

6.4 SEQUENTIAL CIRCUITS

6.4.1 Synthesis of Synchronous Sequential Circuits

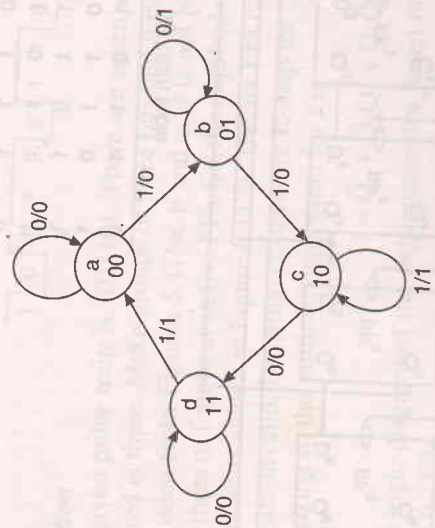
Before going for the design of sequential machines one should be familiar with the following things. In synchronous or clocked sequential circuits, clocked flip-flops are used as memory elements, which change their individual states in synchronism with the periodic clock signal. Therefore, the change 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-flop.
2. Mealy circuit: In this model, the output depends on both the present state of the flip-flop 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. **State diagram:** The state diagram or state graph is a pictorial representation of the relationship between the present state, the input, the next state, and the output of a sequential circuit. A state diagram is a pictorial representation of the behaviour of a sequential circuit.

Figure 6.118a 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 a directed line 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 slash (/). The first input value that causes state transition is labelled first and the output value that occurs when the input is applied during the present state is labelled after the slash (/).



(a) State diagram

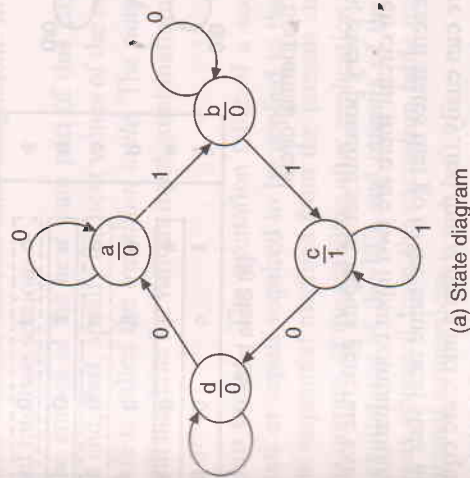
PS	NS, O/P Input X	
	X = 0	X = 1
a	a, 0	b, 0
b	b, 1	c, 0
c	d, 0	c, 1
d	d, 0	a, 1

(b) State table

Figure 6.118 Mealy circuit.

the present state, because the output depends only on the present state and not on the input. 6.119a shows the state diagram of a Moore circuit.

table: Even though the behaviour of a sequential circuit can be conveniently described in a state diagram, for its implementation the information contained in the state diagram is translated into a state table. The state table is a tabular representation of the state diagram. 6.118b shows the state table for the state diagram shown in Figure 6.118a. Figure 6.119b shows the state table for the state diagram shown in Figure 6.119a. The present state designates 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 during the present state.



(a) State diagram

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

(b) State table

Figure 6.119 Moore circuit.

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 any verbal description. The succession of states through which a sequential machine passes in an output sequence which it produces in response to a known input sequence, are specified solely by the state diagram or by the state table and initial state.

The state of a memory element is specified by the value of its output, which may assume 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 next 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 produces exactly the same output and the same next state. When two states are equivalent, one of

technique with an example. Consider the sequential circuit whose state diagram is shown in Figure 6.120a. As shown in the figure, the states are represented by letter symbols instead of 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 6.120b shows the state table for the given state diagram.

Step 2. Find equivalent states.

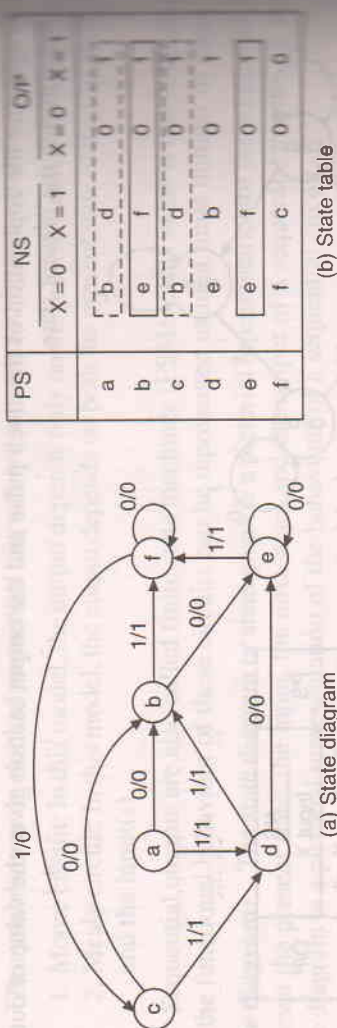


Figure 6.120 Sequential circuit.

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 6.121a. The state diagram for the reduced state table consists of only four states and is shown in Figure 6.121b.

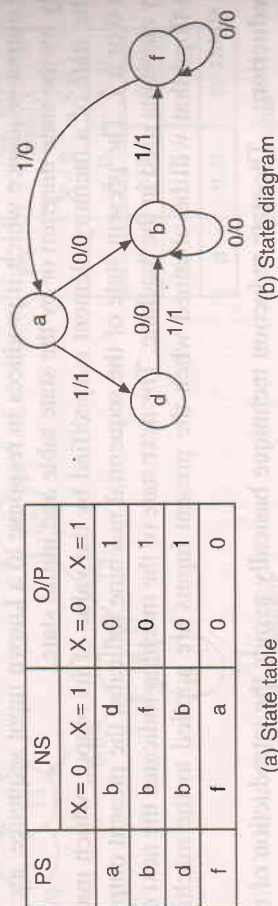


Figure 6.121 Reduced state table and state diagram.

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

to implement flip-flop input functions using minimum logic gates. The output values of the devices are referred to as state variables.

State assignment: There are two basic rules for making state assignments. First, states having the same NEXT STATE for a given input condition should have assignments that are grouped into logically adjacent cells in a K-map.

