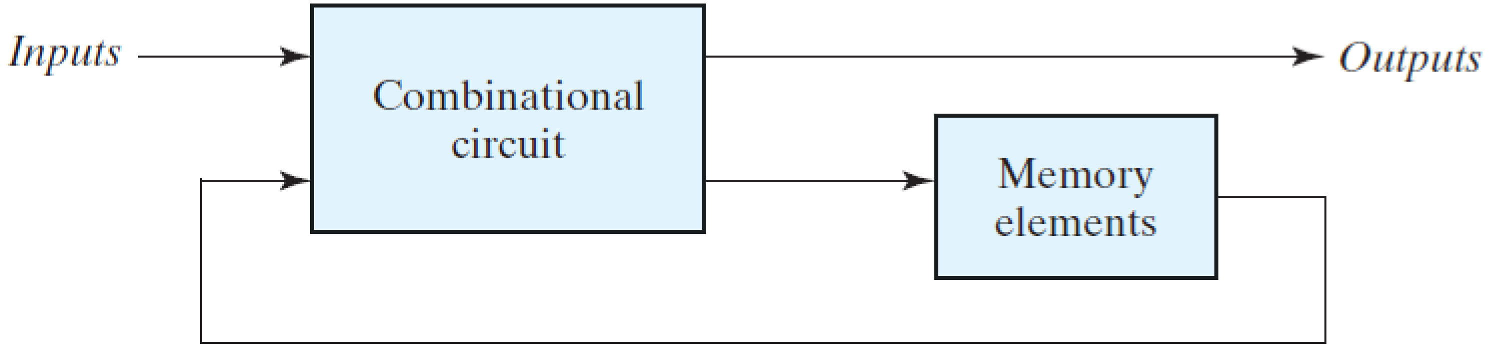
Unit-3: Sequential Logic Circuits

Introduction:

•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.

Sequential circuits:

•A block diagram of a sequential circuit is shown in the following Fig. It consists of a combinational circuit to which storage elements are connected to form a feedback path.



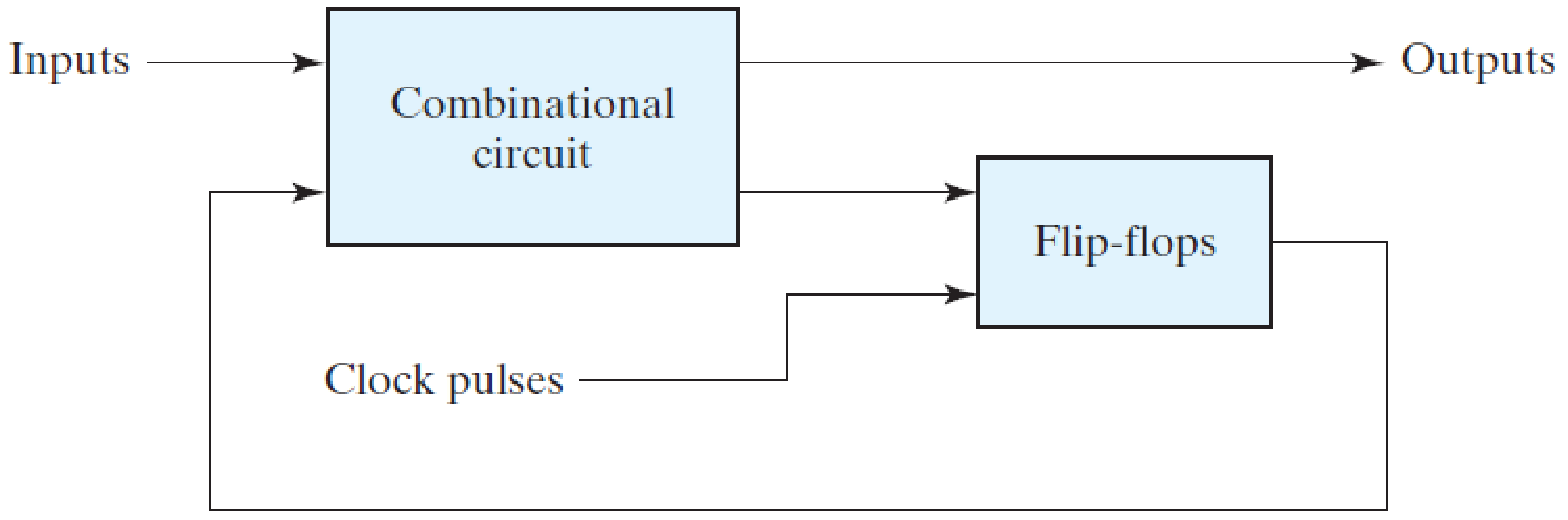
•The block diagram demonstrates that the outputs in a sequential circuit are a function not only of the inputs, but also of the present state of the storage elements. The next state of the storage elements is also a function of external inputs and the present state. Thus, **a sequential circuit is specified 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.*

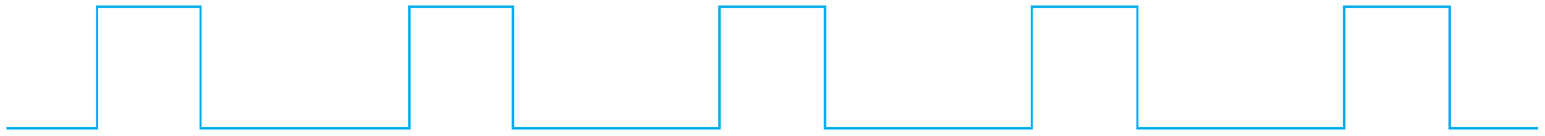
•Synchronous sequential circuits that use clock pulses to control storage elements are called ***clocked sequential circuits*** *and are the type most frequently encountered* in practice. They are called *synchronous circuits because the activity within the* circuit and the resulting updating of stored values is **synchronized to the occurrence of clock pulses**.

•The 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. The block diagram of a synchronous clocked sequential circuit is shown in the following Fig. 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.

•The transition from one state to the next occurs only at predetermined intervals dictated by the clock pulses.



(a) Block diagram



(b) Timing diagram of clock pulses

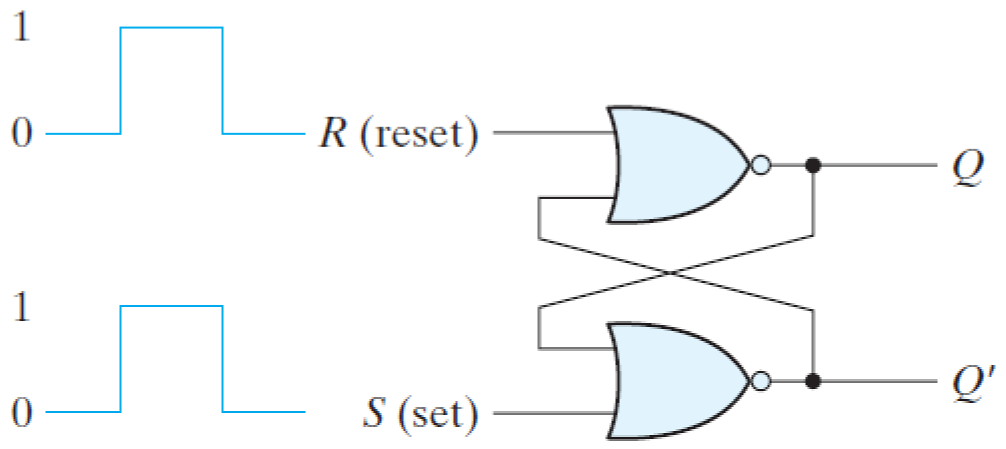
STORAGE ELEMENTS : LATCHES

•*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.

SR Latch:

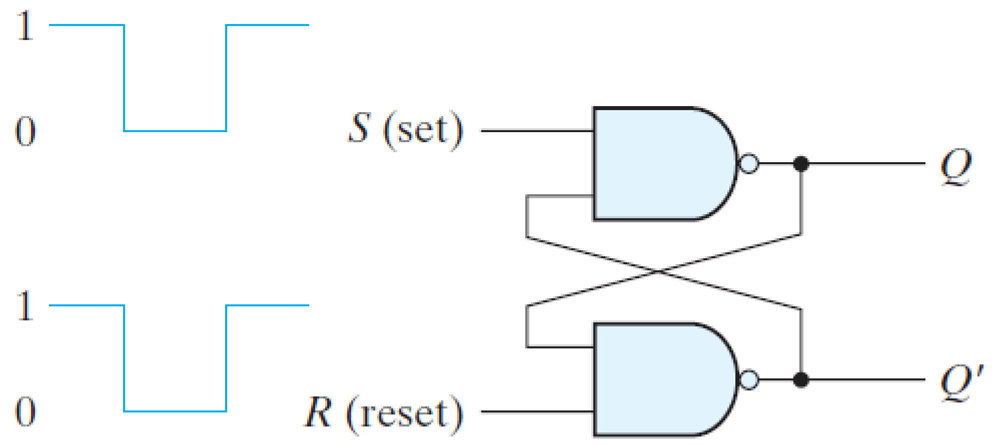
The *SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND* gates, and two inputs labeled *S for set and R for reset. The SR latch constructed with two* cross-coupled NOR gates is shown in the following Fig. The latch has two useful states. When output *Q = 1 and Q' = 0, the 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.



