CHAPTER

7

SEQUENTIAL CIRCUITS–II

7.1 FINITE STATE MACHINE

The most general model of a sequential circuit has inputs, outputs and internal states. A sequential circuit is referred to as a finite state machine (FSM). A finite state machine is an abstract model that describes the synchronous sequential machine. Since in a sequential circuit the output depends on the present input as well as on the past inputs, i.e. on the past histories and since a machine might have an infinite varieties of possible histories, it would need an infinite capacity for storing them. Since it is impossible to implement machines which have infinite storage capabilities, we consider only finite state machines. Finite state machines are sequential circuits whose past histories can affect their future behaviour in only a finite number of ways, i.e. they are 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 7.1 shows the block diagram of a finite state model. $x_1, x_2, ..., x_l$ are inputs. $z_1, z_2, ..., z_m$ are outputs. $y_1, y_2, ..., y_k$ are state variables, and $Y_1, Y_2, ..., Y_k$ represent the next state.

7.1.1 Capabilities and Limitations of Finite State Machines

1. Periodic sequence of finite states: With n-state machine, we can generate a periodic sequence of n states or smaller than n states. For example, in a 6-state machine, we can have a maximum periodic sequence as 0, 1, 2, 3, 4, 5, 0, 1, ...

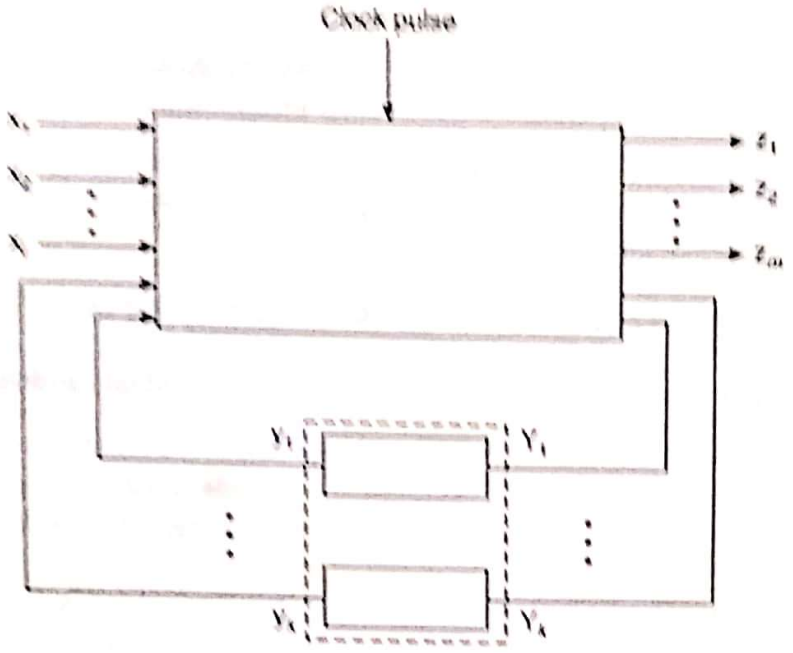


Figure 7.1 Block diagram of a finite state model.

2. No infinite sequence: Consider an infinite sequence such that the output is 1 when and only when the number of inputs received so far is equal to $P(P+1)/2$ for $P = 1, 2, 3, \ldots$, i.e. the desired input-output sequence has the following form:

Input:	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Output:	1	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1

Such an infinite sequence cannot be produced by a finite state machine.

3. Limited memory: The finite state machine has a limited memory and due to limited memory, it cannot produce certain outputs. Consider a binary multiplier circuit for multiplying two arbitrarily large binary numbers. If we implement this with a finite state machine capable of performing serial multiplication, we can find that it is not possible to multiply certain numbers. Such a limitation does occur due to the limited memory available to the machine. This memory is not sufficient to store arbitrarily large partial products resulted during multiplication.

Finite state machines are of two types. They differ in the way the output is generated. They are:

1. **Mealy type model:** In this model, the output is a function of the present state and the present input.
2. **Moore type model:** In this model, the output is a function of the present state only.

7.2 MATHEMATICAL REPRESENTATION OF SYNCHRONOUS SEQUENTIAL MACHINE

We know that the next state of a sequential machine depends upon the present state and the present input. The relation between the present state $S(t)$, present input $x(t)$, and next state $S(t+1)$ can be given as

$$S(t+1) = f\{S(t), x(t)\}$$

The value of output z(t) can be given as

$$z(t) = g\{S(t), x(t)\} \quad \text{for Mealy model}$$
$$z(t) = g\{S(t)\} \quad \text{for Moore model}$$

because in a Mealy machine, the output depends on the present state and input, whereas in a Moore machine, the output depends only on the present state. Table 7.1 shows a comparison between the Moore machine and Mealy machine.

Table 7.1 Comparison between the Moore machine and Mealy machine

Moore machine	Mealy machine
1. Its output is a function of present state only. $z(t) = g\{S(t)\}$	1. Its output is a function of present state as well as present input. $z(t) = g\{S(t), x(t)\}$
2. Input changes do not affect the output.	2. Input changes may affect the output of the circuit.
3. It requires more number of states for implementing same function.	3. It requires less number of states for implementing same function.

7.3 MEALY MODEL

When the output of the sequential circuit depends on both the present state of the flip-flops and on the inputs, the sequential circuit is referred to as Mealy circuit or Mealy machine.

Figure 7.2 shows the logic diagram of a Mealy model. Notice that the output depends upon the present state as well as the present inputs. Looking at the figure, we can easily realize that changes in the input during the clock pulse cannot affect the state of the flip-flop. However, they can affect the output of the circuit. Due to this, if the input variations are not synchronized with a clock, the derived output will also not be synchronized with the clock and we get false outputs. The false outputs can be eliminated by allowing input to change only at the active transition of the clock.