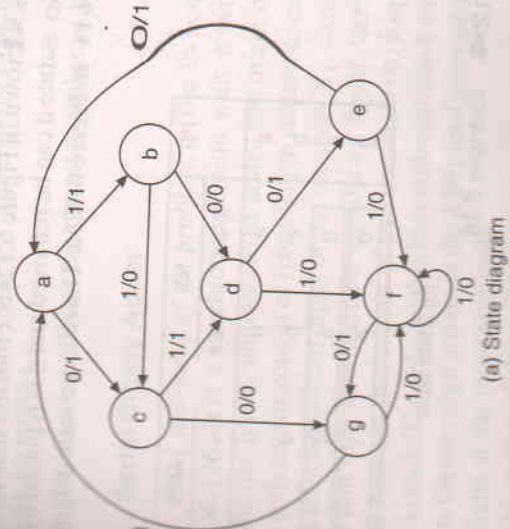
States that are the next states of a single state should have assignments which can be grouped into logically adjacent cells in a K-map.

Next state table: The transition and output table can be obtained from the state table by modifying the entries of the state table to correspond to the states of the machine in the next state with the selected state assignment. In this table, the next state and output entries are grouped into two sections.

The entries of the next state part of this table define the necessary state transitions of the machine and, thus, specify the next values of the outputs of the FFs used. The next state part of the table is called the *transition table*. The output part of the table indicates the output of the machine for various input combinations applied to the machine, which is in the present state.

Excitation table: The excitation table of a sequential machine gives information about the inputs of the machine required to be applied to the memory elements in the sequential circuit to produce the outputs of the machine from the present state to the next state. It also gives information about the excitation of the FFs and outputs of the machine can be obtained by minimizing the minimal expressions from the excitation table using K-maps. The circuit can be realized using

EXAMPLE 6.15 Obtain reduced state table and reduced state diagram for the sequential machine whose state diagram is shown in Figure 6.122a.



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

(h) State table

Solution

The state table for the given sequential circuit will be as shown in Figure 6.122b. From the state table of Figure 6.122b we observe that states g and e are equivalent because they have the same next state and the same output for each one of the inputs. So one of them becomes redundant and can be removed. Let us remove state g. Replacing g by e in the state table we observe that states d and f are equivalent. So one of them becomes redundant and can be removed. Let us remove f and replace f by d. So we are left with five states a, b, c, d and e. The reduced state table is shown in Figure 6.123a. The reduced state diagram is shown in Figure 6.123b.

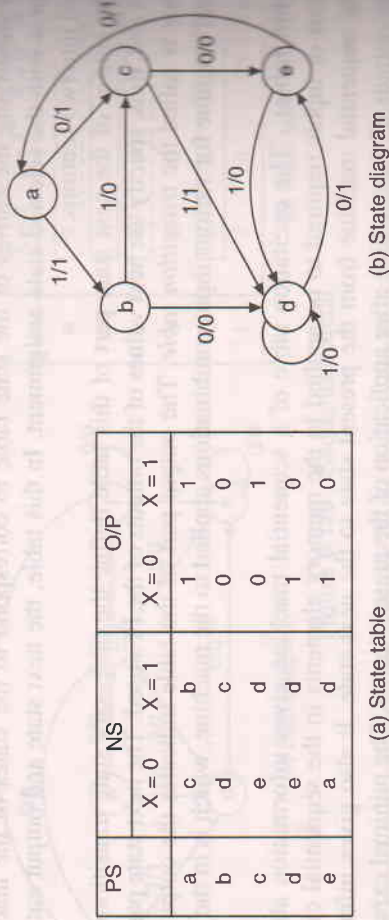


Figure 6.123 Example 6.15: Reduced state table and state diagram.

EXAMPLE 6.16 Obtain a reduced state table and reduced state diagram for the sequential machine whose state diagram is shown in Figure 6.124a.

Solution

The state table for the given state diagram is shown in Figure 6.124b. From the state table we observe that states c and d are equivalent. So state d can be removed and is replaced by c in other places. The reduced state table is shown in Figure 6.125a. The reduced state diagram is shown in Figure 6.125b.

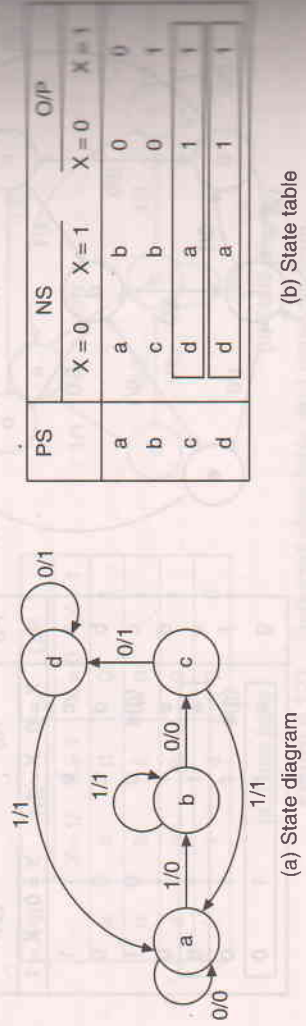
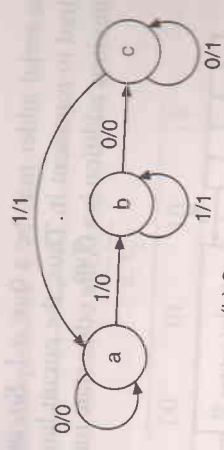


Figure 6.124 Example 6.16.

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 6.125 Example 6.16: Reduced state table and state diagram.

Design steps: 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.

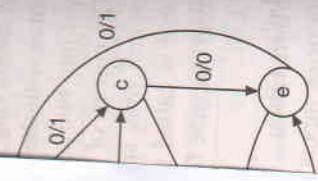
Step 8. Realization: Draw the circuit to realize the minimal expressions obtained above.

6.4.2 Serial Binary Adder

Step 1. Word statement of the problem: The block diagram of a serial binary adder is shown in Figure 6.126. 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

6.122b. equivalent because s. So one of them g by e in the state es redundant and ve states a, b, c, d, diagram is shown



ram gram.

for the sequential

the state table we is replaced by c at ed state diagram is

O/P	
X = 0	X = 1
0	0
0	1
1	1
1	1

ible

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.

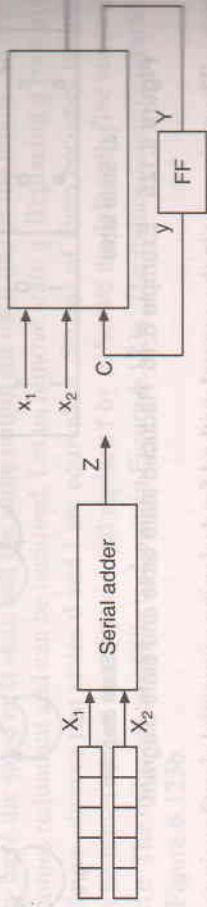


Figure 6.126 Block diagram of the serial binary adder.

