EN1014 Electronic Engineering

Sequential Logic Circuits

Chamira U. S. Edussooriya
B.Sc.Eng. (Moratuwa), M.A.Sc. (UVic), Ph.D. (UVic)

Department of Electronic and Telecommunication Engineering
University of Moratuwa

Some of the tables and figures included in this presentation have been extracted from
Digital Design: With an Introduction to the Verilog HDL (M. M. Mano and M. D. Ciletti, Prentice Hall, 2012)

Introduction

- In contrast to combinational logic circuits, the outputs of a sequential logic circuits depend on the past values of the inputs in addition to the present values of the inputs.

Introduction

- In contrast to combinational logic circuits, the outputs of a sequential logic circuits depend on the past values of the inputs in addition to the present values of the inputs.
- A sequential logic circuit consists of a combinational circuit and storage (memory) elements.

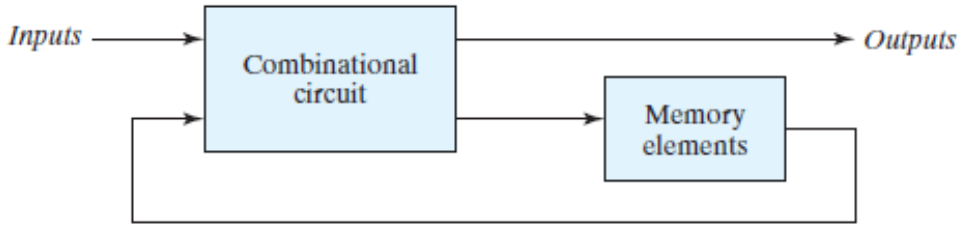


FIGURE 5.1
Block diagram of sequential circuit

Introduction

- In contrast to combinational logic circuits, the outputs of a sequential logic circuits depend on the past values of the inputs in addition to the present values of the inputs.
- A sequential logic circuit consists of a combinational circuit and storage (memory) elements.

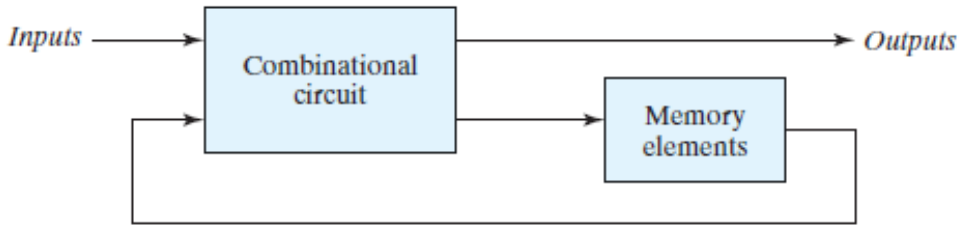


FIGURE 5.1
Block diagram of sequential circuit

- The binary information stored in the storage elements at any given time defines the state of the sequential logic circuit at that time.

Introduction

- In contrast to combinational logic circuits, the outputs of a sequential logic circuits depend on the past values of the inputs in addition to the present values of the inputs.
- A sequential logic circuit consists of a combinational circuit and storage (memory) elements.

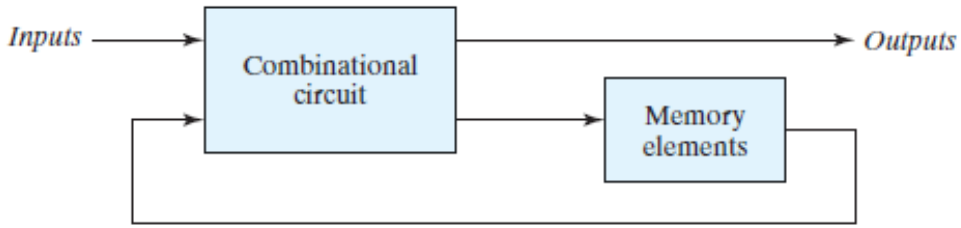


FIGURE 5.1
Block diagram of sequential circuit

- The binary information stored in the storage elements at any given time defines the state of the sequential logic circuit at that time.
- Similar to the outputs of a sequential logic circuit, the next state of the storage elements is a function of both inputs and the present state of the storage elements.