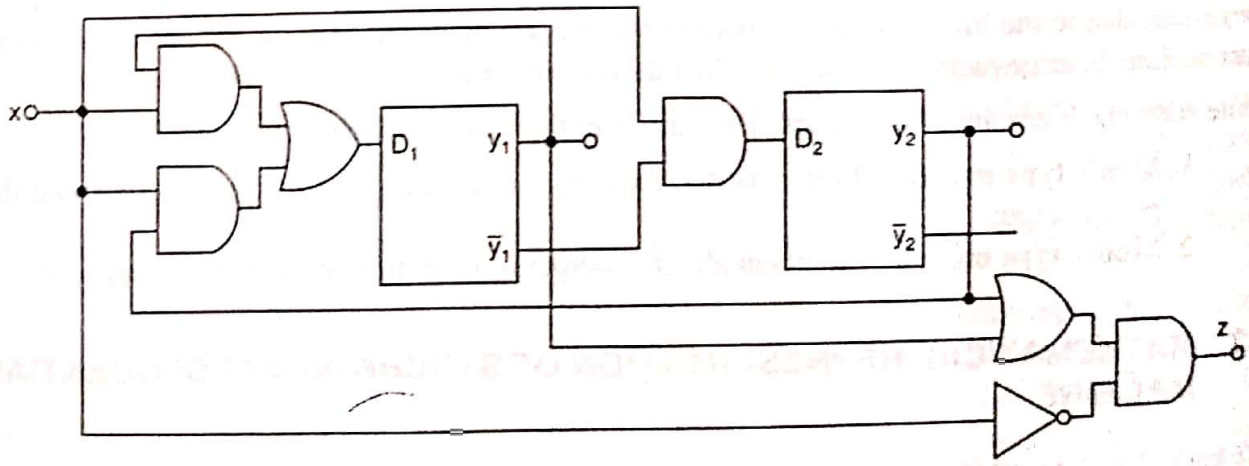


Figure 7.2 Logic diagram of a Mealy model.

The behaviour of a clocked sequential circuit can be described algebraically by means of state equations. A state equation (also called transition equation) specifies the next state as a function of the present state and inputs. The Mealy model shown in the figure consists of two D flip-flops, an

input x, and an output z. Since the D input of a flip-flop determines the value of the next state, the state equations for the model can be written as

$$y_1(t+1) = y_1(t)\,x(t) + y_2(t)\,x(t)$$
$$y_2(t+1) = \bar{y}_1(t)\,x(t)$$

and the output equation is

$$z(t) = \{y_1(t) + y_2(t)\}\,\bar{x}(t)$$

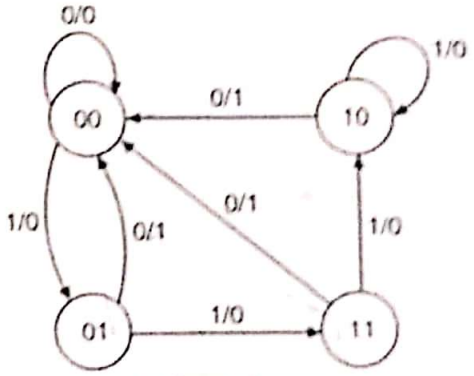
where $y(t+1)$ is the next state of the flip-flop one clock edge later, $x(t)$ is the present input, and $z(t)$ is the present output. If $y_1(t+1)$ and $y_2(t+1)$ are represented by $Y_1(t)$ and $Y_2(t)$, in more compact form, the equations are

$$y_1(t+1) = Y_1 = y_1x + y_2x$$
$$y_2(t+1) = Y_2 = \bar{y}_1x$$
$$z = (y_1 + y_2)\bar{x}$$

The state table of the Mealy model based on the above state equations and output equation is shown in Figure 7.3a. The state diagram based on the state table is shown in Figure 7.3b.

PS		NS				O/P	
		x = 0		x = 1		x = 0	x = 1
y_1	y_2	Y_1	Y_2	Y_1	Y_2	z	z
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

(a) State table



(b) State diagram

Figure 7.3 Mealy model.

In general form, the Mealy circuit can be represented with its block schematic as shown in Figure 7.4.

7.4 MOORE MODEL

As mentioned earlier, when the output of the sequential circuit depends only on the present state of the flip-flop, the sequential circuit is referred to as the Moore circuit or the Moore machine. Figure 7.5 shows the logic diagram of a Moore circuit.

Notice that the output depends only on the present state. It does not depend upon the input at all. The input is used only to determine the inputs of flip-flops. It is not used to determine the output. The circuit shown has two T flip-flops, one input x, and one output z. It can be described algebraically by two input equations and an output equation.

$$T_1 = y_2x$$
$$T_2 = x$$
$$z = y_1\,y_2$$

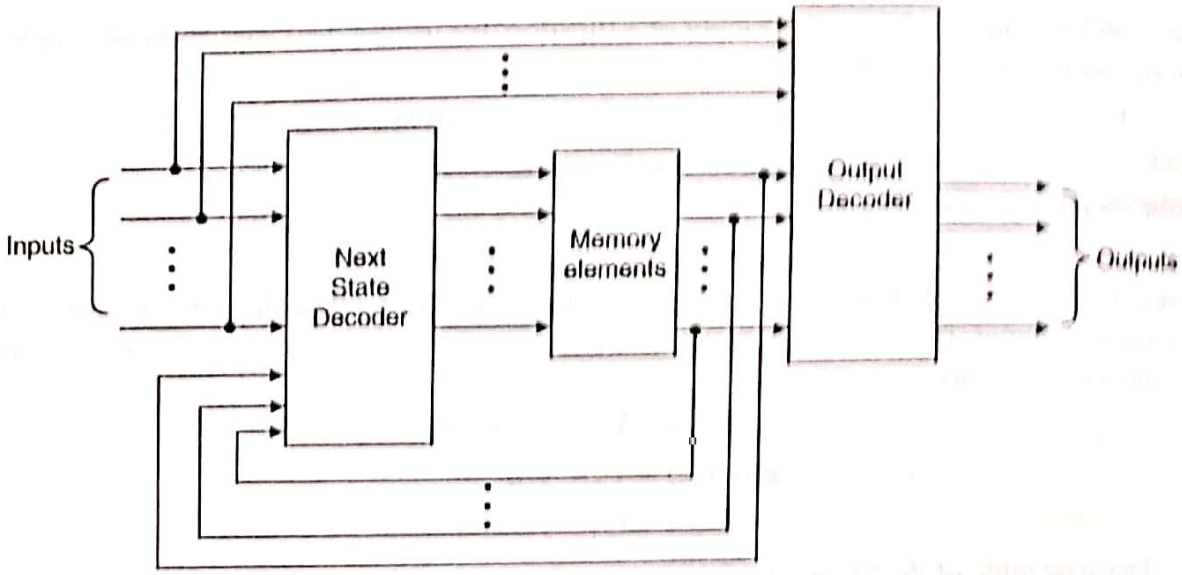


Figure 7.4 Mealy circuit model.