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

The behaviour of a serial adder may be conveniently described by its state diagram and the state table as shown in Figure 6.127. 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. Inputs $x_1 = 1, x_2 = 1$ give sum 0 and carry 1. So, the machine goes to state B and outputs a 0.



Figure 6.127 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 6.14.

Table 6.14 Transition and output table

PS	NS				O/P			
	x_1x_2				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 6.15.

Table 6.15 Excitation table

PS	I/P		NS	Y	I/P to FF		O/P
	y	x_1x_2			D	z	
0	0	0 0	0	0	0	0	0
0	0	0 1	0	0	0	1	1
0	0	1 0	0	0	0	1	1
0	0	1 1	1	1	1	0	0
1	1	0 0	0	0	0	1	1
1	1	0 1	1	1	1	0	0
1	1	1 0	1	1	1	0	0
1	1	1 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 6.128.

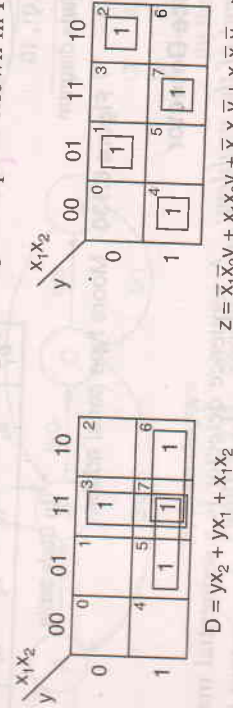


Figure 6.128 K-maps for the serial adder.

Step 8. Implementation: Implement the circuit using those minimal expressions as shown in Figure 6.129.

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 6.129 shows the implementation of a Moore type finite state machine for a serial adder.

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 6.130b 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.

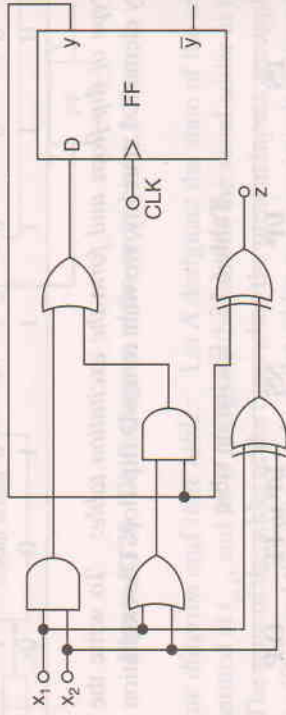
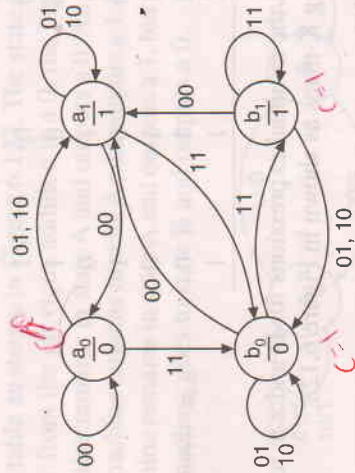


Figure 6.129 Logic diagram of the serial binary adder.



(a) State diagram

PS	NS Input				O/P (sum)
	00	01	10	11	
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 6.130 Moore type serial adder.

6.4.3 The Sequence Detector

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

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 for 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.

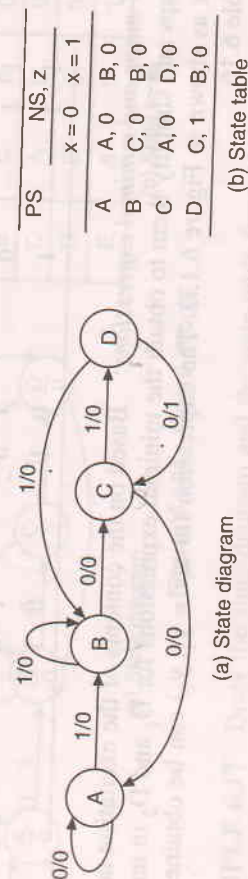


Figure 6.131 Sequence (1010) detector.

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 $A \rightarrow 00$, $B \rightarrow 01$, $C \rightarrow 10$, and $D \rightarrow 11$ be the state assignment. With this assignment draw the transition and output Table 6.16.

Table 6.16 Transition and output table

PS (y_1y_2)	NS (Y_1Y_2)		O/P (z)	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
A $\rightarrow$ 0 0	0 0	0 1	0	0
B $\rightarrow$ 0 1	1 0	0 1	0	0
C $\rightarrow$ 1 0	0 0	1 1	0	0
D $\rightarrow$ 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 6.17.

Table 6.17 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 6.132. The expression for $z(z = y_1, y_2)$ can be obtained directly from Table 6.16.

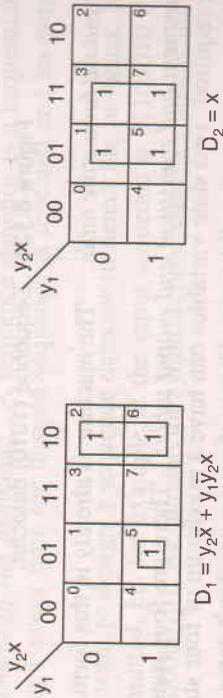


Figure 6.132 K-maps for D_1 and D_2 of the sequence (1010) detector.

Step 8. Implementation: The logic diagram based on these minimal expressions is shown in Figure 6.133.

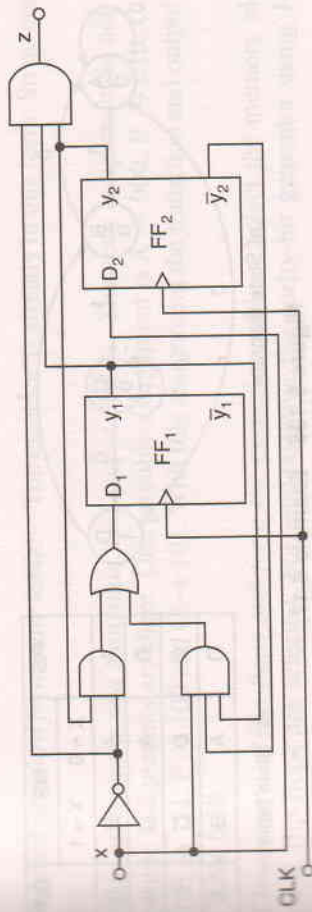


Figure 6.133 Logic diagram of the sequence (1010) detector using D flip-flops.

