UNIT IV - SYNCHRONOUS SEQUENTIAL LOGIC

Storage Elements – Latches – Flip-Flops – Analysis of Clocked Sequential Circuits– Synthesizable HDL Models of Sequential Circuits–State Reduction and Assignment–Design Procedure

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.

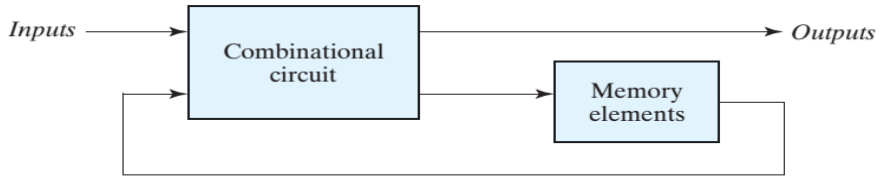


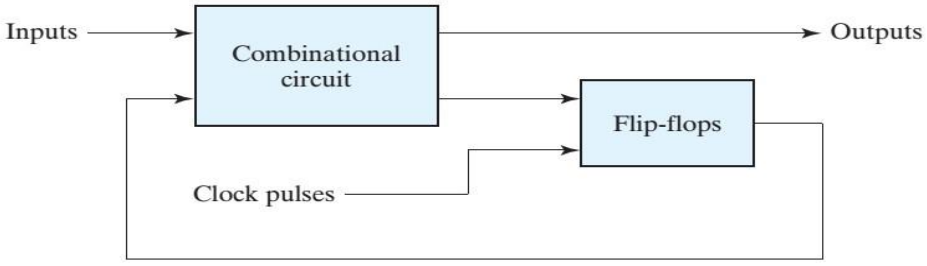
FIGURE 5.1
Block diagram of sequential circuit

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.

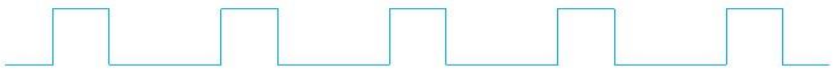
Types in sequential circuit:

1. A ***synchronous sequential*** circuit is a system whose behavior can be defined from the knowledge of its signals at discrete instants of time.
2. 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.

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***.



(a) Block diagram



(b) Timing diagram of clock pulses

FIGURE 5.2
Synchronous clocked sequential circuit

The storage elements (memory) used in clocked sequential circuits are called ***flipflop***s. 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 outputs are formed by a combinational logic function of the inputs to the circuit or the values stored in the flip-flops (or both).

STORAGE ELEMENTS: LATCHES

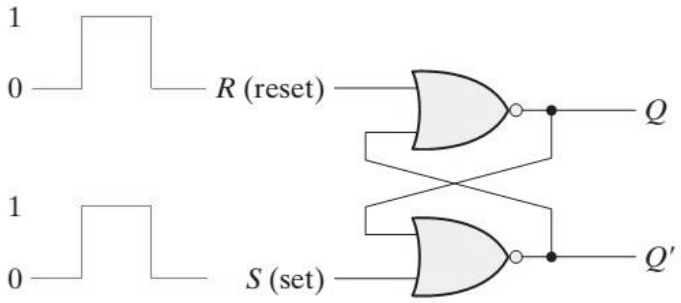
Storage elements that operate with signal levels (rather than signal transitions) are referred to as ***latches***; those controlled by a clock transition are ***flip-flops*** .

Latches are said to be level sensitive devices; flip-flops are edge-sensitive devices. The two types of storage elements are related because latches are the basic circuits from which all flip-flops are constructed. Although latches are useful for storing binary information and for the design of asynchronous sequential circuits, they are not practical for use as storage elements in synchronous sequential circuits. Because they are the building blocks of flip-flops.

SR LATCH

The SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates, and two inputs labeled S for set and R for reset. The SR latch constructed with two cross-coupled NOR gates is shown in Fig. 5.3 . The latch has two useful states. When output Q=1 and Q' =0, the latch is said to be in the set state. When Q=0 and Q' =1, it is in the reset state. If both the input are 0 then there is no change in output.

If a 1 is applied to both the Sand R inputs of the latch, both outputs go to 0. This action produces an undefined next state, because the state that results from the input transitions depends on the order in which they return to 0.



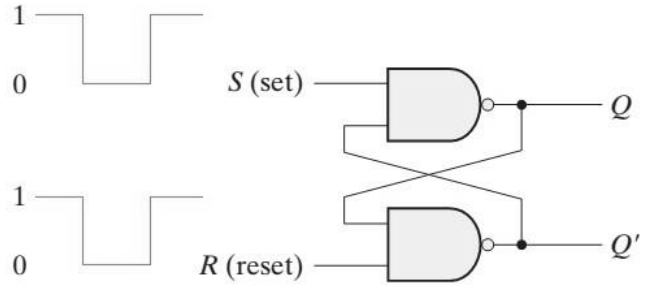
(a) Logic diagram

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

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



(a) Logic diagram

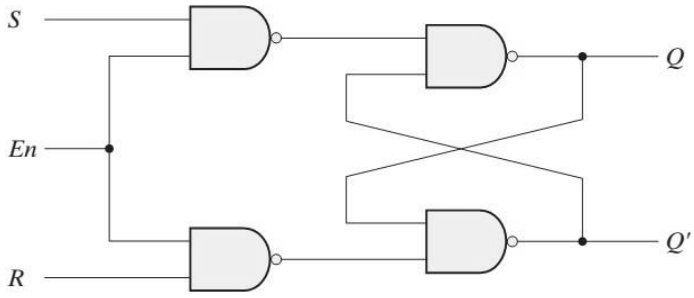
S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	(forbidden)

(b) Function table

FIGURE 5.4
SR **latch with NAND gates**

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 Enacts 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.

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 Sand R



(a) Logic diagram

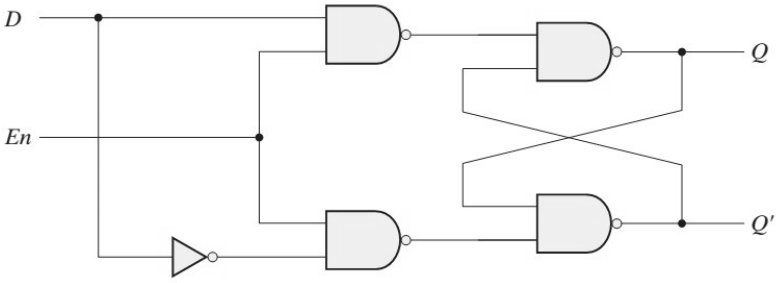
En	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

(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 Sand Rare never equal to 1 at the same time. This is done in the D latch 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. 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.



(a) Logic diagram

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

(b) Function table

FIGURE 5.6
D latch

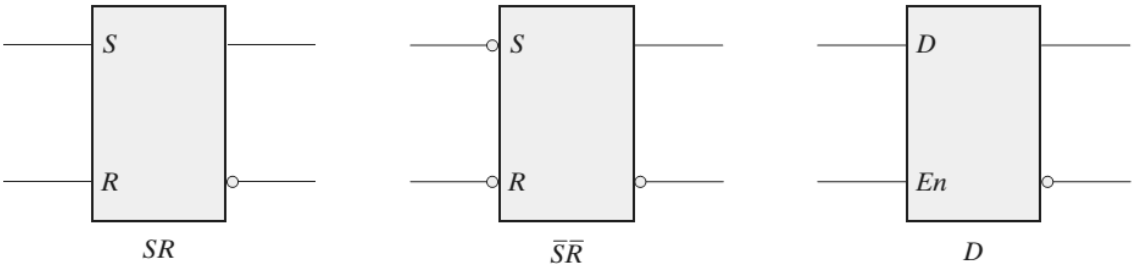


FIGURE 5.7
Graphic symbols for latches

STORAGE ELEMENTS: FLIP-FLOPS

The state of a latch or flip-flop is switched by a change in the control input. This momentary change is called a trigger, and the transition it causes is said to trigger the flip-flop. As long as the pulse input remains at this level, any changes in the data input will change the output and the state of the latch.

The new state of a latch appears at the output while the pulse is still active. This output is connected to the inputs of the latches through the combinational circuit. If the inputs applied to the latches change while the clock pulse is still at the logic-1 level, the latches will respond to new values and a new output state may occur.

The result is an unpredictable situation Flip-flop circuits are constructed in such a way as to make them operate properly when they are part of a sequential circuit that employs a common clock. The problem with the latch is that it responds to a change in the level of a clock pulse. As shown in Fig. 5.8 (a), a positive level response in the enable input allows changes in the output when the D input changes while the clock pulse stays at logic 1.

A clock pulse goes through two transitions: from 0 to 1 and the return from 1 to 0. As shown in Fig. 5.8 , the positive transition is defined as the positive edge and the negative transition as the negative edge.

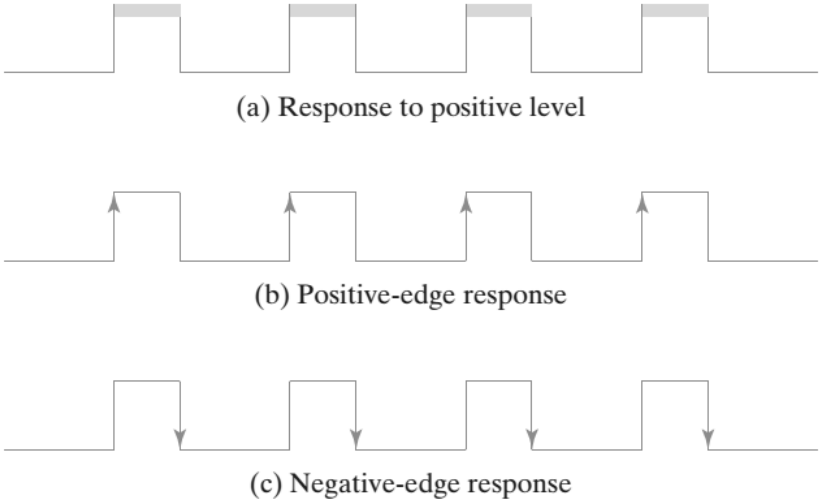


FIGURE 5.8
Clock response in latch and flip-flop

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.

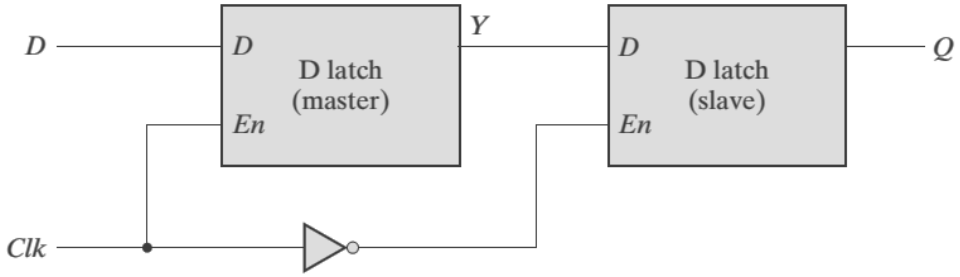
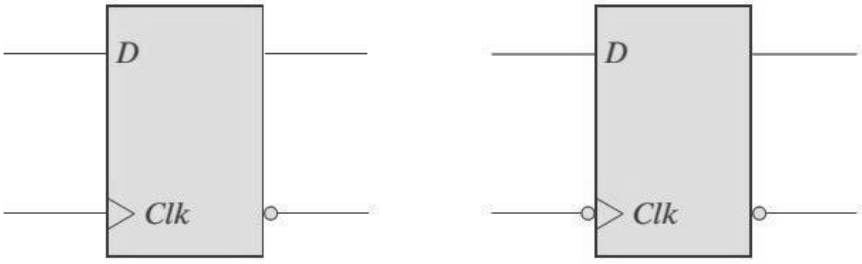


FIGURE 5.9
Master–slave *D* flip-flop

When the clock is 0, the output of the inverter is 1. The slave latch is enabled, and its output Q is equal to the master output Y. The master latch is disabled because Clk=0. When the input pulse changes to the logic-1 level, the data from the external D input are transferred to the master. The slave, however, is disabled as long as the clock remains at the 1 level, because its enable input is equal to 0. Any change in the input changes the master output at Y, but cannot affect the slave output.

When the clock pulse returns to 0, the master is disabled and is isolated from the D input. At the same time, the slave is enabled and the value of Y is transferred to the output of the flip-flop at Q. Thus, a change in the output of the flip-flop can be triggered only by and during the transition of the clock from 1to 0.