(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



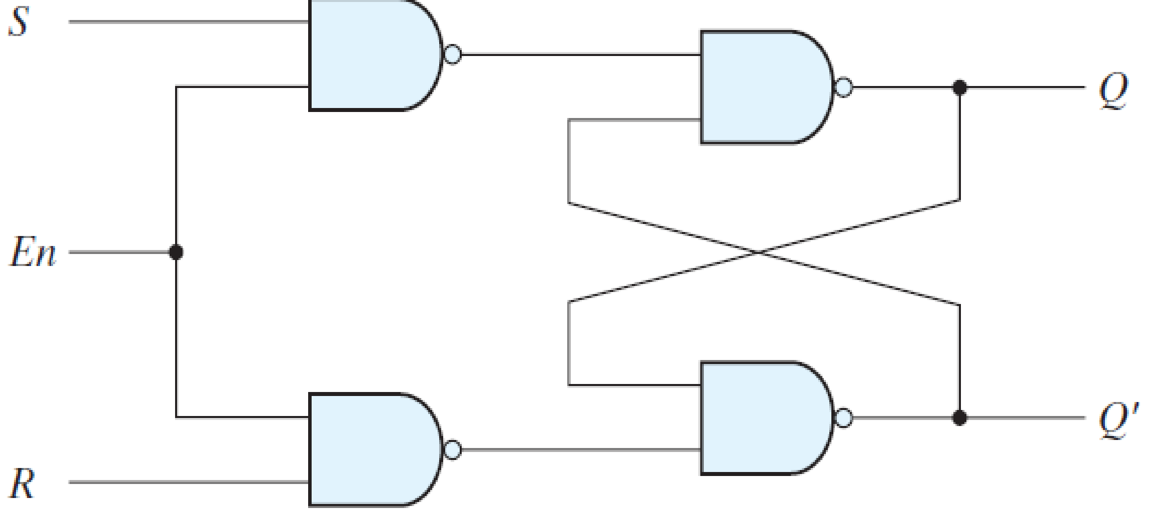
(a) Logic diagram

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

•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.**

•**An *SR latch with a control (enable) input*** *is* shown in the following Fig. 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.

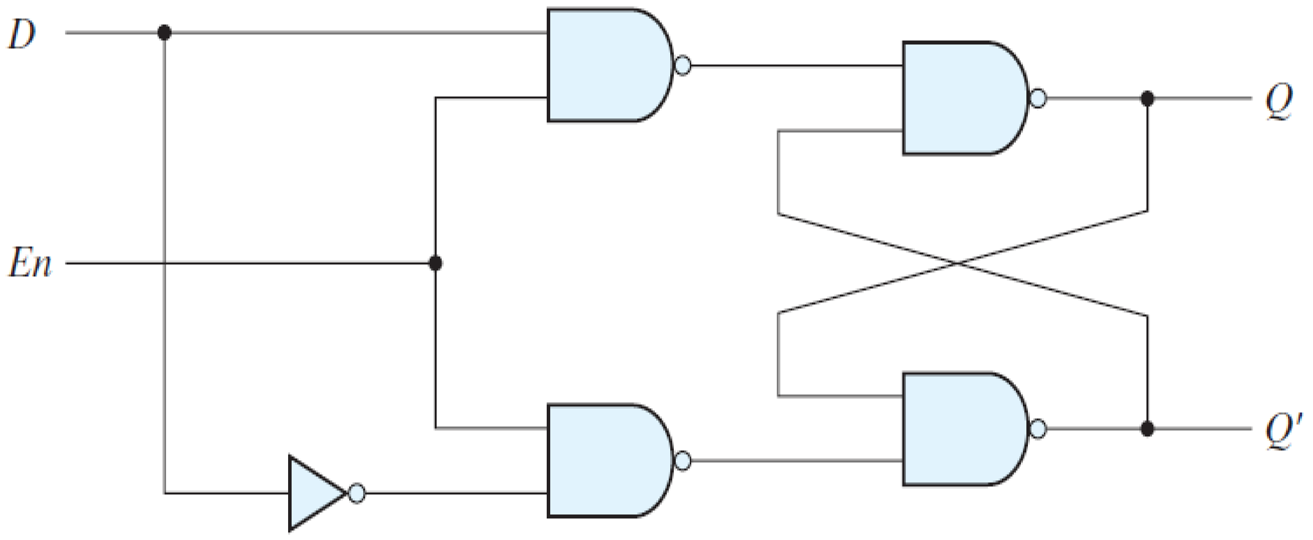


(a) Logic diagram

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

D Latch (Transparent Latch):

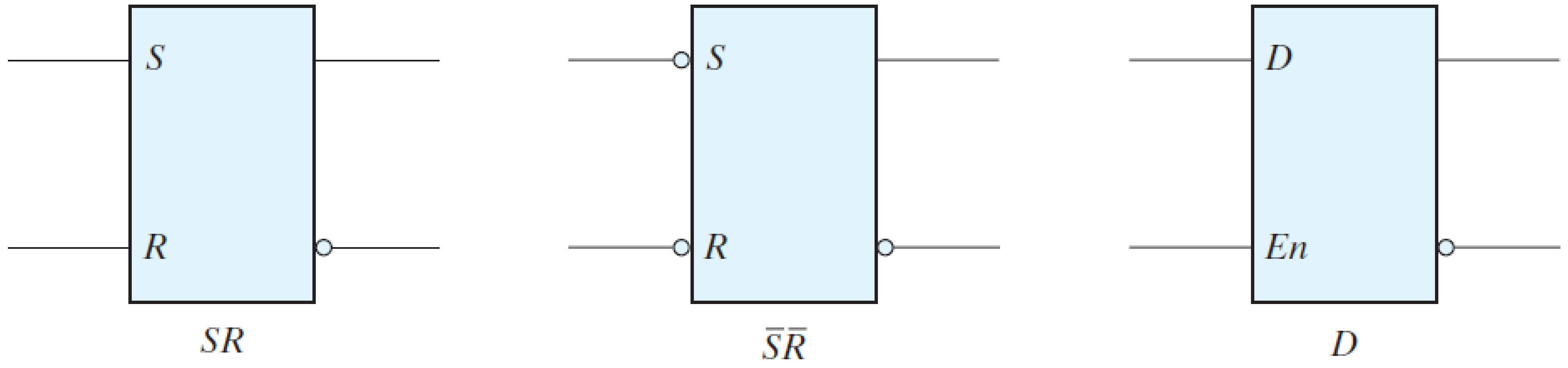


(a) Logic diagram

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

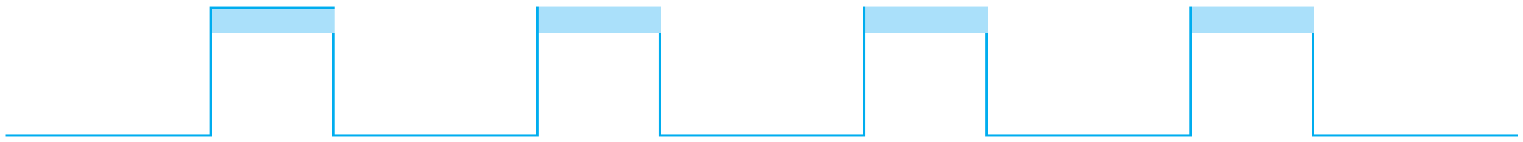
•The graphic symbols for the various latches are shown in the following Fig.

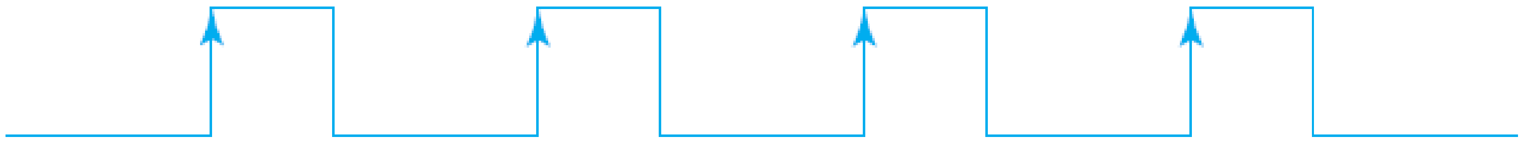


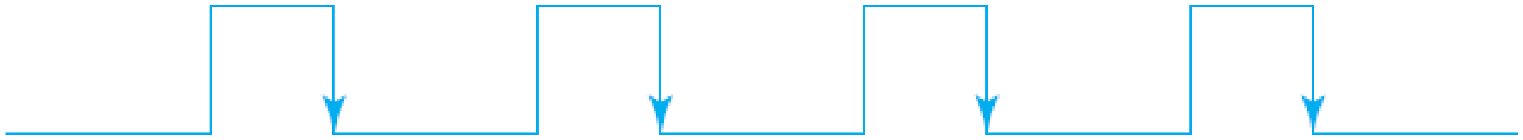
STORAGE ELEMENTS : FLIP – FLOPS:

•The problem with the latch is that it responds to a change in the *level of a clock pulse. As shown in the following* Fig., 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.*


(a) Response to positive level


(b) Positive-edge response


(c) Negative-edge response

•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 (master-slave flip flop).