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 6.134. In the Moore type state diagram the outputs are written inside the state circles below the state name. The state diagram is drawn in the normal way. The machine will be in state A 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. Instead 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 sequence. 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 type circuit can be designed following the normal procedure.

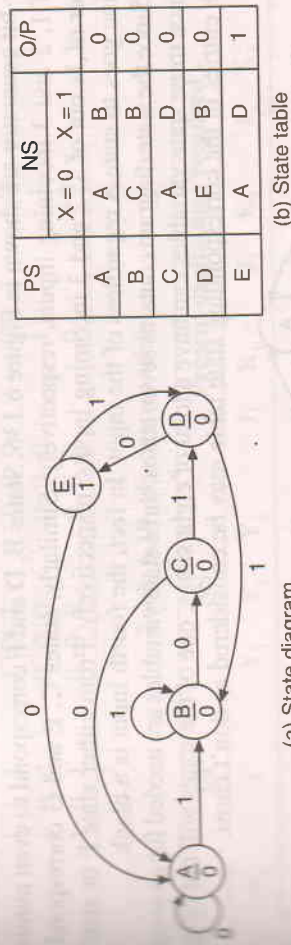


Figure 6.134 Moore circuit.

EXAMPLE 6.17 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 6.135. The state diagram is drawn in the normal way. The machine goes to state A 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 B where the next bit received is a 0, the last four bits will be 1100.

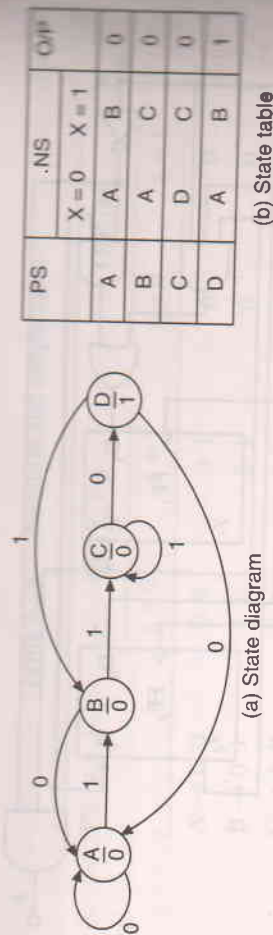


Figure 6.135 Example 6.17.

6.4.4 Parity-bit Generator

Odd parity-bit generator

Step 1. Word statement of the problem: A serial parity-bit generator is a two-terminal circuit which receives coded messages and adds a parity bit to every m bits of the message, so that the resulting outcome is an error-detecting coded message. The inputs are assumed to arrive in strings of three symbols ($m = 3$) and the strings are spaced apart by single time units (i.e. the fourth place is a blank). The parity bits are inserted in the appropriate spaces so that the resulting outcome is a continuous string of symbols without spaces. For even parity, a parity bit 1 is inserted, if and only if the number of 1s in the preceding string of three symbols is odd. For odd parity, a parity bit 1 is inserted, if and only if the number of 1s in the preceding string of three symbols is even.

Steps 2 and 3. State diagram and state table: The state diagram and the state table of an odd parity-bit generator are shown in Figure 6.136. States B, D and F correspond to even number of 1s out of 1, 2 and 3 incoming inputs, respectively. Similarly, states C, E and G correspond to odd number of 1s out of 1, 2 and 3 incoming inputs, respectively. From either state F or state G, the machine goes to state A regardless of the input. In fact, the fourth input is a blank.

Since the state diagram contains seven states, three state variables are needed for an assignment. But since three state variables can have a total of eight states, one of the states will not be assigned and its entries in the corresponding state table may be considered as don't cares.

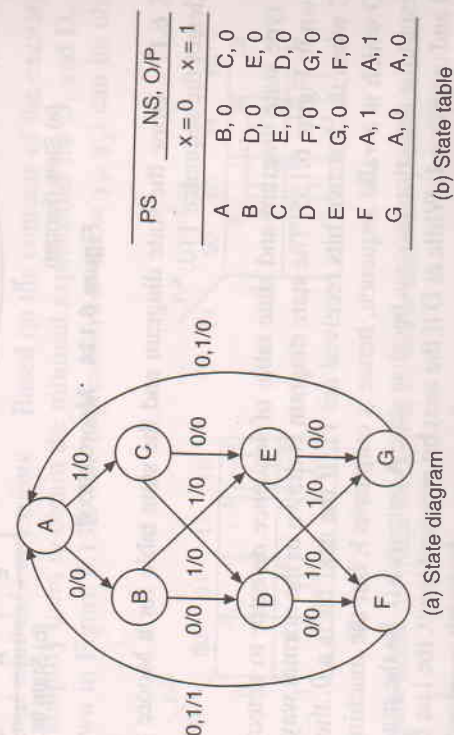


Figure 6.136 3-bit odd-parity generator.

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 state assignment is not unique. Many possible assignments are there. One possible assignment is $A \rightarrow 000$, $B \rightarrow 010$, $C \rightarrow 011$, $D \rightarrow 100$, $E \rightarrow 111$, $F \rightarrow 100$ and $G \rightarrow 101$. With this assignment the transition and output table is given in Table 6.18.

Step 6. *Choose type of flip-flops and form the excitation table:* Select the memory elements. Suppose J-K flip-flops are selected. For implementing the parity-bit generator using J-K flip-flops, draw the excitation table as shown in Table 6.19.

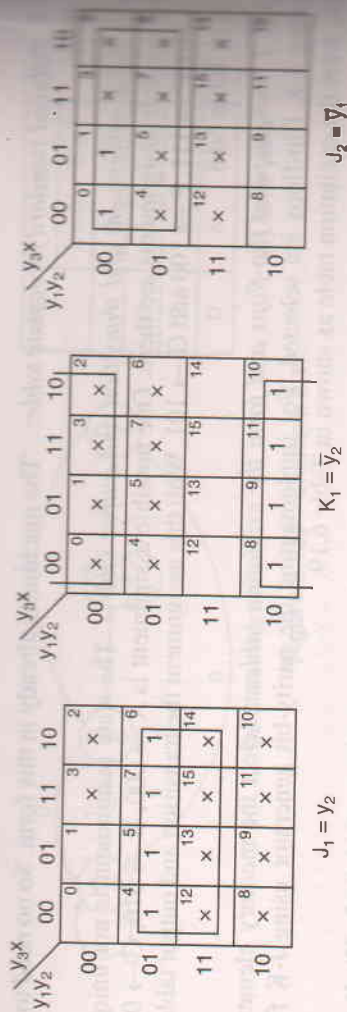
Step 7. *K-maps and minimal expressions:* The minimal expressions for excitations J_1, K_1, J_2, K_2 , and K_3 of flip-flops and the output of the odd-parity generator z in terms of the present state variables Y_1, Y_2, Y_3 , and the input x can be obtained using K-maps as shown in Figure 6.137.