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

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

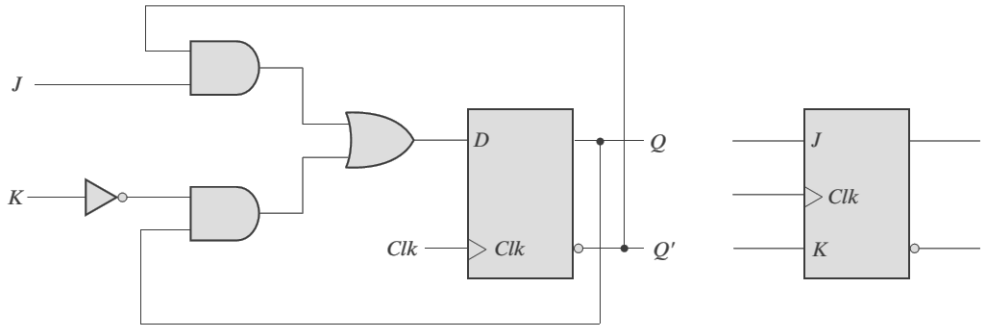
OTHER FLIP-FLOPS

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. 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 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:

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

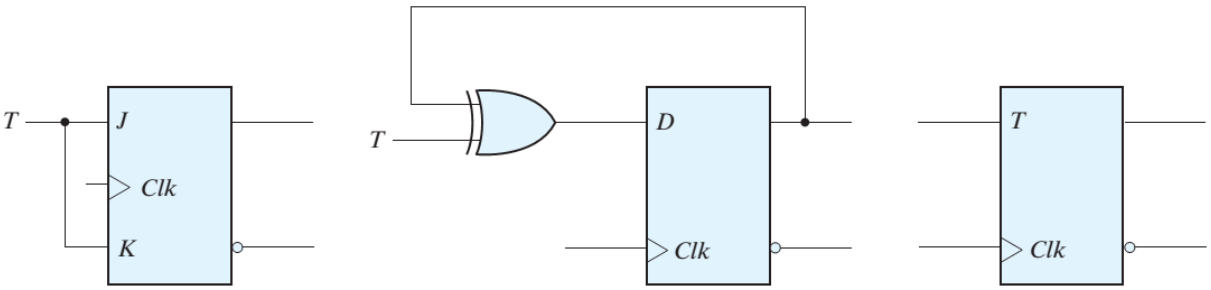


(a) Circuit diagram (b) Graphic symbol

FIGURE 5.12
JK flip-flop

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 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.



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

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

T=0 (J=K=0), a clock edge does not change the output. When T=1 (J=K=1),

a clock edge complements the output. The complementing flip-flop is useful for designing binary counters.

The T flip-flop can be constructed with a D flip-flop and an exclusive-OR gate as shown in Fig. 5.13 (b). The expression for the D input is D=T⊕Q=TQ' +T'Q. When T=0, D=Q and there is no change in the output. When T=1, D=Q' and the output complements. The graphic symbol for this flip-flop has a T symbol in the input.

CHARACTERISTIC TABLES

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

A characteristic table defines the logical properties of a flip-flop by describing its operation in tabular form.

Q(t) refers to the present state Q(t +1) is the next state one clock period later.

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

The next state of a D flip-flop is dependent only on the D input and is independent of the present state. This can be expressed as $Q(t+1)=D$.

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

CHARACTERISTIC EQUATIONS

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

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

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

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

The characteristic equation for the Tflip-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.

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:

$$A(t+1)=A(t)x(t)+B(t)x(t) \quad \text{or} \quad A(t+1)=Ax+Bx$$

$$B(t+1)=A'(t)x(t) \quad \text{or} \quad B(t+1)=A'x$$

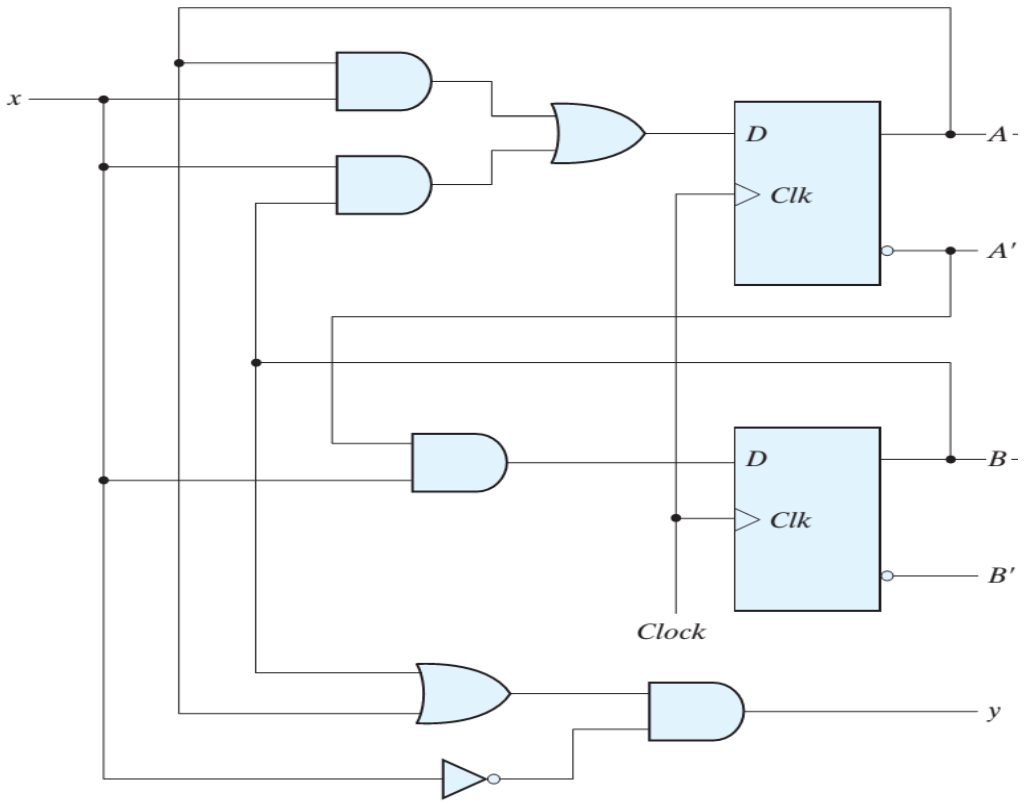


FIGURE 5.15
Example of sequential circuit

A state equation is an algebraic expression that specifies the condition for a flip-flop state transition. The left side of the equation, with (t +1), denotes the next state of the flip-flop one clock edge later. The right side of the equation is a Boolean expression that specifies the present state and input conditions that make the next state equal to 1

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

$$y(t) = [A(t) + B(t)]x'(t) \text{ or } y = (A+B)x'$$

State Table

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

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

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

$$y=Ax' +Bx'$$

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 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 state diagram of the sequential circuit of Fig. 5.15 is shown in Fig. 5.16 .

The state diagram provides the same information as the state table and is obtained directly from Table 5.2 or Table 5.3 . The binary number inside each circle identifies the state of the flip-flops. The directed lines are labeled with two binary numbers separated by a slash. The input value during the present state is labeled first, and the number after the slash gives the output during the present state with the given input.

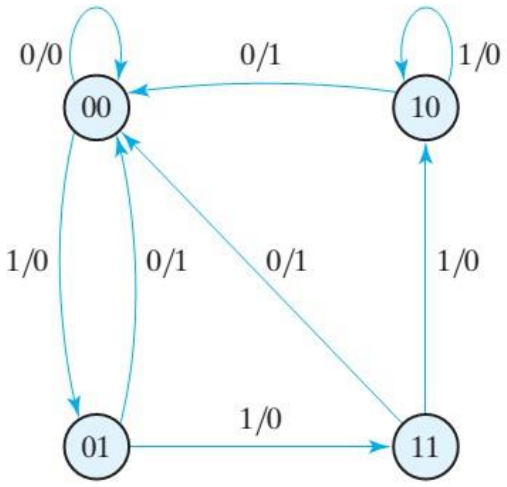
For example, the directed line from state 00 to 01 is labeled 1/0, meaning that when the sequential circuit is in the present state 00 and the input is 1, the output is 0. After the next clock cycle, the circuit goes to the next state, 01. If the input changes to 0, then the output becomes 1, but if the input remains at 1, the output stays at 0. This information is obtained from the state

diagram along the two directed lines emanating from the circle with state 01. A directed line connecting a circle with itself indicates that no change of state occurs.

The steps presented in this example are summarized below:

Circuit diagram→Equations → State table →State diagram

The state diagram gives a pictorial view of state transitions and is the form more suitable for human interpretation of the circuit's operation.

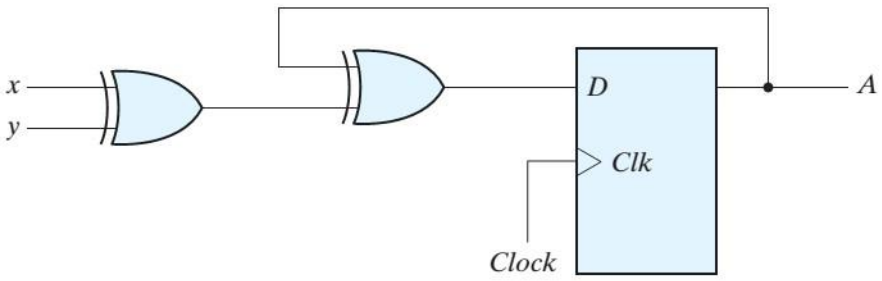


Analysis with D Flip-Flops

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

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

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



(a) Circuit diagram

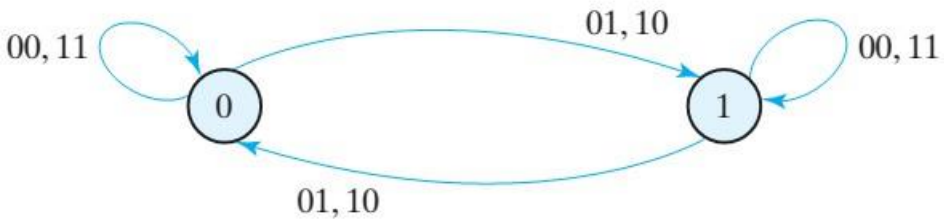
Present state	Inputs		Next state
A	*x*	*y*	*A*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(b) State table

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

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

The circuit has one flip-flop and two states. The state diagram consists of two circles, one for each state as shown in Fig. 5.17 (c). The present state and the output can be either 0 or 1, as indicated by the number inside the circles. The two inputs can have four possible combinations for each state. Two input combinations during each state transition are separated by a comma to simplify the notation.



(c) State diagram

FIGURE 5.17
Sequential circuit with *D* flip-flop

Analysis with JK Flip-Flops

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

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.

$$J_A = B \quad ; \quad K_A = Bx'$$

$$J_B = x' \; ; \; K_B = A'x + Ax' = A \oplus x$$

The state table of the sequential circuit is shown in Table 5.4 . The present-state and input columns list the eight binary combinations. The binary values listed under the columns labeled flip-flop inputs are not part of the state table, but they are needed for the purpose of evaluating the next state as specified in step 2 of the procedure.

The next state of each flip-flop is evaluated from the corresponding J and K inputs and the characteristic table of the JK flip-flop listed in Table 5.1 . There are four cases to consider. When J =1 and K=0, the next state is 1. When J =0 and K=1, the next state is 0. When J =K=0 , there is no change of state and the next-state value is the same as that of the present state. When J =K=1, the next-state bit is the complement of the present-state bit. Examples of the last two cases occur in the table when the present state AB is 10 and input x is 0. J_A and K_A are both equal to 0 and the

present state of A is 1. Therefore, the next state of A remains the same and is equal to 1. In the same row of the table, J_B and K_B are both equal to 1. Since the present state of B is 0, the next state of B is complemented and changes to 1.

