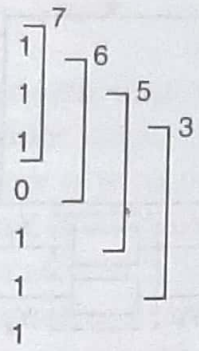
The following pairs of state numbers separated by two states in between are examined to find the difference between their decimal values.

States 7–2 differ by more than 1.
States 6–4 differ by more than 1.
States 5–1 differ by more than 1.
States 2–3 differ by 1. So, this pair fits.

Draw an arrow from the state before 2, that is state 5 to state 3. The required states are, therefore, $7 \rightarrow 6 \rightarrow 5 \rightarrow 3$. The truth table is shown in Figure 6.145a. The K-map and its minimization are shown in Figure 6.145b. The value of f, i.e. the output of the combinational circuit for each state is taken as the next lower bit in the sequence as shown below.



The modified logic diagram based on the value of f is shown in Figure 6.146.

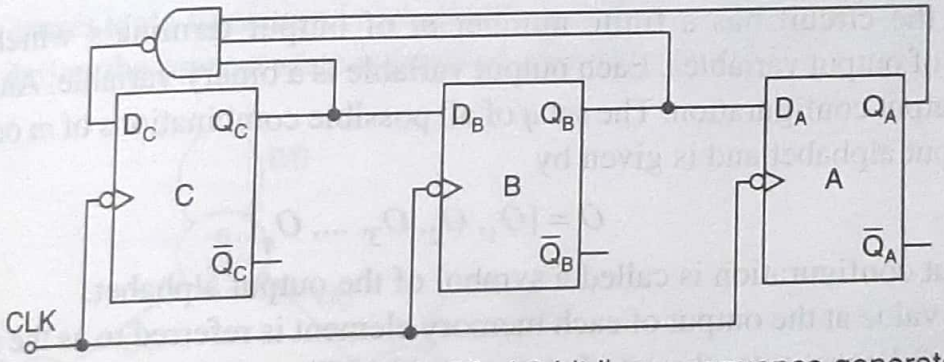


Figure 6.146 Example 6.20: Modified 3-bit linear sequence generator.

Unit-5

6.37 THE FINITE STATE MODEL

A finite state machine is an abstract model describing the synchronous sequential machine. The behaviour of a finite state machine is described as a sequence of events that occur at discrete instants designated as $t = 1, 2, 3 \ldots$ etc. Suppose that the machine has been receiving input signals and also responding by producing output signals. If, now at time t, we were to apply an input signal $x(t)$ to the machine, the response $z(t)$ would depend on $x(t)$ as well as on the past inputs to the machine, and since a given machine might have an infinite varieties of possible histories, it would need an infinite capacity for storing them. Since in practice, it is impossible to implement machines which have infinite storage capabilities, we will concentrate on those machines whose past histories can affect their future behaviour only in a finite number of ways. These are called the *finite state machines*, that is, machines with a fixed number of states. These machines can distinguish among a finite number of classes of input histories. These classes of input histories are referred to

as the internal states of the machine. Every finite state machine, therefore, contains a finite number of memory devices.

Figure 6.147 shows the schematic diagram of a synchronous sequential machine. The circuit has a finite number l of input terminals. The signals entering the circuit via these terminals constitute the set $[x_1, x_2, x_3, ..., x_l]$ of input variables. Each input variable may take one of two possible values, a 0 or a 1. An ordered set of l 0s and 1s is an input configuration. The set p of all possible combinations of l inputs, i.e. $p = 2^l$ is called an input alphabet I, and each one of these configurations is referred to as the symbol of the alphabet

$$I = (I_1, I_2, ..., I_p)$$

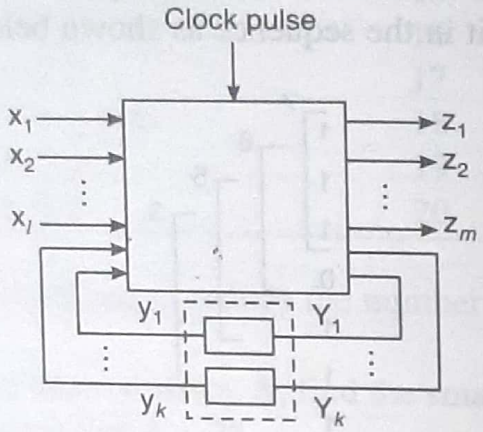


Figure 6.147 Block diagram of a finite state model.

Similarly, the circuit has a finite number m of output terminals which define the set $[z_1, z_2, z_3, ..., z_m]$ of output variables. Each output variable is a binary variable. An ordered set of m 0s and 1s is an output configuration. The set q of all possible combinations of m outputs, i.e. $q = 2^m$ is called the output alphabet and is given by

$$O = [O_1, O_2, O_3, ..., O_q]$$

Each output configuration is called a symbol of the output alphabet.

The signal value at the output of each memory element is referred to as the state variable and the set $[y_1, y_2, ..., y_k]$ constitutes the set of state variables. The combination of values at the outputs of k memory elements $y_1, y_2, ..., y_k$ defines the present internal state or the present state of the machine. The set S of $n = 2^k$ combinations of state variables constitutes the entire set of states of the machine.

$$S = [S_1, S_2, S_3, ..., S_n]$$

The external inputs $x_1, x_2, ..., x_l$ and the values of the state variables $y_1, y_2, ..., y_k$ are supplied to the combinational circuit, which in turn produces outputs $z_1, z_2, ..., z_m$ and the values $Y_1, Y_2, ..., Y_k$. The values of the Ys which appear at the output of the combinational circuit at time t determine the state variables at time $t + 1$, and therefore, the next state of the machine.

Synchronization is achieved by means of clock pulses. The clock pulses may be applied to various AND gates to which input signal is applied. This allows the gates to transmit signals only at instants which coincide with the arrival of clock pulses.

Before going for the design of sequential machines one should be familiar with the following things.

In synchronous or clocked sequential circuits, clocked flip-flops are used as memory elements, which change their individual states in synchronism with the periodic clock signal. Therefore, the change in states of flip-flops and change in state of the entire circuit occurs at the transition of the clock signal.

The synchronous or clocked sequential circuits are represented by two models.

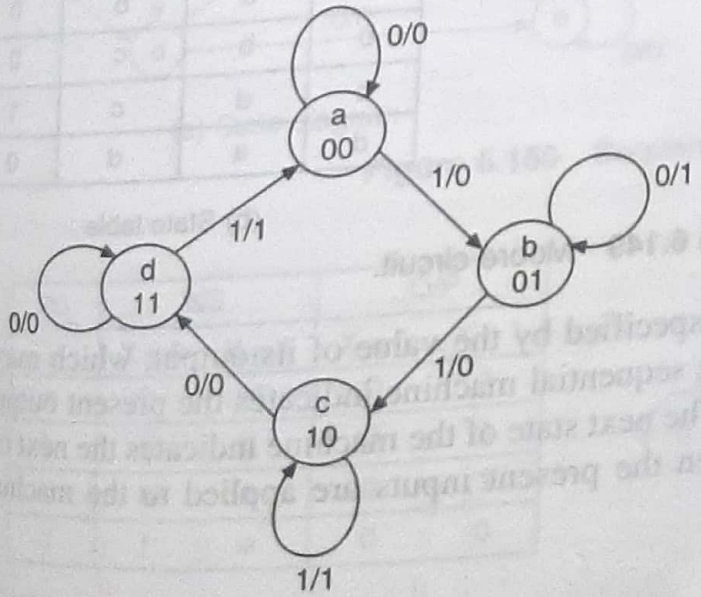
1. *Moore circuit:* In this model, the output depends only on the present state of the flip-flops.
2. *Mealy circuit:* In this model, the output depends on both the present state of the flip-flop(s) and the input(s).