•Another way is to produce a flip-flop that 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 (edge triggered flip flop).

Master–slave *D flip-flop:*

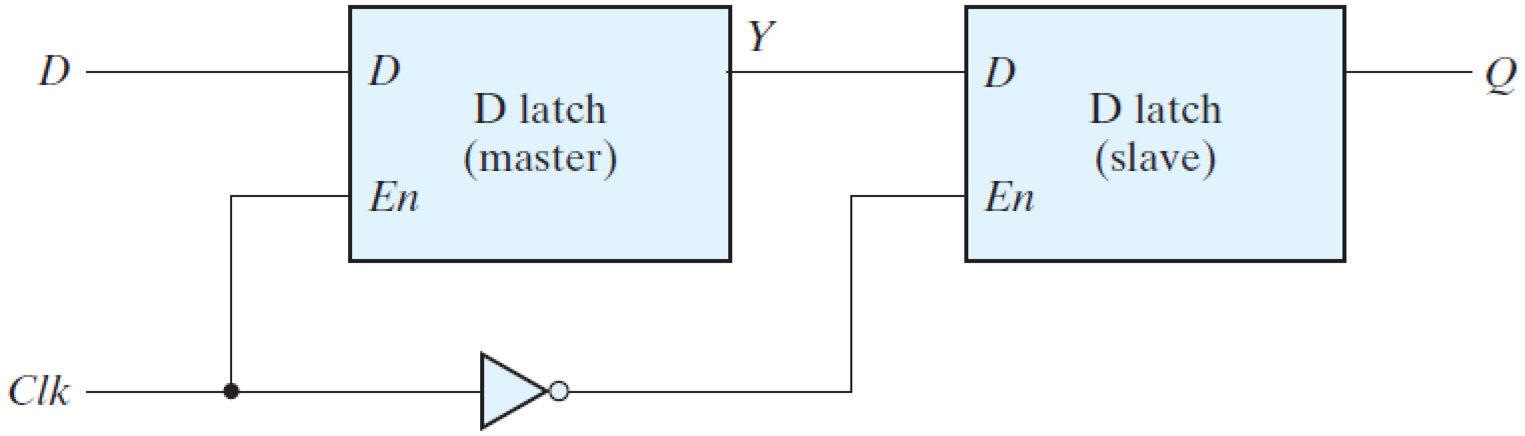
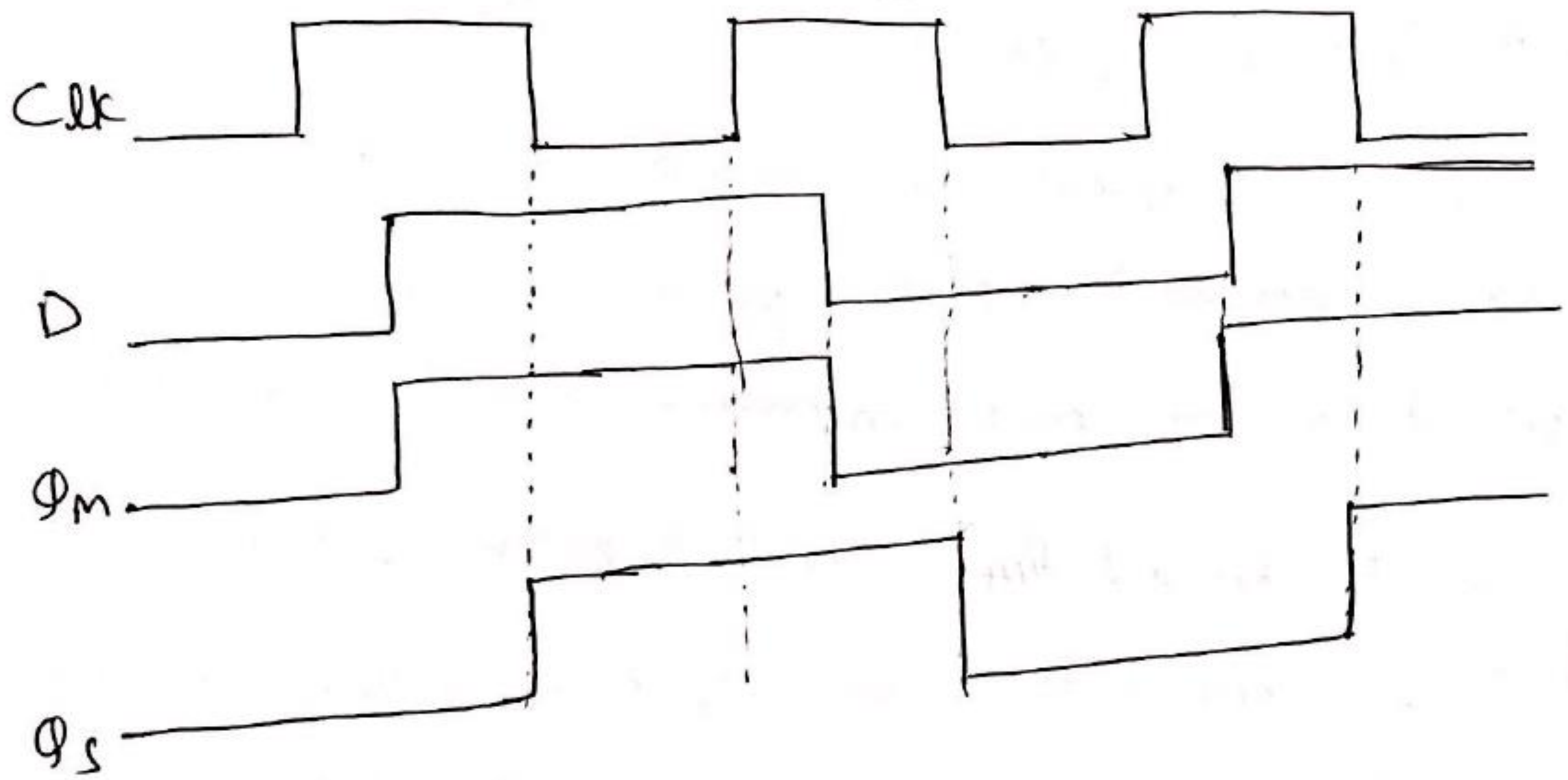


FIGURE: Master–slave *D flip-flop*

•*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.*
- ***Home work***
- It is also possible to design the circuit so that the flip-flop output changes on the positive edge of the clock. **(How?)**

MS D FF Timing Diagram :



D -type positive-edge-triggered flip-flop:

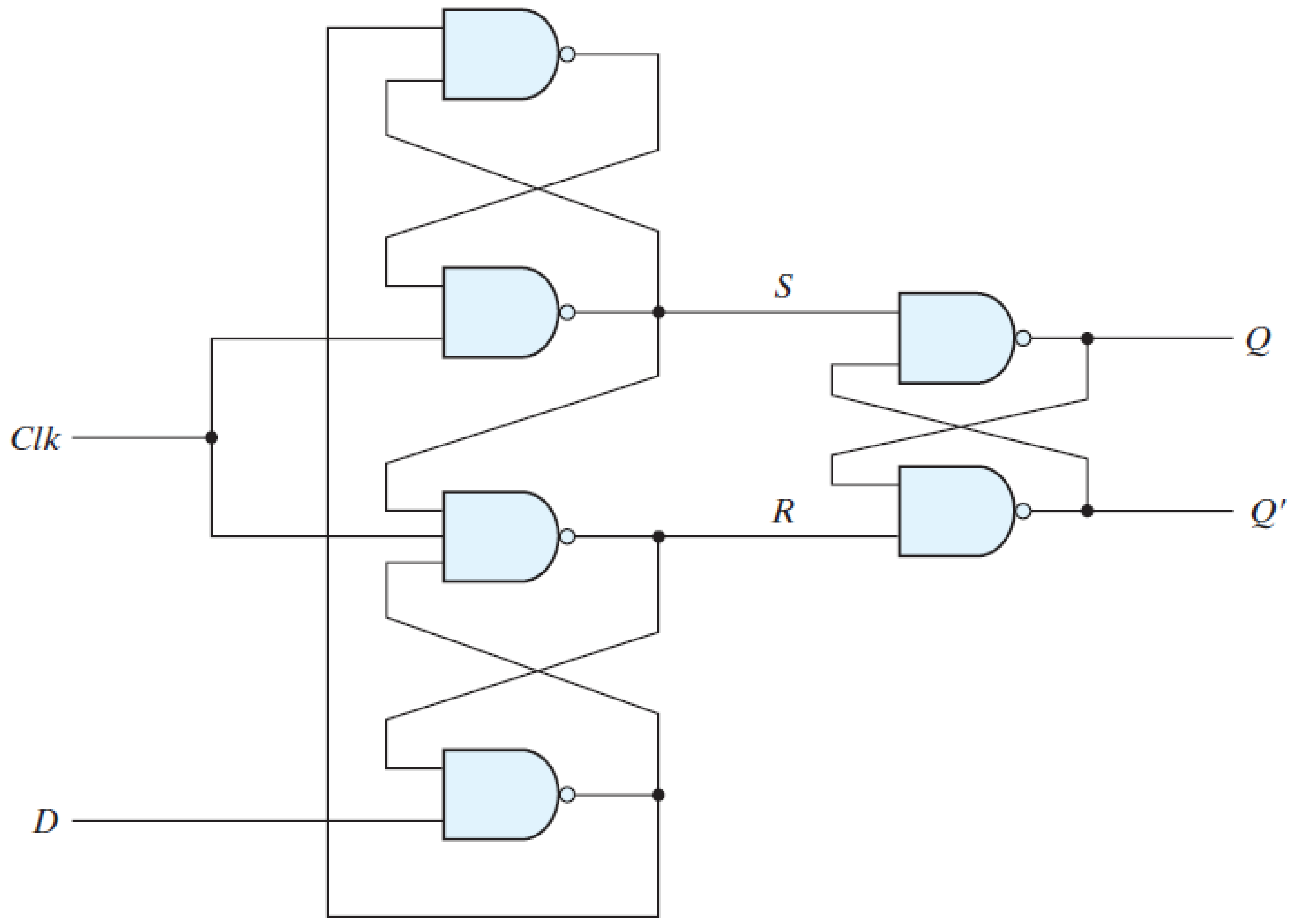


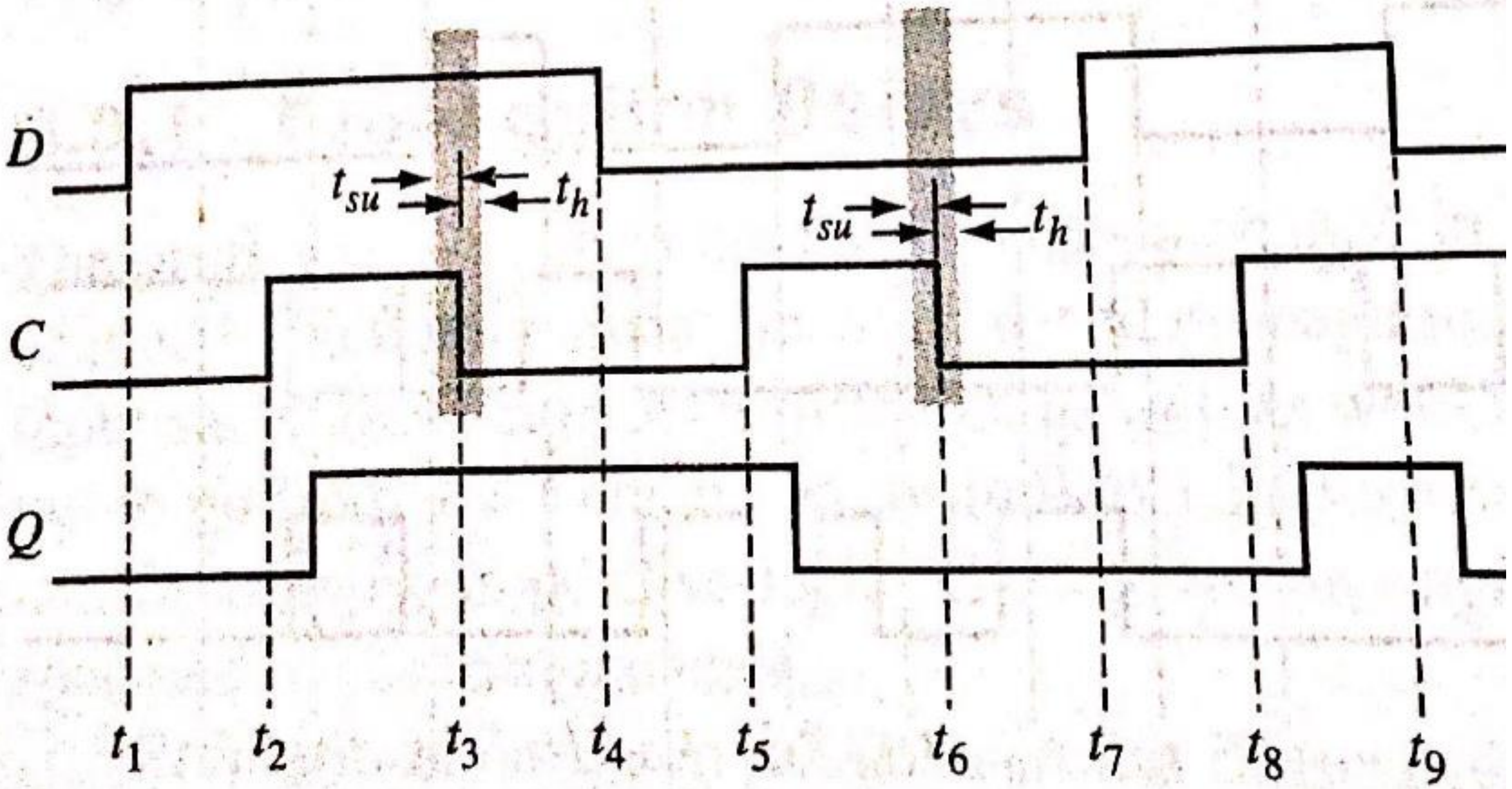
FIGURE: *D -type positive-edge-triggered flip-flop*

•In summary, when the input clock in the positive-edge-triggered flip-flop **makes a positive transition**, the value of *D is transferred to Q .*

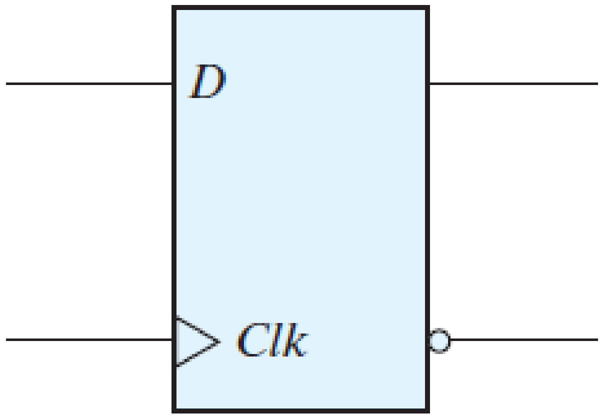
•*A negative transition of the clock (i.e., from* 1 to 0) **does not affect the output,** nor is the output affected by changes in *D when Clk* is in the steady logic-1 level or the logic-0 level. Hence, this type of flip-flop responds to the transition from 0 to 1 and nothing else.

•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 or negative transition* of the clock.

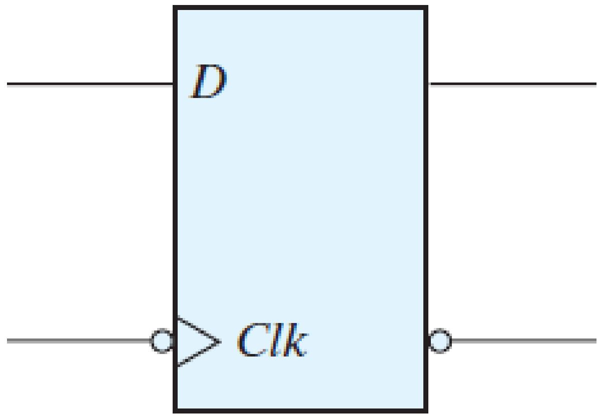
•The propagation delay time of the flip-flop is defined as the interval between the trigger edge and the stabilization of the output to a new state.


D
C
Q
t_{su}
t_h
t_{su}
t_h
t_1
t_2
t_3
t_4
t_5
t_6
t_7
t_8
t_9

<u>Graphic symbol for edge-triggered *D flip-flop:*</u>



(a) Positive-edge



(a) Negative-edge

•The graphic symbol for the edge-triggered *D flip-flop is shown in the above Fig.*

•*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.

Direct Inputs: *D flip-flop with asynchronous reset:*

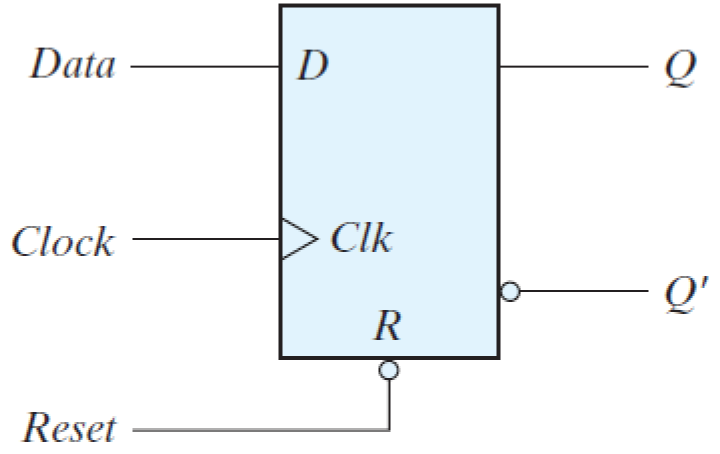
•Some flip-flops have asynchronous inputs that are **used to force the flip-flop to a particular state independently of the clock**. The input that sets the flip-flop to 1 is called ***preset or direct set*** *. The input that clears the flip-flop to 0 is called* ***clear or direct reset****.* When power 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 the following Fig. The circuit diagram is the same as positive edge triggered D FF, 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.*