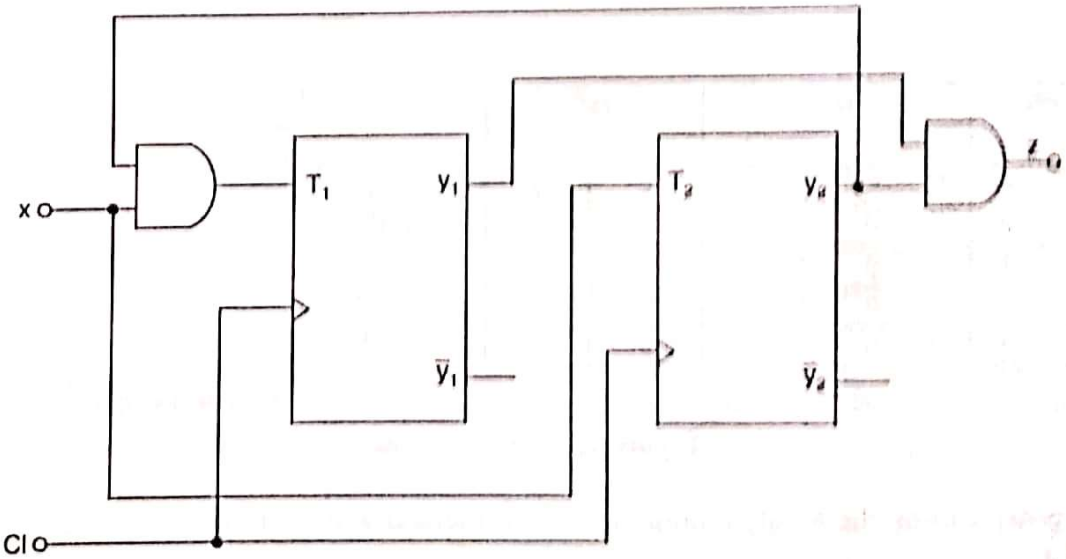


Figure 7.5 Logic diagram of a Moore model.

The characteristic equation of a T flip-flop is

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

The values for the next state can be derived from the state equations by substituting T_1 and T_2 in the characteristic equation yielding

$$y_1(t+1) = Y_1 = (y_2 x) \oplus y_1 = (\overline{y_2 x})y_1 + (y_2 x)\,\bar{y}_1$$

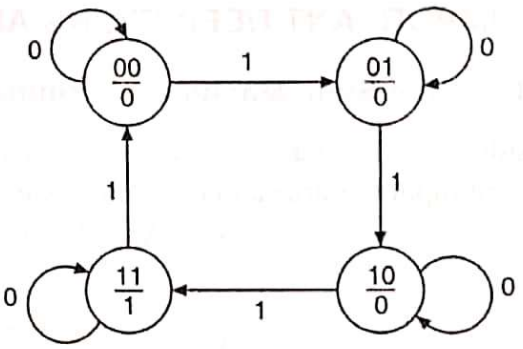
$$= y_1\bar{y}_2 + y_1\bar{x} + \bar{y}_1 y_2 x$$

$$y_2(t+1) = x \oplus y_2 = x\,\bar{y}_2 + \bar{x}y_2$$

The state table of the Moore model based on the above state equations and output equation is shown in Figure 7.6a. The state diagram based on the state table is shown in Figure 7.6b.

PS		NS				O/P
		x = 0		x = 1		
y_1	y_2	Y_1	Y_2	Y_1	Y_2	z
0	0	0	0	0	1	0
0	1	0	1	1	0	0
1	0	1	0	1	1	0
1	1	1	1	0	0	1

(a) State table



(b) State diagram

Figure 7.6 Moore model.

In general form, the Moore circuit can be represented with its block schematic as shown in Figure 7.7. Figure 7.8 shows the Moore circuit model with an output decoder.

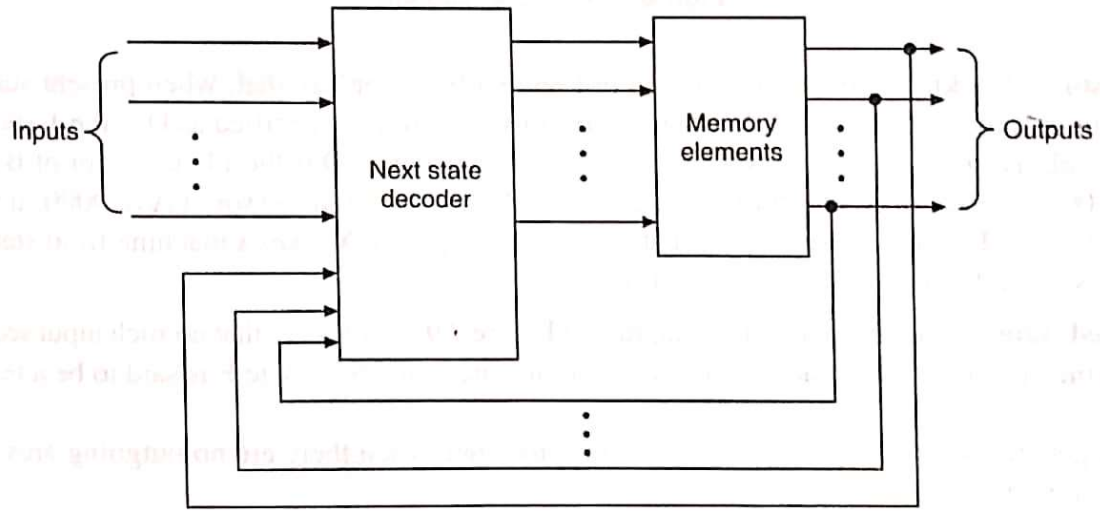


Figure 7.7 Moore circuit model.

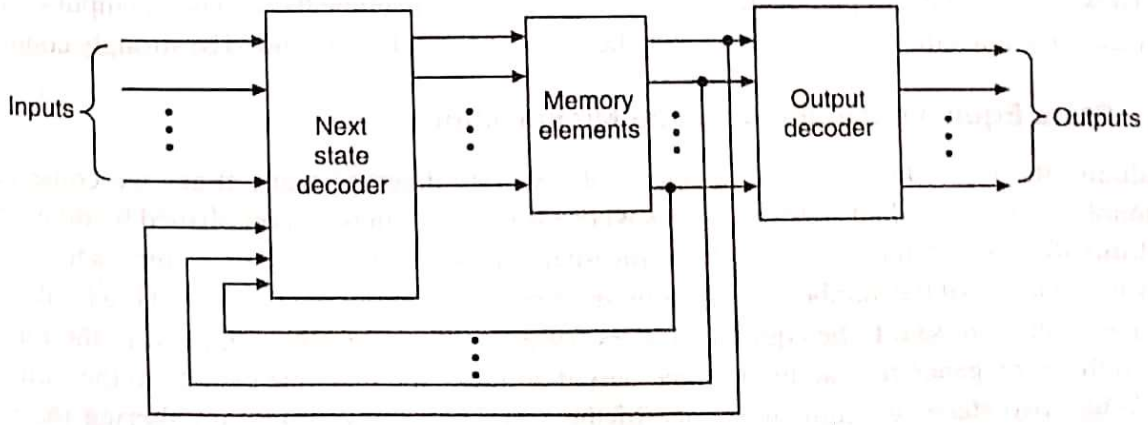


Figure 7.8 Moore circuit model with an output decoder.