Introduction *cont'd*

- Sequential logic circuits can be categorized in two groups, synchronous and asynchronous, based on the timing of their signals.

Introduction *cont'd*

- Sequential logic circuits can be categorized in two groups, synchronous and asynchronous, based on the timing of their signals.
- The behaviour of a synchronous sequential logic circuit can be specified from the knowledge of its signals at discrete instants of time.

Introduction *cont'd*

- Sequential logic circuits can be categorized in two groups, synchronous and asynchronous, based on the timing of their signals.
- The behaviour of a synchronous sequential logic circuit can be specified from the knowledge of its signals at discrete instants of time.
- The behaviour of an asynchronous sequential logic circuit depends on the input signals at any instant of time and the order in which the inputs change.

Synchronous Sequential Circuits

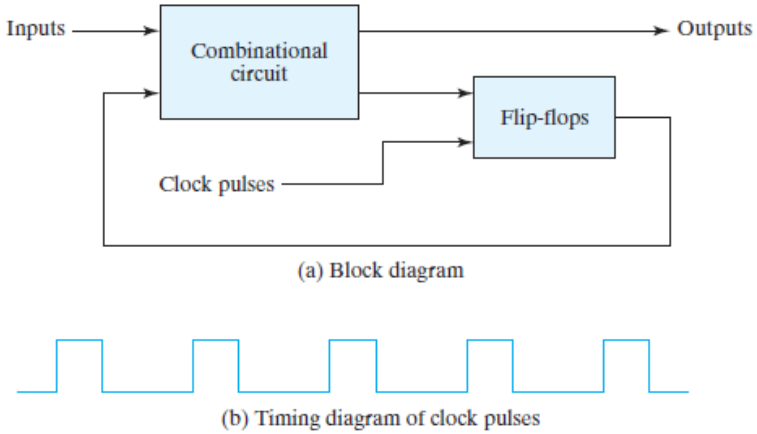


FIGURE 5.2
Synchronous clocked sequential circuit

- In a synchronous sequential circuit, the states of the storage elements are affected at only discrete instants of time.

Synchronous Sequential Circuits

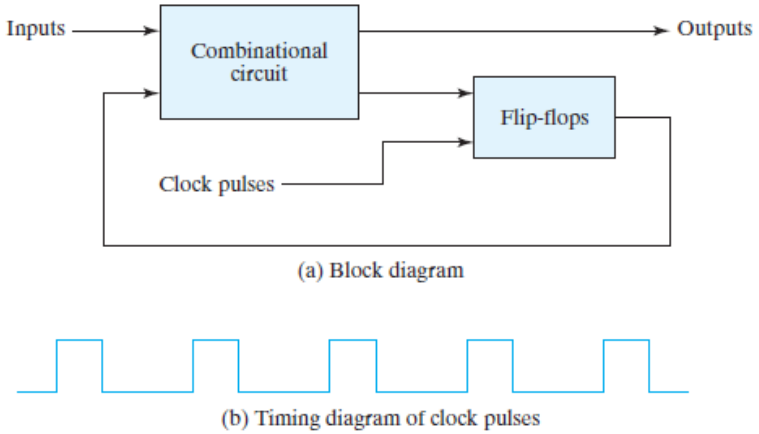
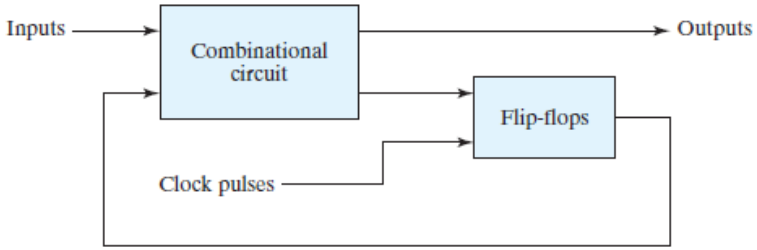


FIGURE 5.2
Synchronous clocked sequential circuit

- In a synchronous sequential circuit, the states of the storage elements are affected 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.

