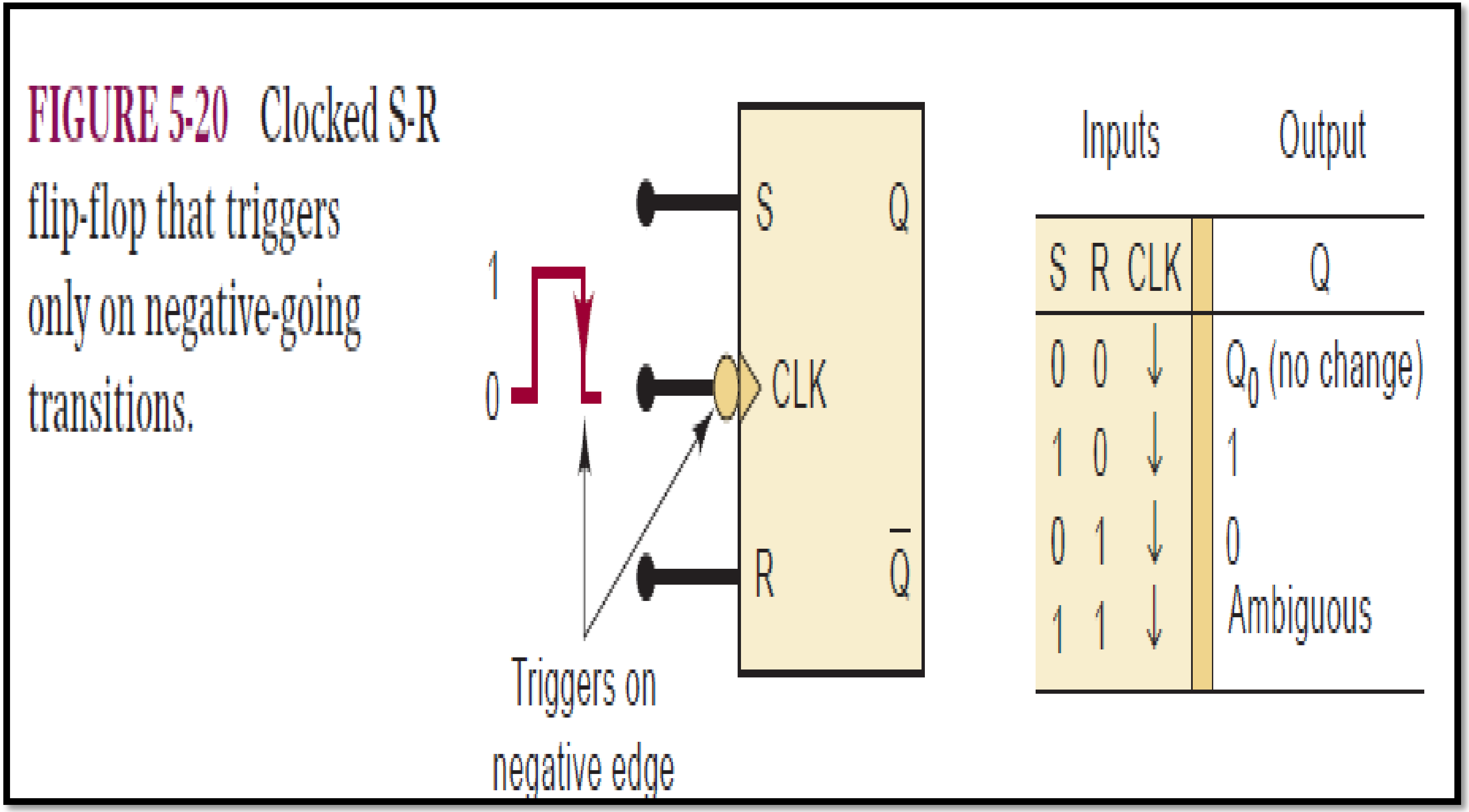


FIGURE 5-20 Clocked S-R flip-flop that triggers only on negative-going transitions.

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

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.

Edge-Triggered *D* Flip-Flop

- 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.

Edge-Triggered *D* Flip-Flop

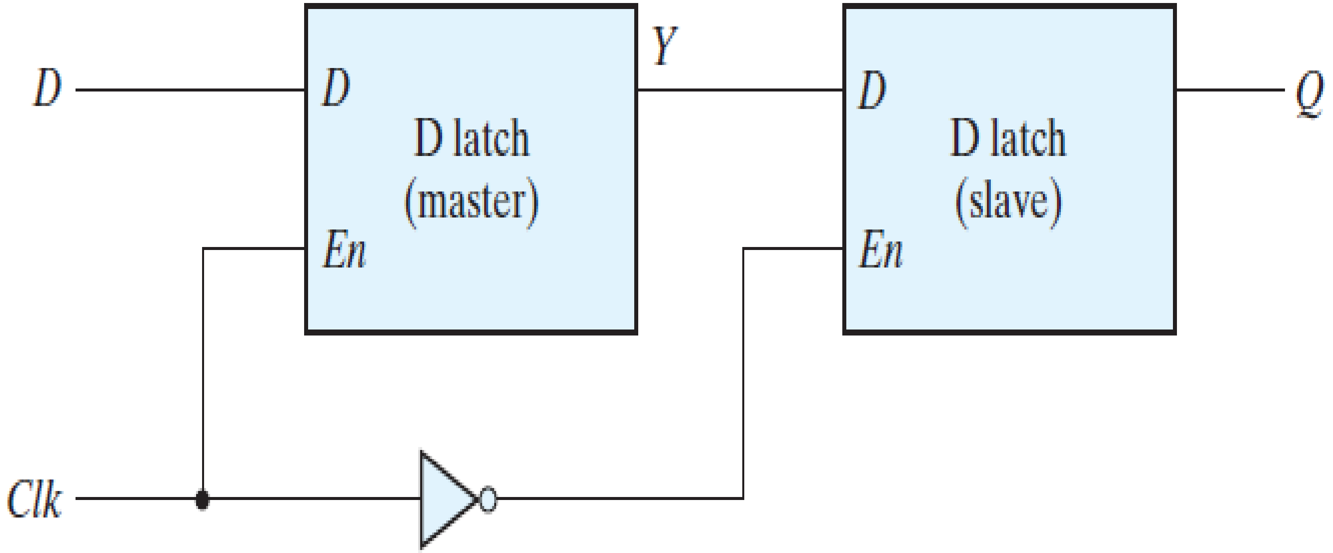


FIGURE 5.9
Master–slave *D* flip-flop

Edge-Triggered *D* Flip-Flop

- 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 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.

Edge-Triggered *D* Flip-Flop

- 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 inactive 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* .