•Flip-flops with a direct set use the symbol *S for the asynchronous set input.*

•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.*



R	Clk	D	Q	Q'
0	X	X	0	1
0	↑	0	0	1
0	↑	1	1	0

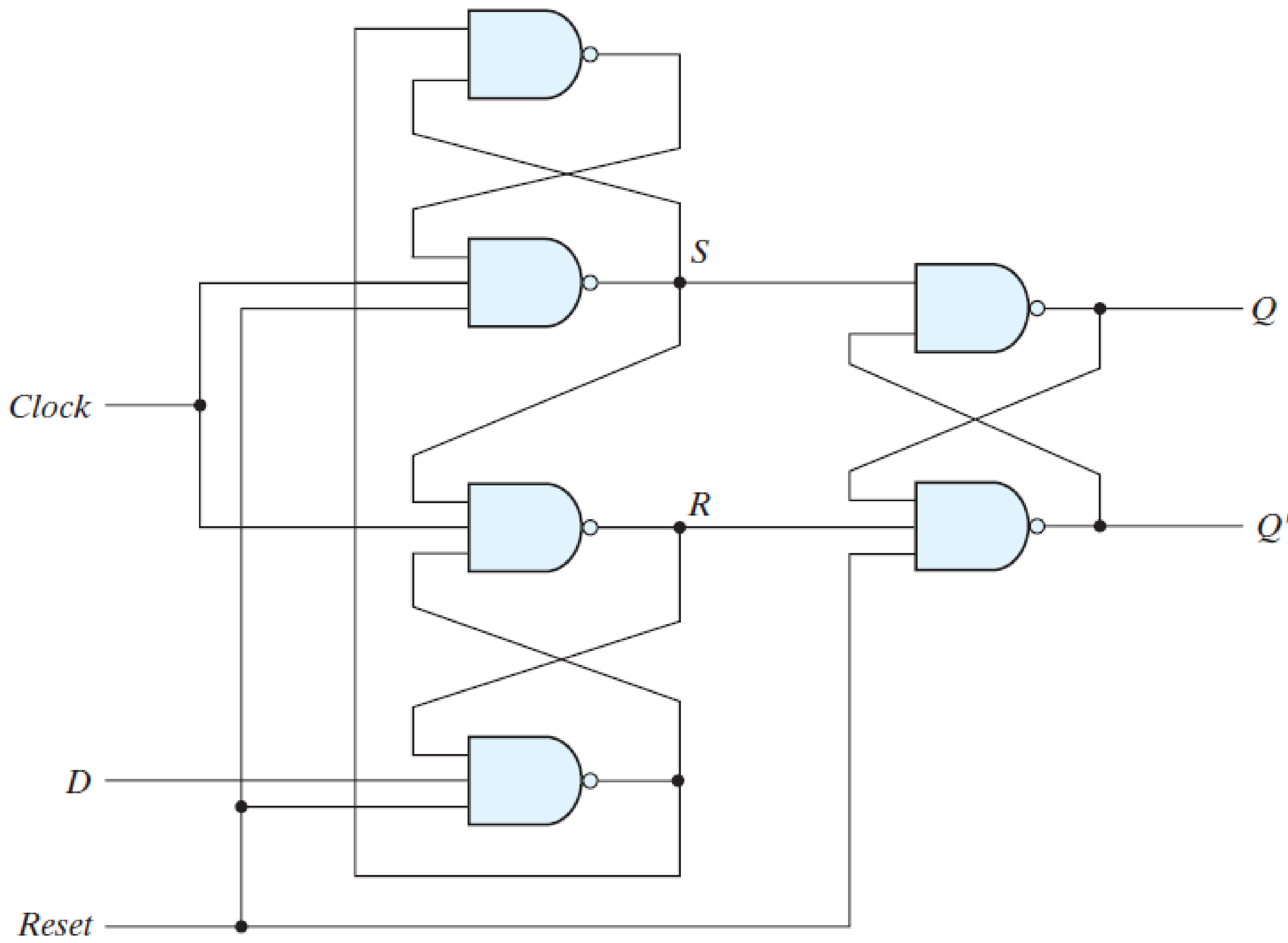


FIGURE: *D flip-flop with asynchronous reset*

Other Flip-Flops:

•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 flipflop*,** 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.

JK flip-flop:

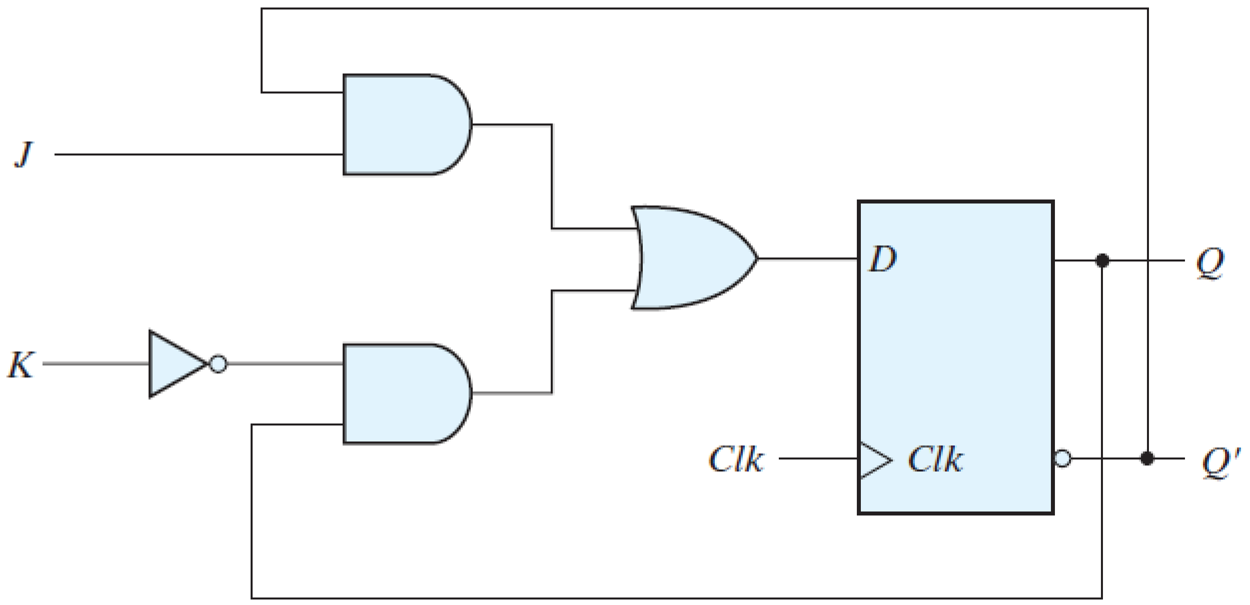
•With only a single input, the ***D flip-flop can set or reset*** the output, depending on the value of the *D input immediately before the clock transition.*

•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 the following Fig.

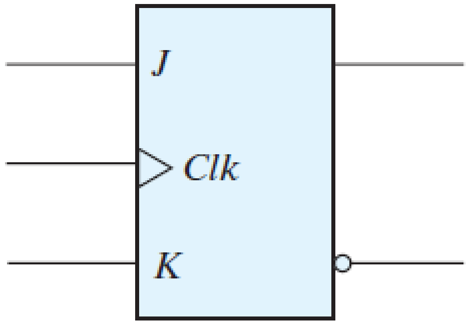
•The *J input sets the flip-flop to 1, the K input resets it to* 0, and when both inputs are enabled, the output is complemented.

•This can be verified by investigating the circuit applied to the *D input:*

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



(a) Circuit diagram

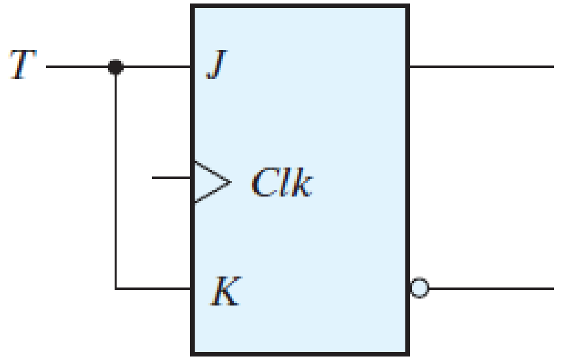


(b) Graphic symbol

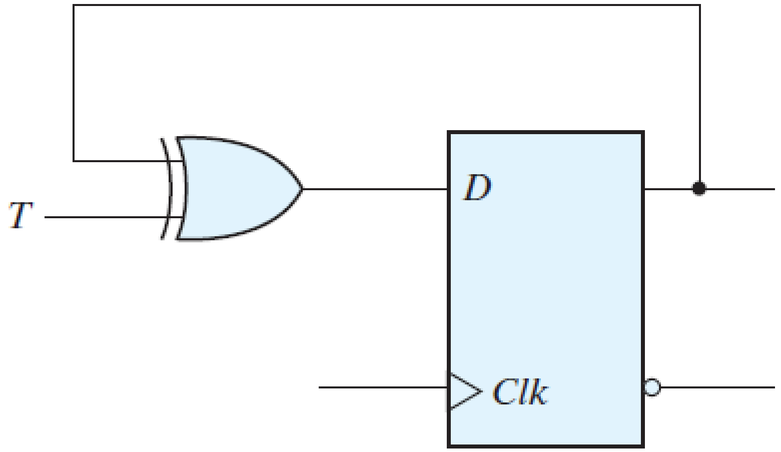
FIGURE : *JK flip-flop*

The *T (toggle) flip-flop:*

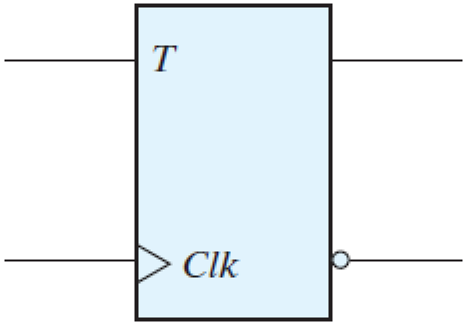
•*It is a complementing flip-flop and can be obtained from a JK* flip-flop when inputs *J and K are tied together as shown in the following Fig.*



(a) From *JK flip-flop*



(b) From *D flip-flop*



(c) Graphic symbol

•The *T flip-flop can also be constructed with a D flip-flop and an exclusive-OR gate as* shown in Fig. (b).