7.5 IMPORTANT DEFINITIONS AND THEOREMS

7.5.1 Finite State Machine—Definitions

Consider the state diagram of a finite state machine shown in Figure 7.9. It is a five-state machine with one input variable and one output variable.

$$S = \{A, B, C, D, E\} \quad I = \{0, 1\} \quad O = \{0, 1\}$$

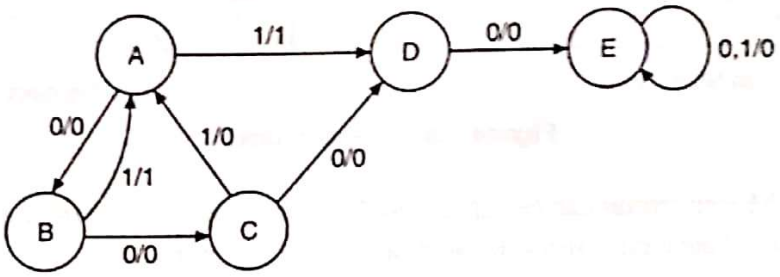


Figure 7. 9 State diagram.

Successor: Looking at the state diagram in Figure 7.9 we can say that, when present state is A and input is 1, the next state is D. In other words, this condition is specified as D is the 1-successor of A. Similarly, we can say that A is the 1 successor of B and C, D is the 11 successor of B and C, C is the 00 successor of A, D is the 000 successor of A, E is the 10 successor of A or 0000 successor of A and so on. In general, we can say that, if an input sequence X takes a machine from state S_i to S_j, then S_j is said to be the X successor of S_i.

Terminal state: Looking at the state diagram of Figure 7.9, we observe that no such input sequence exists which can take the sequential machine out of state E and thus state E is said to be a terminal state.

In general, we can say that a state is a terminal state when there are no outgoing arcs which start from it and terminate in other states.

Strongly-connected machine: In sequential machines many times certain subsets of states may not be reachable from other subsets of states, even if the machine does not contain any terminal state. However, if for every pair of states S_i, S_j of a sequential machine there exists an input sequence which takes the machine M from S_i to S_j, then the sequential machine is said to be strongly connected.

7.5.2 State Equivalence and Machine Minimization

In realizing the logic diagram from a state table or state diagram many times we come across redundant states. Redundant states are states whose functions can be accomplished by other states. The elimination of redundant states reduces the total number of states of the machine which in turn results in reduction of 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 applied to the machine while in this state generate exactly the same output and take the machine exactly to the same next state. When two states are equivalent, one of them can be removed without altering the input-output relationship. Here we will discuss what is meant by state equivalence and how to find equivalent states for machine minimization.

State equivalence theorem: It states that two states S_1 and S_2 are equivalent if for every possible input sequence applied, the machine goes to the same next state and generates the same output. That is, if $S_1(t + 1) = S_2(t + 1)$ and $Z_1 = Z_2$, then $S_1 = S_2$.

7.5.3 Distinguishable States and Distinguishing Sequences

Two states S_A and S_B of a sequential machine are distinguishable, if and only if there exists at least one finite input sequence which when applied to the sequential machine causes different output sequences depending on whether S_A or S_B is the initial state. The sequence which distinguishes these states is called a distinguishing sequence of the pair (S_A, S_B).

Consider states A and B in the state table shown in Table 7.2. When input X is 0, their outputs are 0 and 1 respectively and therefore, states A and B are called 1-distinguishable. Now consider states A and E. The output sequence is as follows:

$$X = 0 \quad \begin{cases} A \rightarrow C, 0 \text{ and } E \rightarrow D, 0\text{: outputs are the same} \\ C \rightarrow E, 0 \text{ and } D \rightarrow B, 1\text{: outputs are different} \end{cases}$$

Here the outputs are different after 2-state transitions and hence states A and E are 2-distinguishable.

Table 7.2 Distinguishable states

PS	NS, Z	
	X = 0	X = 1
A	C, 0	F, 0
B	D, 1	F, 0
C	E, 0	B, 0
D	B, 1	E, 0
E	D, 0	B, 0
F	D, 1	B, 0

Again consider states A and C. The output sequence is as follows:

$$X = 0 \quad \begin{cases} A \rightarrow C, 0 \text{ and } C \rightarrow E, 0\text{: outputs are the same} \\ C \rightarrow E, 0 \text{ and } E \rightarrow D, 0\text{: outputs are the same} \\ E \rightarrow D, 0 \text{ and } D \rightarrow B, 1\text{: outputs are different} \end{cases}$$

Here the outputs are different after 3-state transitions and hence states A and B are 3-distinguishable. In general, we can say that, if two states have a distinguishable sequence of length K, the states are said to be K-distinguishable. The concept of K-distinguishability leads directly to the definition of K-equivalence. States that are not K distinguishable are said to be K-equivalent.

7.6 MINIMIZATION OF COMPLETELY SPECIFIED SEQUENTIAL MACHINES USING PARTITION TECHNIQUE

A procedure for the state minimization and determination of n equivalence is discussed in the following example. Consider the same state table given in Table 7.2.

Step 1. Partition the states into subsets such that all states in the same subset are 0-equivalent.

The first partition P_1 can be obtained by placing those states having the same outputs under all inputs, in the same block. This partitioning gives two subsets.

1. (A, C, E): Their outputs under 0 and 1 inputs are 0 and 0 respectively.
2. (B, D, F): Their outputs under 0 and 1 inputs are 1 and 0 respectively.

$$\therefore \quad P_1 = (A, C, E)(B, D, F)$$

Step 2. Partition the states into subsets such that all states in the same subset are 2- equivalent.

This can be accomplished by observing that two states are 2-equivalent if and only if they are 1-equivalent and their 1_i-successors of all possible I_i are also 1-equivalent. In other words, we can say that, two states are placed in the same block of partition P_2 if and only if they are in the same block of P_1 and for each possible I_i, their I_i successors are also contained in the same block of P_1.