Synchronous Sequential Circuits



(a) Block diagram

(b) Timing diagram of clock pulses

FIGURE 5.2
Synchronous clocked sequential circuit

- In a synchronous sequential circuit, the states of the storage elements are affected 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 pulses determine when computational activity will occur, and the inputs and the present state determine what changes will take place affecting the storage elements and the outputs.

Storage Elements: Latches

- The basic storage elements employed in sequential logic circuits are latches and flip-flops.

Storage Elements: Latches

- The basic storage elements employed in sequential logic circuits are latches and flip-flops.
- Latches are level-sensitive devices where as flip-flops are edge-sensitive devices.

Storage Elements: Latches

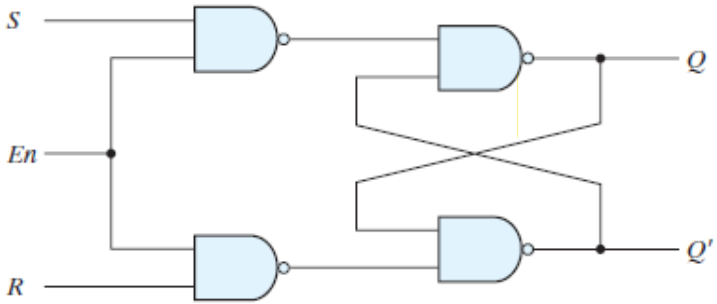
- The basic storage elements employed in sequential logic circuits are latches and flip-flops.
- Latches are level-sensitive devices where as flip-flops are edge-sensitive devices.
- Flip-flops are constructed using latches.

Storage Elements: Latches

- The basic storage elements employed in sequential logic circuits are latches and flip-flops.
- Latches are level-sensitive devices where as flip-flops are edge-sensitive devices.
- Flip-flops are constructed using latches.
- We first consider the basic *SR* latch.

Storage Elements: Latches

- The basic storage elements employed in sequential logic circuits are latches and flip-flops.
- Latches are level-sensitive devices where as flip-flops are edge-sensitive devices.
- Flip-flops are constructed using latches.
- We first consider the basic *SR* latch.
- An *SR* latch with an enable input.



(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

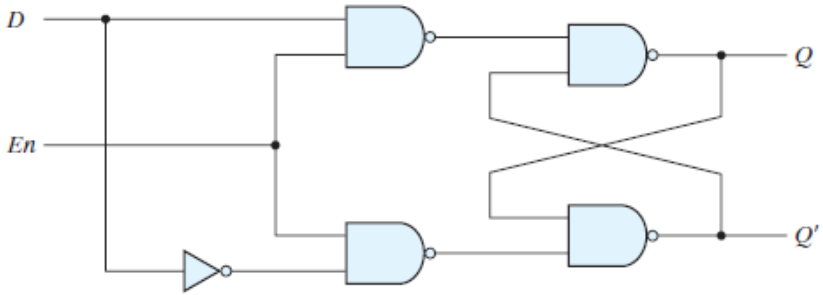
FIGURE 5.5
SR latch with control input

Storage Elements: Latches *cont'd*

- The undesirable condition that may occur with the *SR* latch can be eliminated by adding a NOT gate.

Storage Elements: Latches *cont'd*

- The undesirable condition that may occur with the *SR* latch can be eliminated by adding a NOT gate.
- This modified *SR* latch is called the *D* latch.