Sequential circuits are also called finite state machines (FSMs). This name is due to the fact that the functional behaviour of these circuits can be represented using a finite number of states.

6.37.1 State Diagram

The state diagram or state graph is a pictorial representation of the relationships between the present state, the input, the next state, and the output of a sequential circuit, i.e. the state diagram is a pictorial representation of the behaviour of a sequential circuit.

Figure 6.148a shows the state diagram of a Mealy circuit. The state is represented by a circle also called the node or vertex and the transition between states is indicated by directed lines connecting the circles. A directed line connecting a circle with itself indicates that the next state is the same as the present state. The binary number inside each circle identifies the state represented by the circle. The directed lines are labelled with two binary numbers separated by a symbol /. The input value that causes state transition is labelled first and the output value that occurs when this input is applied during the present state is labelled after the symbol /.



(a) State diagram

PS	NS, O/P — Input X	
	X = 0	X = 1
a	a, 0	b, 0
b	b, 1	c, 0
c	d, 0	c, 1
d	d, 0	a, 1

(b) State table

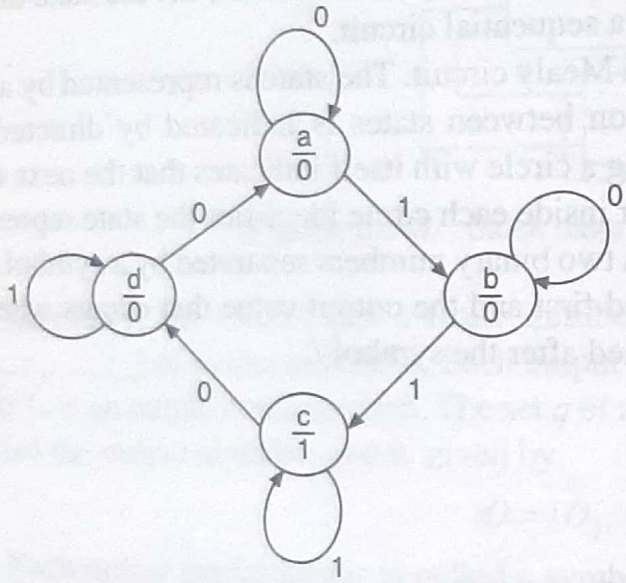
Figure 6.148 Mealy circuit.

In the case of a Moore circuit, the directed lines are labelled with only one binary number representing the input that causes the state transition. The output is indicated within the circle below the present state, because the output depends only on the present state and not on the input. Figure 6.149a shows the state diagram of a Moore circuit.

6.37.2 State Table

Even though the behaviour of a sequential circuit can be conveniently described using a state diagram, for its implementation the information contained in the state diagram is to be translated into a state table. The state table is a tabular representation of the state diagram. Figure 6.148b shows the state table for the state diagram shown in Figure 6.148a. Figure 6.149b shows the state table for the state diagram shown in Figure 6.149a. The present state designates the state of the flip-flops before the application of the clock pulse. The next state is the state of the flip-flops after the application of the clock pulse and the output section gives the value of output variables during the present state.

Both the state diagram and the state table contain the same information and the choice between the two representations is a matter of convenience. Both have the advantage of being precise, unambiguous, and thus, more suitable for describing the operation of a sequential machine than that by any verbal description. The succession of states through which a sequential machine passes and the output sequence which it produces in response to a known input sequence, are specified uniquely by the state diagram or by the state table and initial state.



(a) State diagram

PS	NS, Input X = 0	NS, Input X = 1	O/P
a	a	b	0
b	b	c	0
c	d	c	1
d	a	d	0

(b) State table

Figure 6.149 Moore circuit.

The state of a memory element is specified by the value of its output, which may assume either a 0 or a 1. The present state of the sequential machine indicates the present outputs of the memory elements used in the machine. The next state of the machine indicates the next outputs of the flip-flops that will be obtained when the present inputs are applied to the machine in the present state.

6.37.3 State Reduction

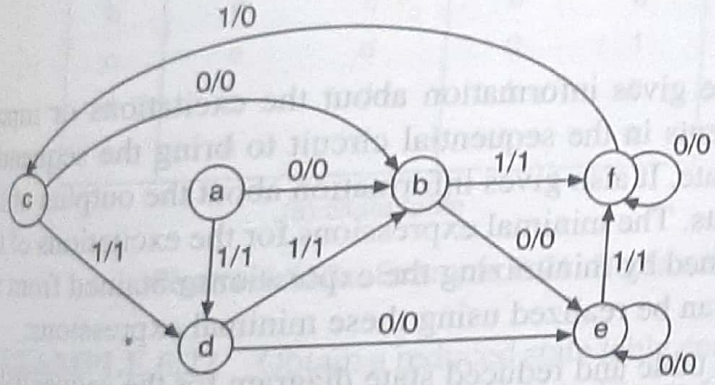
The state reduction technique basically avoids the introduction of redundant states. The reduction in redundant states reduces the number of flip-flops and logic gates, reducing the cost of the final circuit. Two states are said to be equivalent if every possible set of inputs generate exactly the same output and the same next state. When two states are equivalent, one of them can be removed without altering the input-output relationship. Let us illustrate the reduction technique with an example. Consider the sequential circuit whose state diagram is shown in Figure 6.150a. As shown in the figure, the states are represented by letter symbols instead of their binary values because in

state reduction technique, the binary designations of states are not important, but input-output sequences are important. State reduction is done in two steps given as follows.

Step 1. Determine the state table for the given state diagram. Figure 6.150b shows the state table for the given state diagram.

Step 2. Find equivalent states.

As mentioned earlier, in equivalent states, every possible set of inputs generate exactly the same output and the same next state. In the given circuit there are two input combinations $X = 0$, and $X = 1$. Looking at the state table for two present states that go to the same next state and have the same output for both input combinations, we can easily find that states a and c are equivalent. Also states b and e are equivalent. This is because, both states a and c go to states b and d and have outputs 0 and 1 for $X = 0$ and $X = 1$ respectively. Also states b and e both go to states e and f and have outputs 0 and 1 for $X = 0$ and $X = 1$ respectively. Therefore, state c can be removed and replaced by a. Also state e can be removed and replaced by state b. The final reduced state table is shown in Figure 6.151a. The state diagram for the reduced state table consists of only four states and is shown in Figure 6.151b.