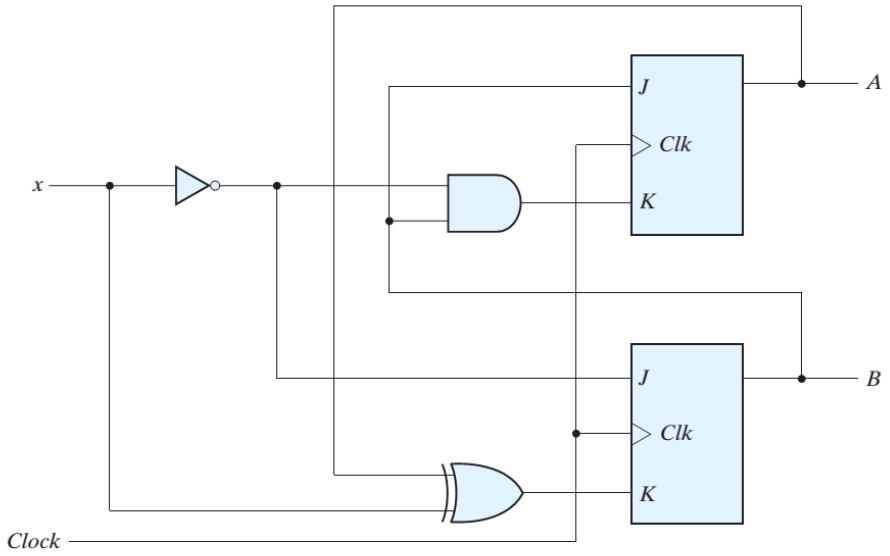


FIGURE 5.18
Sequential circuit with *JK* flip-flop

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

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

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

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

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

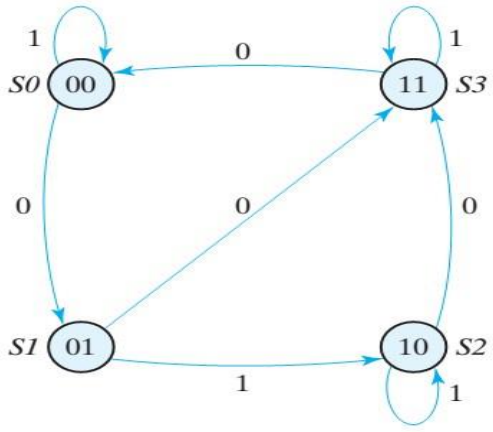
The state equation provides the bit values for the column headed "Next State" for A in the state table. Similarly, the state equation for flip-flop B can be derived from the characteristic equation by substituting the values of JB and KB:

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

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

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 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

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

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

$$TA=Bx$$

$$TB=x$$

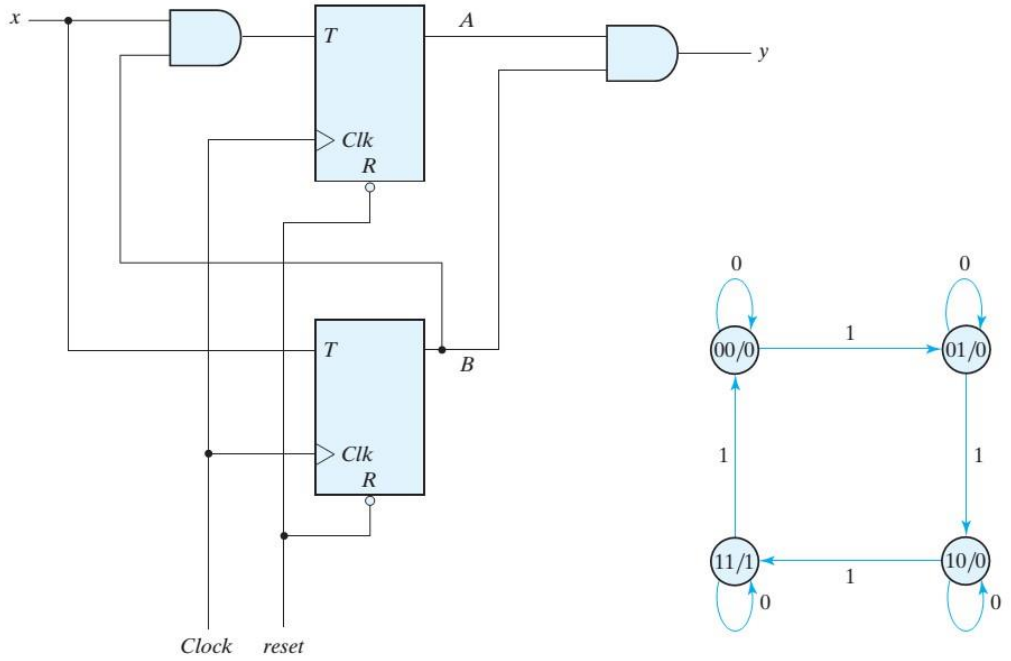
$$y=AB$$

The state table for the circuit is listed in Table 5.5 . The values for yare 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

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

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

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



(a) Circuit diagram (b) State diagram

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

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

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

Mealy and Moore Models of Finite State Machines

The most general model of a sequential circuit has inputs, outputs, and internal states. It is customary to distinguish between two models of sequential circuits: the Mealy model and the Moore model.

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.

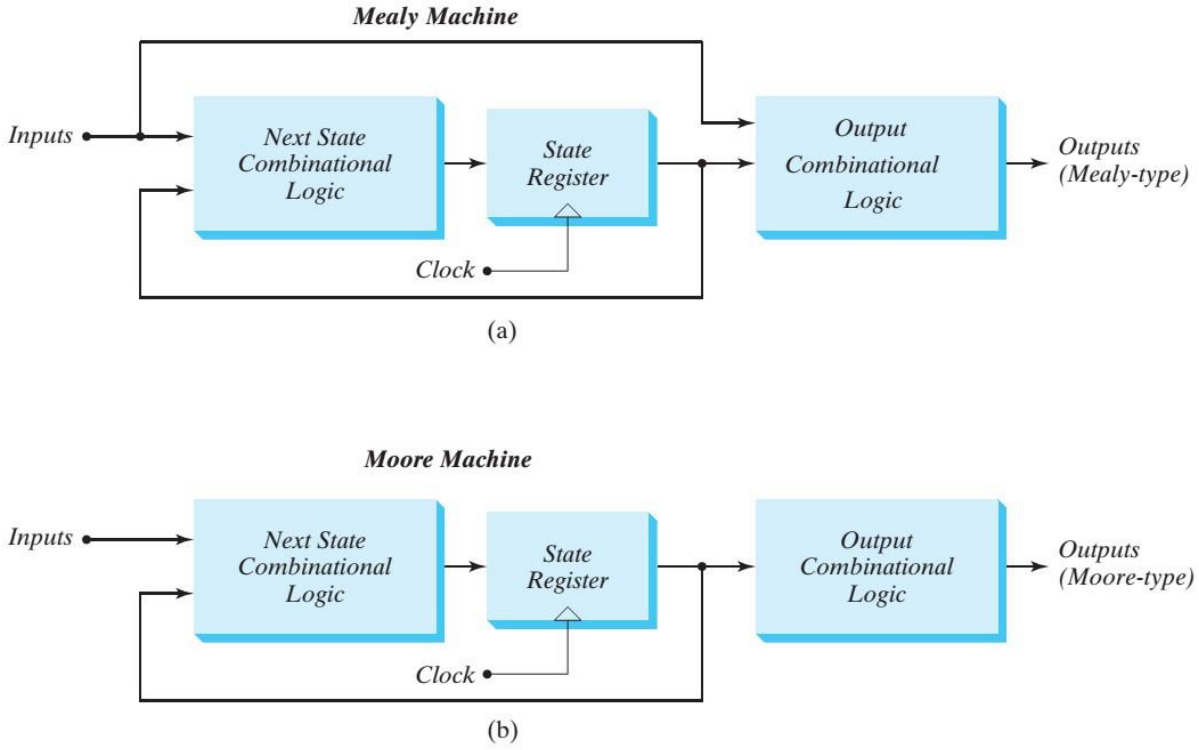


FIGURE 5.21
Block diagrams of Mealy and Moore state machines

STATE REDUCTION AND ASSIGNMENT

The analysis of sequential circuits starts from a circuit diagram and culminates in a state table or diagram. The design(synthesis) of a sequential circuit starts from a set of specifications and culminates in a logic diagram. Design procedures are presented in Section 5.8. Two sequential circuits may exhibit the same input–output behavior, but have a different number of internal states in their state diagram. In general, reducing the number of flip flops reduces the cost of a circuit.

State Reduction

The reduction in the number of flip-flops in a sequential circuit is referred to as the state-reduction problem. State-reduction algorithms are concerned with procedures for reducing the number of states in a state table, while keeping the external input–output requirements unchanged. Since m flip-flops produce 2^m states, a reduction in the number of states may (or may not) result in a reduction in the number of flip-flops. An unpredictable effect in reducing the number of flip-flops

is that sometimes the equivalent circuit (with fewer flip-flops) may require more combinational gates to realize its next state and output logic.

The following ***algorithm*** for the state reduction of a completely specified state table is given here without proof: "*Two states are said to be equivalent if, for each member of the set of inputs, they give exactly the same output and send the circuit either to the same state or to an equivalent state*." When two states are equivalent, one of them can be removed without altering the input–output relationships.

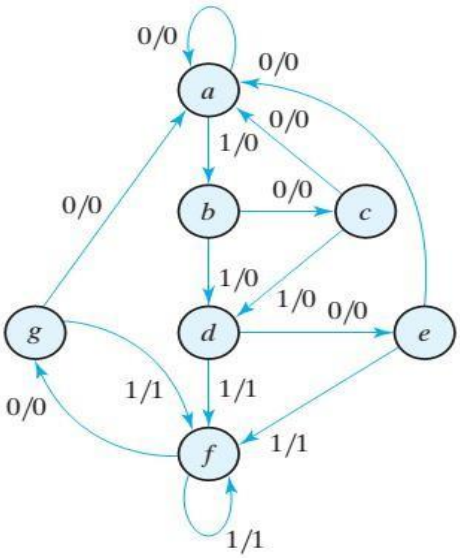


FIGURE 5.25
State diagram

Table 5.6
State Table

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

Now apply this algorithm to Table 5.6 . Going through the state table, we look for two present states that go to the same next state and have the same output for both input combinations. States e and g are two such states: They both go to states a and f and have outputs of 0 and 1 for x=0 and x=1, respectively. Therefore, states g and e are equivalent, and one of these states can be removed. The row with present state g is removed, and state g is replaced by state e each time it occurs in the columns headed "Next State."

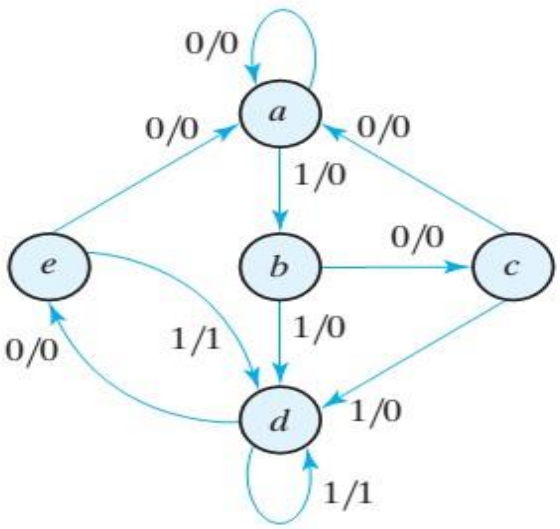
Table 5.7
Reducing the State Table

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

Table 5.8
Reduced State Table

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

Present state f now has next states e and f and outputs 0 and 1 for x=0 and x=1 , respectively. The same next states and outputs appear in the row with present state d. Therefore, states f and d are equivalent, and state f can be removed and replaced by d. The final reduced table is shown in Table 5.8 . The state diagram for the reduced table consists of only five states and is shown in Fig. 5.26. In fact, this sequence is exactly the same as that obtained for Fig. 5.25 if we replace g by e and f by d.



State Assignment

In order to design a sequential circuit with physical components, it is necessary to assign unique coded binary values to the states. For a circuit with m states, the codes must contain n bits, where $2^n \geq m$. For example, with three bits, it is possible to assign codes to eight states, denoted by binary numbers 000 through 111. If the state table of Table 5.6 is used, we must assign binary values to seven states; the remaining state is unused.

The simplest way to code five states is to use the first five integers in binary counting order, as shown in the first assignment of Table 5.9 . Another similar assignment is the Gray code shown in assignment 2. Here, only one bit in the code group changes when going from one number to the next. This code makes it easier for the Boolean functions to be placed in the map for simplification.

Table 5.9
Three Possible Binary State Assignments

State	Assignment 1, Binary	Assignment 2, Gray Code	Assignment 3, One-Hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

Another possible assignment often used in the design of state machines to control data-path units is the one-hot assignment. This configuration uses as many bits as there are states in the circuit. At any given time, only one bit is equal to 1 while all others are kept at 0. One-hot encoding usually leads to simpler decoding logic for the next state and output. One-hot machines can be faster than machines with sequential binary encoding.

Table 5.10
Reduced State Table with Binary Assignment 1

	Next State		Output	
Present State	$x = 0$	$x = 1$	$x = 0$	$x = 1$
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

Table 5.10 is the reduced state table with binary assignment 1 substituted for the letter symbols of the states. A different assignment will result in a state table with different binary values for the states. The binary form of the state table is used to derive the next state and output-forming

combinational logic part of the sequential circuit. The complexity of the combinational circuit depends on the binary state assignment chosen.

DESIGN PROCEDURE

Design procedures or methodologies specify hardware that will implement a desired behavior. The design effort for small circuits may be manual, but industry relies on automated synthesis tools for designing massive integrated circuits. The sequential building block used by synthesis tools is the D flip-flop. Together with additional logic, it can implement the behavior of JK and T flip-flops.