Edge-Triggered *D* Flip-Flop

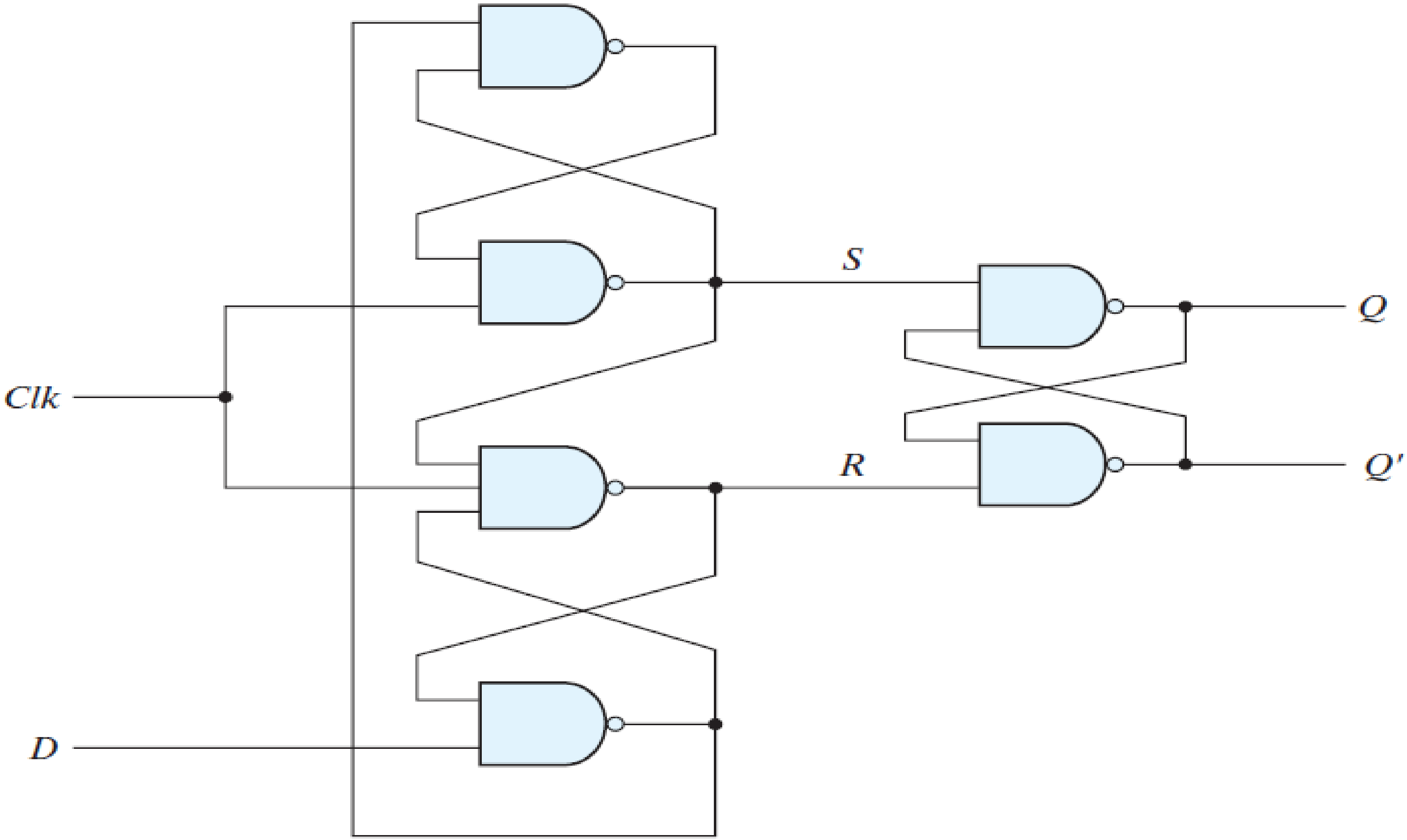
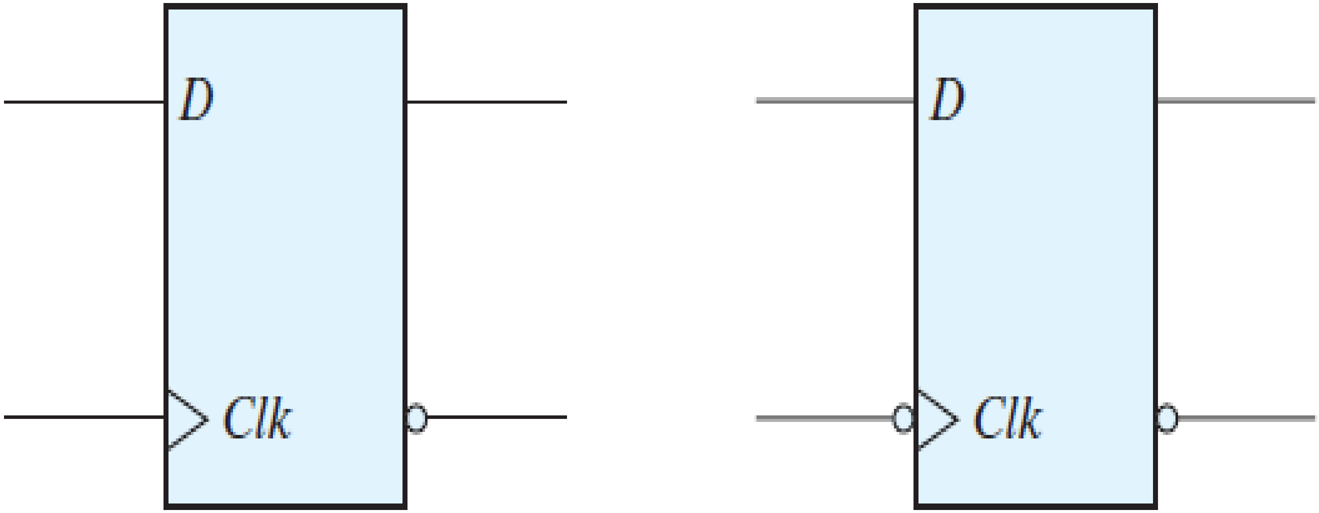


FIGURE 5.10
D-type positive-edge-triggered flip-flop

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.

Edge-Triggered *D* Flip-Flop



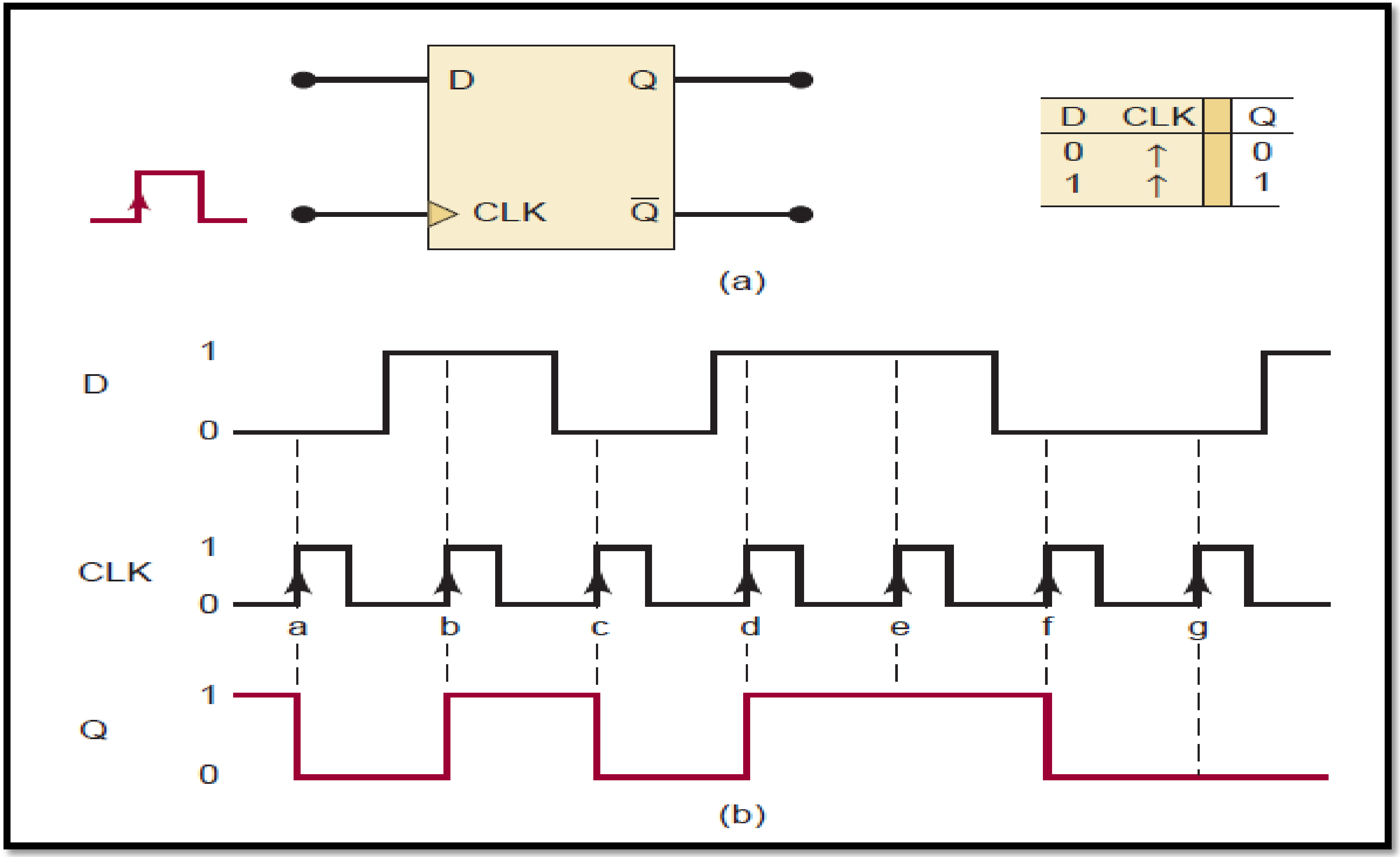
(a) Positive-edge

(a) Negative-edge

FIGURE 5.11

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

Edge-Triggered *D* Flip-Flop



D	CLK	Q
0	↑	0
1	↑	1

(a)

(b)

Other Flip-Flops

- 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.