(a) State diagram

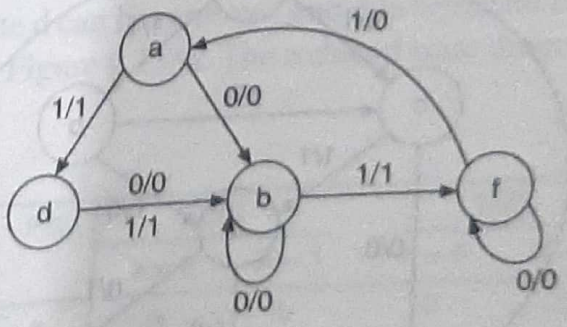
PS	NS		O/P	
	X = 0	X = 1	X = 0	X = 1
a	b	d	0	1
b	e	f	0	1
c	b	d	0	1
d	e	b	0	1
e	e	f	0	1
f	f	c	0	0

(b) State table

Figure 6.150 Sequential circuit.

PS	NS		O/P	
	X = 0	X = 1	X = 0	X = 1
a	b	d	0	1
b	b	f	0	1
d	b	b	0	1
f	f	a	0	0

(a) State table



(b) State diagram

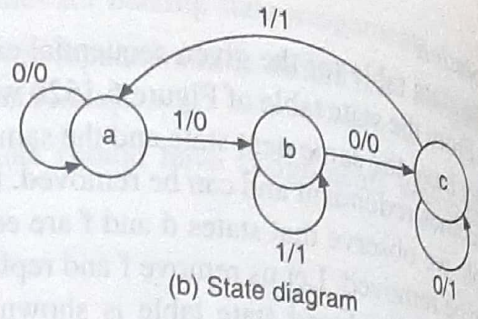
Figure 6.151 Reduced state table and state diagram.

6.37.4 State Assignment

The process of assigning binary values to the states of the sequential machine is known as *state assignment*. The binary values are to be assigned to the states in such a way that it is possible to implement flip-flop input functions using minimum logic gates. The output values of the physical devices are referred to as state variables.

PS	NS		O/P	
	X = 0	X = 1	X = 0	X = 1
a	a	b	0	0
b	c	b	0	1
c	c	a	1	1

(a) State table



(b) State diagram

Figure 6.155 Example 6.22: Reduced state table and state diagram.

6.38 MEMORY ELEMENTS

6.38.1 D Flip-Flop

The excitation table, the state diagram (Mealy model), and the state table of a D flip-flop are shown in Figures 6.156a, b, and c respectively.

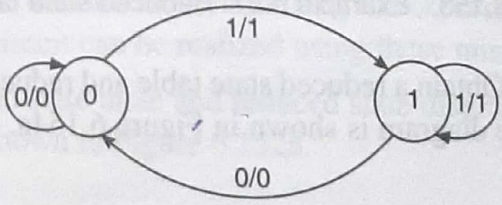
The general state diagram, the K-map for the next state, and the Moore model of the state diagram are shown in Figures 6.156d, e, and f respectively. In the general state diagram, S_1 and S_2 represent the states of the FF and I_1 and I_2 correspond to the input conditions.

The expression for the next state of the flip-flop is

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

Present state (PS)	Input to FF	Next state (NS)
Q(t)	D(t)	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

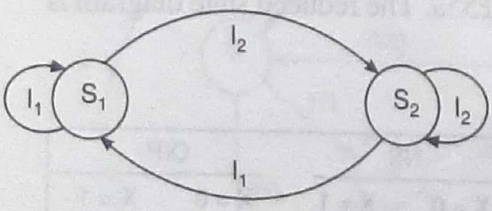
(a) Excitation table



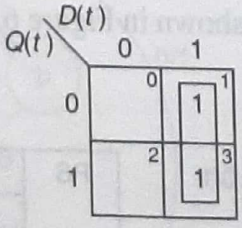
(b) State diagram (Mealy model)

PS	N/S, O/P	
	D = 0	D = 1
0	0, 0	1, 1
1	0, 0	1, 1

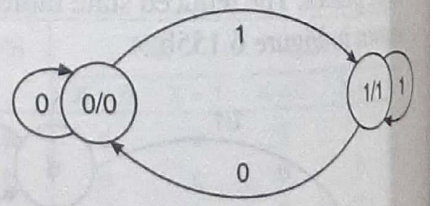
(c) State table



(d) General state diagram



(e) K-map for $Q(t+1)$



(f) Moore model

Figure 6.156 D flip-flop.

6.38.2 T Flip-Flop

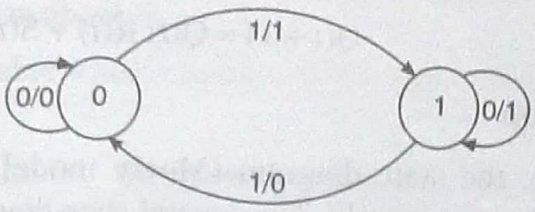
The excitation table, the state diagram(Mealy model) and the state table of the T flip-flop are shown in Figures 6.157a, b and c respectively. The general state diagram, the K-map for the next state, and the Moore model of the state diagram are shown in Figures 6.157d, e, and f respectively.

The expression for the next state of the flip-flop is

$$Q(t+1) = \bar{Q}(t)\,T(t) + Q(t)\,\bar{T}(t) = Q(t) \oplus T(t)$$

Present state (PS) Q(t)	Input to FF T(t)	Next state (NS) Q(t + 1)
0	0	0
0	1	1
0	0	1
1	1	0
1		

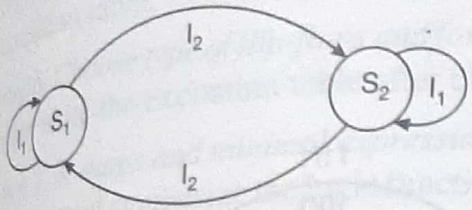
(a) Excitation table



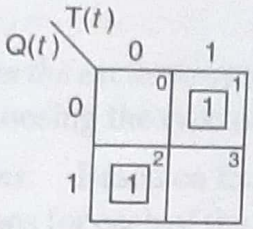
(b) State diagram (Mealy model)

PS	N/S, O/P	
	T = 0	T = 1
0	0, 0	1, 1
1	1, 1	0, 0

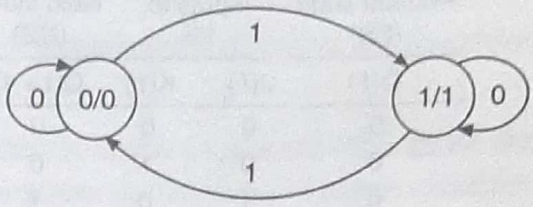
(c) State table



(d) General state diagram



(e) K-map for Q(t + 1)



(f) Moore model

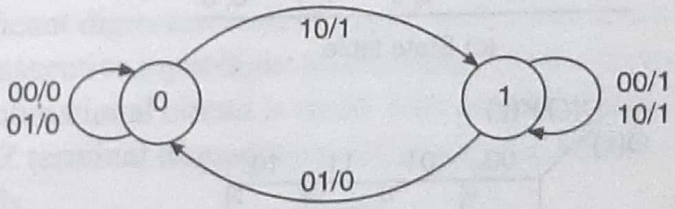
Figure 6.157 T flip-flop.

6.38.3 S-R Flip-Flop

The excitation table, the state diagram(Mealy model), and the state table of the S-R flip-flop are shown in Figures 6.158a, b, and c respectively. The general state diagram, the K-map for the next state, and the Moore model of the state diagram are shown in Figures 6.158d, e and f respectively.