Table 6.18 Transition and output

PS $Y_1 Y_2 Y_3$	NS ($Y_1 Y_2 Y_3$)			O/P (z)	
	$x = 0$	$x = 1$	$x = 1$	$x = 0$	$x = 1$
0 0 0	0 1 0	0 1 1	0	0	0
0 1 0	1 1 0	1 1 1	0	0	0
0 1 1	1 1 1	1 1 0	0	0	0
1 1 0	1 0 0	1 0 1	0	0	0
1 1 1	1 0 1	1 0 0	0	0	0
1 0 0	0 0 0	0 0 0	1	1	1
1 0 1	0 0 0	0 0 0	0	0	0

Table 6.19 Excitation table

PS $Y_1 Y_2 Y_3$	I/P		NS			Present excitations required					O/P	
	x		Y_1	Y_2	Y_3	J_1	K_1	J_2	K_2	J_3	K_3	z
0 0 0	0		0	1	0	0	x	1	x	0	x	0
0 0 0	1		0	1	1	0	x	1	x	1	x	0
0 1 0	0		1	1	0	1	x	x	0	0	x	0
0 1 0	1		1	1	1	1	x	x	0	1	x	0
0 1 1	0		1	1	1	1	x	x	0	x	0	0
0 1 1	1		1	1	0	1	x	x	0	x	1	0
1 1 0	0		1	0	0	x	0	x	1	0	x	0
1 1 0	1		1	0	1	x	0	x	1	1	x	0
1 1 1	0		1	0	1	x	0	x	1	x	0	0
1 1 1	1		1	0	0	x	0	x	1	x	1	0
1 0 0	0		0	0	0	x	1	0	x	0	x	1
1 0 0	1		0	0	0	x	1	0	x	0	x	1
1 0 1	0		0	0	0	x	1	0	x	1	1	0
1 0 1	1		0	0	0	x	1	0	x	1	1	0



Also from K-maps for K_2 , J_3 and z

$K_2 = y_1$; $J_3 = y_2x + \bar{y}_1x$; $K_3 = x + \bar{y}_2$; $z = y_1y_2y_3$

Figure 6.137 K-maps for excitations of 3-bit odd-parity generator using J-K flip-flops.

Step 8. Implementation: The logic diagram based on those expressions is shown in Figure 6.138.

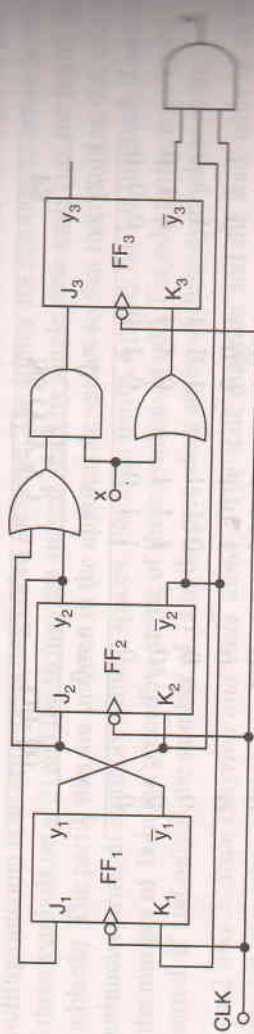


Figure 6.138 Logic diagram of a 3-bit odd-parity generator using J-K flip-flops.

EXAMPLE 6.18 A long sequence of pulses enters a 2-input 2-output synchronous sequential circuit which is required to produce an output $z = 1$, whenever the sequence 1111 occurs. Overlapping sequences are accepted. For example, if the input is 0101111, the required output is 00000011. Design the circuit.

Solution

Step 1. Word statement of the problem: The block diagram of the 2-input, 2-output synchronous sequential circuit (sequence detector) with one input terminal and one output terminal is shown in Figure 6.139. The sequence detector has to detect and produce an output 1 whenever the sequence 1111 occurs. Overlapping sequences are accepted.



Figure 6.139 Example 6.18: Block diagram of a 2-input 2-output synchronous sequential machine

Steps 2 and 3. State diagram and state table: Let the machine be initially in state A. While at A the first bit received may be a 0 or a 1. If it is a 0, it is invalid, so it will not start the detection process and so it will remain at A itself and outputs a 0. If the first bit is a 1, it is valid and so the detection process starts and the machine goes to next state B and outputs a 0.

While at B the next bit received may be a 0 or a 1. If the second bit is a 0, the first two bits become 10 which is not a part of the valid sequence and so the detection has to start afresh. So machine goes back to A the starting state and outputs a 0. If the second bit is a 1, the first two bits become 11 which is a part of the valid sequence. So machine goes to next state C and outputs a 0. While at C, the machine may receive the next bit as a 0 or a 1. If the third bit is a 0, the sequence becomes 110 which is not a part of the valid sequence. So machine will output a 0 and goes to the initial state A. If the third bit is a 1, the sequence becomes 111 which is a part of the valid sequence. So the machine goes to the next state D and outputs a 0. While at D, the machine may receive the next bit as a 0 or a 1. If the fourth bit is a 0, the sequence becomes 1110, which is not valid and so the machine outputs a 0 and goes to state A. If the fourth bit is a 1, the four bits become 1111 which is a valid sequence. The machine outputs a 1 and remains at D itself because overlapping is permitted. It can utilize the second, third and fourth bits, i.e. 111 and continue the detection process. So if the fifth bit is a 0, the last four bits become 1110, so it will output a 0 and goes to state A. If the fifth bit is a 1, the last four bits become 1111, so it will output a 1 and remain at D itself and so on.

Based on the above description of the working of the machine, the state diagram and the state table indicating the transition of states are shown in Figure 6.140.

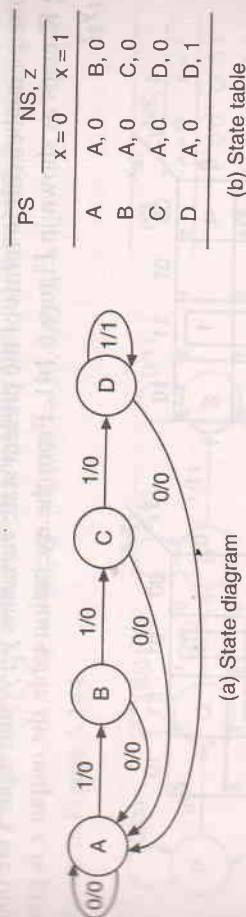


Figure 6.140 Example 6.18: Sequence (1111) detector.