The procedure for designing synchronous sequential circuits can be summarized by

a list of recommended steps:

1. From the word description and specifications of the desired operation, derive a state diagram for the circuit.

2. Reduce the number of states if necessary.

3. Assign binary values to the states.

4. Obtain the binary-coded state table.

5. Choose the type of flip-flops to be used.

6. Derive the simplified flip-flop input equations and output equations.

7. Draw the logic diagram.

Example : Detect a sequence of three or more Consecutive 1's in a string

The state diagram for this type of circuit is shown in Fig. 5.27 . It is derived by starting with state S0 , the reset state. If the input is 0, the circuit stays in S0 , but if the input is 1, it goes to state S1 to indicate that a 1 was detected. If the next input is 1, the change is to state S2 to indicate the arrival of two consecutive 1's, but if the input is 0, the state goes back to S0. The third consecutive 1 sends the circuit to state S3. If more 1's are detected, the circuit stays in S3. Any 0 input sends the circuit back to S0 This is a Moore model sequential circuit, since the output is 1 when the circuit is in state S3and is 0 otherwise.

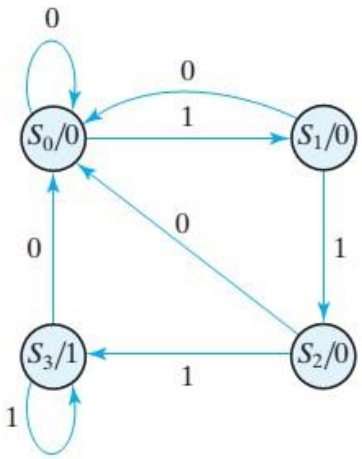


FIGURE 5.27
State diagram for sequence detector

Synthesis Using D Flip-Flops

To design the circuit by hand, we need to assign binary codes to the states and list the state table. This is done in Table 5.11 . The table is derived from the state diagram of Fig. 5.27 with a sequential binary assignment. We choose two D flip-flops to represent the four states, and we label their outputs A and B. There is one input x and one output y. The characteristic equation of the D flip-flop is Q(t +1) =DQ, which means that the next-state values in the state table specify the D input condition for the flip-flop. The flip-flop input equations can be obtained directly from the next-state columns of A and Band expressed in sum-of-minterms form as

$$A(t+1)=D_A(A,B,x)=\sum(3,5,7)$$

$$B(t+1)=D_B(A,B,x)=\sum(1,5,7)$$

$$y(A,B,x)=\sum(6,7)$$

Table 5.11
State Table for Sequence Detector

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	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

where A and Bare the present-state values of flip-flops A and B, x is the input, and D_A and D_B are the input equations.

The Boolean equations are simplified by means of the maps plotted in Fig. 5.28 . The simplified equations are

$$DA=Ax+Bx$$

$$DB=Ax+B'x$$

$$y=AB$$

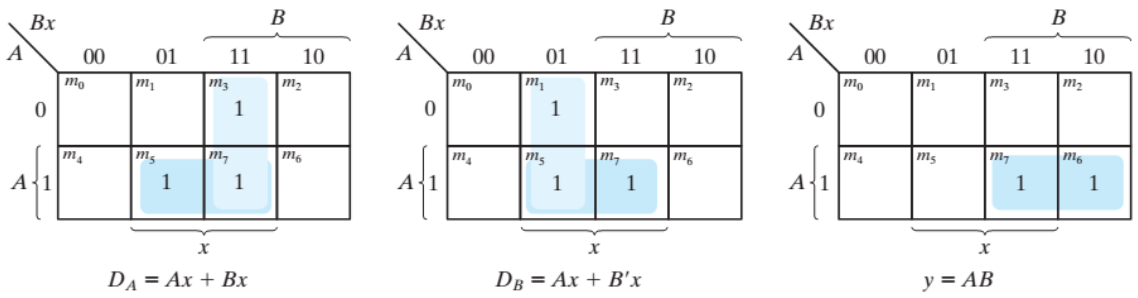


FIGURE 5.28
K-Maps for sequence detector

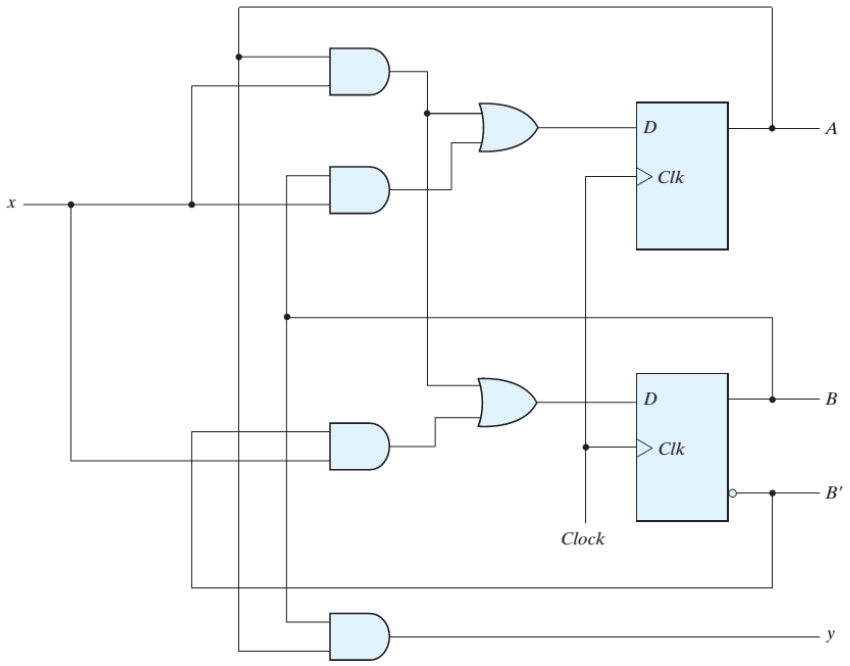


FIGURE 5.29
Logic diagram of a Moore-type sequence detector

Excitation Tables

The design of a sequential circuit with flip-flops other than the D type is complicated by the fact that the input equations for the circuit must be derived indirectly from the state table. When D-type flip-flops are employed, the input equations are obtained directly from the next state. This is not the case for the JK and T types of flip-flops. In order to determine the input equations for these flip-flops, it is necessary to derive a functional relationship between the state table and the input equations.

we need a table that lists the required inputs for a given change of state. Such a table is called an ***excitation table.***

Table 5.12 shows the excitation tables for the two flip-flops (JKand T). Each table has a column for the present state Q(t), a column for the next state Q(t +1), and a column for each input to show how the required transition is achieved

Table 5.12
Flip-Flop Excitation Tables

Q(t)	Q(t = 1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(a) *JK* Flip-Flop

Q(t)	Q(t = 1)	T
0	0	0
0	1	1
1	0	1
1	1	0

(b) *T* Flip-Flop

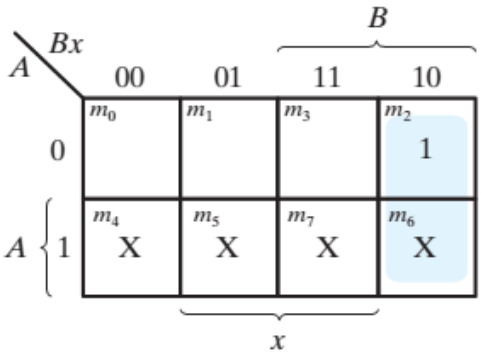
Synthesis Using JK Flip-Flops

The manual synthesis procedure for sequential circuits with JK flip-flops is the same as with D flip-flops, except that the input equations must be evaluated from the present state to the next-state transition derived from the excitation table. To illustrate the procedure, we will synthesize the sequential circuit specified by Table 5.13 .

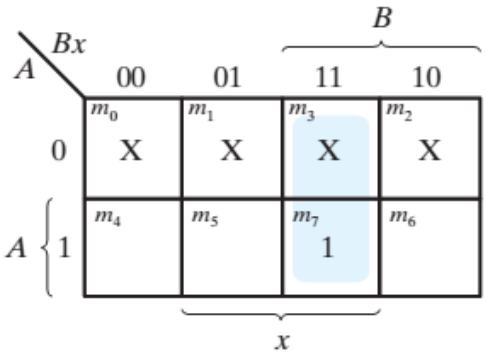
Table 5.13
State Table and JK Flip-Flop Inputs

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	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

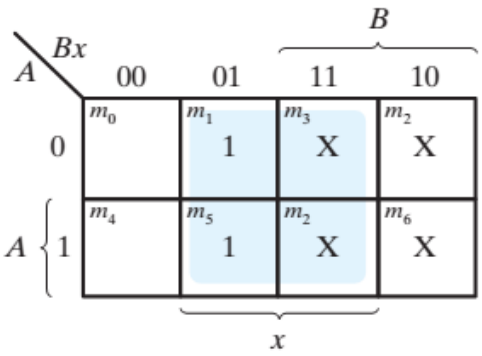
The flip-flop inputs in Table 5.13 specify the truth table for the input equations as a function of present state A, present state B, and input x. The input equations are simplified in the maps of Fig. 5.30



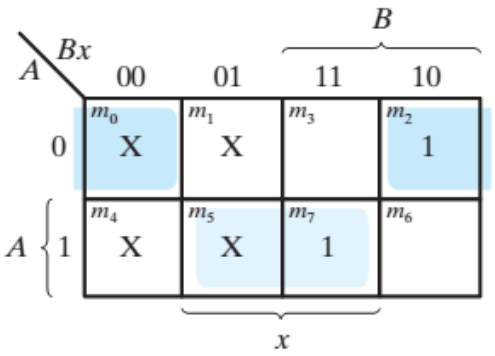
$J_A = Bx'$



$K_A = Bx$



$J_B = x$



$K_B = (A \oplus x)'$

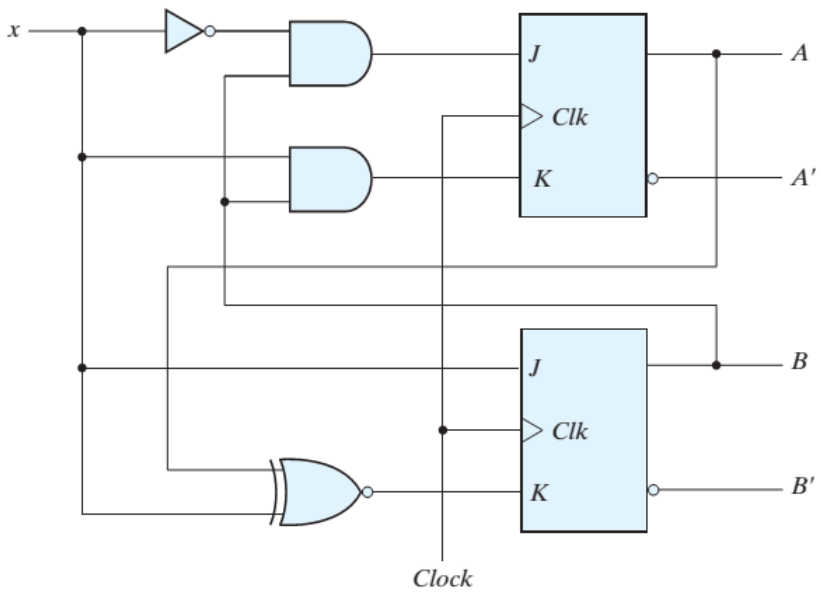


FIGURE 5.31
Logic diagram for sequential circuit with *JK* flip-flops

Synthesis Using T Flip-Flops

The procedure for synthesizing circuits using T flip-flops will be demonstrated by designing a binary counter. An n-bit binary counter consists of n flip-flops that can count in binary from 0 to 2^n-1. The state diagram of a three-bit counter is shown in Fig. 5.32 .

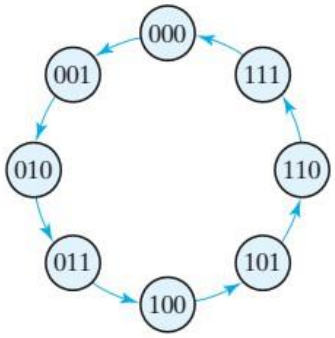


FIGURE 5.32
State diagram of three-bit binary counter

Table 5.14
State Table for Three-Bit Counter

Present State			Next State			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

The flip-flop input equations are simplified in the maps of Fig. 5.33 . Note that T_{A0} has 1's in all eight minterms because the least significant bit of the counter is complemented with each count. A Boolean function that includes all minterms defines a constant value of 1.