Present state (PS) Q(t)	Inputs to FF S(t)	R(t)	Next state (NS) Q(t + 1)
0	0	0	0
0	0	1	0
0	1	0	1
1	0	0	1
1	0	1	0
1	1	0	1

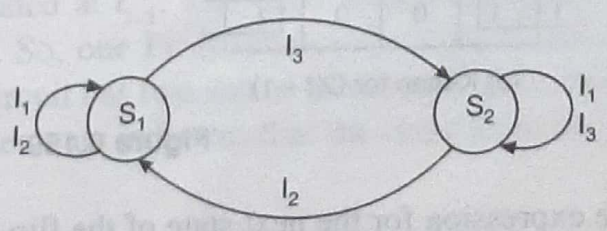
(a) Excitation table



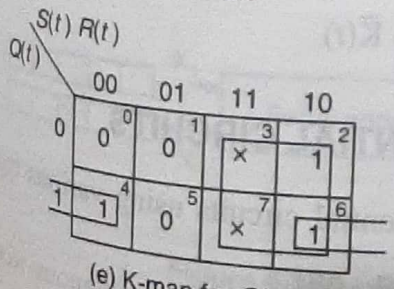
(b) State diagram (Mealy model)

PS	NS, O/P		
	S R = 0 0	S R = 0 1	S R = 1 0
0	0, 0	0, 0	1, 1
1	1, 1	0, 0	1, 1

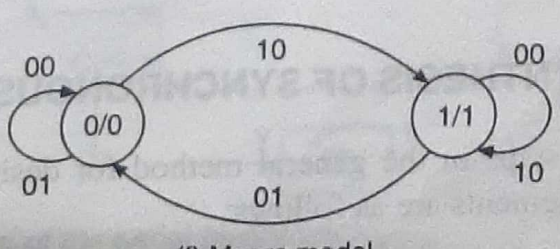
(c) State table



(d) General state diagram



(e) K-map for Q(t + 1)



(f) Moore model

Figure 6.158 S-R flip-flop.

The expression for the next state of the flip-flop is

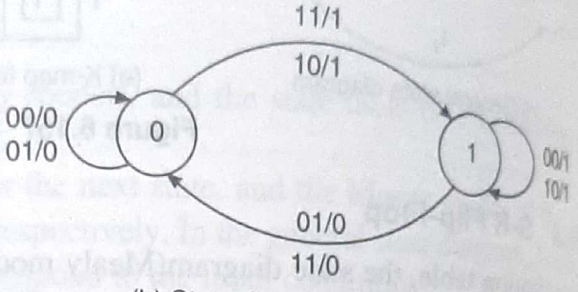
$$Q(t+1) = Q(t)\,\bar{R}(t) + S(t)$$

6.38.4 J-K Flip-Flop

The excitation table, the state diagram (Mealy model), and the state table are shown in Figures 6.159a, b and c respectively. The general state diagram, the K-map for the next state and the Moore model of the state diagram are shown in Figures 6.159d, e and f respectively.

Present state (PS)	Inputs to FF		Next state (NS)
$Q(t)$	$J(t)$	$K(t)$	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

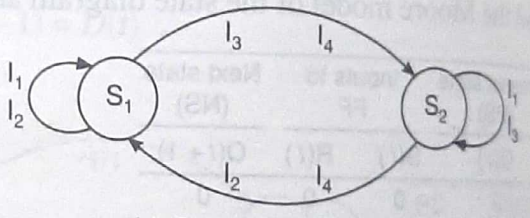
(a) Excitation table



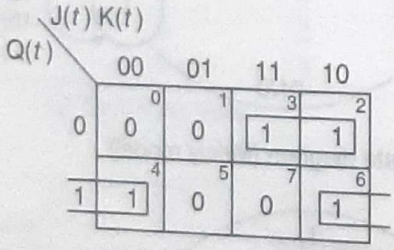
(b) State diagram (Mealy model)

PS	NS, O/P			
	J K	J K	J K	J K
	0 0	0 1	1 0	1 1
0	0, 0	0, 0	1, 1	1, 1
1	1, 1	0, 0	1, 1	0, 0

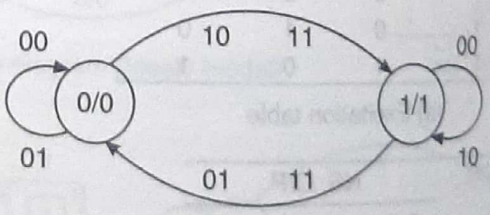
(c) State table



(d) General state diagram



(e) K-map for $Q(t+1)$



(f) Moore model

Figure 6.159 J-K flip-flop.

The expression for the next state of the flip-flop is

$$Q(t+1) = J(t)\,\bar{Q}(t) + Q(t)\,\bar{K}(t)$$

6.39 SYNTHESIS OF SYNCHRONOUS SEQUENTIAL CIRCUITS

The main steps in the general method for designing sequential circuits using various types of memory elements are as follows:

Step 1. *Word statement:* State the purpose of the machine in simple, unambiguous words. It should be realizable with a finite number of memory elements.

Step 2. State diagram: Based on the word description of the machine, draw the state diagram which depicts the complete information about it.

Step 3. State table: Write the state table which contains all the information of the state diagram in tabular form.

Step 4. Reduced standard form state table: Remove the redundant states if any in step 2 and write the reduced standard form state table.

Step 5. State assignment and transition table: Assign binary names to the states and write the transition table.

Step 6. Choose type of flip-flops and form the excitation table: Based on the entries in the transition table write the excitation table after choosing the type of FFs.

Step 7. K-maps and minimal expressions: Based on the contents of the excitation table draw the K-maps and synthesize the logic functions for each of the excitations as functions of input variables and state variables.

Step 8. Realization: Draw the circuit to realize the minimal expressions obtained above.

6.40 SERIAL BINARY ADDER

Step 1. Word statement of the problem: The block diagram of a serial binary adder is shown in Figure 6.160. It is a synchronous circuit with two input terminals designated X_1 and X_2 which carry the two binary numbers to be added and one output terminal Z which represents the sum. The inputs and outputs consist of fixed-length sequences of 0s and 1s. The addition is performed serially, i.e. the least significant digits of the numbers X_1 and X_2 arrive at the corresponding input terminals at t_1; a unit time later the next significant digits arrive at the input terminals, and so on. The time interval between the arrivals of two consecutive input digits is determined by the frequency of the circuit's clock. The delay within the combinational circuit is small with respect to the clock frequency and thus the sum digit arrives at the Z terminal immediately following the arrival of the corresponding input digits at the input terminals.

The output of the serial adder z_i at time t_i is a function of the inputs $x_1(t_i)$ and $x_2(t_i)$ at that time t_i and of a carry which had been generated at t_{i-1}. The carry which represents the past history of the serial adder may be a 0 or a 1. So, one FF is required to store it and one state variable is required to represent it. Thus, the circuit has two states. If one state indicates that the carry from the previous addition is a 0, the other state indicates that the carry from the previous addition is a 1.