1. The 0-successors of (A, C, E) are (C, E, D): They are in different blocks of P_1. So the block (A, C, E) must be split into (A, C) and (E).
2. The 1-successors of (B, D, F) are (F, E, B): They are in different blocks of P_1. Therefore, (B, D, F) must be split into (B, F) and (D).

$$\therefore \quad P_2 = (A, C)(E)(B, F)(D)$$

3. The 1-successors of (A, C, E) are (F, B, B): They are in the same block of P_1. The 0-successors of (B, D, F) are (D, B, D). They are in the same block of P_1. So, no partitioning is possible.

Step 3. Partition the states into subsets such that all states in the same block are 3-equivalent. For this, consider the states which are 2-equivalent, i.e. blocks in P_2.

1. The 0-successors of (A, C) are (C, E): They are in different blocks of P_2. So partition (A, C) into (A) and (C).
2. The 1-successors of (A, C) are (F, B): They are in the same block of P_2.

$$\therefore \quad P_3 = (A)(C)(E)(B, F)(D)$$

Further partitioning of states is not possible because we find that the 0- and 1-successors of (B, F), i.e. (D, D) and (B, F) are in the same block of P_3.

The states in the same blocks of P_3 are equivalent. So states B and F are equivalent. One of them is redundant and can be eliminated. Let us remove F and replace F by B in the other places in the table. The minimized state table is shown in Table 7.3.

Table 7.3 Minimized state table

PS	NS, Z	
	X = 0	X = 1
A	C, 0	F, 0
B	D, 1	F, 0
C	E, 0	B, 0
D	B, 1	E, 0
E	D, 0	B, 0

In general, we can say that the P_{k+1} partition is obtained from the P_k partition by placing in the same block of P_{k+1}, those states which are in the same block of P_k and whose I_i successors for every possible I_i are also in a common block of P_k.

When $P_{k+1} = P_k$, the partitioning process terminates and P_k defines the sets of equivalent states of the sequential machine. The P_k is thus called the *equivalence partition* and the partitioning procedure discussed above is referred to as the *Moore reduction procedure.*

Note:

1. The equivalent partition is unique.
2. If two states, S_i and S_j of sequential machine M are distinguishable, then they are distinguishable by a sequence of length $n-1$ or less, where n is the number of states in M.

Machine equivalence: Two machines, M_1 and M_2 are said to be equivalent if and only if for every state in M_1, there is a corresponding equivalent state in M_2 and vice versa.

EXAMPLE 7.1 For the machine given in Table 7.4, find the equivalence partition and a corresponding reduced machine in standard form and also explain the procedure.

Table 7.4 Example 7.1: State table

PS	NS, Z	
	X = 0	X = 1
A	B, 0	E, 0
B	E, 0	D, 0
C	D, 1	A, 0
D	C, 1	E, 0
E	B, 0	D, 0

Solution

1. For equivalence partition, group the states having the same output under all input conditions (i.e. for X = 0 and X = 1) into blocks.

In the given table, states (A, B, E) and states (C, D) have same outputs under all input conditions.

$$\therefore \quad P_1 = (A, B, E)\ (C, D)$$

2. See whether the 0 and 1-successors of states in each block of P_1 are in the same block of P_1 or not. If they are in different blocks partition the states.

Here the 0-successors of (A, B, E), i.e (B, E, B) are in the same block, but 1-successors of (A, B, E), i.e. (E, D, D) are in different blocks of P_1. So, partition (A, B, E) into (A) and (B, E). 0-successors of (C, D), i.e. (D, C) are in the same block. Also 1-successors of (C, D), i.e. (A, E) are in the same block. So, no partitioning is possible.

$$\therefore \quad P_2 = (A)\ (B, E)\ (C, D)$$

3. See whether the 0 and 1-successors of states in each block of P_2 are in the same blocks of P_2 or not. If they are in different blocks of P_2, partition them.

Here, the 0- and 1-successors of (B, E), i.e. (E, B) and (D, D) are in same blocks of P_2. The 0-successors of (C, D), i.e. (D, C) are also in one block, but the 1-successors of (C, D), i.e. (A, E) are in different blocks. So, partition (C, D) into (C) and (D).

$$\therefore \quad P_3 = (A)\ (B, E)\ (C)\ (D)$$

4. See whether the 0 and 1-successors of (B, E) are in same blocks of P_3. If they are in different blocks of P_3 partition them.

The 0 and 1-successors of P_3, i.e. (E, B) and (D, D) are in the same blocks of P_3.
So, no further partitioning is possible.

$$\therefore \quad P_4 = (A)\ (B, E)\ (C)\ (D)$$

Thus, equivalent states are

$$B = E$$

So, state E is redundant and can be removed. Also state E can be replaced by state B in the table. A corresponding reduced machine in standard form is shown in Table 7.5.

Table 7.5 Example 7.1: Reduced state table

PS	NS, Z	
	X = 0	X = 1
A	B, 0	B, 0
B	B, 0	D, 0
C	D, 1	A, 0
D	C, 1	B, 0

EXAMPLE 7.2

(a) Explain the limitations of finite state machines.

(b) Find the equivalence partition and a corresponding reduced machine in standard form for the machine given in Table 7.6.

Table 7.6 Example 7.2: State table

PS	NS, Z	
	X = 0	X = 1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

Solution

1. States having the same output under all input conditions can be grouped as

$$P_1 = (A, C, E)(B, D, F)$$

2. The 0- and 1-successors of (A, C, E), i.e. (E, E, C) and (D, B, F) are in the same block of P_1. 0-successors of (B, D, F), i.e. (F, F, B) are also in the same block of P_1. So, no partitioning is required, but 1-successors of (B, D, F), i.e. (D, B, C) are in different blocks of P_1. So, partition (B, D, F) into (B, D) and (F).

$$\therefore \quad P_2 = (A, C, E)(B, D)(F)$$

3. 0-successors of (A, C, E), i.e. (E, E, C) and the 0- and 1-successors of (B, D), i.e. (F, F) and (D, B) are in same blocks of P_2. So, no partitioning is required. The 1-successors of (A, C, E), i.e. (D, B, F) are in different blocks of P_2. So, partition (A, C, E) into (A, C) and (E).

$$\therefore \quad P_3 = (A, C)(E)(B, D)(F)$$

4. The 0- and 1-successors of (A, C), and (B, D) i.e. (E, E), (D, B) and (F, F), (D, B) are in the same blocks of P_3. So, no partitioning is required.

$$\therefore \quad P_4 = (A, C)(E)(B, D)(F)$$

Thus, equivalent states are

$$A = C \text{ and } B = D$$

So, states C and D are redundant and can be removed. C and D can be replaced by A and B respectively in the rest of the table. The resultant minimized state table is as shown in Table 7.7.