•The expression for the *D input is*

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

Characteristic Tables:

•A characteristic table defines the logical properties of a flip-flop by describing its operation in tabular form. They define the next state (i.e., the state that results from a clock transition) as a function of the inputs and the present state.

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

T Flip-Flop

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

D Flip-Flop

D	$Q(t+1)$	
0	0	Reset
1	1	Set

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:

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

•The characteristic equation for the *JK flip-flop:*

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

•The characteristic equation for the *T flip-flop:*

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

<u>ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS:</u>

•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.**

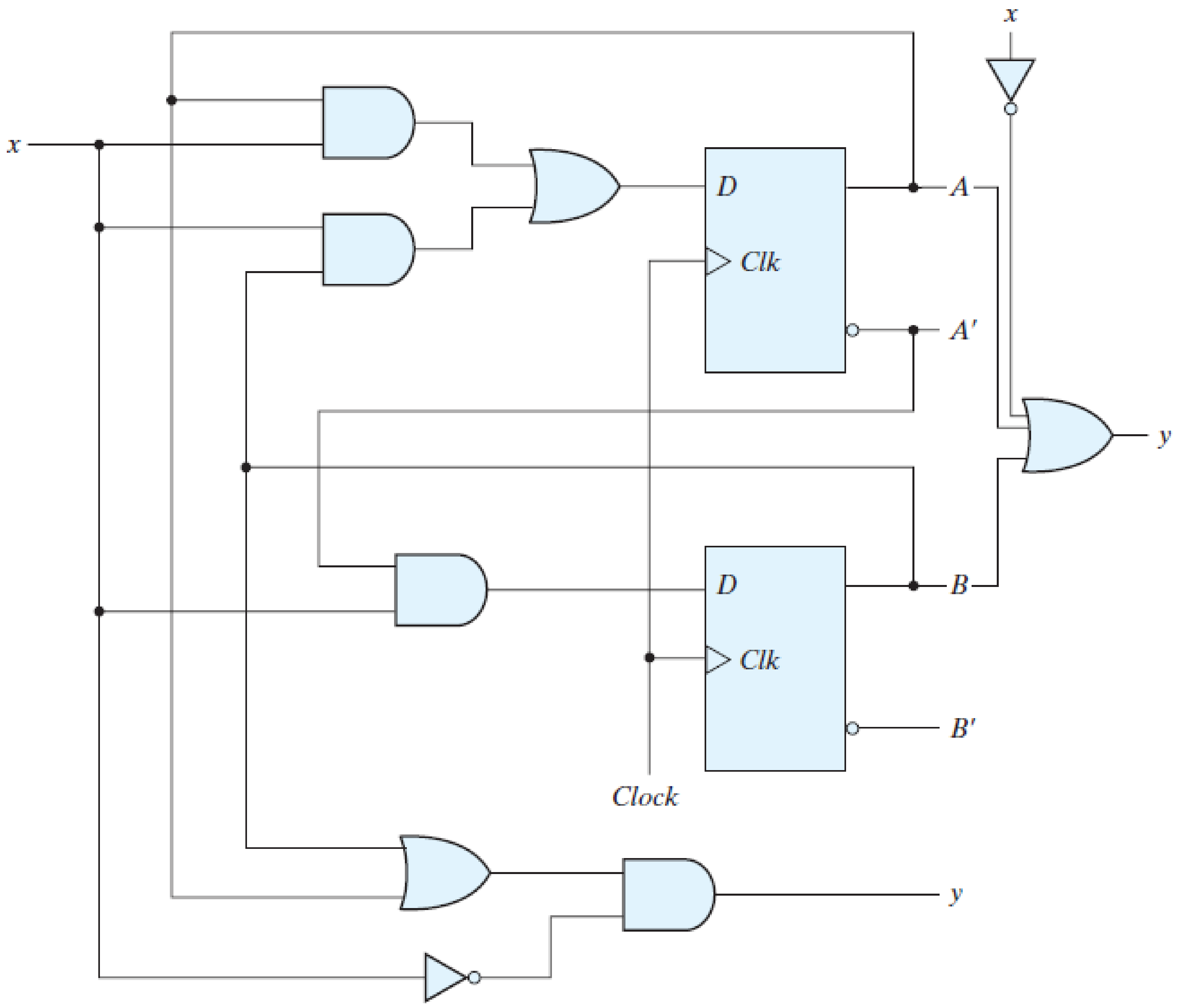
•The outputs and the next state are both a function of the inputs and the present state. The **analysis** of a sequential circuit consists of obtaining **a table or a diagram** for the time sequence of inputs, outputs, and internal states.

•Here an **algebraic expression** is written 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.

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.

•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:

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

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

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

•Similarly, the present-state value of the output can be expressed algebraically as

$y = (A + B)x'$

State Table or Transition Table:

•The time sequence of inputs, outputs, and flip-flop states can be enumerated in a *state table (sometimes called a transition table).* This 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.*

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

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 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.

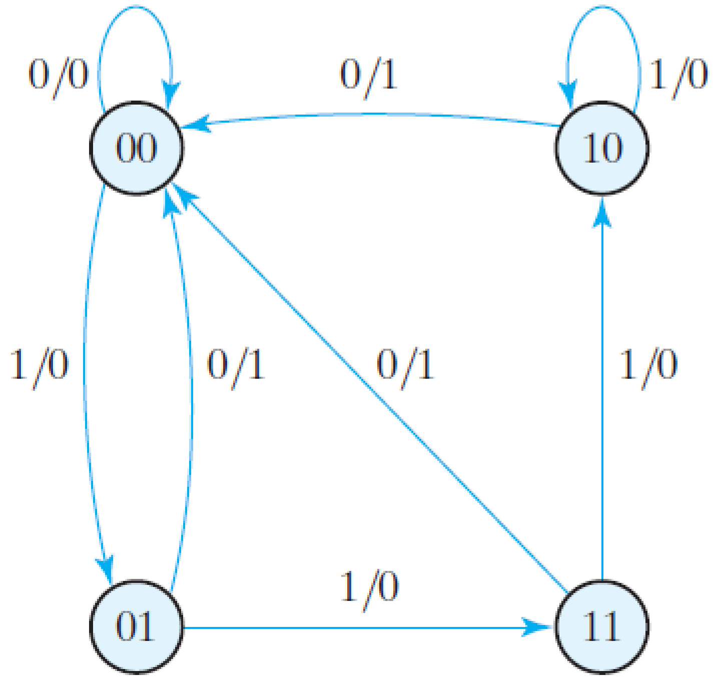


FIGURE: State diagram

•The state diagram provides the same information as the state table and is obtained directly from State Table.

•The steps presented in this example are summarized below:

Circuit diagram – Equations – State table - 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 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).***

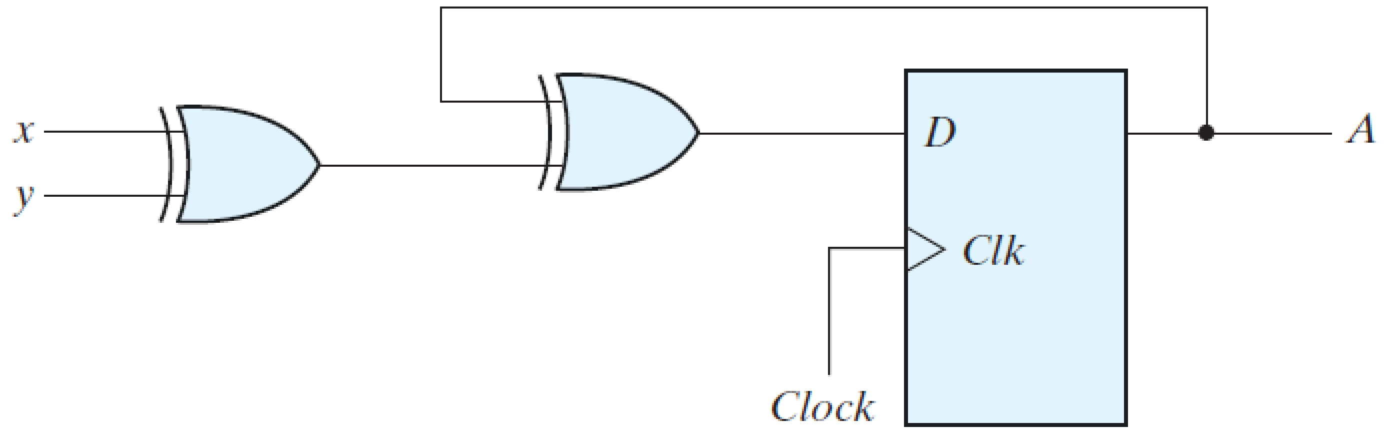
Analysis with *D Flip-Flops:*

Example: Write the circuit diagram, state table and state diagram for the following input equation:

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

•The *DA 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 the following Fig.