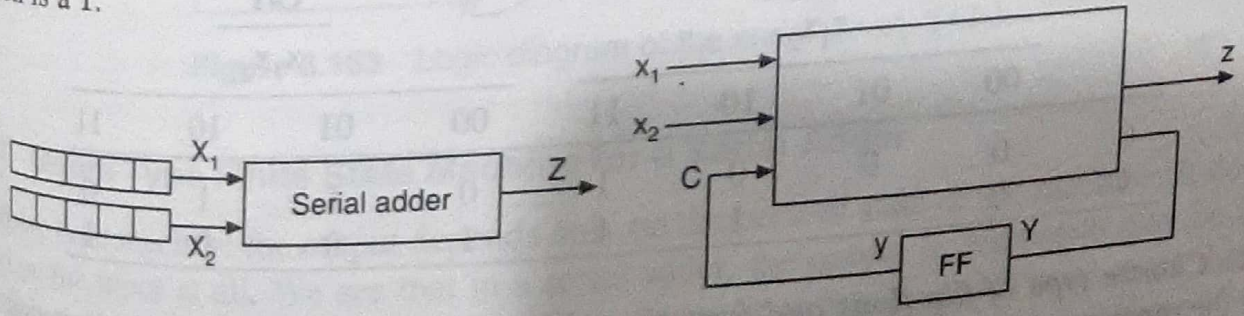
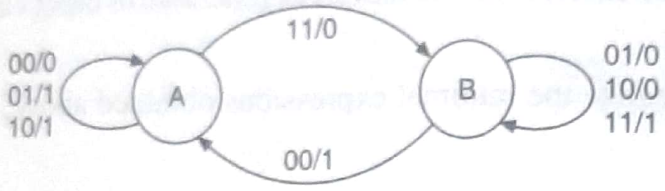


Figure 6.160 Block diagram of the serial binary adder.

Steps 2 and 3. State diagram and state table: Let A designate the state of the serial adder at t_{i-1} if a carry 0 was generated at t_{i-1}, and let B designate the state of the serial adder at t_{i-1} if a carry 1 was generated at t_{i-1}. The state of the adder at the time when the present inputs are applied is referred as the *present state* (PS), and the state to which the adder goes as a result of the new carry value referred to as the *next state* (NS).

The behaviour of a serial adder may be conveniently described by its state diagram and state table as shown in Figure 6.161. The state diagram shows that if the machine is in state A, i.e. carry from the previous addition is a 0, the inputs $x_1 = 0$, $x_2 = 0$ give sum 0 and carry 0. So, the machine remains in state A and outputs a 0. An input $x_1 = 0$ and $x_2 = 1$ give sum 1 and carry 0. So, the machine remains in state A and outputs a 1. Inputs $x_1 = 1$, $x_2 = 0$ give sum 1 and carry 0. So, the machine remains in state A and outputs a 1, but the inputs $x_1 = 1$, $x_2 = 1$ give sum 0 and carry 1. So, the machine goes to state B and outputs a 0.



(a) State diagram

PS	NS, O/P x_1x_2			
	00	01	10	11
A	A, 0	A, 1	A, 1	B, 0
B	A, 1	B, 0	B, 0	B, 1

(b) State table

Figure 6.161 Serial adder.

If the machine is in state B, i.e. carry from the previous addition is a 1, inputs $x_1 = 0$, $x_2 = 1$ give sum 0 and carry 1. So, the machine remains in state B and outputs a 0. Inputs $x_1 = 1$, $x_2 = 0$ give sum 0 and carry 1. So, the machine remains in state B and outputs a 0. Inputs $x_1 = 1$, $x_2 = 1$ give sum 1 and carry 1. So, the machine remains in state B and outputs a 1. Inputs $x_1 = 0$, $x_2 = 0$ give sum 1 and carry 0. So, the machine goes to state A and outputs a 1. The state table also gives the same information.

Step 4. Reduced standard form state table: The machine is already in this form. So no need to do anything.

Step 5. State assignment and transition and output table: The states, A = 0 and B = 1 have already been assigned. So, the transition and output table is as shown in Table 6.15.

Table 6.15 Transition and output table

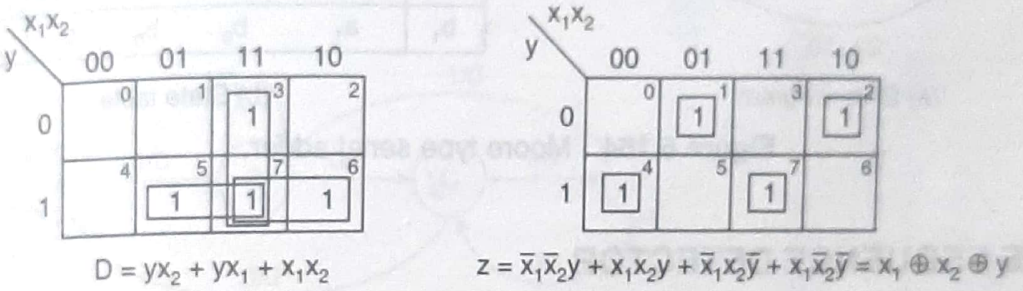
PS	NS x_1x_2				O/P x_1x_2			
	00	01	10	11	00	01	10	11
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Step 6. Choose type of flip-flops and form the excitation table: To write the excitation table, select the memory element. Let us say, we want to use D flip-flop. The excitation table is shown in Table 6.16.

Table 6.16 Excitation table

PS	I/P		NS	I/P to FF	O/P
y	x_1	x_2	Y	D	z
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1

Step 7. K-maps and minimal expressions: Obtain the minimal expressions for D and z in terms of the state variable y and inputs x_1, and x_2 by using K-maps as shown in Figure 6.162.



$D = yx_2 + yx_1 + x_1x_2$

$z = \bar{x}_1\bar{x}_2y + x_1x_2y + \bar{x}_1x_2\bar{y} + x_1\bar{x}_2\bar{y} = x_1 \oplus x_2 \oplus y$

Figure 6.162 K-maps for the serial adder.

Step 8. Implementation: Implement the circuit using those minimal expressions as shown in Figure 6.163.

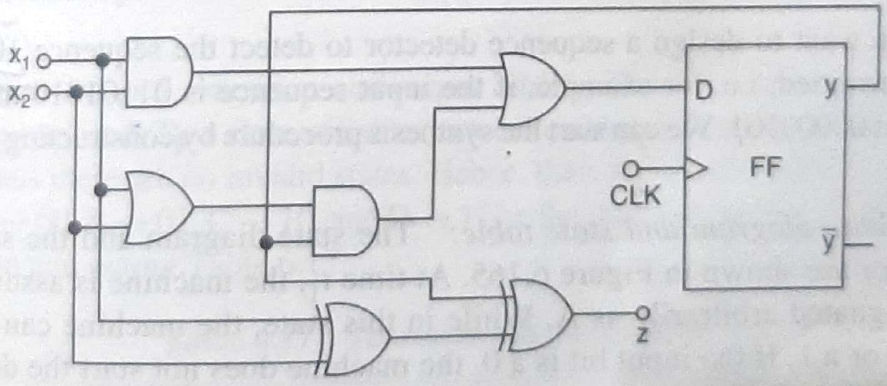
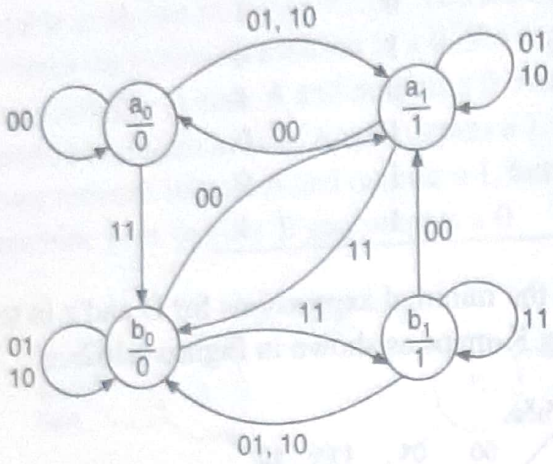


Figure 6.163 Logic diagram of the serial binary adder.