Table 7.7 Example 7.2: Reduced state table

PS	NS, Z	
	X = 0	X = 1
A	E, 0	B, 1
B	F, 0	B, 0
E	A, 0	F, 1
F	B, 0	A, 0

EXAMPLE 7.3 What are the conditions for two machines to be equivalent? For the machine given in Table 7.8, find the equivalence partition and a corresponding reduced machine in standard form.

Table 7.8 Example 7.3: State table

PS	NS, Z	
	X = 0	X = 1
A	F, 0	B, 1
B	G, 0	A, 1
C	B, 0	C, 1
D	C, 0	B, 1
E	D, 0	A, 1
F	E, 1	F, 1
G	E, 1	G, 1

Solution

1. States having the same output under all input conditions can be grouped as

$$P_1 = (A, B, C, D, E)(F, G)$$

2. 1-successors of (A, B, C, D, E), i.e. (B, A, C, B, A) and the 0- and 1-successors of (F, G), i.e. (E, E), and (F, G) are in the same blocks of P_1. So, no partitioning is required. The 0-successors of (A, B, C, D, E), i.e. (F, G, B, C, D) are in different blocks of P_1. So, partition (A, B, C, D, E) into (A, B) and (C, D, E).

$$\therefore \quad P_2 = (A, B)(C, D, E)(F, G)$$

3. The 0- and 1-successors of (A, B) and (F, G), i.e. (F, G), (B, A) and (E, E), (F, G) are in the same blocks of P_2. So, no partitioning is required.

The 1-successors of (C, D, E), i.e. (C, B, A) are in different blocks of P_2. Also the 0-successors of (C, D, E), i.e. (B, C, D) are in different blocks of P_2. So, partition (C, D, E) into (C) and (D, E).

$$\therefore \quad P_3 = (A, B)(C)(D, E)(F, G)$$

4. The 0- and 1-successors of (A, B) and (F, G), i.e. (F, G), (B, A) and (E, E), (F, G) and the 1-successors of (D, E), i.e. (B, A) are in the same blocks of P_3. So, no partitioning is possible. But the 0-successors of (D, E), i.e. (C, D) are in different blocks of P_3. So, partition (D, E) into (D) and (E).

$$\therefore \quad P_4 = (A, B)(C)(D)(E)(F, G)$$

5. The 0- and 1-successors of (A, B) and (F, G), i.e. (F, G), (B, A) and (E, E) (F, G) are in the same blocks of P_4. So, no further partitioning is possible.

Thus, equivalent states are

$$A = B \text{ and } F = G$$

So, states B and G are redundant and can be removed. In the rest of the table, B and G are replaced by A and F respectively. The resultant minimized state table is shown in Table 7.9.

Table 7.9 Example 7.3: Reduced state table

PS	NS, Z	
	X = 0	X = 1
A	F, 0	A, 1
C	A, 0	C, 1
D	C, 0	A, 1
E	D, 0	A, 1
F	E, 1	F, 1

EXAMPLE 7.4

(a) Define the state equivalence and machine equivalence with reference to sequential machines.

(b) Reduce the number of states in the state table given in Table 7.10, and tabulate the reduced state table and give proper assignment.

Table 7.10 Example 7.4: State table

PS	NS, Z	
	X = 0	X = 1
A	F, 0	B, 0
B	D, 0	C, 0
C	F, 0	E, 0
D	G, 1	A, 0
E	D, 0	C, 0
F	F, 1	B, 1
G	G, 0	H, 0
H	G, 1	A, 0

Solution

1. States having the same output under all input conditions can be grouped as

$$P_1 = (A, B, C, E, G)\ (D, H)\ (F)$$

2. The 1-successors of (A, B, C, E, G), i.e. (B, C, E, C, H) are in different blocks. So, split (A, B, C, E, G) into (A, B, C, E) and (G). The 0-successors of (A, B, C, E, G), i.e. (F, D, F, D, G) are in different blocks. So, partition (A, B, C, E, G) into (A, C) (B, E) and (G).

$$\therefore \quad P_2 = (A, C)\ (B, E)\ (G)\ (D, H)\ (F)$$

3. The 0-successors of (A, C), i.e. (F, F) and the 1-successors of (A, C), i.e. (B, E) are in the same blocks of P_2. The 0-successors of (B, E), i.e. (D, D) and the 1-successors of (D, H), i.e. (A, A) are also in the same block of P_2. So, no further partitioning is possible.

$$\therefore \quad P_3 = (A, C)\ (B, E)\ (G)\ (D, H)\ (F)$$

Thus, the equivalent states are

$$A = C,\ B = E,\ \text{and}\ D = H$$

So, states C, E and H are redundant and can be removed from the state table. Also states C, E and H can be replaced by A, B and D in the rest of the table. The resultant minimized state table is shown in Table 7.11.

Table 7.11 Example 7.4: Reduced state table

PS	NS, Z	
	X = 0	X = 1
A	F, 0	B, 0
B	D, 0	A, 0
D	G, 1	A, 0
F	F, 1	B, 1
G	G, 0	D, 0

There are five states. So three state variables are required. The state assignment can be

$$A \rightarrow 000,\ B \rightarrow 001,\ D \rightarrow 011,\ F \rightarrow 101,\ G \rightarrow 111.$$

EXAMPLE 7.5 For the machine described by Table 7.12 obtain
(a) The corresponding reduced machine table in standard form.
(b) Find a minimum length that distinguishes state A from state B.

Table 7.12 Example 7.5: State table

PS	NS, Z	
	X = 0	X = 1
A	B, 1	H, 1
B	F, 1	D, 1
C	D, 0	E, 1
D	C, 0	F, 1
E	D, 1	C, 1
F	C, 1	C, 1
G	C, 1	D, 1
H	C, 0	A, 1

Solution

(a)

1. States having the same output under all input conditions are grouped as

$$P_1 = (A, B, E, F, G)\ (C, D, H)$$

2. The 1-successors of (A, B, E, F, G), i.e. (H, D, C, C, D) are in the same block of P_1. Also the 0- and 1-successors of (C, D, H), i.e. (D, C, C) and (E, F, A) are in the same blocks of P_1. So, no partitioning is possible. The 0-successors of (A, B, E, F, G), i.e. (B, F, D, C, C) are in different blocks of P_1. So, partition (A, B, E, F, G) into (A, B) and (E, F, G).