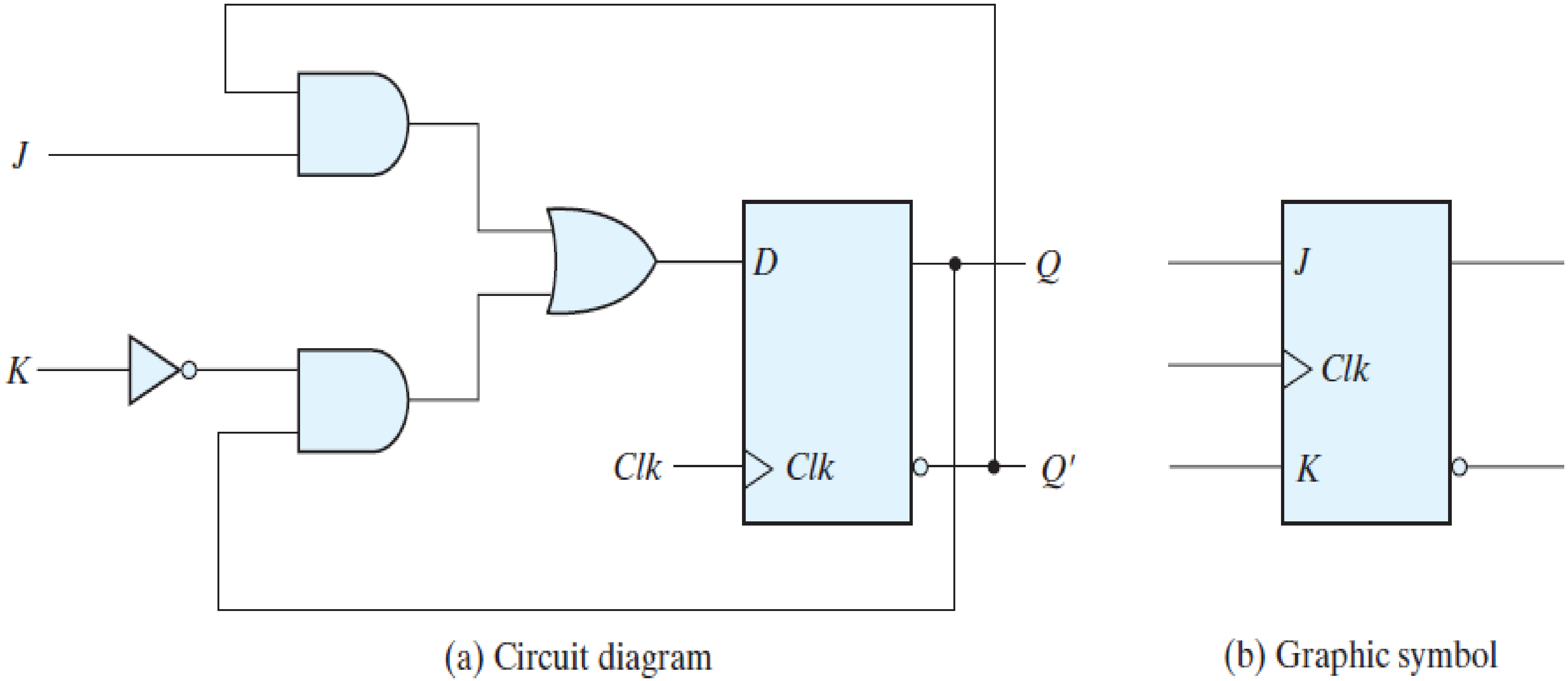
J-K FF

- 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**

J-K FF

- 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* .

J-K FF

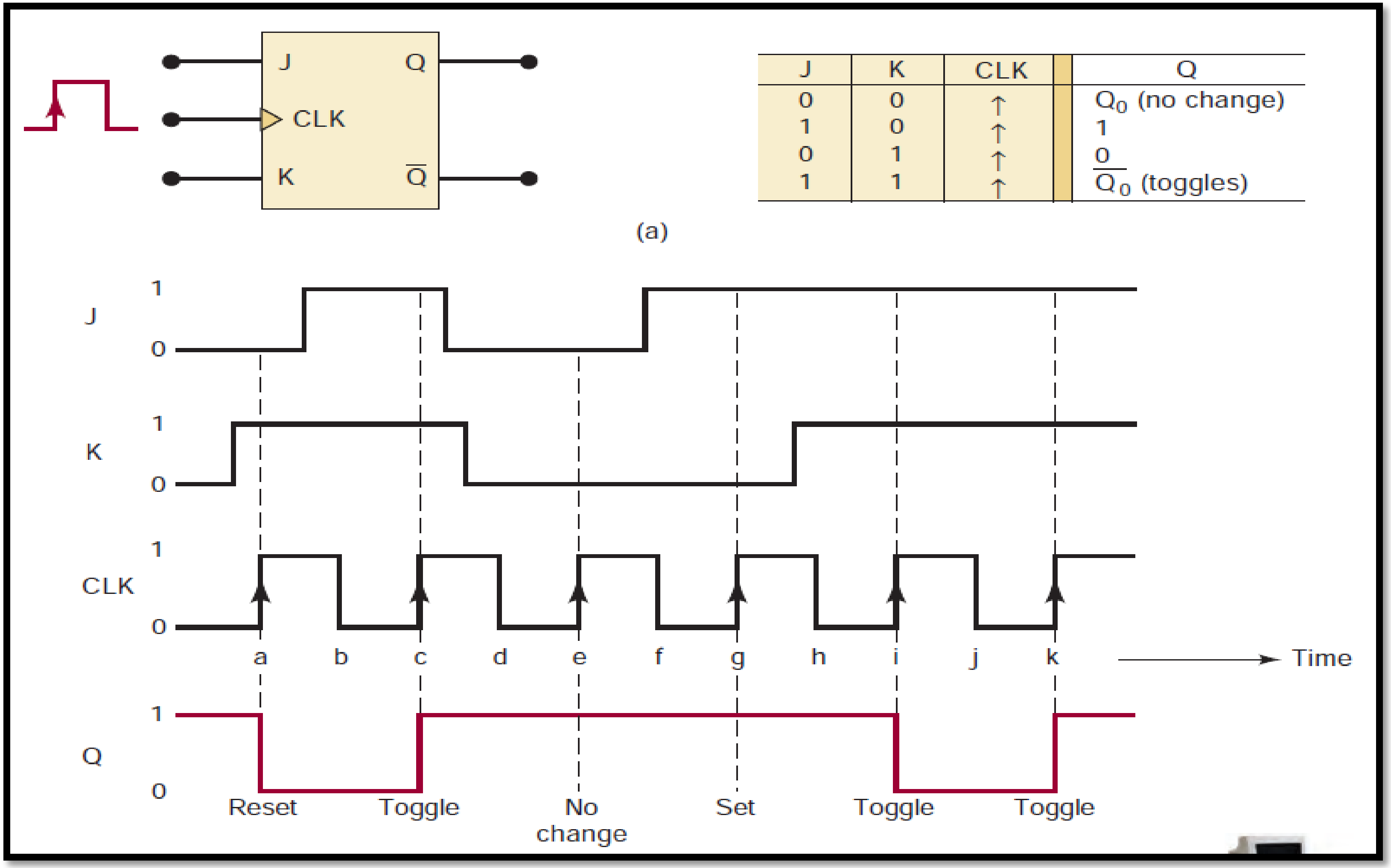


(a) Circuit diagram

(b) Graphic symbol

FIGURE 5.12
JK flip-flop

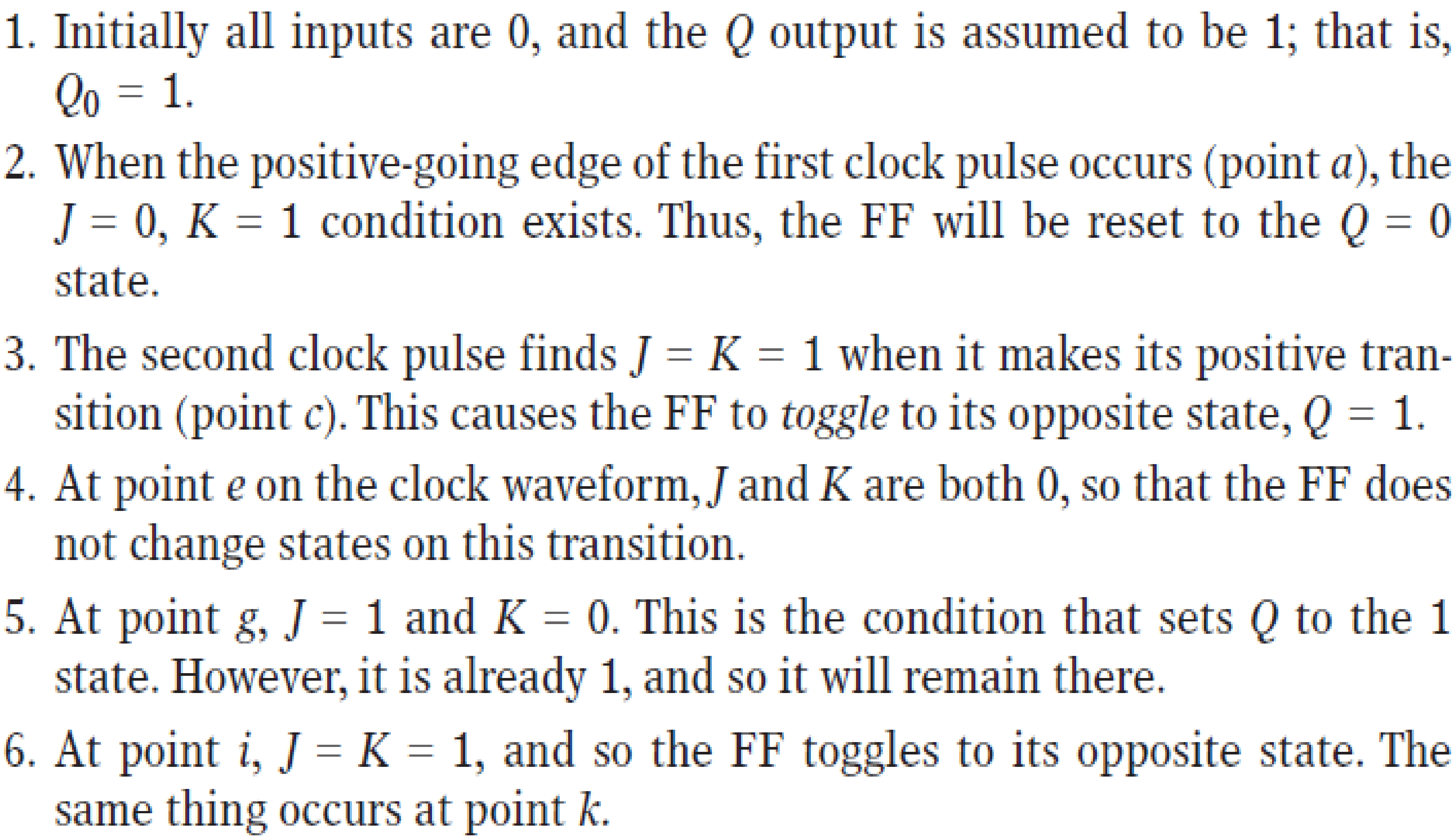
J-K FF



J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

(a)