6.40.1 Moore Type Finite State Machine for a Serial Adder

In a Moore type machine the output depends only on the present state of the machine. It does not depend on the input at all. We see that in a serial adder, the output in each state may be 0 or 1 depending on the values of the inputs. So a Moore type state machine will need more than two states. Figure 6.164a shows a Moore type state diagram for a serial adder. We can define additional states by splitting both the states a and b into two states each. Instead of a we will use a_0 and a_1 to

denote the fact that the carry is 0 and the sum is either 0 or 1 respectively. When the state is splitted, we have to direct transition associated with output 0 to state 0 and transition associated with output 1 to state 1. So state a with output 1 is directed to state a_1 and state a without 0 is directed to state a_0. Figure 6.164b shows the state table for the Moore type serial adder. Once, the Moore type state diagram and state table are drawn the Moore circuit can be designed following the normal procedure.



(a) State diagram

PS	NS Input y = 00	01	10	11	O/P (sum)
a_0	a_0	a_1	a_1	b_0	0
a_1	a_0	a_1	a_1	b_0	1
b_0	a_1	b_0	b_0	b_1	0
b_1	a_1	b_0	b_0	b_1	1

(b) State table

Figure 6.164 Moore type serial adder.

6.41 THE SEQUENCE DETECTOR

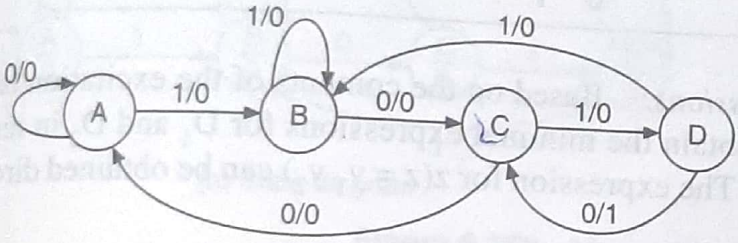
6.41.1 Mealy Type Model

Step 1. Word statement of the problem: A sequence detector is a sequential machine which produces an output 1 every time the desired sequence is detected and an output 0 at all other times.

Suppose we want to design a sequence detector to detect the sequence 1010 and say that overlapping is permitted, i.e. for example, if the input sequence is 01101010 the corresponding output sequence is 00000101. We can start the synthesis procedure by constructing the state diagram of the machine.

Steps 2 and 3. State diagram and state table: The state diagram and the state table of the sequence detector are shown in Figure 6.165. At time t_1, the machine is assumed to be in the initial state designated arbitrarily as A. While in this state, the machine can receive first bit input, either a 0 or a 1. If the input bit is a 0, the machine does not start the detection process because the first bit in the desired sequence is a 1. So, it remains in the same state and outputs a 0. Hence, an arc labelled 0/0 starting and terminating at A is drawn. If the input bit is a 1, the detection process starts. So, the machine goes to state B and outputs a 0. Hence, an arc labelled 1/0 starting at A and terminating at B is drawn. While in state B, the machine may receive a 0 or a 1 bit. If the bit is a 0, the machine goes to the next state, say state C, because the previous two bits are 10 which are a part of the valid sequence, and outputs a 0. So, draw an arc labelled 0/0 from B to C. If the bit is a 1, the two bits become 11 and this is not a part of the valid sequence. Since overlapping is permitted, the second 1 may be used to start the sequence,

so, the machine remains in state B only and outputs a 0. So, an arc labelled 1/0 starting and terminating at B is drawn. While in state C, the machine may receive a 0 or a 1 bit. If it receives a 0, the last three bits received will be 100 and this is not a part of the valid sequence and also none of the last two bits can be used to start the new sequence. So, the machine goes to state A to restart the detection process and outputs a 0. Hence, an arc labelled 0/0 starting at C and terminating at A is drawn. If it receives a 1 bit, the last three bits will be 101 which are a part of the valid sequence. So, the machine goes to the next state, say state D, and outputs a 0. So, an arc labelled 1/0 is drawn from C to D. While in state D, the machine may receive a 0 or a 1. If it receives a 0, the last four bits become 1010 which is a valid sequence and the machine outputs a 1. Since overlapping is permitted, the machine can utilize the last two bits 10 to get another 1010 sequence. So, the machine goes to state C (if overlapping is not permitted or the machine has to restart after outputting a 1, the machine goes to state A). So, an arc labelled 0/1 is drawn from D to C. If it receives a 1 bit, the last four bits received will be 1011 which is not a valid sequence. So, the machine outputs a 0. Since the fourth bit 1 can become the starting bit for the valid sequence, the machine goes to state B. So, an arc labelled 1/0 is drawn from D to B.



(a) State diagram

PS	NS, z	
	x = 0	x = 1
A	A, 0	B, 0
B	C, 0	B, 0
C	A, 0	D, 0
D	C, 1	B, 0

(b) State table

Figure 6.165 Sequence (1010) detector.

Step 4. Reduced standard form state table: The machine is already in this form. So no need to do anything.

Step 5. State assignment and transition and output table: There are four states; therefore, two state variables are required. Two state variables can have a maximum of four states. So, all states are utilized and thus there are no invalid states. Hence, there are no don't cares. Assign the states arbitrarily. Let A → 00, B → 01, C → 10, and D → 11 be the state assignment. With this assignment draw the transition and output Table 6.17.

Table 6.17 Transition and output table

PS (y_1y_2)	NS (Y_1Y_2)		O/P (z)	
	x = 0	x = 1	x = 0	x = 1
A → 0 0	0 0	0 1	0	0
B → 0 1	1 0	0 1	0	0
C → 1 0	0 0	1 1	0	0
D → 1 1	1 0	0 1	1	0

Step 6. Choose type of Flip-flops and form the excitation table: Select the D flip-flops as memory elements and draw the excitation Table 6.18.

Table 6.18 Excitation table

PS		I/P	NS		I/P to FFs		O/P
y_1	y_2	x	Y_1	Y_2	D_1	D_2	z
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	0	0	0	0	0
1	0	1	1	1	1	1	0
1	1	0	1	0	1	0	1
1	1	1	0	1	0	1	0

Step 7. K-maps and minimal expressions: Based on the contents of the excitation table, draw the K-maps and simplify them to obtain the minimal expressions for D_1 and D_2 in terms of y_1, y_2, and x as shown in Figure 6.166. The expression for $z(z = y_1\, y_2)$ can be obtained directly from Table 6.17.