(a) Logic diagram

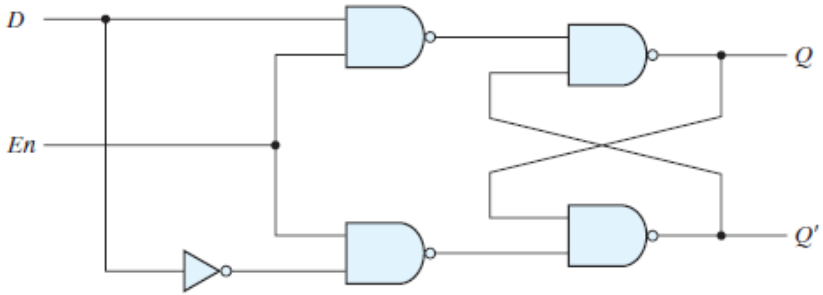
En	*D*	Next state of *Q*
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

(b) Function table

FIGURE 5.6
D latch

Storage Elements: Latches *cont'd*

- The undesirable condition that may occur with the *SR* latch can be eliminated by adding a NOT gate.
- This modified *SR* latch is called the *D* 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

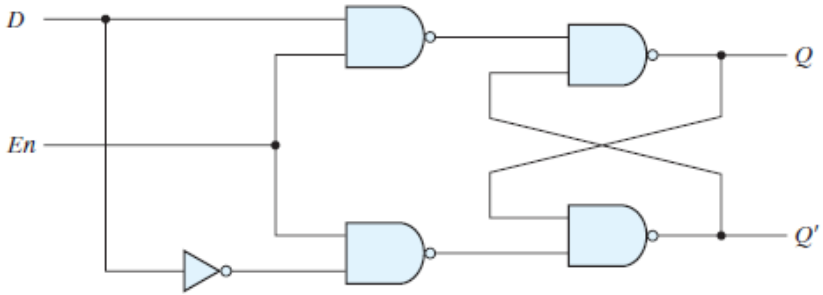
(b) Function table

FIGURE 5.6
D latch

- The binary information present at the data input (input *D*) of the *D* latch is transferred to the *Q* output when the enable input is asserted.

Storage Elements: Latches *cont'd*

- The undesirable condition that may occur with the *SR* latch can be eliminated by adding a NOT gate.
- This modified *SR* latch is called the *D* 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

- The binary information present at the data input (input *D*) of the *D* latch is transferred to the *Q* output when the enable input is asserted.
- In general, the enable input can be either active-high or active-low.

Storage Elements: Flip-Flops

- The state of a flip-flop is changed (or triggered) only during a signal transition.

Storage Elements: Flip-Flops

- The state of a flip-flop is changed (or triggered) only during a signal transition.
- A *D* flip-flop can be constructed by employing two latches in a special configuration, called master-slave configuration.

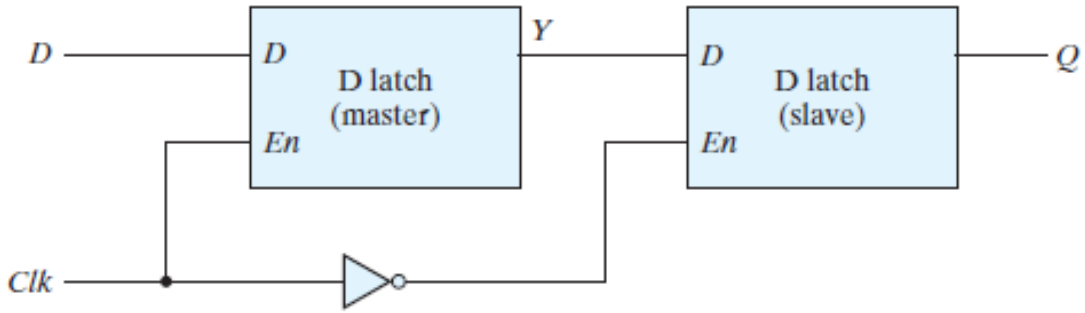


FIGURE 5.9
Master–slave *D* flip-flop

Storage Elements: Flip-Flops

- The state of a flip-flop is changed (or triggered) only during a signal transition.
- A *D* flip-flop can be constructed by employing two latches in a special configuration, called master-slave configuration.