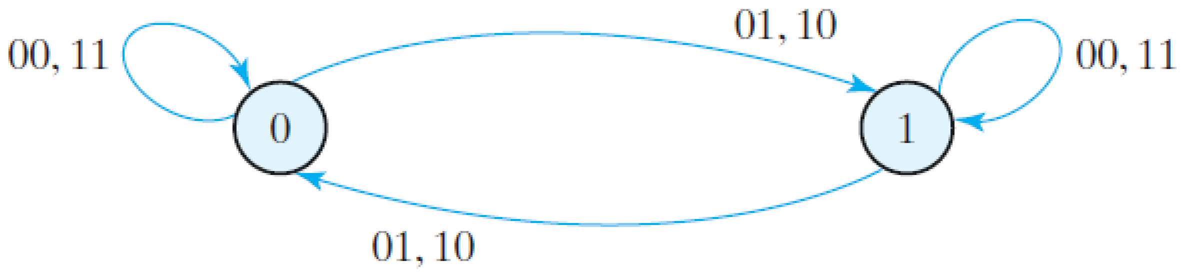
•The state table has one column for the present state of flip-flop *A, two columns for* the two inputs, and one column for the next state of *A* .

•The next-state values are obtained from the state equation

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

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

•The circuit has one flip-flop and two states. The **state diagram** consists of two circles, one for each state as shown in Fig.



Analysis with *JK Flip-Flops:*

•Consider the sequential circuit with two *JK flip-flops A and B and* one input *x, as shown in the following Fig.*

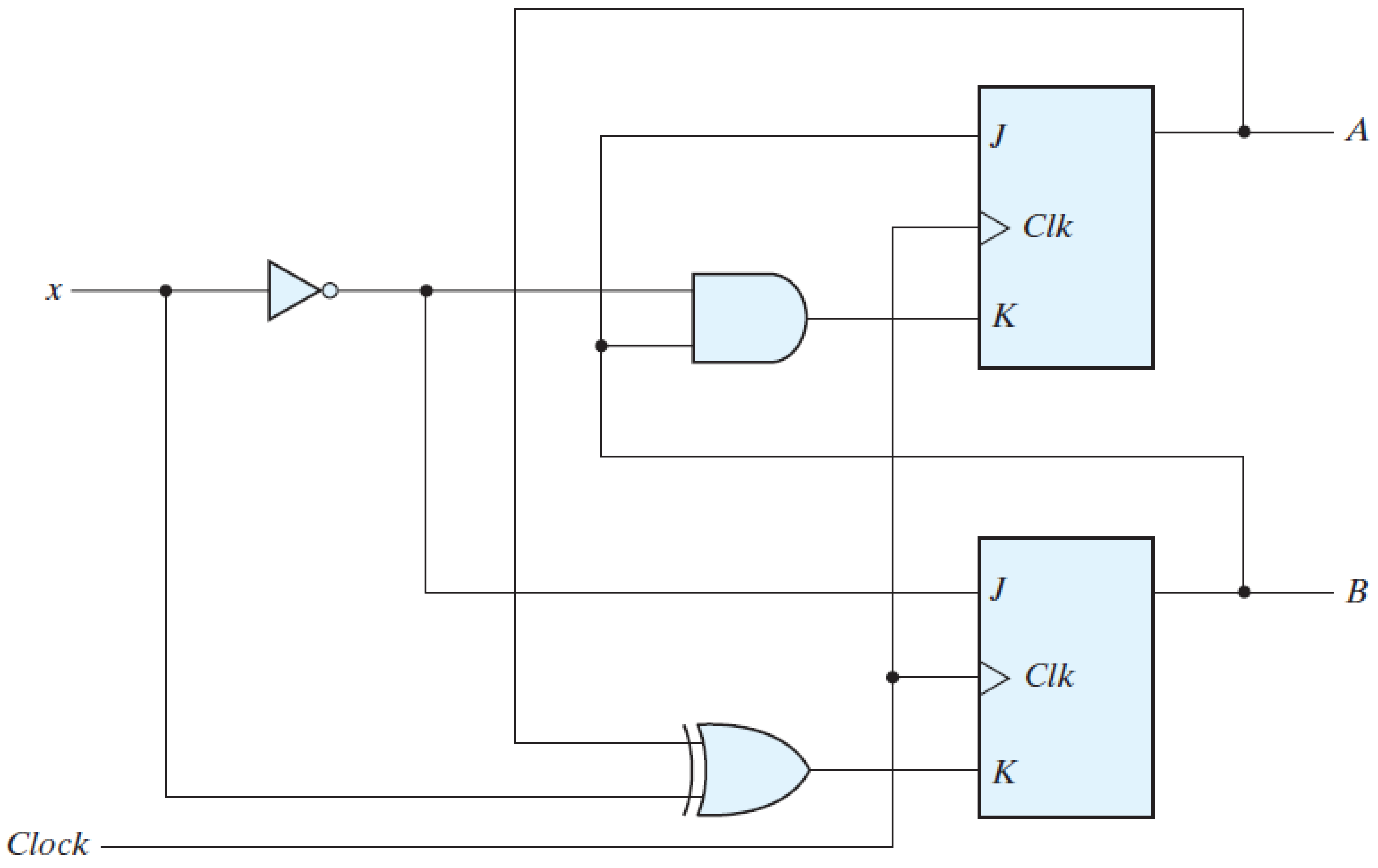


FIGURE: Sequential circuit with *JK flip-flop*

- The circuit can be specified by the flip-flop input equations

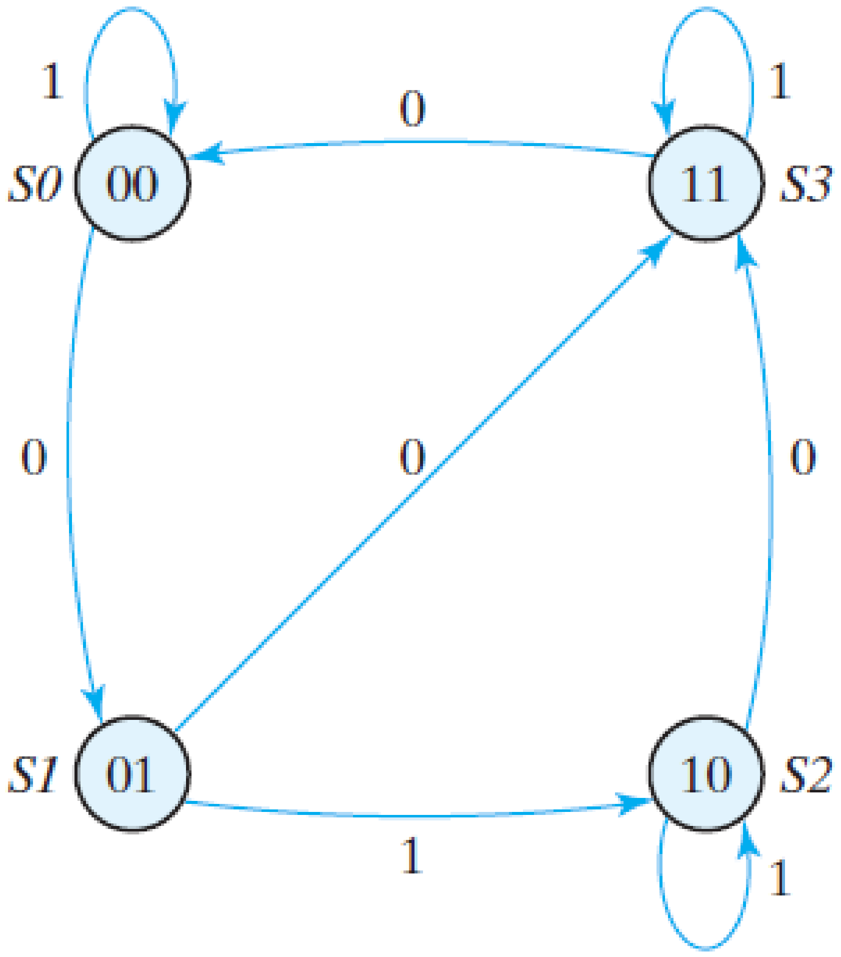
$$J_A = B \quad K_A = Bx'$$
$$J_B = x' \quad K_B = A'x + Ax' = A \oplus x$$

- The state table of the sequential circuit is shown in the following Table (using the characteristic table or equations)