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: The state assignment is arbitrary. There are four states. So, two state variables are needed, which can give a maximum of four states. Therefore, there are no invalid states. Let the states be assigned as $A \rightarrow 00$, $B \rightarrow 01$, $C \rightarrow 10$, and $D \rightarrow 11$. With this state assignment, the transition and output table is as shown in Table 6.20.

Table 6.20 Example 6.18: Transition and output table

PS	NS ($Y_1 Y_2$)		Output, z	
	Y_1	Y_2	x = 0	x = 1
0	0	0	0	0
0	1	0	0	0
1	0	0	0	0
1	1	0	0	1

Step 6. Choose type of flip-flops and form the excitation table: Let us say D flip-flops are used as memory elements. With D flip-flops as memory elements the excitation table is as shown in Table 6.21.

Table 6.21 Excitation table

PS		I/P		NS		I/P to FF		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		0	0	0	0	0
0	1	1		1	0	1	0	0
1	0	0		0	0	0	0	0
1	0	1		1	1	1	1	0
1	1	0		0	0	0	0	0
1	1	1		1	1	1	1	1

Step 7. K-maps and minimal expressions: The minimal expressions for excitations of FFs of the sequential circuit in terms of the present state variables y_1 , y_2 and input x are obtained using K-maps as shown in Figure 6.141. From the excitation table the output z is given by $z = y_1 y_2 x$.

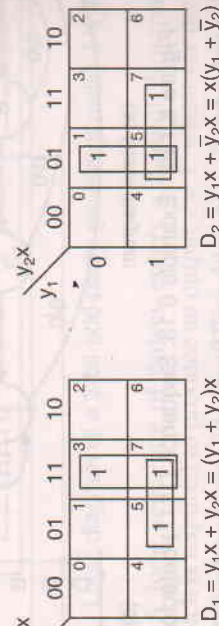


Figure 6.141 Example 6.18: K-maps for the sequence (1111) detector using D flip-flops.

Step 8. Implementation: The logic diagram of the sequence detector based on those minimal expressions is shown in Figure 6.142.

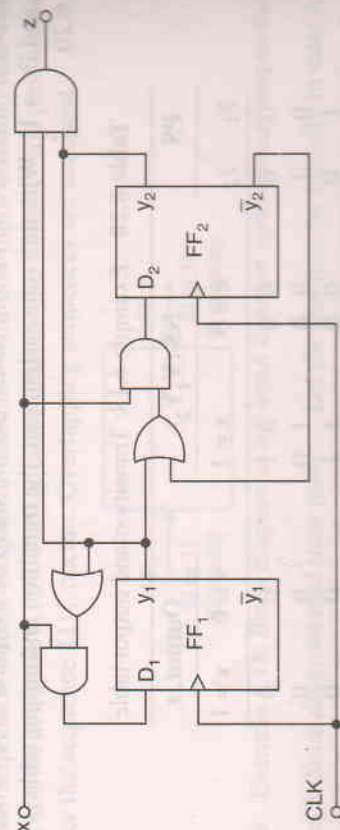


Figure 6.142 Example 6.18: Logic diagram of the sequence (1111) detector using D flip-flops.

machine cannot start processing the next string from here (since it is non overlapping type) it goes to state D just to provide a time delay and then goes to A whether the third bit received is a 0 or a 1. While at E, if the next bit is a 0, the string is 010 which is invalid and so the output is a 0; if it is a 1, the string is 011 which is valid and, so, the machine outputs a 1. In both the cases, the machine goes to A. While at C, the machine may receive a 0 or a 1. If it is a 0, the machine goes to E because the first two bits are 10. If it is a 1, it goes to state F. The first two bits are now 11. While at F, whether the next bit is a 0 or a 1 the machine outputs a 1 because both the sequences 110 and 111 are valid and goes to state A because overlapping is not permitted.

EXAMPLE 6.23

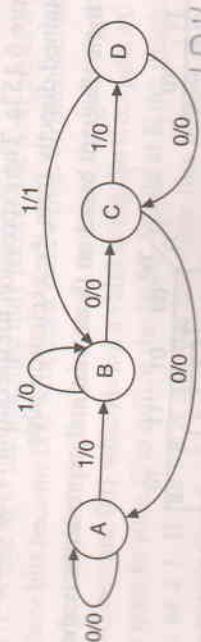
A clocked sequential circuit with single input x and single output z produces an output $z = 1$ whenever the input x completes the sequence 1011 and overlapping is allowed.

- Obtain the state diagram.
- Obtain its minimum state table and design the circuit with D flip-flops.

Solution

Step 1. Word statement of the problem: A single input (x), single output (z) synchronous sequential circuit to detect the sequence 1011 and to produce an output $z = 1$, whenever the input x completes the sequence is to be designed. It is given that overlapping is permitted.

Steps 2 and 3. State diagram and state table: The state diagram is drawn as shown in Figure 6.152. Let the machine be initially in state A. While at A it receives the first bit which may be a 0 or a 1. If it is a 0, the detection does not start. So the machine remains in state A itself and outputs a 0. If the first bit is a 1, it outputs a 0 and goes to state B. While at B, it receives the next bit which may be either a 0 or a 1. If it is a 0, the last two bits will be 10 which is a part of the valid sequence. So it outputs a 0 and goes to the next state C. If it is a 1, the last two bits become 11 which is not a part of the valid sequence. So it outputs a 0 and since overlapping is permitted it remains at state B (to utilize the last 1 bit). While at C it receives the third bit which may be a 0 or a 1. If it is a 0 the last 3 bits will be 100 which is not a part of the valid sequence and so it outputs a 0 and goes to state A because the detection has to start afresh. If it is a 1, the last 3 bits will be 101 which is part of the valid sequence. So the machine outputs a 0 and goes to state D. While at D it receives the fourth bit which may be either a 0 or a 1. If it is a 0, the last four bits become 1010 which is not a valid sequence. So it outputs a zero and since overlapping is permitted it goes to state C to utilize the last two bits 10. If it is a 1, the last 4 bits become 1011 which is a valid sequence. So the machine outputs a 1 and goes to state B to utilize the last bit 1. The state table is also shown in Figure 6.152.