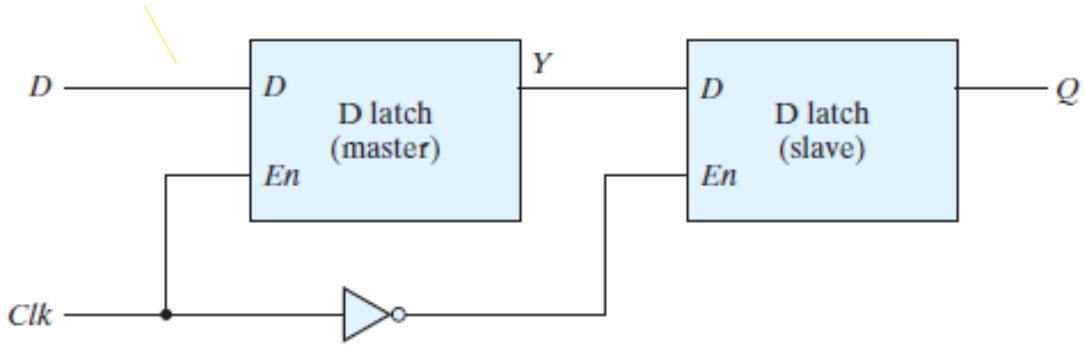


FIGURE 5.9
Master–slave *D* flip-flop

- The value at the output of the flip-flop is the value that was stored in the master stage immediately before the negative edge occurred.

Storage Elements: Flip-Flops *cont'd*

- Another way of constructing a *D* flip-flop is by using three *SR* latches.

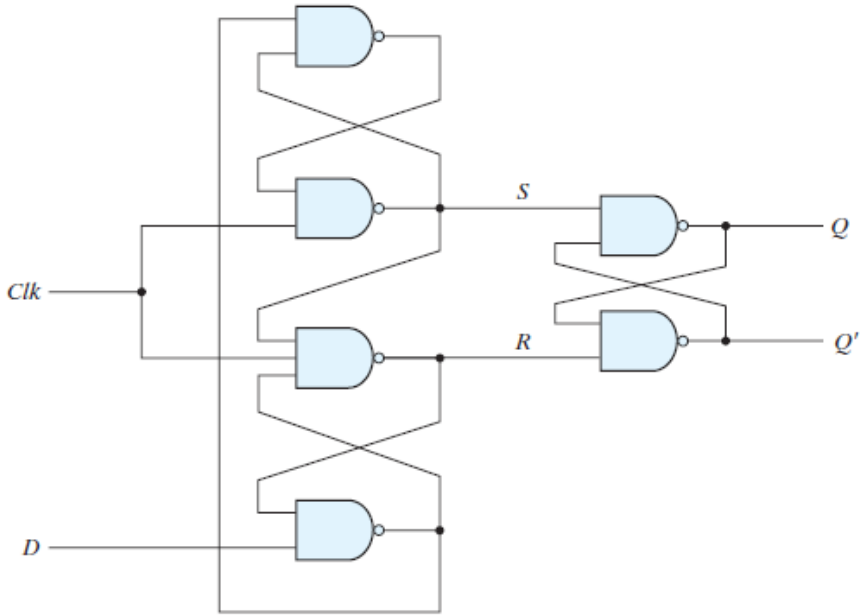


FIGURE 5.10
D-type positive-edge-triggered flip-flop

Storage Elements: Flip-Flops *cont'd*

- Another way of constructing a *D* flip-flop is by using three *SR* latches.