$$\therefore \quad P_2 = (A, B)\ (E, F, G)\ (C, D, H)$$

3. The 1-successors of (A, B), i.e. (H, D) and the 0-successors of (C, D, H), i.e. (D, C, C) are in the same blocks of P_2. So, no partitioning is possible.
The 0-successors of (A, B), i.e. (B, F) are in different blocks of P_2. So, partition (A, B) into (A) and (B). The 1-successors of (C, D, H), i.e. (E, F, A) are in different blocks of P_2. So, partition (C, D, H) into (C, D) and (H).

$$\therefore \quad P_3 = (A)\ (B)\ (E, F, G)\ (C, D)\ (H)$$

4. The 0- and 1-successors of (E, F, G) and (C, D), i.e. (D, C, C), (C, C, D) and (D, C), (E, F) are in the same blocks of P_3. So, no further partitioning is possible.

$$\therefore \quad P_4 = (A)\ (B)\ (E, F, G)\ (C, D)\ (H)$$

Thus, equivalent states are

$$E = F = G \quad \text{and} \quad C = D$$

So, states F, G and D are redundant and can be removed. Also states F, G and D can be replaced by states E, E and C respectively in the remaining table. The corresponding reduced machine table in the standard form is shown in Table 7.13.

Table 7.13 Example 7.5: Reduced state table

PS	NS, Z	
	X = 0	X = 1
A	B, 1	H, 1
B	E, 1	C, 1
C	C, 0	E, 1
E	C, 1	C, 1
H	C, 0	A, 1

(b) Distinguishability of states A and B:

$$X=0\begin{cases} A \to B,\ 1 & B \to F,\ 1 & \text{outputs are the same} \\ B \to F,\ 1 & F \to C,\ 1 & \text{outputs are the same} \\ F \to C,\ 1 & C \to D,\ 0 & \text{outputs are different} \end{cases}$$

$$X=1\begin{cases} A \to H,\ 1 & B \to D,\ 1 & \text{outputs are the same} \\ H \to A,\ 1 & D \to F,\ 1 & \text{outputs are the same} \\ A \to H,\ 1 & F \to C,\ 1 & \text{outputs are the same} \\ H \to A,\ 1 & C \to E,\ 1 & \text{outputs are the same} \end{cases}$$

So, the minimum length that distinguishes state A from state B is 3, i.e. states A and B are 3-distinguishable.

EXAMPLE 7. 6

(a) Obtain a minimal state table using partition technique for the state table shown in Table 7.14.

(b) Find a minimal length sequence that distinguishes state q_1 from state q_2.

Table 7.14 Example 7.6: State table

PS	NS, Z	
	X = 0	X = 1
q_1	q_2, 0	q_8, 1
q_2	q_6, 0	q_4, 1
q_3	q_4, 1	q_5, 0
q_4	q_3, 1	q_6, 0
q_5	q_4, 0	q_5, 1
q_6	q_3, 0	q_5, 1
q_7	q_3, 0	q_4, 1
q_8	q_3, 1	q_1, 0

Solution

(a)

1. States having the same output under all input conditions can be grouped as

$$P_1 = (q_1, q_2, q_5, q_6, q_7)\ (q_3, q_4, q_8)$$

2. The 0-successors of (q_3, q_4, q_8), i.e. (q_4, q_3, q_3) and the 1-successors of (q_3, q_4, q_8), i.e. (q_5, q_6, q_1) are in the same blocks of P_1. So, no partitioning is required.

The 0-successors of $(q_1, q_2, q_5, q_6, q_7)$, i.e. $(q_2, q_6, q_4, q_3, q_3)$ are in different blocks of P_1. So, partition $(q_1, q_2, q_5, q_6, q_7)$ into (q_1, q_2) and (q_5, q_6, q_7). The 1-successors of $(q_1, q_2, q_5, q_6, q_7)$, i.e. $(q_8, q_4, q_5, q_5, q_4)$ are in different blocks. So, partition $(q_1, q_2, q_5, q_6, q_7)$ into (q_1, q_2, q_7) and (q_5, q_6).

$$\therefore \quad P_2 = (q_1, q_2)\ (q_5, q_6)\ (q_7)\ (q_3, q_4, q_8)$$

3. The 1-successors of (q_1, q_2), i.e. (q_8, q_4) are in the same block of P_2 but the 0-successors of (q_1, q_2), i.e. (q_2, q_6) are in different blocks of P_2. So, partition (q_1, q_2) into (q_1) and (q_2). The 0-successors of (q_5, q_6), i.e. (q_4, q_3) are in the same block of P_2. Also the 1-successors of (q_5, q_6), i.e.(q_5, q_5) are in the same block of P_2. So, no partitioning of (q_5, q_6) is possible. The 0-successors of (q_3, q_4, q_8), i.e. (q_4, q_3, q_3) are in the same block of P_2. The 1-successors of (q_3, q_4, q_8), i.e. (q_5, q_6, q_1) are in different blocks of P_2. So, partition (q_3, q_4, q_8) into (q_3, q_4) and (q_8).

$$\therefore \quad P_3 = (q_1)\ (q_2)\ (q_5, q_6)\ (q_7)\ (q_3, q_4)\ (q_8)$$

4. The 0- and 1-successors of (q_5, q_6) i.e. (q_4, q_3) and (q_5, q_5), and the 0- and 1-successors of (q_3, q_4), i.e. (q_4, q_3) and (q_5, q_6) are in the same blocks of P_3. So, no further partitioning is possible.

$$\therefore \quad P_4 = (q_1)\ (q_2)\ (q_5, q_6)\ (q_7)\ (q_3, q_4)\ (q_8)$$

Thus, equivalent states are

$$q_5 = q_6 \text{ and } q_3 = q_4$$

So, states q_6 and q_4 are redundant and can be removed. In the remaining table q_6 can be replaced by q_5, and q_4 can be replaced by q_3. So, the resultant minimized state table is as shown in Table 7.15.

Table 7.15 Example 7.6: Reduced state table

PS	NS, Z	
	X = 0	X = 1
q_1	q_2, 0	q_8, 1
q_2	q_5, 0	q_3, 1
q_3	q_3, 1	q_5, 0
q_5	q_3, 0	q_5, 1
q_7	q_3, 0	q_3, 1
q_8	q_3, 1	q_1, 0

(b) The minimum length sequence that distinguishes state q_1 from state q_2 is determined as follows:

$$X = 0 \begin{cases} q_1 \rightarrow q_2, 0 & q_2 \rightarrow q_6, 0 & \text{outputs are the same} \\ q_2 \rightarrow q_6, 0 & q_6 \rightarrow q_3, 0 & \text{outputs are the same} \\ q_6 \rightarrow q_3, 0 & q_3 \rightarrow q_4, 1 & \text{outputs are different} \end{cases}$$

$$X = 1 \begin{cases} q_1 \rightarrow q_8, 1 & q_2 \rightarrow q_4, 1 & \text{outputs are the same} \\ q_8 \rightarrow q_1, 0 & q_4 \rightarrow q_6, 0 & \text{outputs are the same} \\ q_1 \rightarrow q_8, 1 & q_6 \rightarrow q_5, 1 & \text{outputs are the same} \\ q_8 \rightarrow q_1, 0 & q_5 \rightarrow q_5, 1 & \text{outputs are different} \end{cases}$$

So, the minimum length of the sequence that distinguishes state q_1 from state q_2 is 3.