(a) State diagram

Figure 6.152 Example 6.23

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

(b) State table

Step 5. State assignment and transition and output table: There are four states. The state variables are required. Two state variables can have a maximum of four states. If the states are utilized and there are no invalid states. Hence there are no don't care. Assign the states arbitrarily. Let $A \rightarrow 00$, $B \rightarrow 01$, $C \rightarrow 10$, and $D \rightarrow 11$ be the state assignment. With this state assignment, draw the transition and output Table 6.28.

Table 6.28 Example 6.23: Transition and output table

PS	$y_1 y_2$	NS ($Y_1 Y_2$)		O/P (z)	
		$x = 0$	$x = 1$	$x = 0$	$x = 1$
A $\rightarrow$	0 0	0 0	0 1	0	0
B $\rightarrow$	0 1	1 0	0 1	0	0
C $\rightarrow$	1 0	0 0	1 1	0	0
D $\rightarrow$	1 1	1 0	0 1	0	1

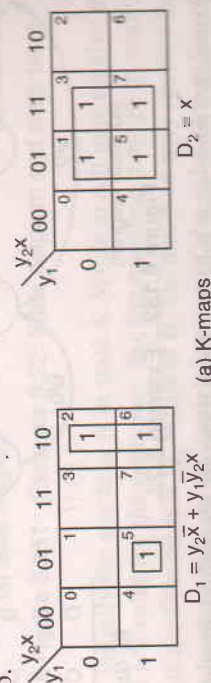
Step 6. Choose type of flip-flops and form the excitation table: It is given that the memory elements are D flip-flops. With D flip-flops as memory elements draw the excitation Table 6.29.

Table 6.29 Example 6.23: Excitation table

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

Step 7. K-maps and minimal expressions: Draw the K-maps and simplify them to obtain the minimal expressions for excitations D_2 and D_1 in terms of present state outputs y_1 and y_2 and the input x as shown in Figure 6.153a. The expression for the output of the machine in terms of y_1 , y_2 and x can be obtained directly from the excitation table.

Step 8. Implementation: The logic diagram based on those minimal expressions is shown in Figure 6.153b.



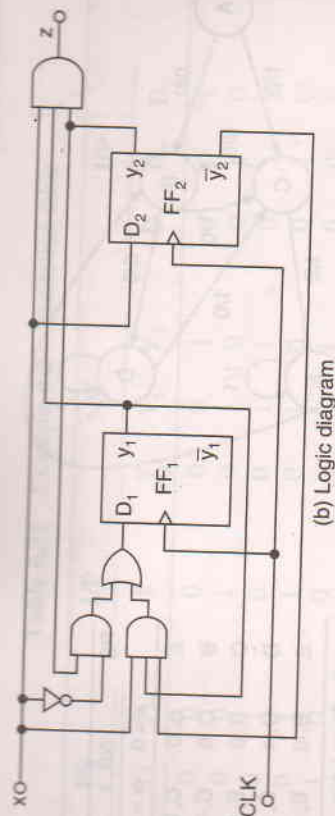


Figure 6.153 Example 6.23: K-maps and logic diagram.

EXAMPLE 6.24 A clocked sequential circuit is provided with a single input x and a single output z . whenever the input produces a string of pulses 111 or 000 and at the end of the sequence it produces an output $z = 1$ and overlapping is also allowed.

- Obtain the state diagram.
- Obtain the state table.
- Design the sequence detector.

Solution

Step 1. Word statement of the problem: A sequential circuit with one input terminal and one output terminal to detect the sequence 111 or 000 is to be designed. Overlapping is permitted.

Steps 2 and 3. State diagram and state table: Let the machine be initially in state A. While at A the machine may receive the first bit as a 0 or as a 1. Both are valid and the machine outputs a 0 and will go to state B if the first bit is a 0 and to state C if the first bit is a 1. While at B the machine may receive the next bit as a 0 or as a 1. If the bit is a 0, the first two bits become 00 which is a part of the valid string and so it outputs a 0 and goes to next state D. If the bit is a 1, the first two bits become 01 which is not a part of the valid string. It outputs a 0 and since overlapping is permitted, to utilize the last bit 1 the machine will go to state C. While at C the machine may receive the next bit as a 0 or as a 1. If the bit is a 0, the last two bits will be 10 which is not a part of the valid string. It outputs a 0 and since overlapping is permitted to utilize the last 0, the machine will go to state B. If the bit is a 1, the last two bits will be 11 which is a part of the valid sequence. So the machine goes to state E and outputs a 0. While at D the machine may receive the next bit as a 0 or as a 1. If the bit is a 0, the last three bits become 000 which is a valid string. So the machine outputs a 1 and since overlapping is permitted it remains on state D to utilize the last two zeros. If the bit is a 1, the last three bits become 001 which is not a valid string. So it outputs a 0 and since overlapping is permitted it goes to state C to utilize the last 1. While at E the machine may receive the next bit as a 0 or as a 1. If the bit is a 0, the last three bits become 110 which is not a valid string. So it outputs a 0 and since overlapping is permitted it goes to state B to utilize the last 0. If the bit is a 1, the last three bits will be 111 which is a valid sequence. So the machine outputs a 1 and remains in the same state to utilize the last two 1s. The state diagram and the state table are as shown in Figure 6.154.

* Finite State Machine (FSM)

Types

- 1) Moore State M/c $\rightarrow$ o/p is only depends on P.S
- 2) Mealy " " $\rightarrow$ o/p is depending on $\boxed{P.S + i/p}$

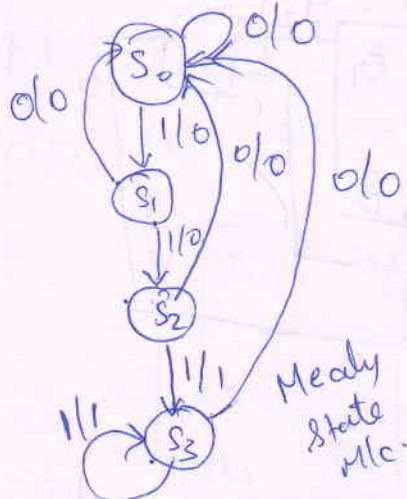
* Seq./ Pattern Detectors

- Steps required to implement S.D

a) Design a seq. detector to detect three (or) more consecutive 1's in the string of bits coming through an i/p line?