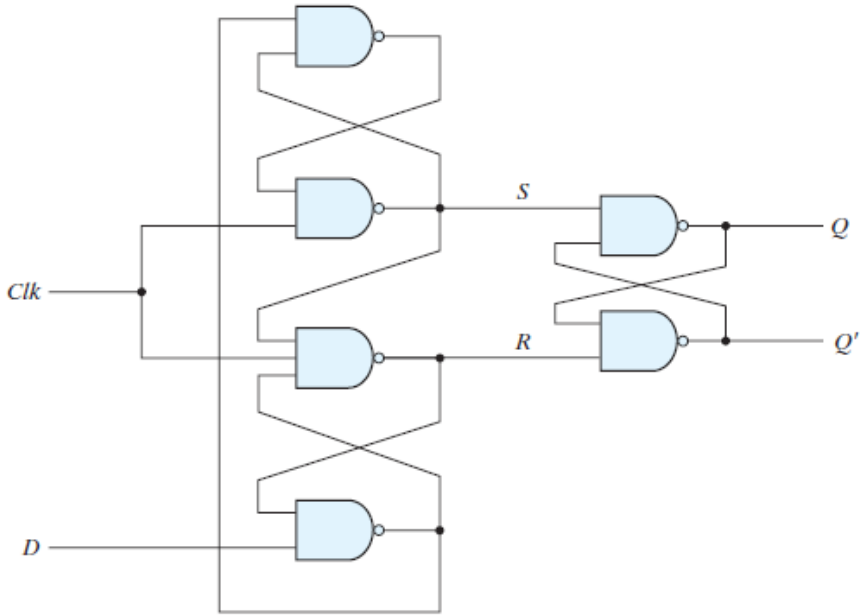


FIGURE 5.10
D-type positive-edge-triggered flip-flop

- Two *SR* latches respond to the *D* and the *Clk* inputs while the third *SR* latch provides the output of the flip-flop.

Storage Elements: Flip-Flops *cont'd*

- Another way of constructing a *D* flip-flop is by using three *SR* latches.

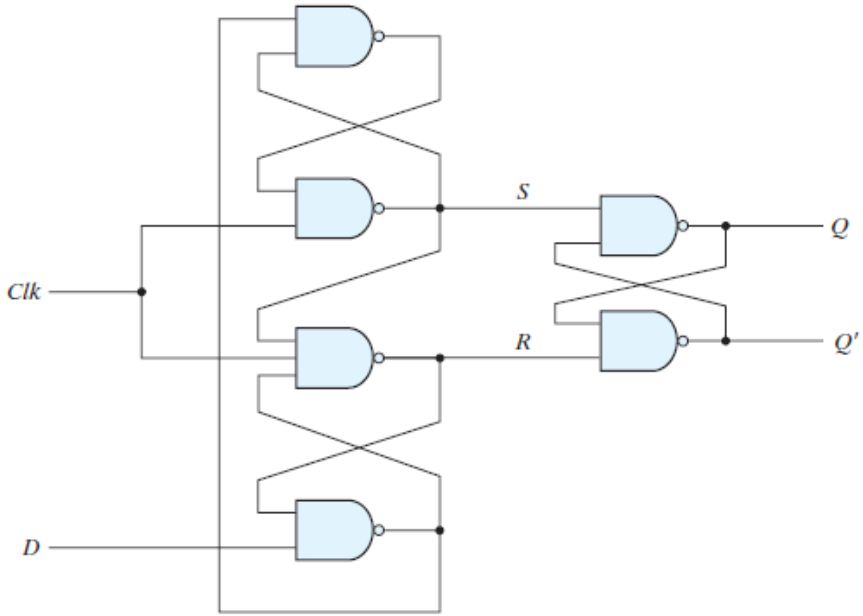
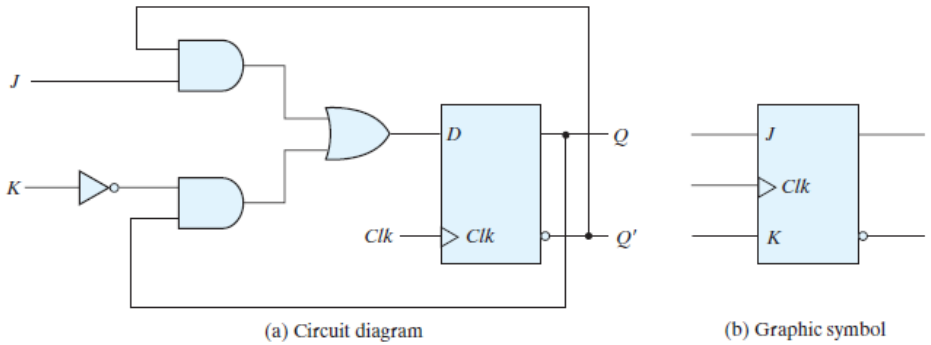