EXAMPLE 7.7 Determine the minimal state equivalent of the state table shown in Table 7.16 using partition technique. Also determine the minimum length of sequence that distinguishes state B from state C.

Table 7.16 Example 7.7: State table

PS	NS, Z	
	X = 0	X = 1
A	A, 0	E, 1
B	A, 1	E, 1
C	B, 1	F, 1
D	B, 1	F, 1
E	C, 0	G, 0
F	C, 0	G, 0
G	D, 0	H, 0
H	D, 0	H, 0

Solution

1. States having the same output under all input conditions can be grouped as

$$P_1 = (A)\ (B, C, D)\ (E, F, G, H)$$

2. The 1-successors of (B, C, D), i.e. (E, F, F) are in the same block of P_1. Also, the 0- and 1-successors of (E, F, G, H), i.e. (C, C, D, D) and (G, G, H, H) are in the same blocks of P_1. So, no partitioning is possible.

The 0-successors of (B, C, D), i.e. (A, B, B) are in different blocks of P_1. So, partition (B, C, D) into (B) and (C, D).

$\therefore$
$$P_2 = (A)\ (B)\ (C, D)\ (E, F, G, H)$$

3. The 0- and 1-successors of (C, D), i.e. (B, B) and (F, F) are in the same blocks of P_2. Also, the 0- and 1-successors of (E, F, G, H), i.e. (C, C, D, D) and (G, G, H, H) are in the same blocks of P_2. So, no further partitioning is possible.

$\therefore$ $P_3 = (A)\ (B)\ (C, D)\ (E, F, G, H)$

Thus, the equivalent states are

$$C = D \quad \text{and} \quad E = F = G = H$$

So, state D and states F, G, H are redundant and can be removed. In the reduced table, D may be replaced by C and F, G and H may be replaced by E. The resultant minimized state table is shown in Table 7.17.

Table 7.17 Example 7.7: Reduced state table

PS	NS, Z	
	X = 0	X = 1
A	A, 0	E, 1
B	A, 1	E, 1
C	B, 1	E, 1
E	C, 0	E, 0

The minimum length of sequence:

$X = 0$

$B \rightarrow A, 1$ $\quad C \rightarrow B, 1$ $\quad$ outputs are the same

$A \rightarrow A, 0$ $\quad B \rightarrow A, 1$ $\quad$ outputs are different

$X = 1$

$B \rightarrow E, 1$ $\quad C \rightarrow F, 1$ $\quad$ outputs are the same

$E \rightarrow G, 0$ $\quad F \rightarrow G, 0$ $\quad$ outputs are the same

$G \rightarrow H, 0$ $\quad G \rightarrow H, 0$ $\quad$ outputs are the same

So, the minimum length of the sequence that distinguishes state B from state C is 2.

EXAMPLE 7.8 Determine a minimal state table equivalent to the state table given in Table 7.18 using the partition technique.

Table 7.18 Example 7.8: State table

PS	NS, Z	
	X = 0	X = 1
S_1	S_1, 1	S_1, 0
S_2	S_1, 1	S_6, 1
S_3	S_2, 0	S_5, 0
S_4	S_1, 0	S_7, 0
S_5	S_4, 1	S_3, 1
S_6	S_2, 0	S_5, 0
S_7	S_4, 1	S_3, 1

Solution

1. States having the same output under all input conditions can be grouped as

$$P_1 = (S_1)\ (S_2, S_5, S_7)\ (S_3, S_4, S_6)$$

2. The 1-successors of (S_2, S_5, S_7), i.e. (S_6, S_3, S_3) are in the same block of P_1. Also the 1-successors of (S_3, S_4, S_6), i.e. (S_5, S_7, S_5) are in the same block of P_1. So, no partitioning is required.

The 0-successors of (S_2, S_5, S_7), i.e. (S_1, S_4, S_4) are in different blocks of P_1. So, partition (S_2, S_5, S_7) into (S_2) and (S_5, S_7). Also, the 0-successors of (S_3, S_4, S_6), i.e. (S_2, S_1, S_2) are in different blocks of P_1. So, split (S_3, S_4, S_6) into (S_4) and (S_3, S_6).

$$\therefore \quad P_2 = (S_1)\ (S_2)\ (S_5, S_7)\ (S_4)\ (S_3, S_6)$$

3. The 0- and 1-successors of (S_5, S_7) and (S_3, S_6), i.e. (S_4, S_4), (S_3, S_3) and (S_2, S_2), (S_5, S_5) are in the same blocks of P_2. So, no further partitioning is possible.

$$\therefore \quad P_3 = (S_1)\ (S_2)\ (S_5, S_7)\ (S_4)\ (S_3, S_6)$$

Thus, the equivalent states are

$$S_5 = S_7 \text{ and } S_3 = S_6$$

So, states S_7 and S_6 are redundant and can be removed. Also, S_7 can be replaced by S_5 and S_6 can be replaced by S_3 in the remaining table. The resultant minimized state table is shown in Table 7.19.

Table 7.19 Example 7.8: Reduced state table

PS	NS, Z	
	X = 0	X = 1
S_1	S_1, 1	S_1, 0
S_2	S_1, 1	S_3, 1
S_3	S_2, 0	S_5, 0
S_4	S_1, 0	S_5, 0
S_5	S_4, 1	S_3, 1

7.7 SIMPLIFICATION OF INCOMPLETELY SPECIFIED MACHINES

In designing combinational logic circuits, we often encountered situations where the truth table was incompletely specified, which resulted in don't care terms. The same thing can happen in sequential circuits if the state transitions or output variables are not completely specified. The machines in which the state transitions or output variables are not completely specified are called incompletely specified machines.

When the state transitions are not specified, the sequential machine is not predictable. It is desirable to avoid such situations either by specifying the input sequence so that no next state is unspecified or by assigning next states that are not contrary to the desired results. This is illustrated in Table 7.20 and Table 7.21. Table 7.20 gives the incomplete description of a sequential machine. However Table 7.21 adds state T to describe specified behaviour of the sequential machine in