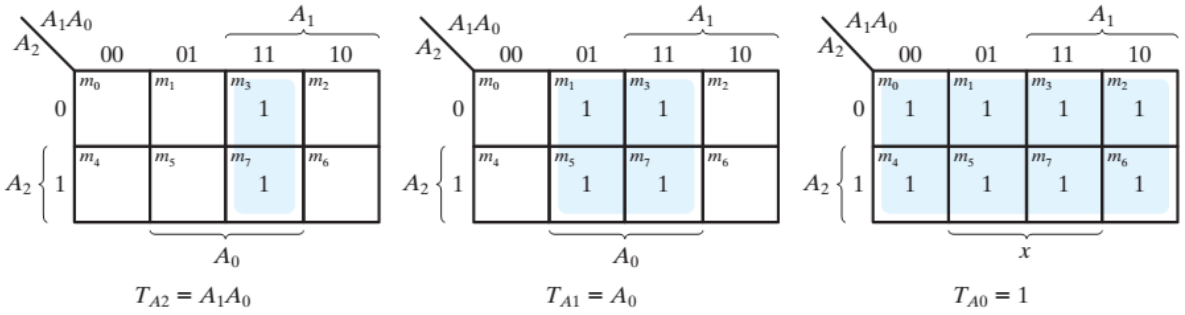


FIGURE 5.33
Maps for three-bit binary counter

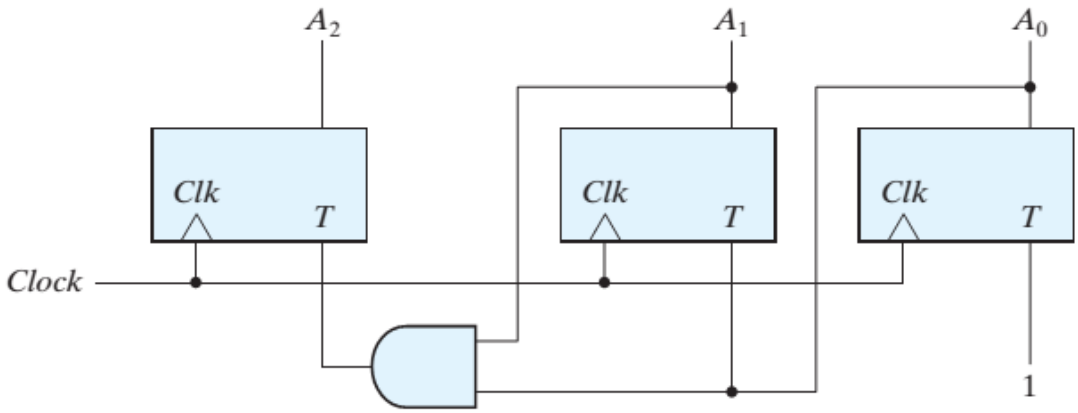


FIGURE 5.34
Logic diagram of three-bit binary counter

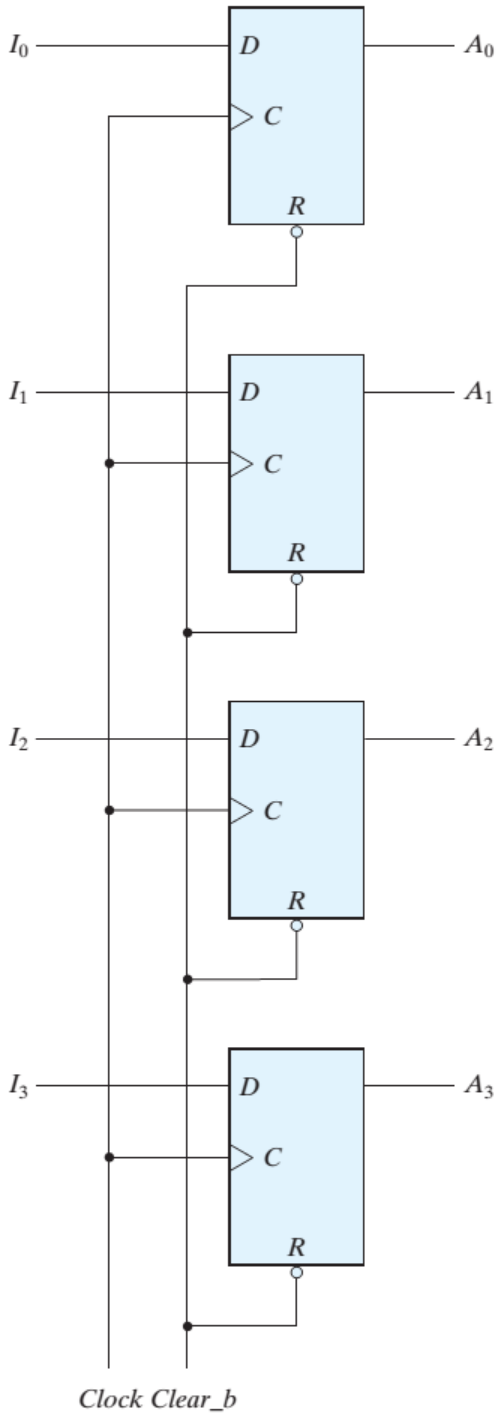
UNIT V - REGISTERS AND COUNTERS

Registers–Shift Registers–Ripple Counters–Synchronous, Counters–Other Counters–HDL for Registers and Counters

Registers

A register is a group of flip-flops, each one of which shares a common clock and is capable of storing one bit of information. An n-bit register consists of a group of n flip-flops capable of storing n bits of binary information. In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks.

A register constructed with four D-type flip-flops to form a four-bit data storage register. The common clock input triggers all flip-flops on the positive edge of each pulse, and the binary data available at the four inputs are transferred into the register. The value of (I3, I2, I1, I0) immediately before the clock edge determines the value of (A3, A2, A1, A0) after the clock edge. The four outputs can be sampled at any time to obtain the binary information stored in the register.



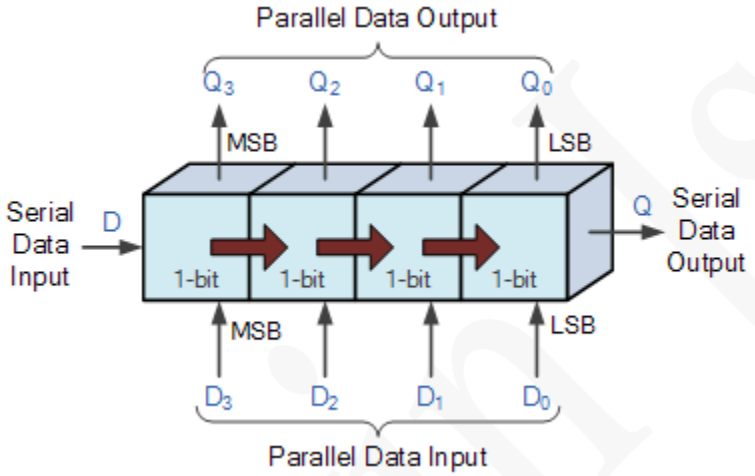
SHIFT REGISTERS

A register capable of shifting the binary information held in each cell to its neighboring cell, in a selected direction, is called a shift register. The logical configuration of a shift register consists of a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop. All flip-flops receive common clock pulses, which activate the shift of data from one stage to the next.

- Serial-in to Parallel-out (SIPO) - the register is loaded with serial data, one bit at a time, with the stored data being available at the output in parallel form.

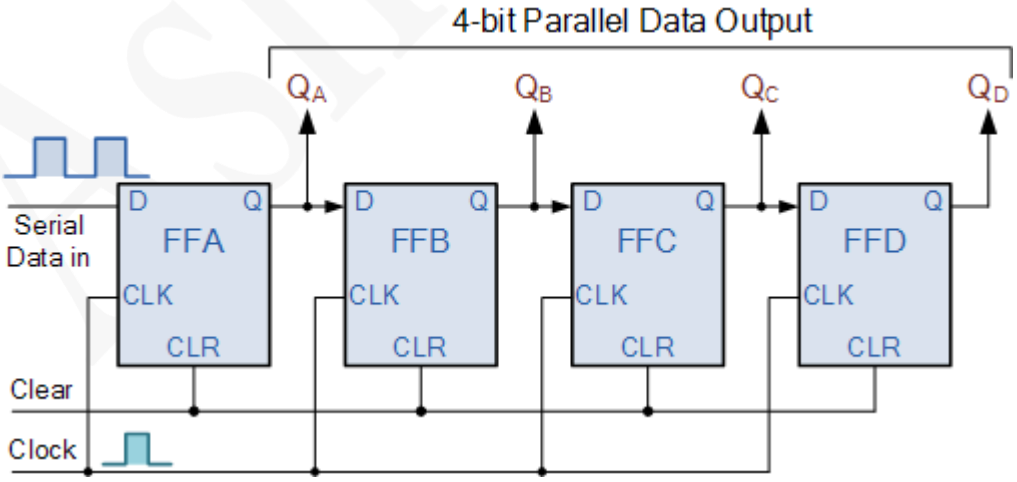
- Serial-in to Serial-out (SISO) - the data is shifted serially "IN" and "OUT" of the register, one bit at a time in either a left or right direction under clock control.
- Parallel-in to Serial-out (PISO) - the parallel data is loaded into the register simultaneously and is shifted out of the register serially one bit at a time under clock control.
- Parallel-in to Parallel-out (PIPO) - the parallel data is loaded simultaneously into the register, and transferred together to their respective outputs by the same clock pulse.

The effect of data movement from left to right through a shift register can be presented graphically as:



Also, the directional movement of the data through a shift register can be either to the left, (left shifting) to the right, (right shifting) left-in but right-out, (rotation) or both left and right shifting within the same register thereby making it *bidirectional*. In this tutorial it is assumed that all the data shifts to the right, (right shifting).

Serial-in to Parallel-out (SIPO) Shift Register



If a logic "1" is connected to the DATA input pin of FFA then on the first clock pulse the output of FFA and therefore the resulting Q_A will be set HIGH to logic "1" with all the other outputs still remaining LOW at logic "0". Assume now that the DATA input pin of FFA has returned LOW again to logic "0" giving us one data pulse or 0-1-0.

The second clock pulse will change the output of FFA to logic "0" and the output of FFBand Q_B HIGH to logic "1" as its input D has the logic "1" level on it from Q_A. The logic "1" has now moved or been "shifted" one place along the register to the right as it is now at Q_A.

When the third clock pulse arrives this logic "1" value moves to the output of FFC (Q_C) and so on until the arrival of the fifth clock pulse which sets all the outputs Q_A to Q_D back again to logic level "0" because the input to FFA has remained constant at logic level "0".

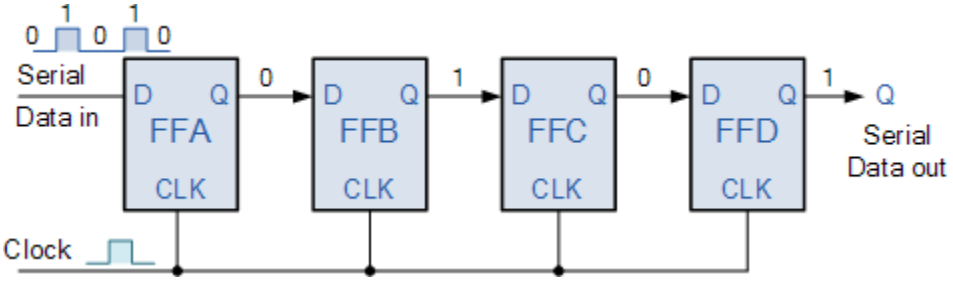
The effect of each clock pulse is to shift the data contents of each stage one place to the right, and this is shown in the following table until the complete data value of 0-0-0-1 is stored in the register. This data value can now be read directly from the outputs of Q_A to Q_D.

Serial-in to Serial-out (SISO) Shift Register

This **shift register** is very similar to the SIPO above, except were before the data was read directly in a parallel form from the outputs Q_A to Q_D, this time the data is allowed to flow straight through the register and out of the other end. Since there is only one output, the DATA leaves the shift register one bit at a time in a serial pattern, hence the name **Serial-in to Serial-Out Shift Register** or **SISO**.

The SISO shift register is one of the simplest of the four configurations as it has only three connections, the serial input (SI) which determines what enters the left hand flip-flop, the serial output (SO) which is taken from the output of the right hand flip-flop and the sequencing clock signal (Clk). The logic circuit diagram below shows a generalized serial-in serial-out shift register.

4-bit Serial-in to Serial-out Shift Register



Parallel-in to Serial-out (PISO) Shift Register

The Parallel-in to Serial-out shift register acts in the opposite way to the serial-in to parallel-out one above. The data is loaded into the register in a parallel format in which all the data bits enter their inputs simultaneously, to the parallel input pins P_A to P_D of the register. The data is then read out sequentially in the normal shift-right mode from the register at Q representing the data present at P_A to P_D.

This data is outputted one bit at a time on each clock cycle in a serial format. It is important to note that with this type of data register a clock pulse is not required to parallel load the register as it is already present, but four clock pulses are required to unload the data.