J-K FF

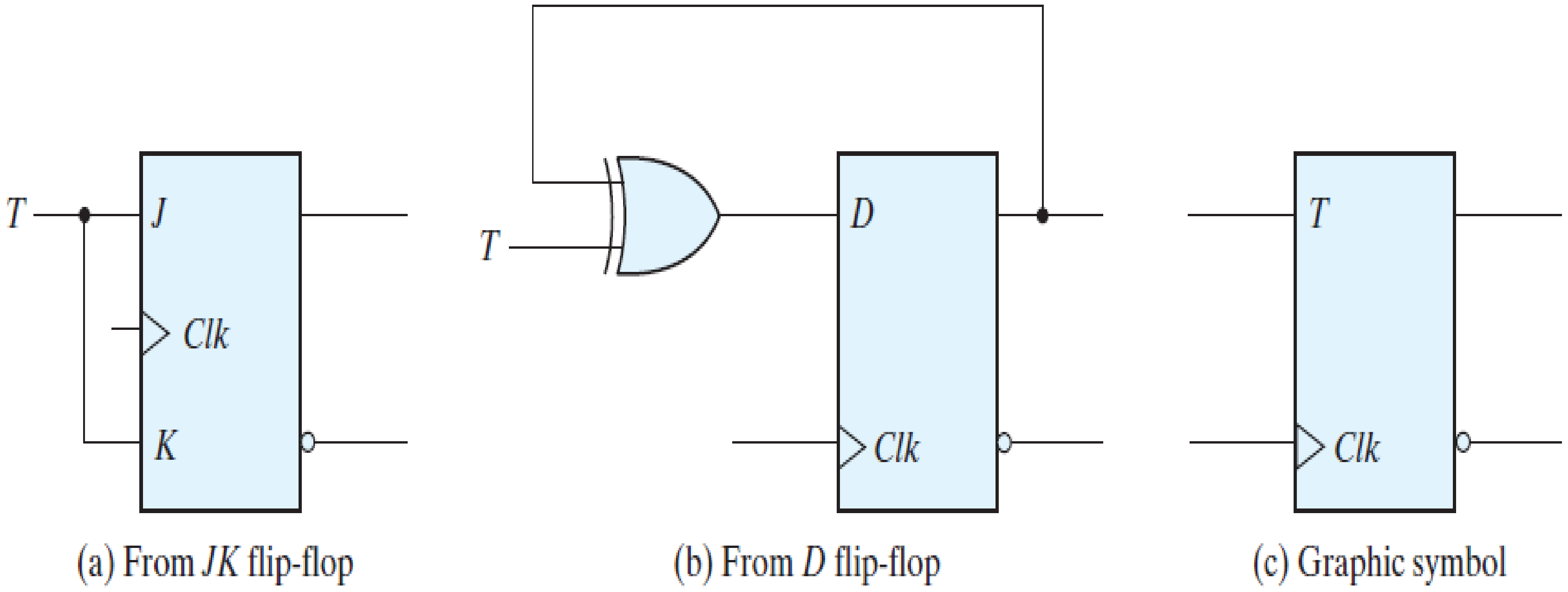


1. Initially all inputs are 0, and the Q output is assumed to be 1; that is, $Q_0 = 1$.
2. When the positive-going edge of the first clock pulse occurs (point *a*), the $J = 0$, $K = 1$ condition exists. Thus, the FF will be reset to the $Q = 0$ state.
3. The second clock pulse finds $J = K = 1$ when it makes its positive transition (point *c*). This causes the FF to *toggle* to its opposite state, $Q = 1$.
4. At point *e* on the clock waveform, J and K are both 0, so that the FF does not change states on this transition.
5. At point *g*, $J = 1$ and $K = 0$. This is the condition that sets Q to the 1 state. However, it is already 1, and so it will remain there.
6. At point *i*, $J = K = 1$, and so the FF toggles to its opposite state. The same thing occurs at point *k*.

T Flip-Flops

- 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).
- 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* **X-OR** *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.