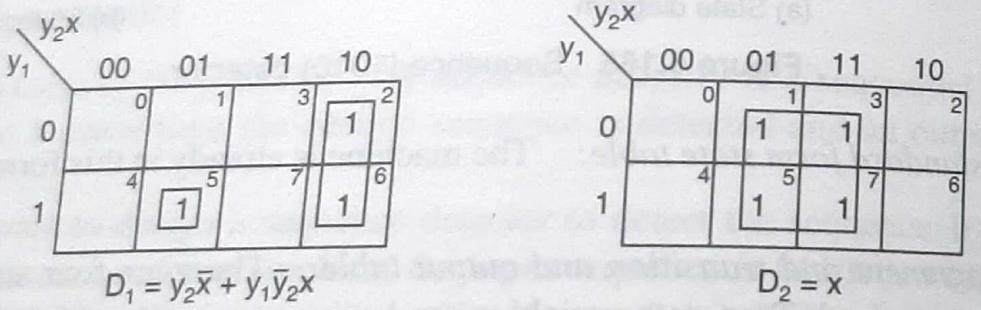


Figure 6.166 K-maps for D_1 and D_2 of the sequence (1010) detector.

Step 8. Implementation: The logic diagram based on these minimal expressions is shown in Figure 6.167.

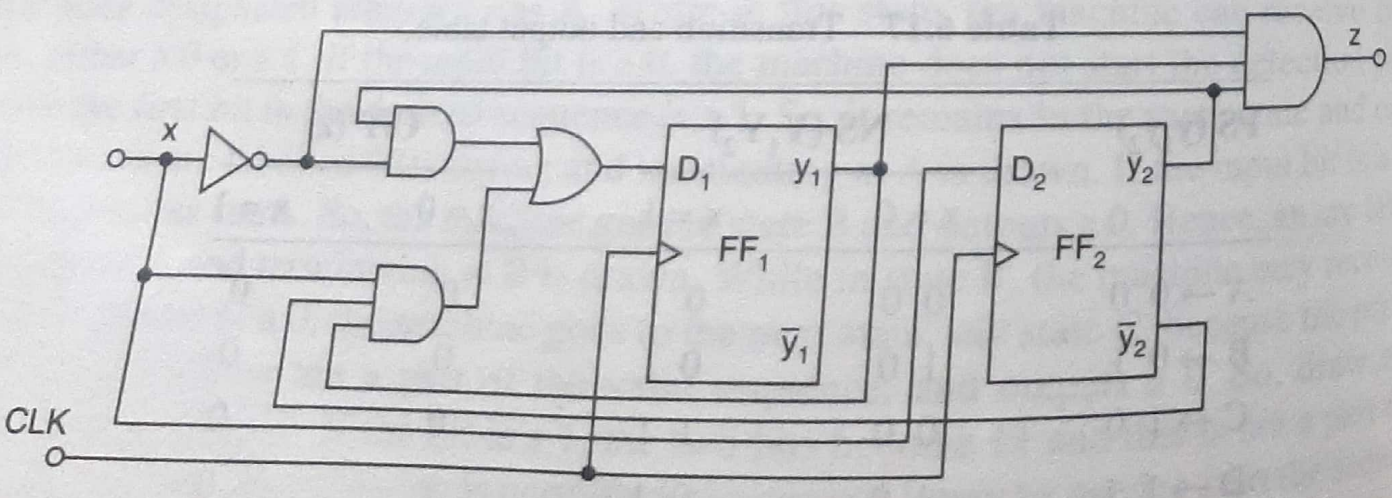
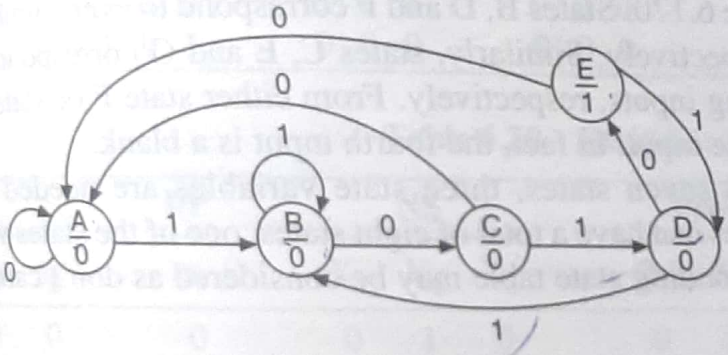


Figure 6.167 Logic diagram of the sequence (1010) detector using D flip-flops.

6.41.2 Moore Type Circuit

For the design of Moore type of circuit, the same steps are to be followed. The state diagram and the state table of a Moore type sequence detector to detect the sequence 1010 are shown in Figure 6.168. In the Moore type state diagram the outputs are written inside the circle below the state name. The state diagram is drawn in the normal way. The machine will be in state D if the last three bits are 101. If the next bit is a 0, the last four bits will be 1010 which is a valid sequence. So the machine outputs a 1, but to utilize overlapping it cannot go to state C because the output at C is 0. Instead it goes to a new state E where output is equal to 1. While at E, if the next bit is a 0 the last five bits become 10100. So the machine goes to state A to restart the detection. If the next bit is a 1, the last five bits become 10101. So to utilize the last three bits, i.e. 101 it goes to state D. Once, the Moore type state diagram and state table are drawn, the Moore circuit can be designed following the normal procedure.



(a) State diagram

PS	NS		O/P
	X = 0	X = 1	
A	A	B	0
B	C	B	0
C	A	D	0
D	E	B	0
E	A	D	1

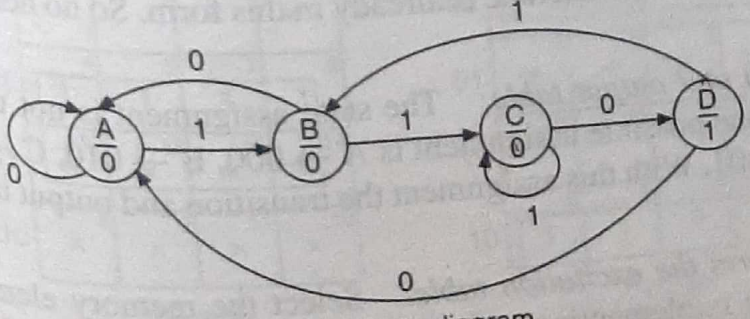
(b) State table

Figure 6.168 Moore circuit.

EXAMPLE 6.23 Draw the state diagram and the state table for a Moore type sequence detector to detect the sequence 110.

Solution

The Moore type state diagram and state table of sequence detector to detect the sequence 110 are shown in Figure 6.169. The state diagram is drawn in the normal way. The machine is in state C when the last two bits received are 11. If the next bit is a 0, the last three bits become 110 which is a valid sequence, hence it outputs a 1, but the machine cannot go to state A to restart the detection process because state A outputs a 0. So the machine goes to a new state D and outputs a 1. While at D if the next bit received is a 0, the last four bits will be 1100. So the machine goes to state A to restart the process. If the bit received is a 1, the last four bits will be 1101. So the machine goes to state B to utilize the last bit.



(a) State diagram

PS	NS		O/P
	X = 0	X = 1	
A	A	B	0
B	A	C	0
C	D	C	0
D	A	B	1

(b) State table

Figure 6.169 Example 6.23.