4-bit Parallel-in to Serial-out Shift Register

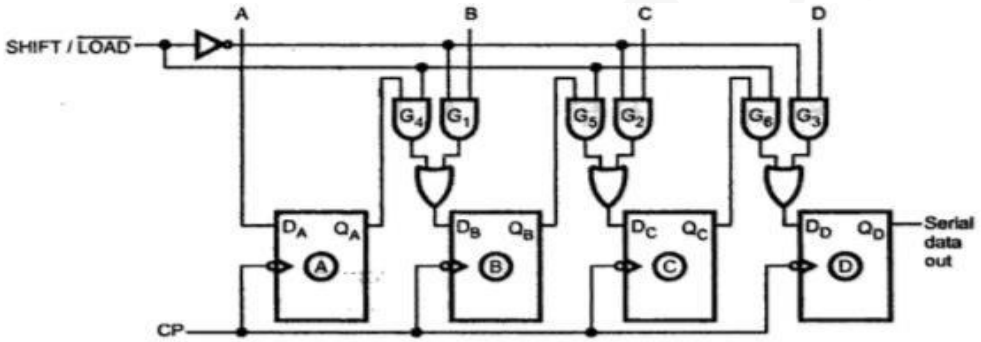
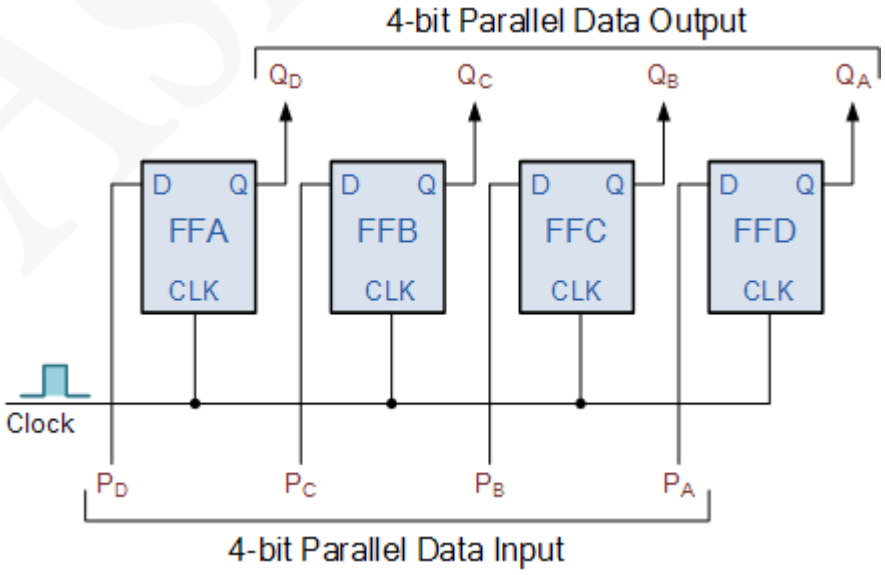


Fig. 8.15 Parallel in serial out shift register

Parallel-in to Parallel-out (PIPO) Shift Register

The final mode of operation is the Parallel-in to Parallel-out Shift Register. This type of shift register also acts as a temporary storage device or as a time delay device similar to the SISO configuration above. The data is presented in a parallel format to the parallel input pins P_A to P_D and then transferred together directly to their respective output pins Q_A to Q_A by the same clock pulse. Then one clock pulse loads and unloads the register. This arrangement for parallel loading and unloading is shown below.

4-bit Parallel-in to Parallel-out Shift Register



The PIPO shift register is the simplest of the four configurations as it has only three connections, the parallel input (PI) which determines what enters the flip-flop, the parallel output (PO) and the sequencing clock signal (Clk).

Similar to the Serial-in to Serial-out shift register, this type of register also acts as a temporary storage device or as a time delay device, with the amount of time delay being varied by the frequency of the clock pulses. Also, in this type of register there are no interconnections between the individual flip-flops since no serial shifting of the data is required.

8.4.5 Bidirectional Shift Register

This type of register allows shifting of data either to the left or to the right side. It can be implemented by using logic gate circuitry that enables the transfer of data from one stage to the next stage to the right or to the left, depending on the level of a control line. Fig. 8.17 illustrates a four-bit bidirectional register. The RIGHT/$\overline{\text{LEFT}}$ is the control input signal which allows data shifting either towards right or towards left. A high on this line enables the shifting of data towards right and a low enables it towards left. When RIGHT/$\overline{\text{LEFT}}$ signal is high, gates G_1, G_2, G_3, G_4 are enabled. The state of the Q output of each flip-flop is passed through the D input of the following flip-flop. When a clock pulse arrives, the data are shifted one place to the right. When the RIGHT/$\overline{\text{LEFT}}$ signal is low, gates G_5, G_6, G_7, G_8 are enabled. The Q output of each flip-flop is passed through the D input of the preceding flip-flop. When clock pulse arrives, the data are shifted one place to the left.

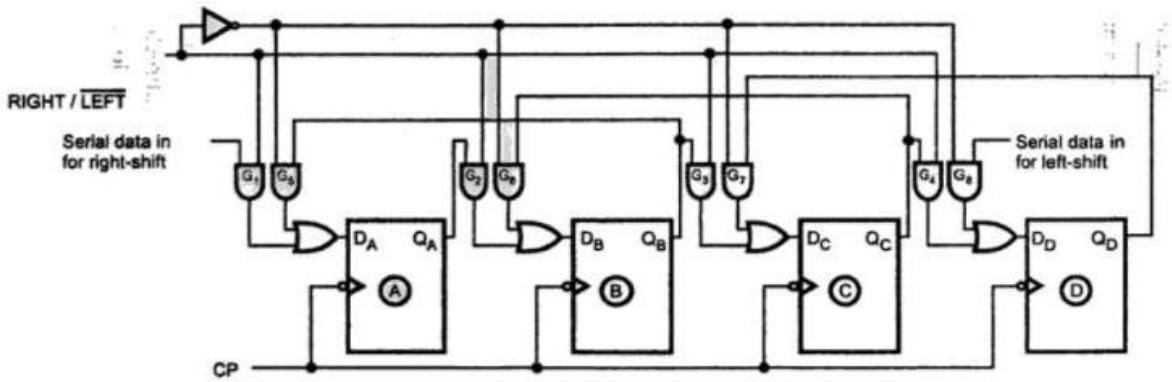


Fig. 8.17 4-bit bidirectional shift register

8.5 Universal Shift Register

A register capable of shifting in one direction only is a undirectional shift register. A register capable of shifting in both directions is a bidirectional shift register. If the register has both shifts (right-shift and left-shift) and parallel load capabilities, it is referred to as **Universal shift register.**

The Fig. 8.19 shows the 4-bit universal shift register. It has all the capabilities listed above. It consists of four flip-flops and four multiplexers. The four multiplexers have two common selection inputs S_1 and S_0, and they select appropriate input for D flip-flop. The Table 8.2 shows the register operation depending on the selection inputs of multiplexers. When S_1S_0 = 00, input 0 is selected and the present value of the register is applied to the D inputs of the flip-flops. This results no change in the register value. When S_1S_0 = 01, input 1 is selected and circuit connections are such that it operates as a right shift register. When S_1S_0 = 10, input 2 is selected and circuit connections are such that it operates as a left-shift register. Finally, when S_1S_0 = 11, the binary information on the parallel input lines is transferred into the register simultaneously and it is a parallel load operation.

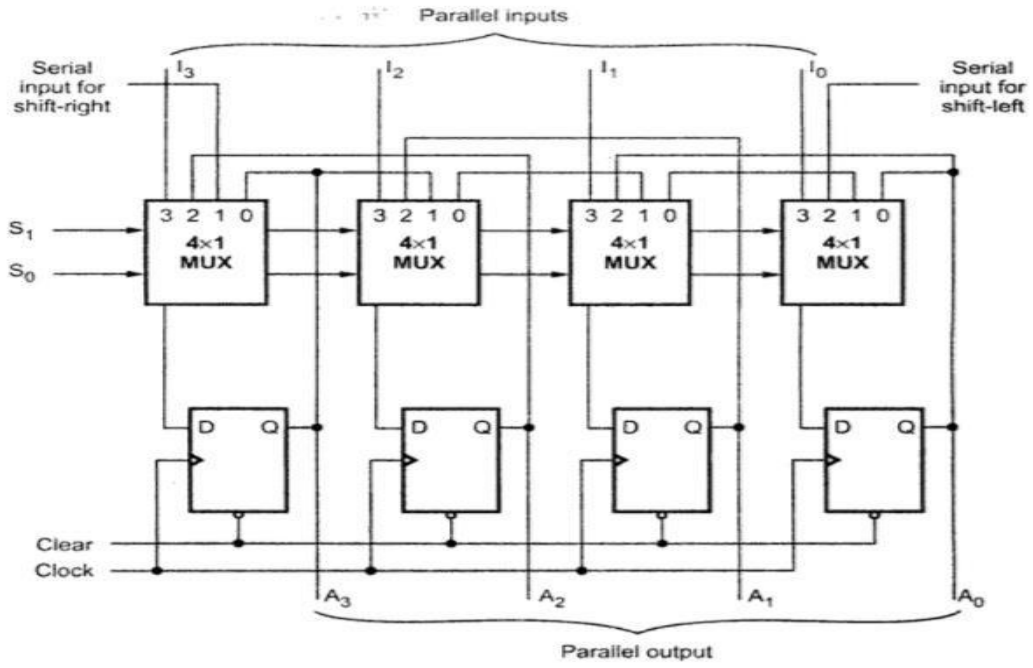


Fig. 8.19 4-bit universal shift register

Mode control		Register operation
S_1	S_0	
0	0	No change
0	1	Shift-right
1	0	Shift-left
1	1	Parallel load

Table 8.2 Mode control and register operation

Counters

7.1 Introduction

A group of flip-flops connected together forms a **register.** A register is used solely for storing and shifting data which is in the form of 1s and/or 0s, entered from an external source. It has no specific sequence of states except in certain very specialized applications. A **counter** is a register capable of counting the number of clock pulses arriving at its clock input. Count represents the number of clock pulses arrived. A specified sequence of states appears as the counter output. This is the main difference between a register and a counter. A specified sequence of states is different for different types of counters.

There are two types of counters, synchronous and asynchronous. In synchronous counter, the common clock input is connected to all of the flip-flops and thus they are clocked simultaneously. In asynchronous counter, commonly called, ripple counters, the first flip-flop is clocked by the external clock pulse and then each successive flip-flop is clocked by the Q or $\overline{Q}$ output of the previous flip-flop. Therefore in an asynchronous counter, the flip-flops are not clocked simultaneously. Let us start with asynchronous counters.

7.2 Asynchronous / Ripple Up Counters

Fig. 7.1 shows 2-bit asynchronous counter using JK flip-flops. As shown in Fig. 7.1, the clock signal is connected to the clock input of only first stage flip-flop. The clock input of the second stage flip-flop is triggered by the Q_A output of the first stage . Because of the inherent propagation delay time through a flip-flop, a transition of the input clock pulse and a transition of the Q_A output of first stage can never occur at exactly the same time. Therefore, the two flip-flops are never simultaneously triggered, which results in asynchronous counter operation.

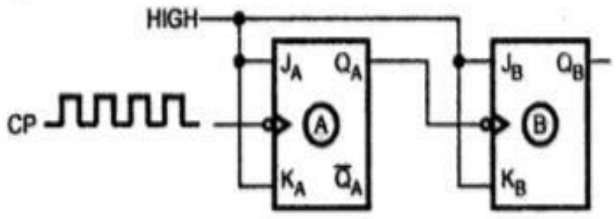


Fig. 7.1 A two-bit asynchronous binary counter

Fig. 7.1 (a) shows the timing diagram for two-bit asynchronous counter. It illustrates the changes in the state of the flip-flop outputs in response to the clock. J and K input of JK flip-flops are tied to logic HIGH hence output will toggle for each negative edge of the clock input.