T Flip-Flops



(a) From *JK* flip-flop

(b) From *D* flip-flop

(c) Graphic symbol

FIGURE 5.13

T flip-flop

Characteristic Tables

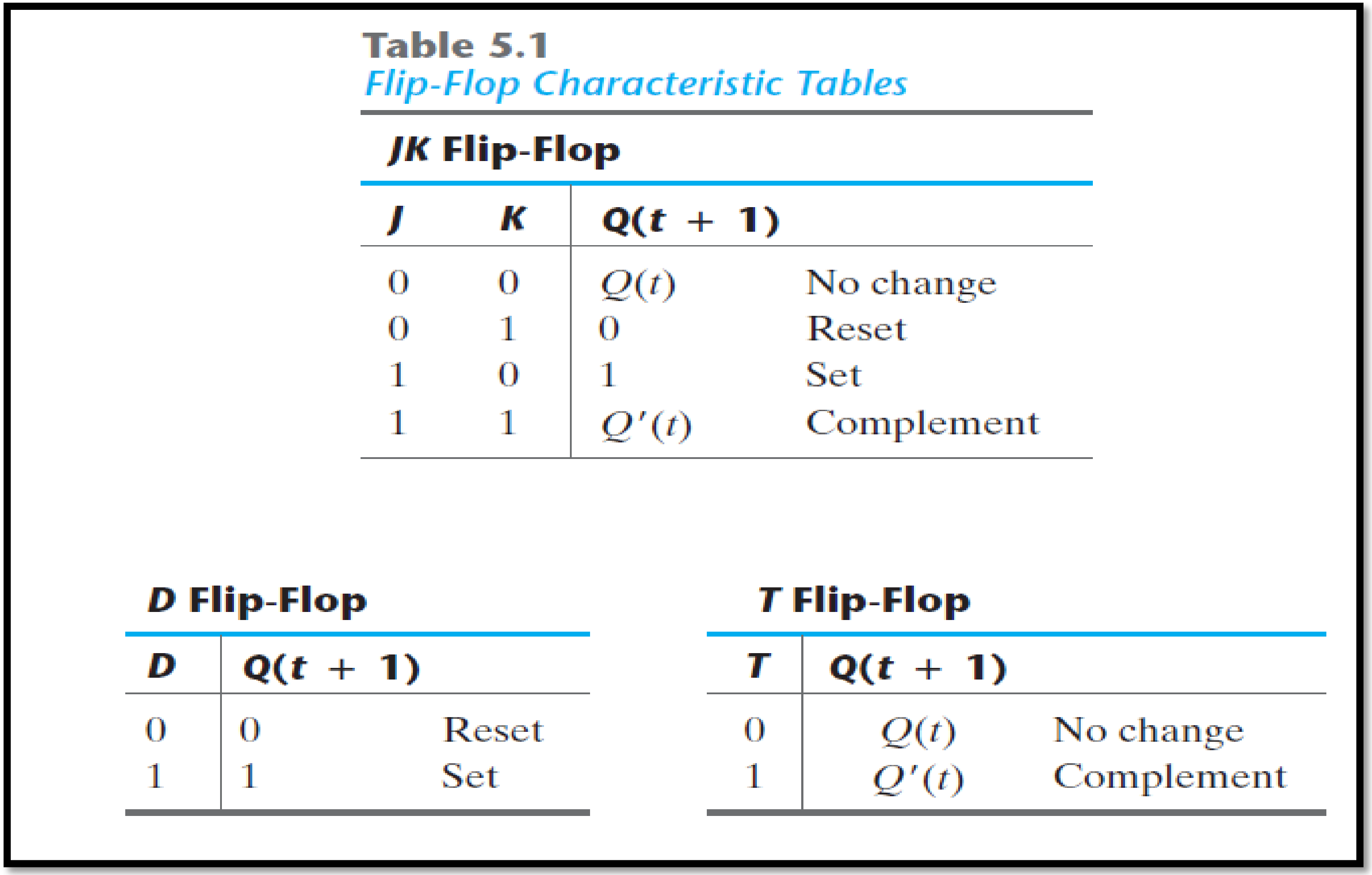


Table 5.1
Flip-Flop Characteristic Tables

***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

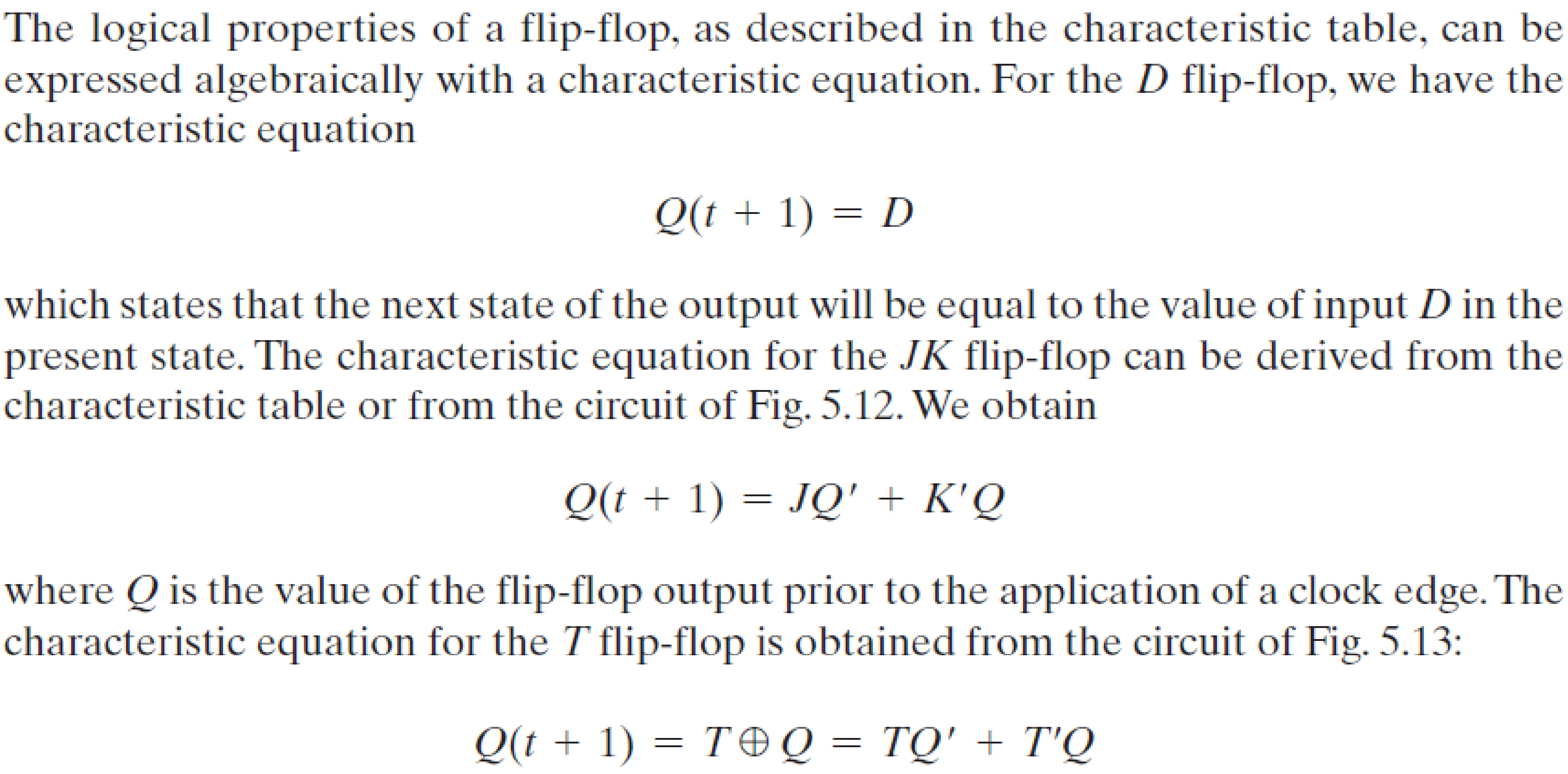
***D* Flip-Flop**

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

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:

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

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.

ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS

- 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.

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 .
- 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:

State Equations

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

State Equations

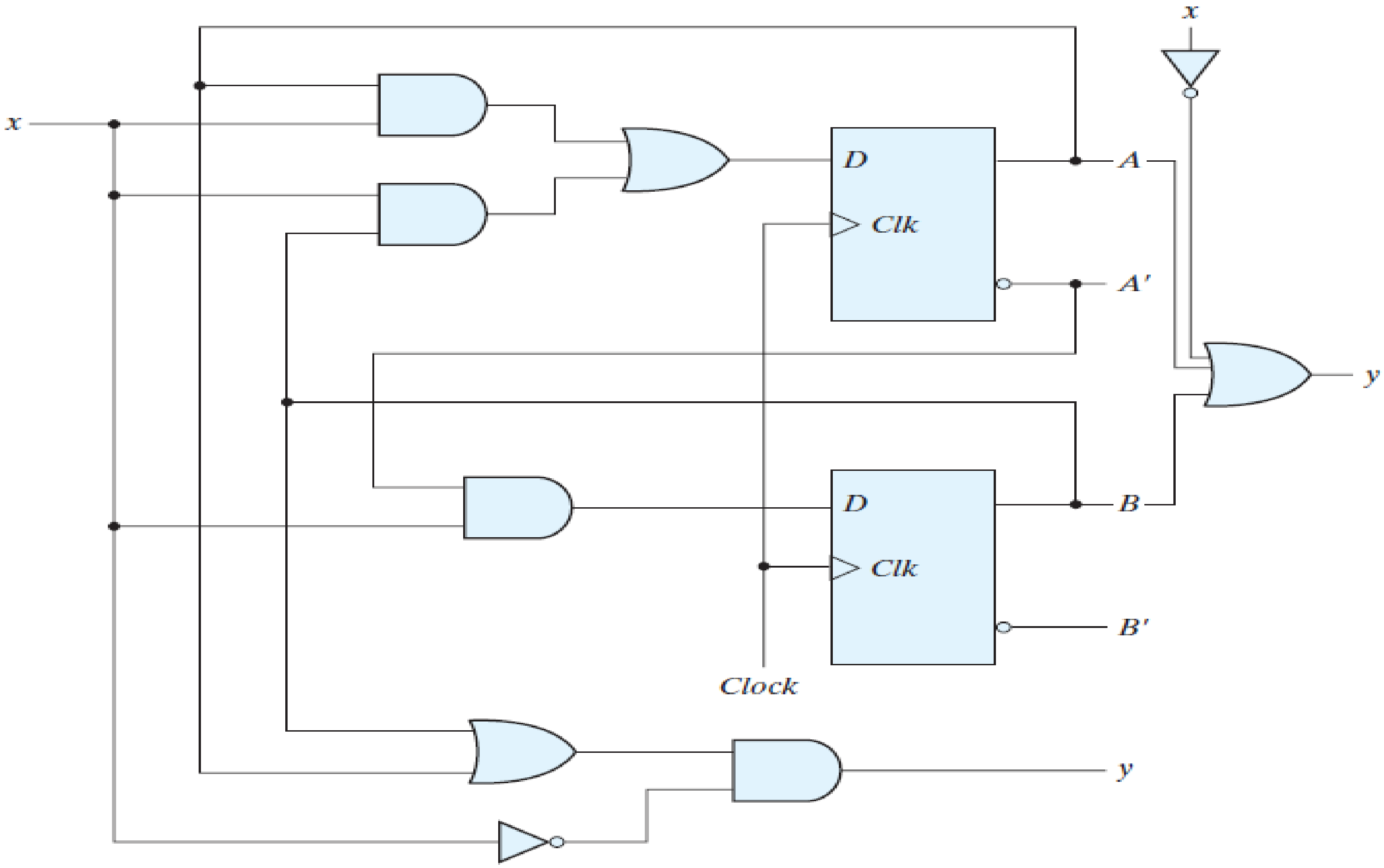


FIGURE 5.15
Example of sequential circuit

State Equations

- 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.