X - 001110111100---

Y - 0000100011---



$S_0 = \text{Reset}$

$S_1 = 1$

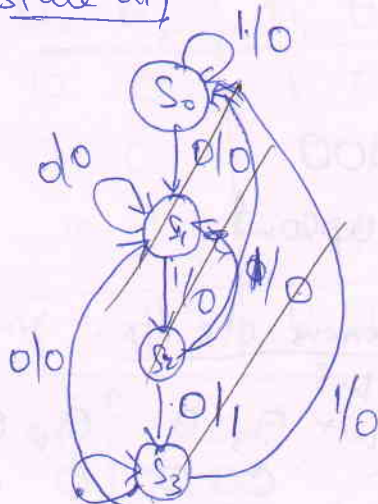
$S_2 = 11$

$S_3 = 111$

111111
11111

a) 010 seq. detector

State dig



$S_0 = \text{Reset}$

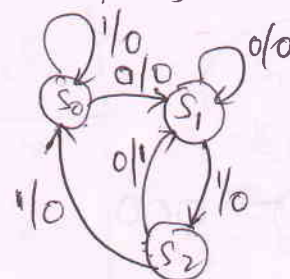
$S_1 = 0$

$S_2 = 01$

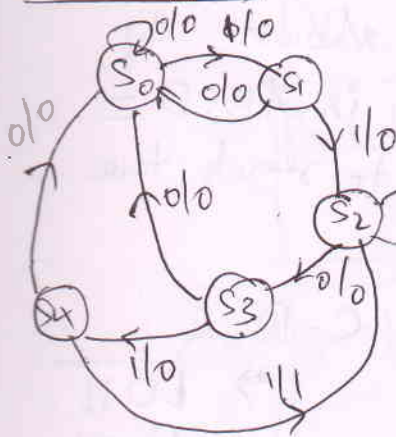
$S_3 = 010$

Overlapping allowed

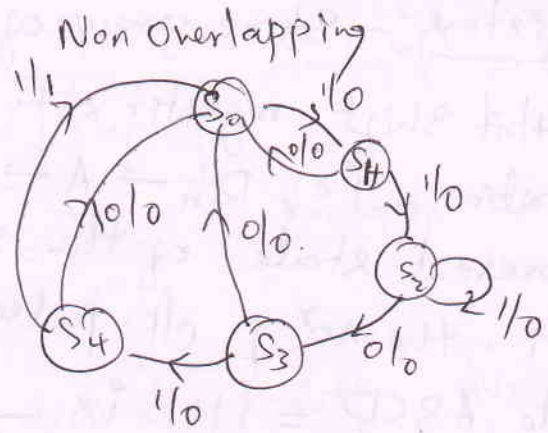
Non-Overlapping



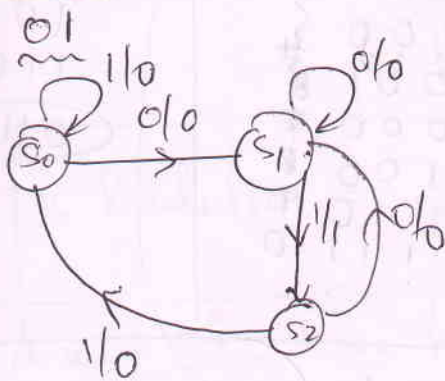
Q) 11011
overlapping



$S_0 = \text{Reset}$
 $S_1 = 1$
 $S_2 = 11$
 $S_3 = 110$
 $S_4 = 1101$



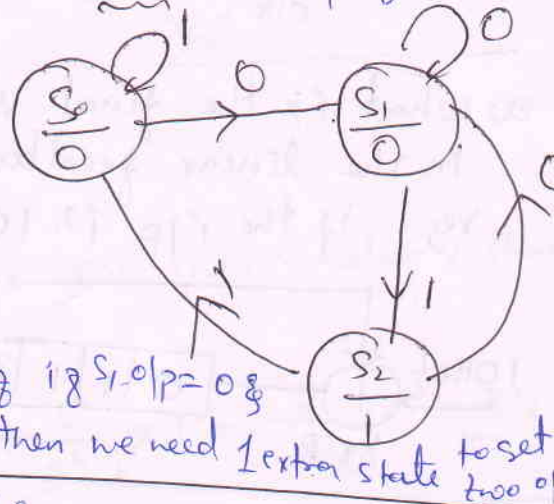
* Mealy



$S_0 = \text{Reset}$
 $S_1 = 0$
 $S_2 = 01$

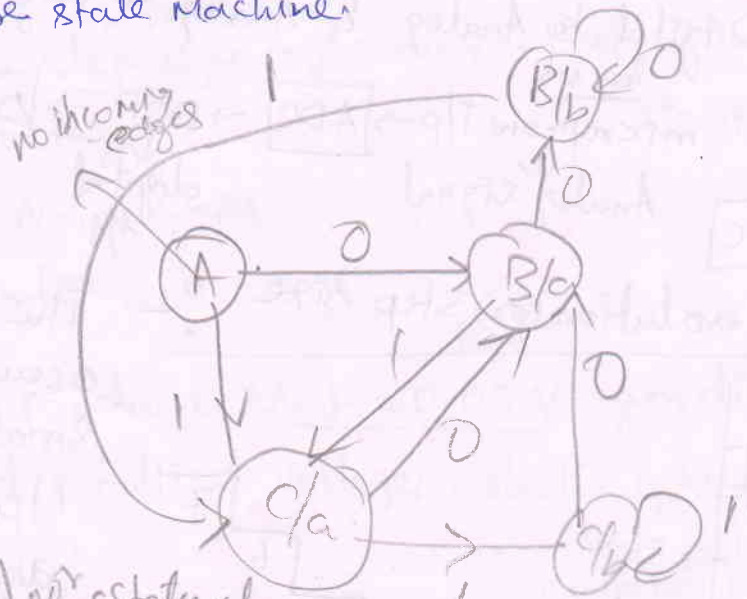
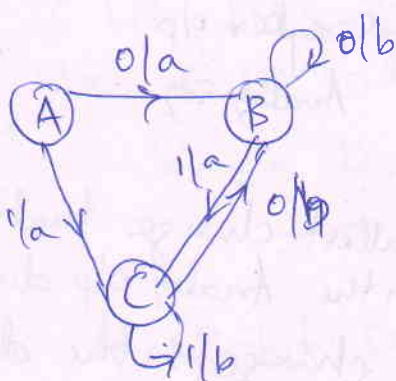
Moore

→ Some time Moore requires more states compared to Mealy



because if $S_1, 0/p = 0/0$
 1 then we need 1 extra state to set two 0

ex Draw equivalent Moore state machine.



States

⇒ (2x4) no. of states at max
 → Moore M/c.