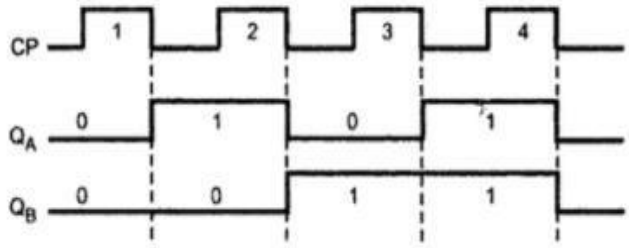


Fig 7.1 (a) Timing diagram for the counter of Fig. 7.1

Example 7.1 : *Extend the counter shown in Fig. 7.1 for 3-stages, and draw output waveforms.*

Solution :

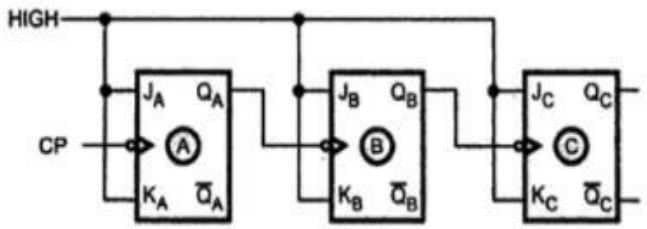


Fig. 7.2 (a) Logic Diagram

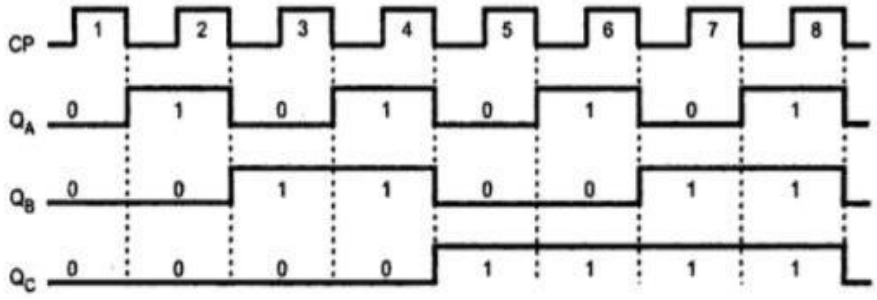


Fig. 7.2 (b) Output waveforms for 3-bit asynchronous counter

In Fig. 7.2 (b), timing diagram for 3-bit asynchronous counter we have not considered the propagation delays of flip-flops, for simplicity. If we consider the propagation delays of flip-flops we get timing diagram as shown in Fig. 7.3.

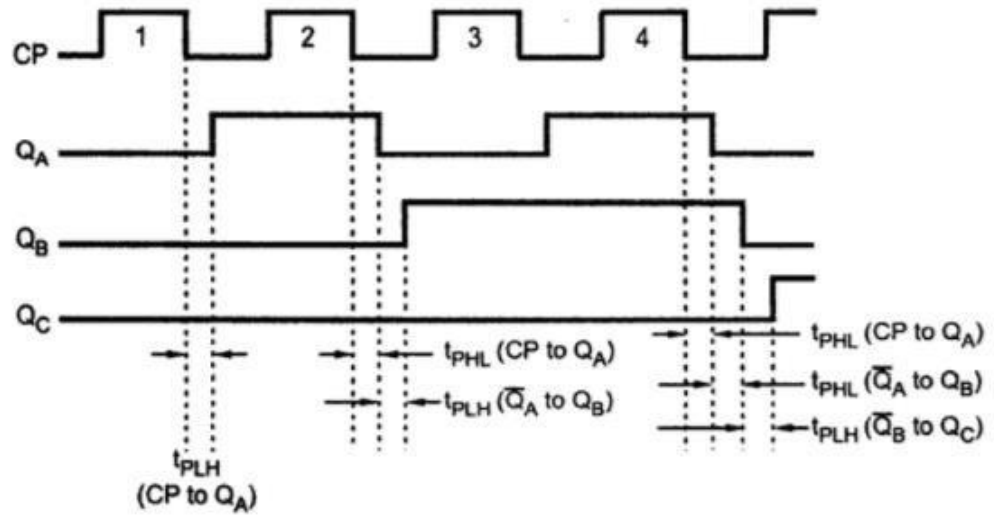


Fig. 7.3 Propagation delays in a ripple clocked binary counter

The timing diagram shows propagation delays. We can see that propagation delay of the first stage is added in the propagation delay of second stage to decide the transition time for third stage. This cumulative delay of an asynchronous counter is a major disadvantage in many applications because it limits the rate at which the counter can be clocked and creates decoding problems.

➠ **Example 7.2 :** *Draw the logic diagram for 3-stage asynchronous counter with negative edge triggered flip-flops.*

Solution : When flip-flops are negatively edge triggered. The Q output of previous stage is connected to the clock input of the next stage. Fig. 7.4 shows 3-stage asynchronous counter with negative edge triggered flip-flops.

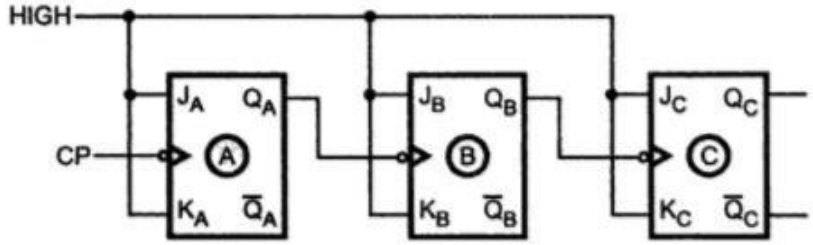


Fig. 7.4 Logic diagram of 3-stage negative edge triggered counter

➠ **Example 7.3 :** *A counter has 14 stable states 0000 through 1101. If the input frequency is 50 kHz what will be its output frequency?*

Solution :

$$\frac{50\text{ kHz}}{14} = 3.57\text{ kHz}$$

Example 7.4 : *The t_{pd} for each flip-flop is 50 ns , determine the maximum operating frequency for MOD-32 ripple counter.*

Solution : We know that MOD-32 uses five flip-flops. With t_{pd} = 50 ns, the f_{max} for ripple counter can be given as,

$$f_{max\ (ripple)} = \frac{1}{5 \times 50\,ns} = 4\ MHz$$

7.3 Asynchronous/Ripple Down Counter

In the last section we have seen that the output of counter is incremented by one for each clock transition. Therefore, we call such counters as up **counters.** In this section we see the asynchronous/ripple down counter. The down counter will count downward from a maximum count to zero.

The Fig. 7.5 shows the 4-bit asynchronous down counter using JK flip-flops. Here, the clock signal is connected to the clock input of only first flip-flop. This connection is same as asynchronous/ripple up counter. However, the clock input of the remaining flip-flops is triggered by the $\overline{Q}_A$ output of the previous stage instead of Q_A output of the previous stage.

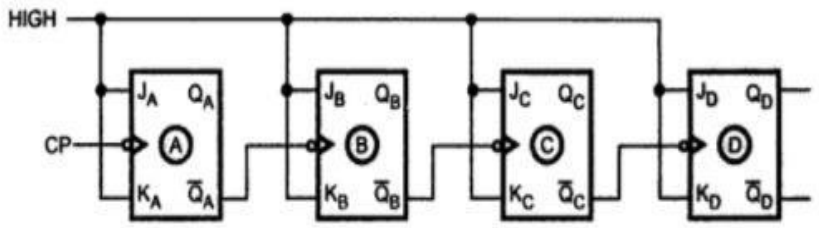


Fig. 7.5 4-bit asynchronous down counter

The Fig. 7.6 shows the timing diagram for 4-bit asynchronous down counter. It illustrates the changes in the state of the flip-flop outputs in response to the clock. Again the J and K inputs of JK flip-flops are tied to logic HIGH hence output will toggle for each negative edge of the clock input.

Down counters are not as widely used as up counters. They are used in situation where it must be known when a desired number of input pulses has occurred. In these situations the down counter is preset to the desired number and then allowed to count down as the pulses are applied. When the counter reaches the zero state it is detected by a logic gate whose output then indicates that the preset number of pulses has occurred.

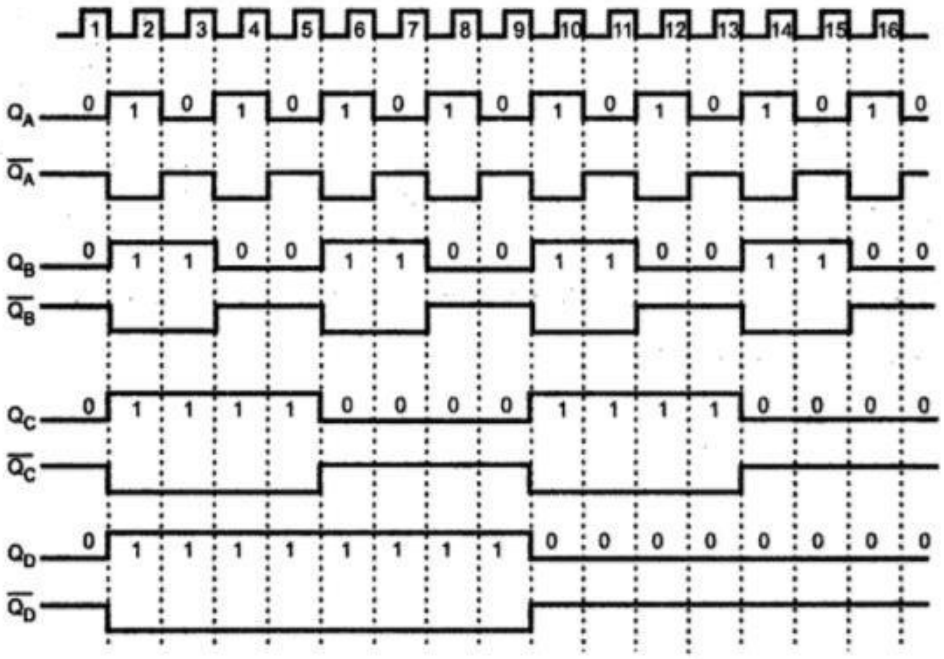


Fig. 7.6 Timing diagram of 4-bit asynchronous down counter

7.4 Synchronous Up Counters

When counter is clocked such that each flip-flop in the counter is triggered at the same time, the counter is called as synchronous counter. Fig. 7.7 shows two stage synchronous counter.

Here, clock signal is connected in parallel to clock inputs of both the flip-flops. But the Q_A output of first stage is used to drive the J and K inputs of the second stage. Let us see the operation of the circuit. Initially, we assume that the $Q_A = Q_B = 0$. When positive edge of the first clock pulse is applied, flip-flop A will toggle because $J_A = K_A = 1$, whereas flip-flop B output will remain zero because $J_B = K_B = 0$. After first clock pulse $Q_A = 1$ and $Q_B = 0$. At negative going edge of the second clock pulse both flip-flops will toggle because they both have a toggle condition on their J and K inputs ($J_A = K_A = J_B = K_B = 1$). Thus after second clock pulse, $Q_A = 0$ and $Q_B = 1$. At negative going edge of the third clock pulse flip-flop A toggles making $Q_A = 1$, but flip-flop B remains set i.e. $Q_B = 1$. Finally, at the leading edge of the fourth clock pulse both flip-flops toggle as their JK inputs are at logic 1. This results $Q_A = Q_B = 0$ and counter recycled back to its original state. The timing details of above operation is shown in Fig. 7.8.

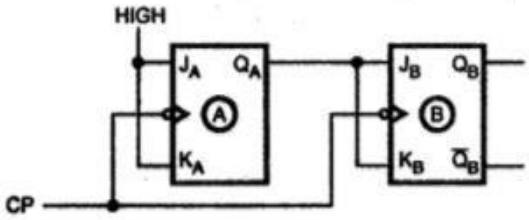
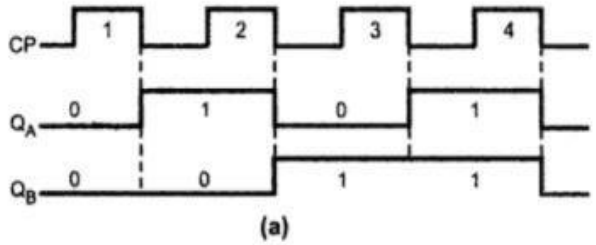


Fig. 7.7 A two-bit synchronous binary counter



CP	Q_B	Q_A
0	0	0
1	0	1
2	1	0
3	1	1

(b)

Fig. 7.8 Timing diagram and state sequence for the 2-bit synchronous counter

A 3-bit Synchronous Binary Counter

Fig. 7.9 (a) shows 3-bit synchronous binary counter and its timing diagram. The state sequence for this counter is shown in Table 7.1.