FIGURE 5.10
D-type positive-edge-triggered flip-flop

- Two *SR* latches respond to the *D* and the *Clk* inputs while the third *SR* latch provides the output of the flip-flop.
- The value of *D* is transferred to *Q* during the positive edge of the clock.

Storage Elements: Flip-Flops *cont'd*

- Other types of flip-flops:
  - *JK* flip-flop

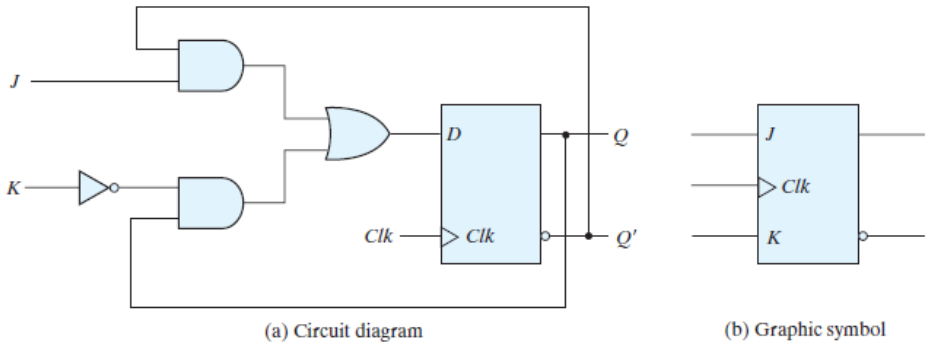


(a) Circuit diagram (b) Graphic symbol

FIGURE 5.12
JK flip-flop

Storage Elements: Flip-Flops *cont'd*

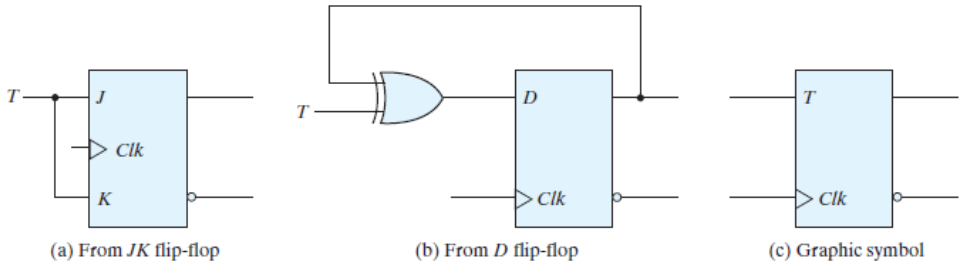
- Other types of flip-flops:
  - *JK* flip-flop



(a) Circuit diagram (b) Graphic symbol

FIGURE 5.12
JK flip-flop

  - *T* (toggle) flip-flop



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

FIGURE 5.13
T flip-flop

Storage Elements: Flip-Flops *cont'd*

- Asynchronous inputs such as preset and clear can be used to force a flip-flop to a particular state independently of a clock.

Storage Elements: Flip-Flops *cont'd*

- Asynchronous inputs such as preset and clear can be used to force a flip-flop to a particular state independently of a clock.
- They are useful for bringing a flip-flop to a known starting state prior to the clock operation.

Storage Elements: Flip-Flops *cont'd*

- Asynchronous inputs such as preset and clear can be used to force a flip-flop to a particular state independently of a clock.
- They are useful for bringing a flip-flop to a known starting state prior to the clock operation.
- *D* flip-flop with an asynchronous reset (or clear).