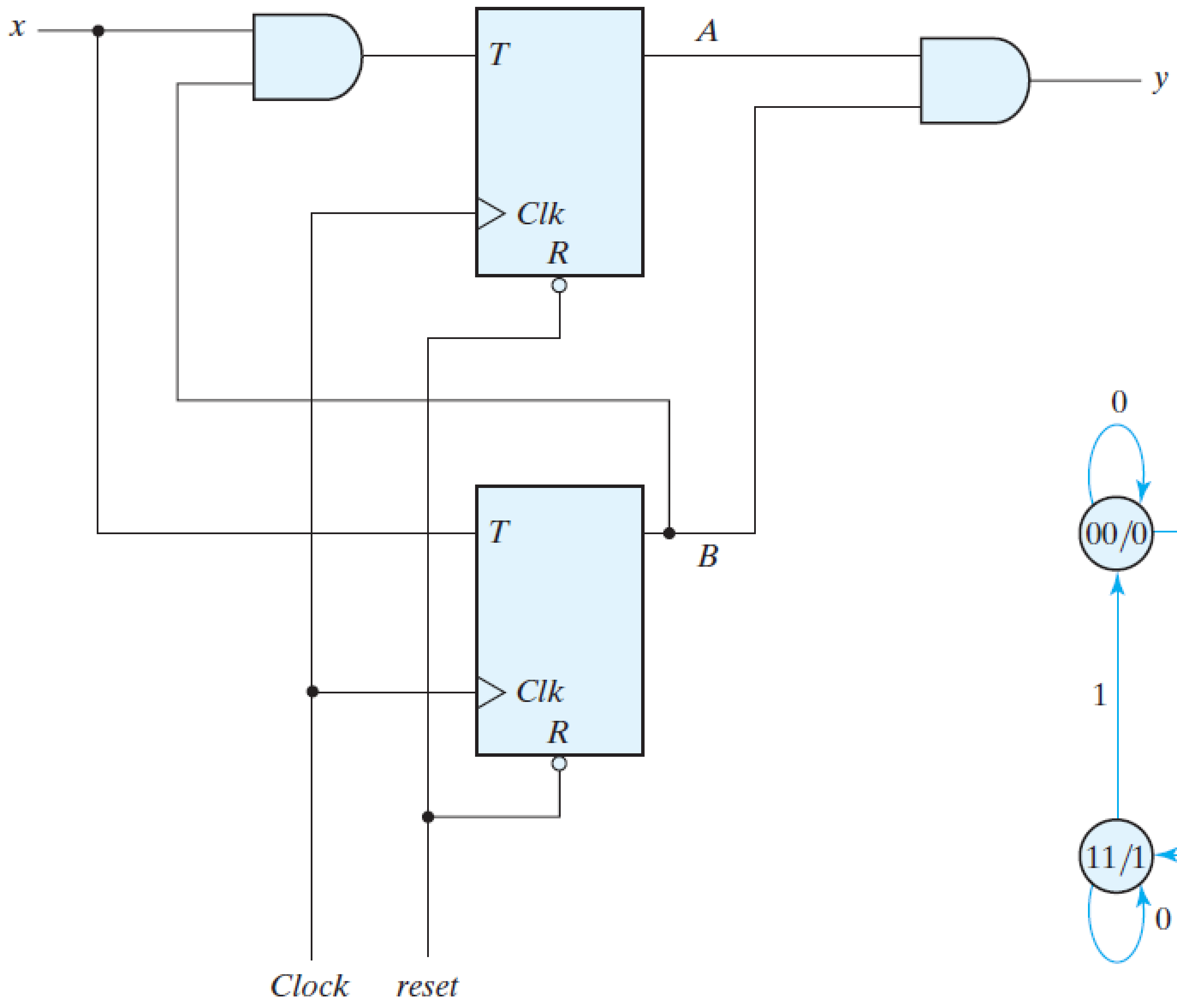
State Table for Sequential Circuit with JK Flip-Flops

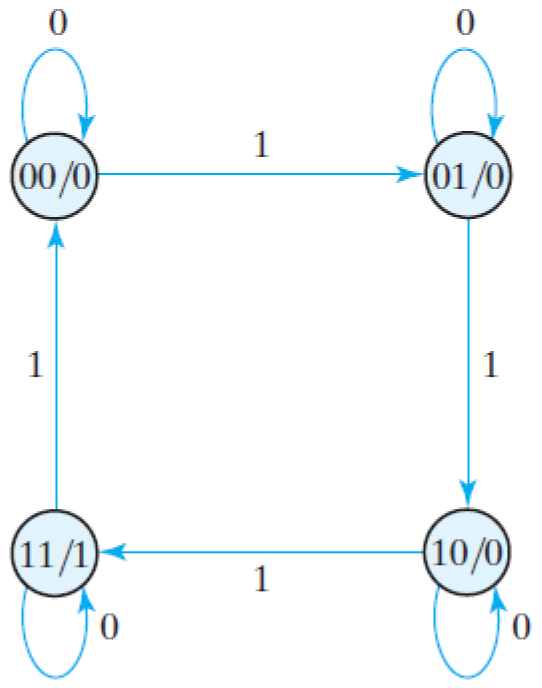
Present State		Input	Next State		Flip-Flop Inputs			
A	**B**	**x**	**A**	**B**	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

•The state diagram of the sequential circuit is shown in the following Fig.



Analysis with *T Flip-Flops:*





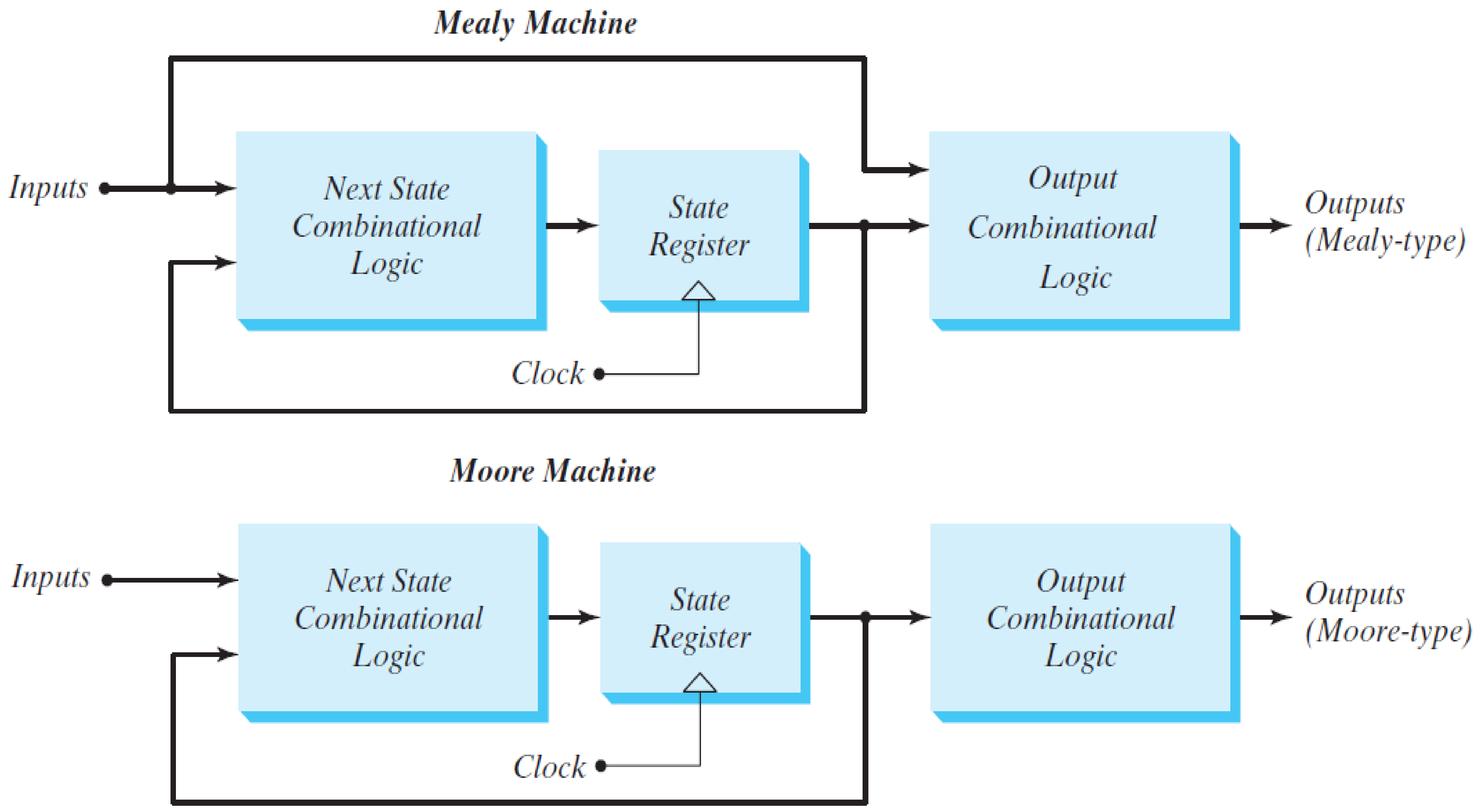
State Table for Sequential Circuit with T Flip-Flops

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

- 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:

•It is customary to distinguish between two models of sequential circuits: the Mealy model and the Moore model. Both are shown in Fig.



•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.

•**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.**

Verilog Models of SLC:

•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.

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. 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.