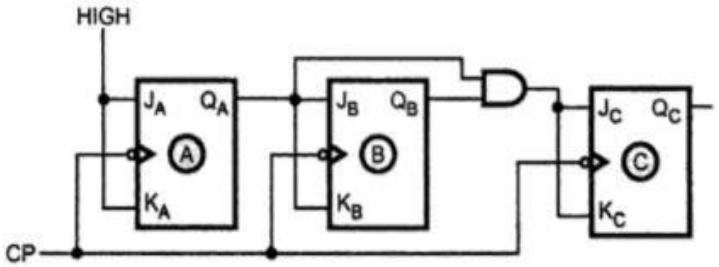


Fig. 7.9 (a) A three-bit synchronous binary counter

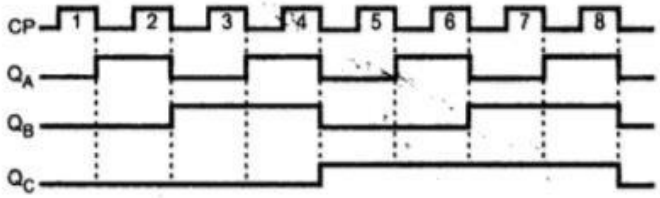


Fig. 7.9 (b) Timing diagram for 3-bit synchronous binary counter

CP	Q_C	Q_B	Q_A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Table 7.1 State sequence for 3-bit binary counter

Looking at Fig. 7.9 (b), we can see that Q_A changes on each clock pulse as we progress from its original state to its final state and then back to its original state. To produce this operation, flip-flop A is held in the toggle mode by connecting J and K inputs to HIGH. Now let us see what flip-flop B does. Flip-flop B toggles, when Q_A is 1. When Q_A is a 0, flip-flop B is in the no-change mode and remains in its present state. Looking at the Table 7.1, we can notice that flip-flop C has to change its state only when Q_B and Q_A both are at logic 1. This condition is detected by AND gate and applied to the J and K inputs of flip-flop C. Whenever both Q_A and Q_B are HIGH, the output of the AND gate makes the J and K inputs of flip-flop C HIGH, and flip-flop C toggles on the following clock pulse. At all other times, the J and K inputs of flip-flop C are held LOW by the AND gate output, and flip-flop does not change state.

A Four-Bit Synchronous Binary Counter

Fig. 7.10 (a) shows logic diagram and timing diagram for 4-bit synchronous binary counter. As counter is implemented with negative edge triggered flip-flops, the transitions occur at the negative edge of the clock pulse. In this circuit, first three flip-flops work same as 3-bit counter discussed previously.

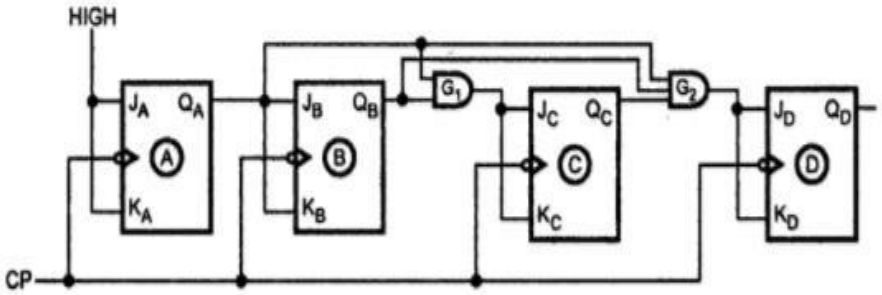


Fig. 7.10 (a)

For the fourth stage, flip-flop has to change the state when $Q_A = Q_B = Q_C = 1$. This condition is decoded by 3-input AND gate G_2. Therefore, when $Q_A = Q_B = Q_C = 1$, flip-flop D toggles and for all other times it is in no change condition.

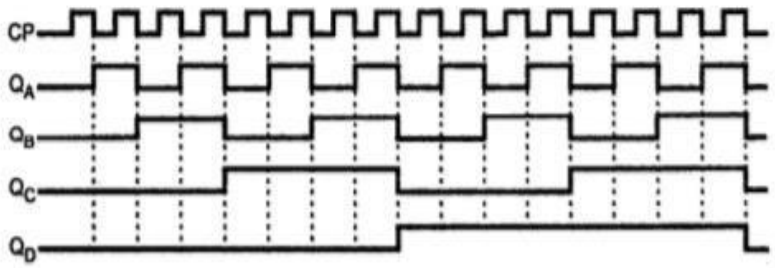


Fig. 7.10 (b) A four-bit synchronous binary counter and timing diagram

Example 7.5 : *Determine f_{max} for the 4-bit synchronous counter if t_{pd} for each flip-flop is 50 ns and t_{pd} for each AND gate is 20 ns. Compare this with f_{max} for a MOD - 16 ripple counter.*

Solution : For a synchronous counter the total delay that must be allowed between input clock pulses is equal to flip-flop t_{pd} + AND gate t_{pd}. Thus $T_{clock} \geq 50 + 20 = 70$ ns, and so the counter has

$$f_{max} = \frac{1}{70\,ns} = 14.3\ MHz$$

We know that MOD-16 ripple counter used four flip-flops. With flip-flop t_{pd} = 50 ns, the f_{max} for ripple counter can be given as

$$f_{max\ (ripple)} = \frac{1}{4 \times 50\,ns} = 5\ MHz$$

7.5 Synchronous Down and Up/Down Counters

We have seen that a ripple counter could be made to count down by using the inverted output of each flip-flop to drive the next flip-flops in the counter. A parallel/synchronous down counter can be constructed in a similar manner-that is, by using the inverted FF outputs to drive the following JK inputs. For example, the parallel up counter of Fig. 7.10 (a) can be converted to a down counter by connecting the $\overline{Q}_A$, $\overline{Q}_B$, $\overline{Q}_C$ and $\overline{Q}_D$ outputs in place of Q_A, Q_B, Q_C and Q_D respectively. The counter will then proceed through the following sequence as input pulses are applied :

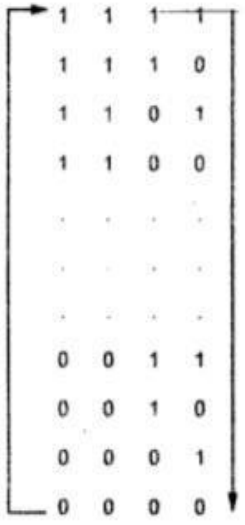


Fig. 7.11

To form a parallel up/down counter the control input (Up/$\overline{Down}$) is used to control whether the normal flip-flop outputs or the inverted flip-flop outputs are fed to the J and K inputs of the following flip-flops. The Fig. 7.11 shows 3-bit up/down counter that will count from 000 up to 111 when the Up/$\overline{Down}$ control input is 1 and from 111 down to 000 when the Up/$\overline{Down}$ control input is 0.

A logic 1 on the Up/$\overline{Down}$ enables AND gates 1 and 2 and disables AND gates 3 and 4. This allows the Q_A and Q_B outputs through to the J and K inputs of the next flip-flops so that the counter will count up as pulses are applied. When Up/$\overline{Down}$ line is logic 0, AND gates 1 and 2 are disables and AND gates 3 and 4 are enabled. This allows the $\overline{Q}_A$ and $\overline{Q}_B$ outputs through to the J and K inputs of the next flip-flops so that the counter will count down as pulses are applied.

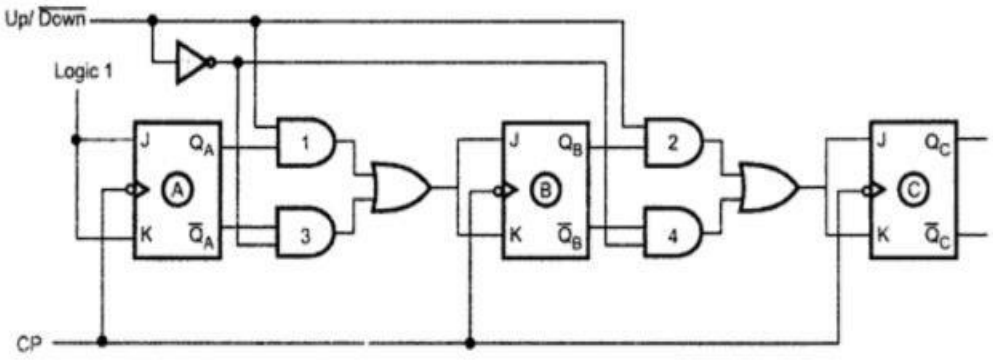


Fig. 7.12 3-bit synchronous/parallel up/down counter

7.6 Synchronous Vs Asynchronous Counters

The Table 7.2 shows the comparison between synchronous and asynchronous counters.

Asynchronous Counters	Synchronous Counters
1) In this type of counter flip-flops are connected in such a way that output of first flip-flop drives the clock for the next flip-flop.	1) In this type there is no connection between output of first flip-flop and clock input of the next flip-flop.
2) All the flip-flops are not clocked simultaneously.	2) All the flip-flops are clocked simultaneously.
3) Logic circuit is very simple even for more number of states.	3) Design involves complex logic circuit as number of states increases.
4) Main drawback of these counters is their low speed as the clock is propagated through number of flip-flops before it reaches last flip-flop.	4) As clock is simultaneously given to all flip-flops there is no problem of propagation delay. Hence they are preferred when number of flip-flops increases in the given design.

Table 7.2 Synchronous Vs Asynchronous counters

7.7 MOD Counters using Reset Input

Fig. 7.14 shows basic 3-bit ripple counter. In its basic form it is a MOD-8 binary counter which count in sequence from 000 to 111. However, the presence of NAND gate will alter this sequence as follows :

1. The NAND gate output is connected to the asynchronous RESET inputs of each flip-flop. As long as the NAND output is HIGH, it will have no effect on the counter. When it goes LOW, it will reset all the flip-flops so that counter immediately goes to the 000 state.
2. The inputs for the NAND gate are the outputs of the A and C flip-flops, and so the NAND output will go LOW whenever $Q_A = Q_C = 1$. This condition will occur when the counter goes from the 100 state to the 101 state (input pulse 5 on waveforms). The LOW at the NAND output will immediately (generally within a few nanoseconds) reset the counter to the 000 state. Once the flip-flops have been reset, the NAND output goes back HIGH, since the $Q_A = Q_C = 1$ condition no longer exists.

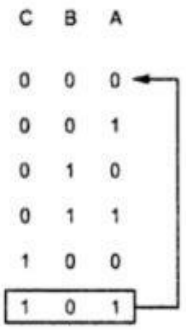


Fig. 7.13

3. The counting sequence is therefore from 000 to 100. Although the counter does go to the 101 state, it remains there for only a few nanoseconds before it recycles to 000. Thus, we can essentially say that the counter counts from 000 to 100 and then recycles to 000. Due to this counter skips 101, 110, and 111 states and it goes through only five different states; thus it is a MOD-5 counter.

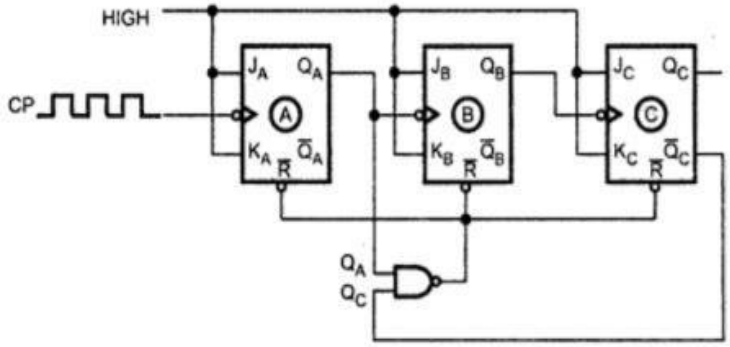


Fig. 7.14 MOD-5 counter using RESET input

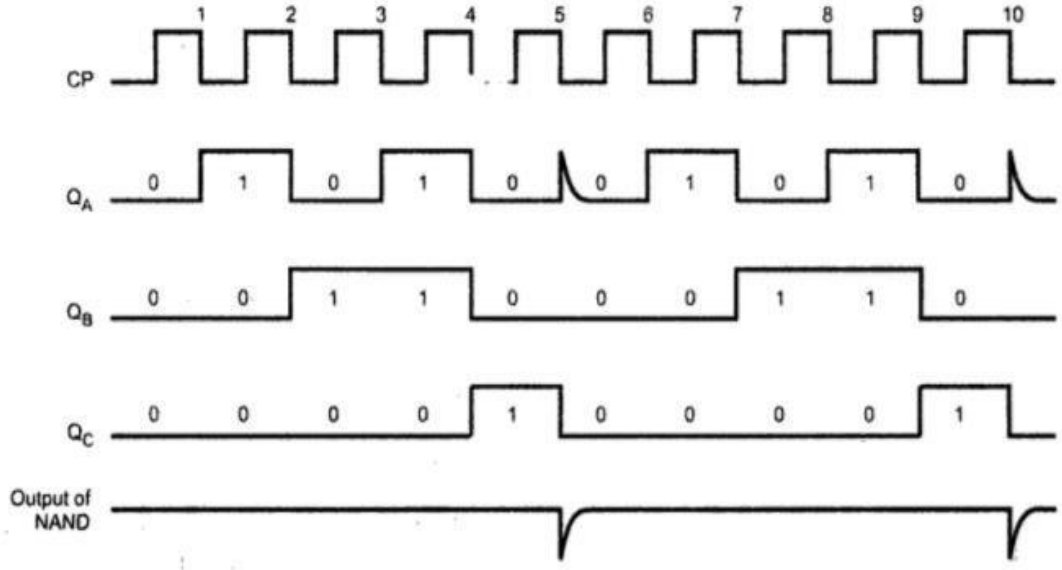


Fig. 7.15 Waveforms for MOD-5 counter