State Table

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

State Table

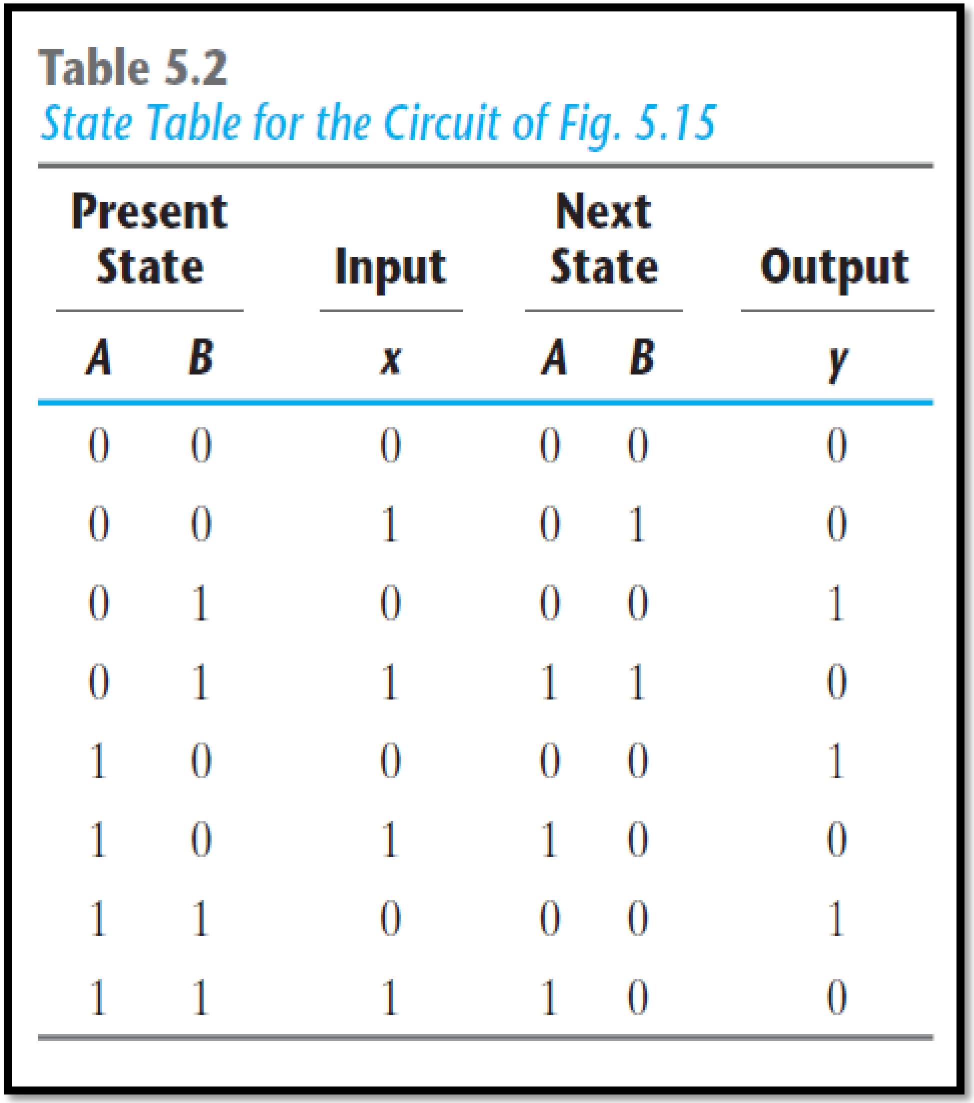


Table 5.2
State Table for the Circuit of Fig. 5.15

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

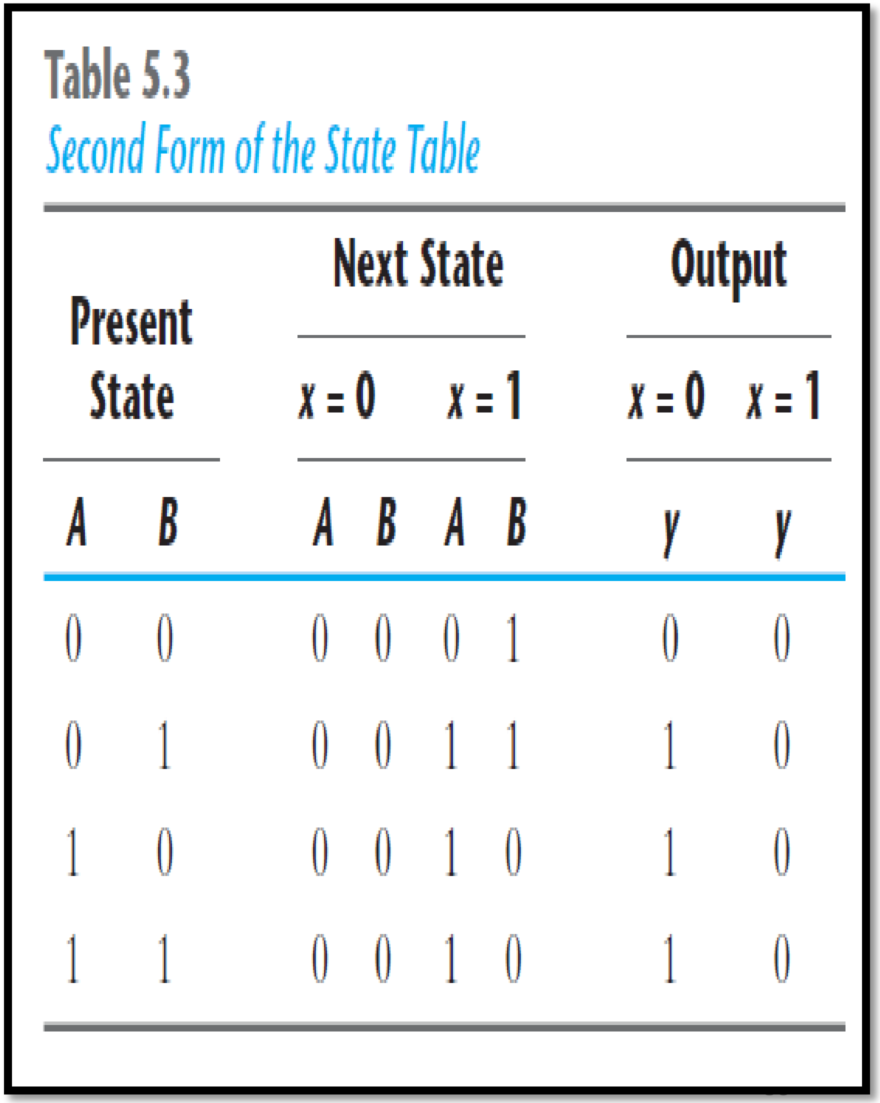


Table 5.3
Second Form of the State Table

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

State Table

- 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.

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 the circles.
- 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.

State Diagram

- 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.
- The steps presented in this example are summarized below:
- Circuit diagram S Equations – State table S State diagram
- *The state diagram gives a pictorial view of state transitions and is the form more suitable for human interpretation of the circuit's operation* .

State Diagram

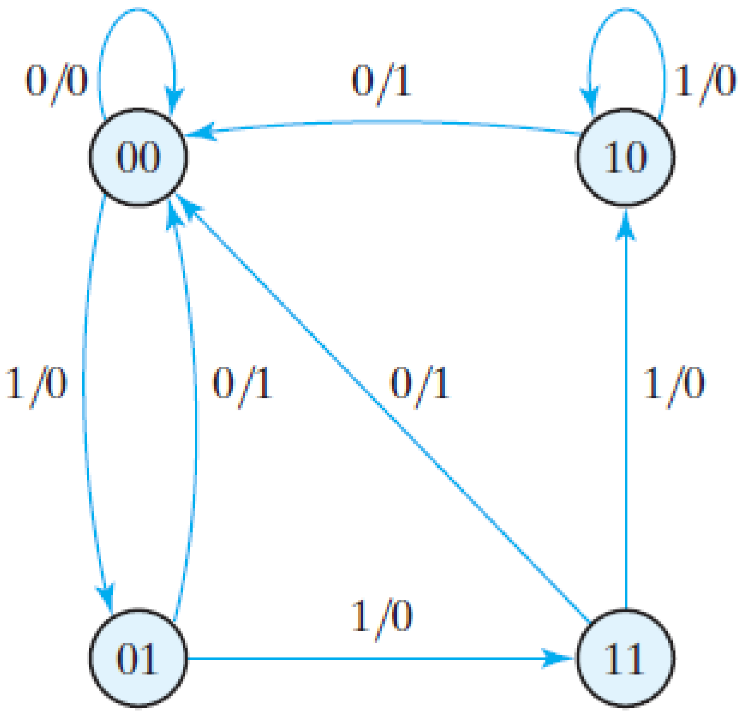


FIGURE 5.16
State diagram of the circuit of Fig. 5.15

Flip-Flop Input Equations

- 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* (sometimes, *excitation equations*).
- 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* .

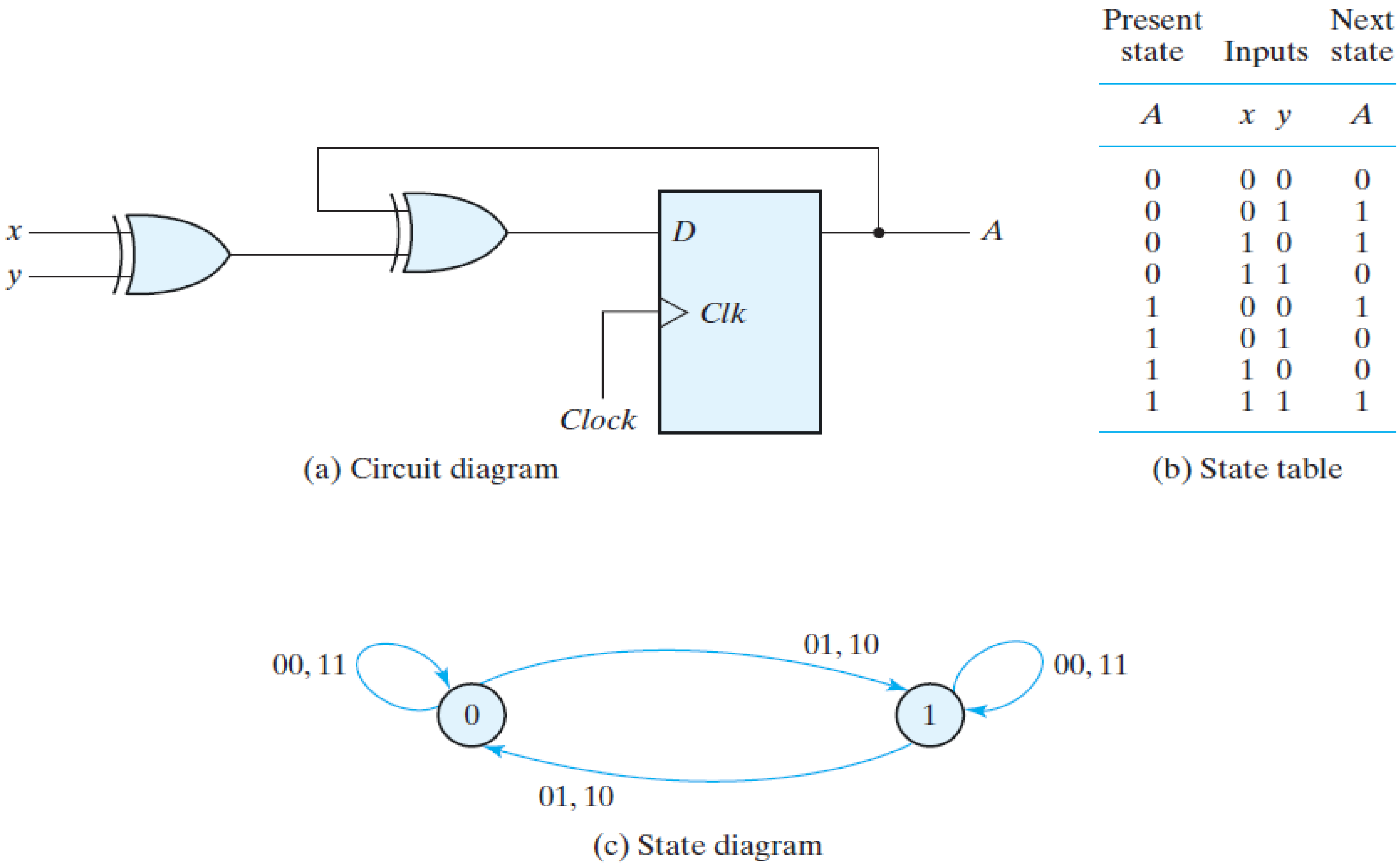
Flip-Flop Input Equations

- The logic diagram of the circuit can be expressed algebraically with two flip-flop input equations and an output equation:
- $D_A = Ax + Bx$
- $D_B = A'x$
- $y = (A + B)x'$
- 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.