Step 8. *Implementation:* The logic diagram based on those expressions is shown in Figure 6.17

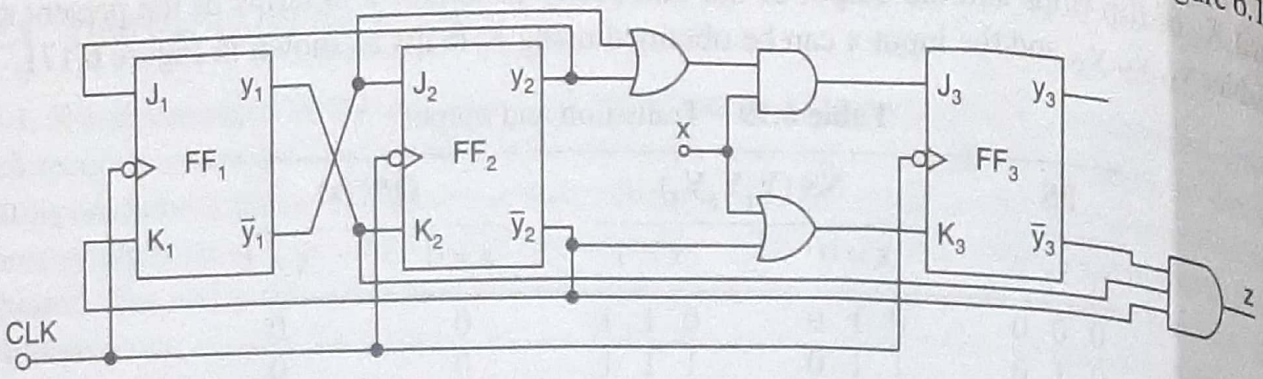


Figure 6.172 Logic diagram of a 3-bit odd-parity generator using J-K flip-flops.

6.43 COUNTERS

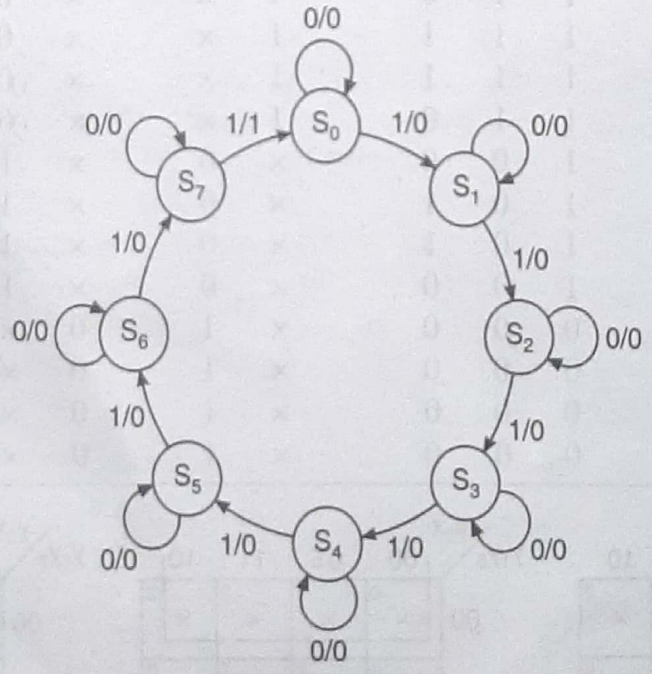
6.43.1 Design of a 3-bit Gray Code Counter

Step 1. *Word statement of the problem:* The counter is to be designed with one input terminal (which receives pulse signals) and one output terminal. It should be capable of counting in the Gray system up to 7 and producing an output pulse for every 8 input pulses. After the count 7 is reached, the next pulse will reset the counter to its initial state, i.e. to a count of zero.

Steps 2 *and* 3. *State diagram and state table:* The state diagram, and the state table of the 3-bit Gray code counter are shown in Figure 6.173.

State assignment	
S_0	0 0 0
S_1	0 0 1
S_2	0 1 1
S_3	0 1 0
S_4	1 1 0
S_5	1 1 1
S_6	1 0 1
S_7	1 0 0

(a) State assignment table



(b) State diagram

PS	NS, O/P	
	x = 0	x = 1
S_0	S_0, 0	S_1, 0
S_1	S_1, 0	S_2, 0
S_2	S_2, 0	S_3, 0
S_3	S_3, 0	S_4, 0
S_4	S_4, 0	S_5, 0
S_5	S_5, 0	S_6, 0
S_6	S_6, 0	S_7, 0
S_7	S_7, 0	S_0, 1

(c) State table

Figure 6.173 A 3-bit Gray code counter.

Step 4. *Reduced standard form state table:* The machine is already in this form. So no need to do anything.

Step 5. *State assignment and transition and output table:* There are eight states for a 3-bit counter. So, three state variables are required which can give a maximum of eight possible states. So, no

valid states exist. The state assignment cannot be arbitrary, since the counter has to change the states in a definite manner. Hence, the state assignment is

$S_0 \to 000, S_1 \to 001, S_2 \to 011, S_3 \to 010, S_4 \to 110, S_5 \to 111, S_6 \to 101, S_7 \to 100$

The transition and output table for the counter is shown in Table 6.21.

Table 6.21 Transition and output table

PS	NS		O/P	
	x = 0	x = 1	x = 0	x = 1
0 0 0	0 0 0	0 0 1	0	0
0 0 1	0 0 1	0 1 1	0	0
0 1 1	0 1 1	0 1 0	0	0
0 1 0	0 1 0	1 1 0	0	0
1 1 0	1 1 0	1 1 1	0	0
1 1 1	1 1 1	1 0 1	0	0
1 0 1	1 0 1	1 0 0	0	0
1 0 0	1 0 0	0 0 0	0	1

Step 6. Choose type of flip-flops and form the excitation table: Select T type flip-flops. The Excitation table is as shown in Table 6.22.

Table 6.22 Excitation table

PS	I/P	NS	Inputs to FFs	O/P
y_3 y_2 y_1	x	Y_3 Y_2 Y_1	T_3 T_2 T_1	z
0 0 0	0	0 0 0	0 0 0	0
0 0 0	1	0 0 1	0 0 1	0
0 0 1	0	0 0 1	0 0 0	0
0 0 1	1	0 1 1	0 1 0	0
0 1 1	0	0 1 1	0 0 0	0
0 1 1	1	0 1 0	0 0 1	0
0 1 0	0	0 1 0	0 0 0	0
0 1 0	1	1 1 0	1 0 0	0
1 1 0	0	1 1 0	0 0 0	0
1 1 0	1	1 1 1	0 0 1	0
1 1 1	0	1 1 1	0 0 0	0
1 1 1	1	1 0 1	0 1 0	0
1 0 1	0	1 0 1	0 0 0	0
1 0 1	1	1 0 0	0 0 1	0
1 0 0	0	1 0 0	0 0 0	0
1 0 0	1	0 0 0	1 0 0	1

Step 7. K-maps and minimal expressions: The minimal expressions for excitation functions to T flip-flops, T_1, T_2 and T_3 in terms of the present state variables y_1, y_2, y_3, and the input x can be obtained using K-maps as shown in Figure 6.174. From the excitation table, the expression for output z is

$$z = y_3 \bar{y}_2 \bar{y}_1 x$$