Analysis with *D* Flip-Flops

- We will summarize the procedure for analyzing a clocked sequential circuit with *D* flipflops by means of a simple example.
- The circuit we want to analyze is described by the input equation
- $D_A = A \otimes x \otimes y$
- The D_A symbol implies a *D* flip-flop with output *A* .
- The *x* and *y* variables are the inputs to the circuit.
- The logic diagram is obtained from the input equation and is drawn in Fig. 5.17 (a).

Analysis with *D* Flip-Flops



(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

Analysis with *JK* Flip-Flops

- 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.

Analysis with *JK* Flip-Flops

- The circuit can be specified by the flip-flop input equations
- $J_A = B$ $K_A = Bx'$
- $J_B = x'$ $K_B = A'x + Ax' = A \otimes x$
- The state table of the sequential circuit is shown in Table 5.4 .
- There are four cases to consider.
- When *J* = 1 and *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.

Analysis with *JK* Flip-Flops

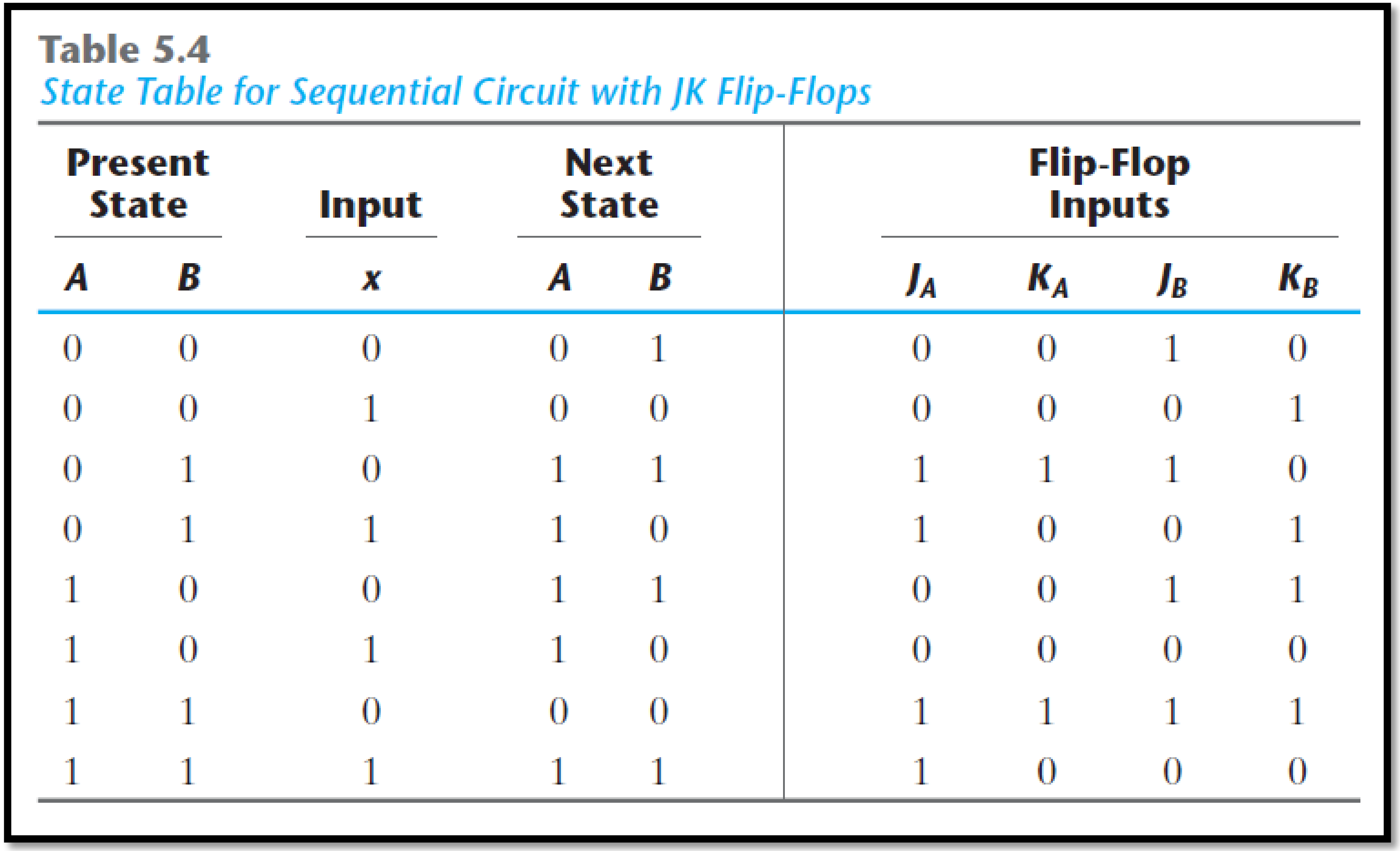


Table 5.4
State Table for Sequential Circuit with JK Flip-Flops

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

Analysis with *JK* Flip-Flops

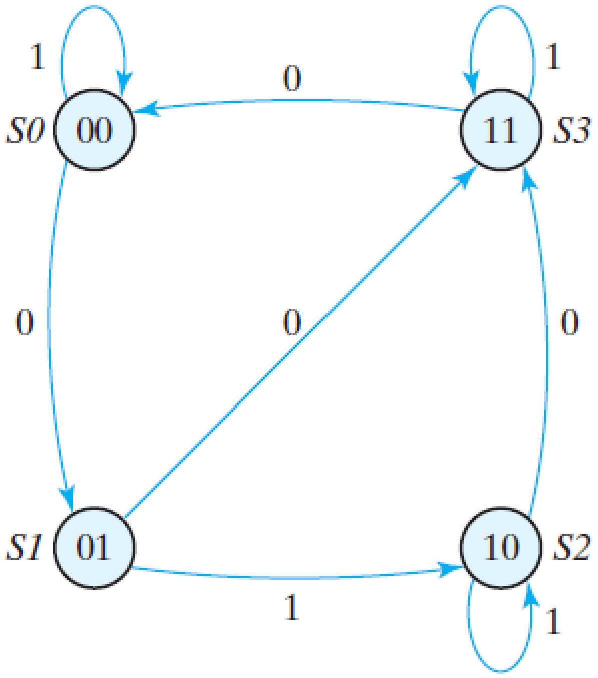


FIGURE 5.19
State diagram of the circuit of Fig. 5.18

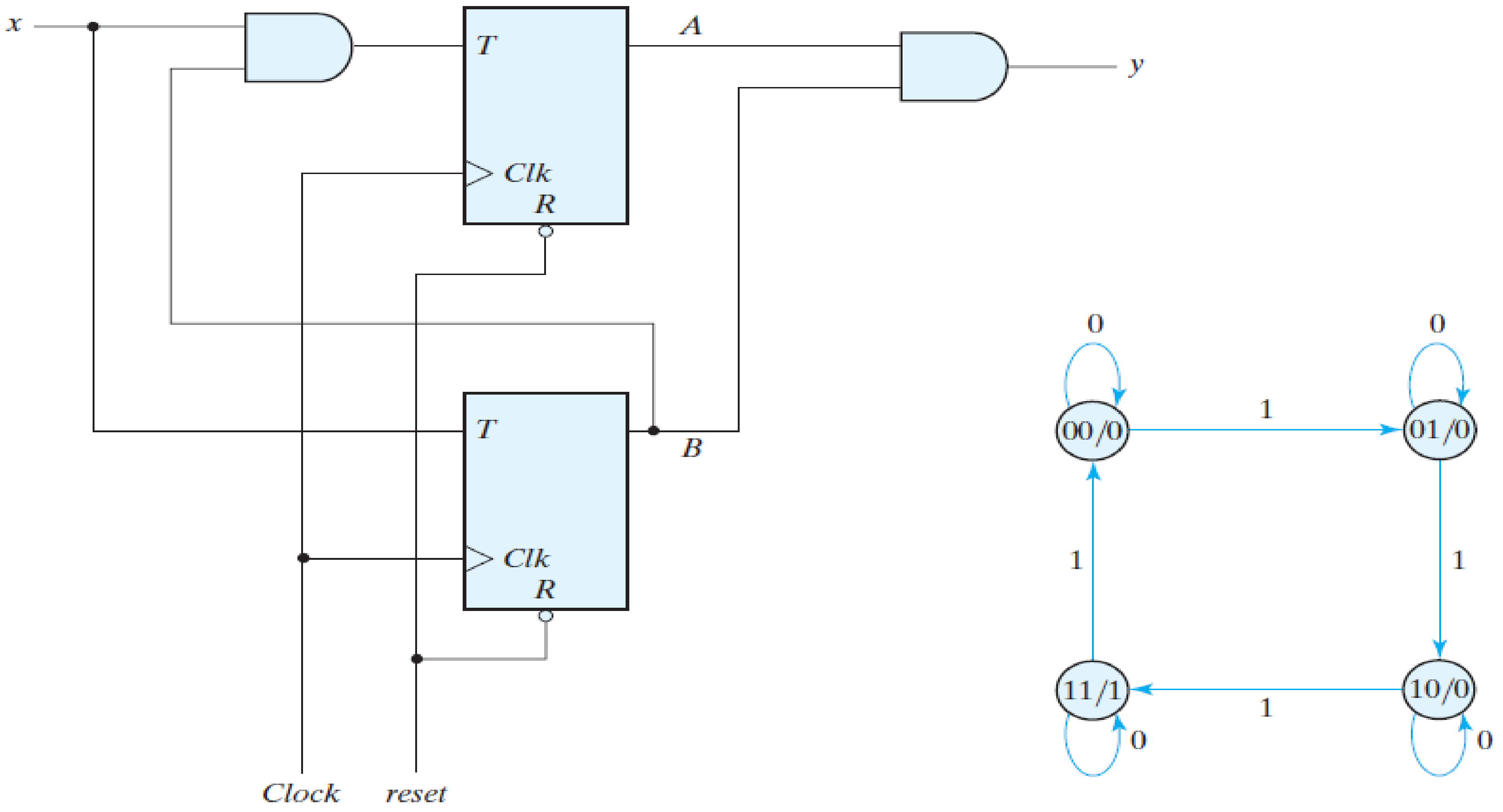
Analysis with *T* Flip-Flops

- 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 the characteristic table listed in Table 5.1 or the characteristic equation
- $Q(t + 1) = T \otimes Q = T'Q + TQ'$
- 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:

Analysis with *T* Flip-Flops

- $T_A = Bx$
- $T_B = x$
- $y = AB$
- 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
- $A(t + 1) = (Bx)'A + (Bx)A' = AB' + Ax' + A'Bx$
- $B(t + 1) = x \otimes B$

Analysis with *T* Flip-Flops



(a) Circuit diagram

(b) State diagram

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

Analysis with *T* Flip-Flops

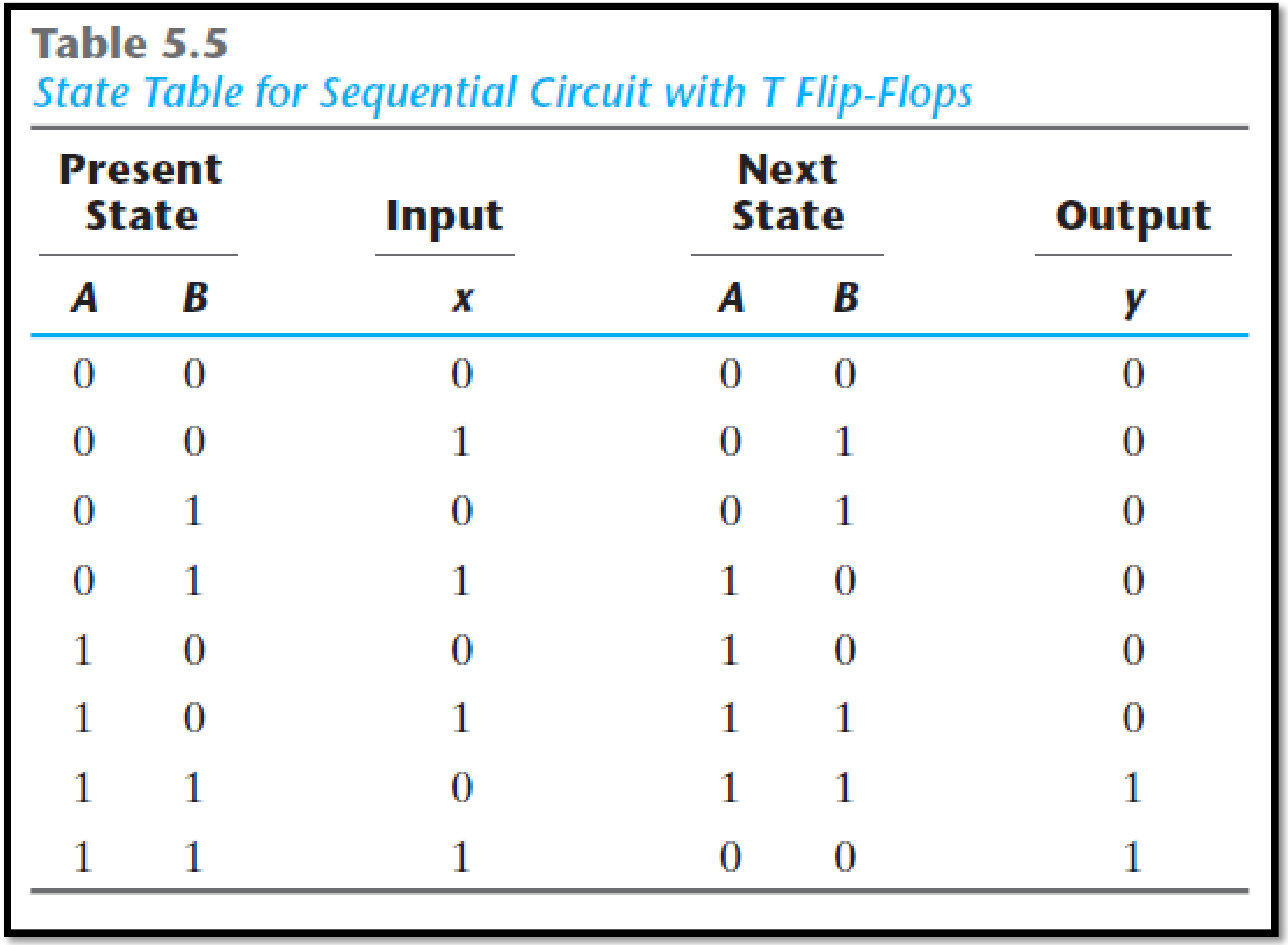


Table 5.5
State Table for Sequential Circuit with T Flip-Flops

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

Mealy and Moore Models of Finite State Machines

- 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 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.

Mealy and Moore Models of Finite State Machines

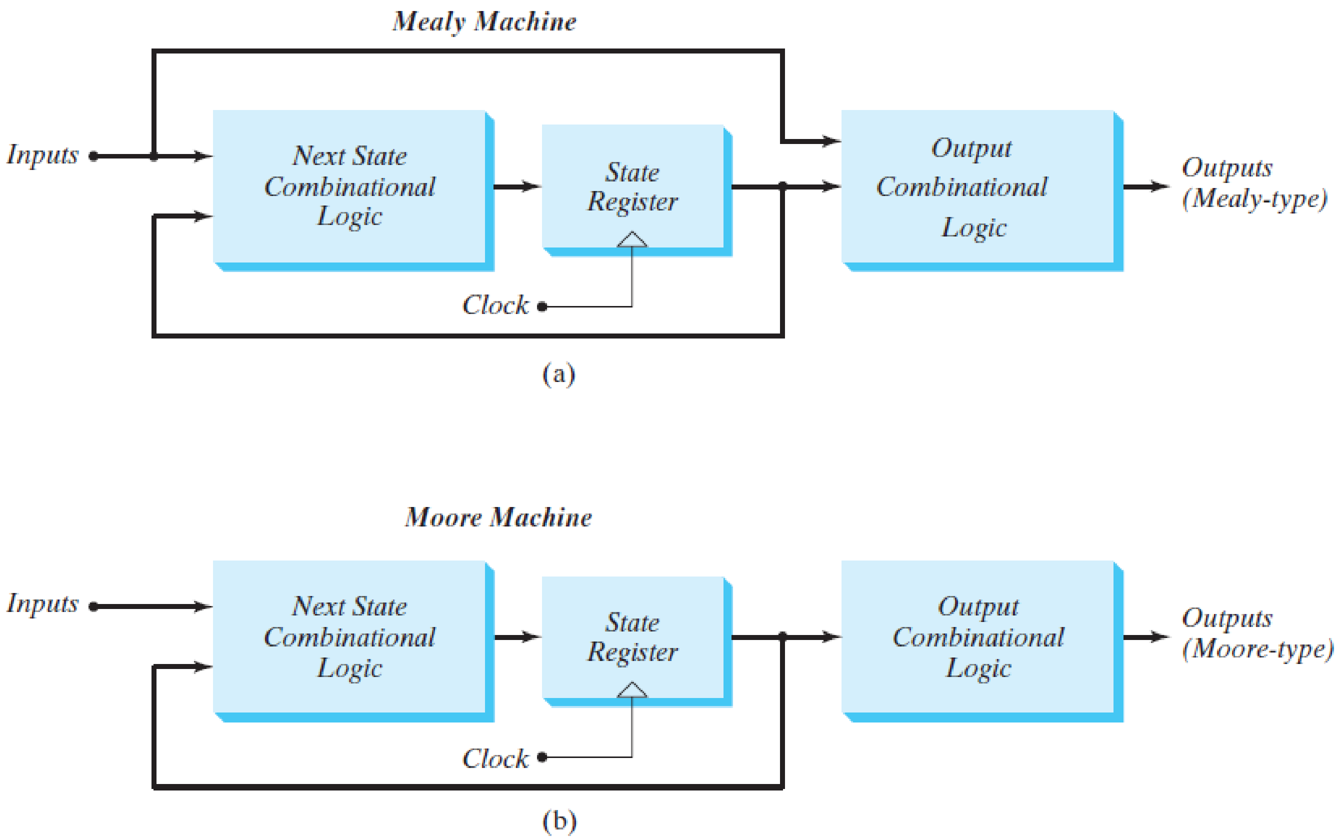


FIGURE 5.21
Block diagrams of Mealy and Moore state machines