Finite State machine :-

A state machine is said to be finite, if it can be represented with finite number of states.

Ex: Sequential circuits. (contains finite states).

consider the state diagram shown in fig. below.

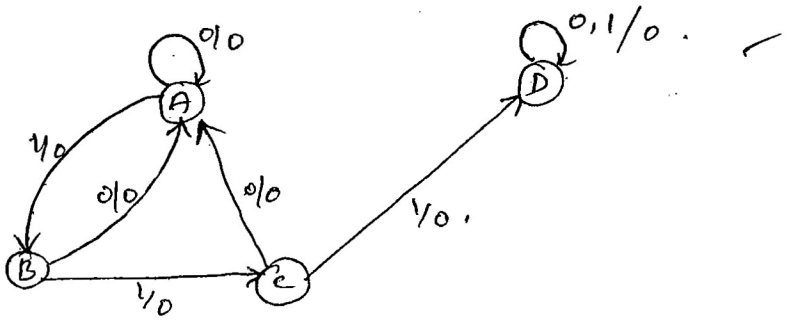


fig: state diagram

→ It is a 4 states machine with one i/p variable & one o/p variable such that;

(States) $S = \{A, B, C, D\}$

I/p's $I = \{0, 1\}$

o/p's $O = \{0, 1\}$.

Successor :-

From state dgm, we can say that, when present state is 'A' & i/p is '1', the next state is B. In other words this condition is specified as B is 1-successor of A.

Similarly we can say that A is 0-successor of C, C is a 11-successor of A, D is a 111-successor of A & so on.

→ In general, we can say that, if an i/p sequence 'x' takes a machine from state S_i to state S_j, then S_j is said to be the x-successor of S_i.

UNIT – VII

SEQUENTIAL CIRCUITS

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 14.1 shows the block diagram of a finite state model. $x_1, x_2, ..., x_i$ 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.

CAPABILITIES AND LIMITATIONS OF FINITE STATE MACHINES

1. Periodic sequence of finite states: With an *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, ...

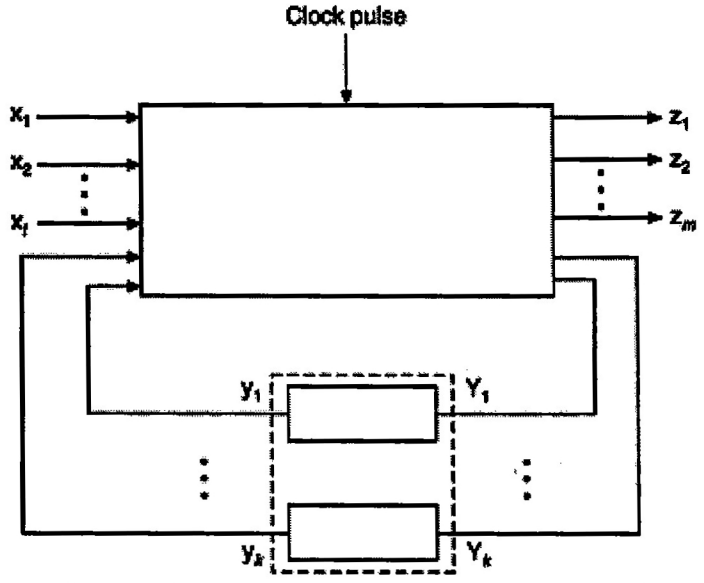


Figure 14.1 Block diagram of a finite state model.

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, ..., 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.

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 14.1 shows a comparison between the Moore machine and Mealy machine.

Table 14.1 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.

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 14.2 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.

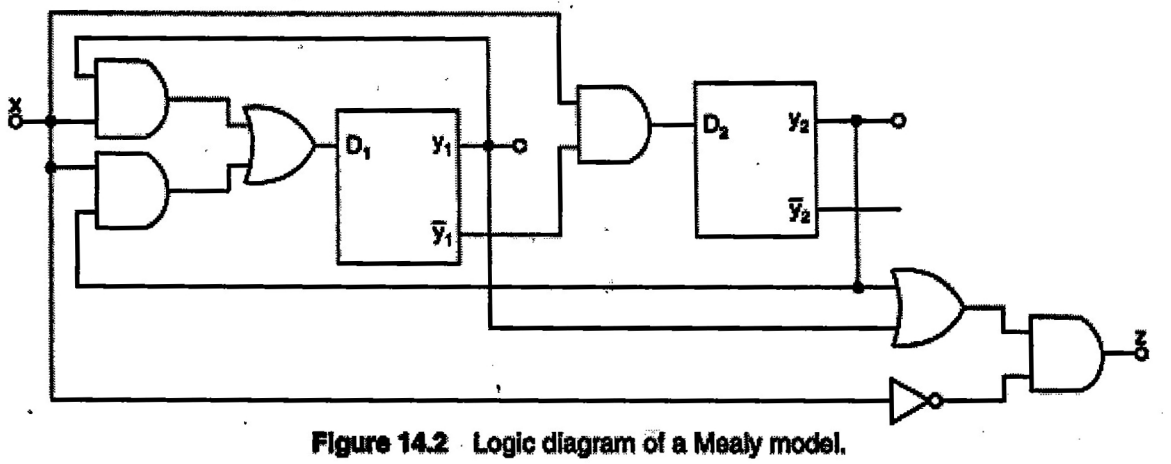


Figure 14.2 Logic diagram of a Mealy model.

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

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

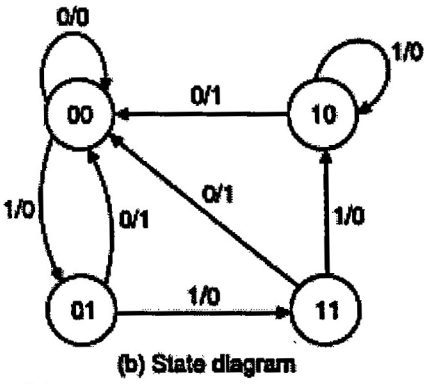
where y(t+1) is the next state of the flip-flop one clock edge later, x(t) is the present input, and z(t) is the present output. If $y_1(t+1)$ and $y_2(t+1)$ are represented by $Y_1(t)$ and $Y_2(t)$, in more compact form, the equations are

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

The state table of the Mealy model based on the above state equations and output equation is shown in Figure 14.3a. The state diagram based on the state table is shown in Figure 14.3b.

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 14.3 Mealy model.

In general form, the Mealy circuit can be represented with its block schematic as shown in Figure 14.4.

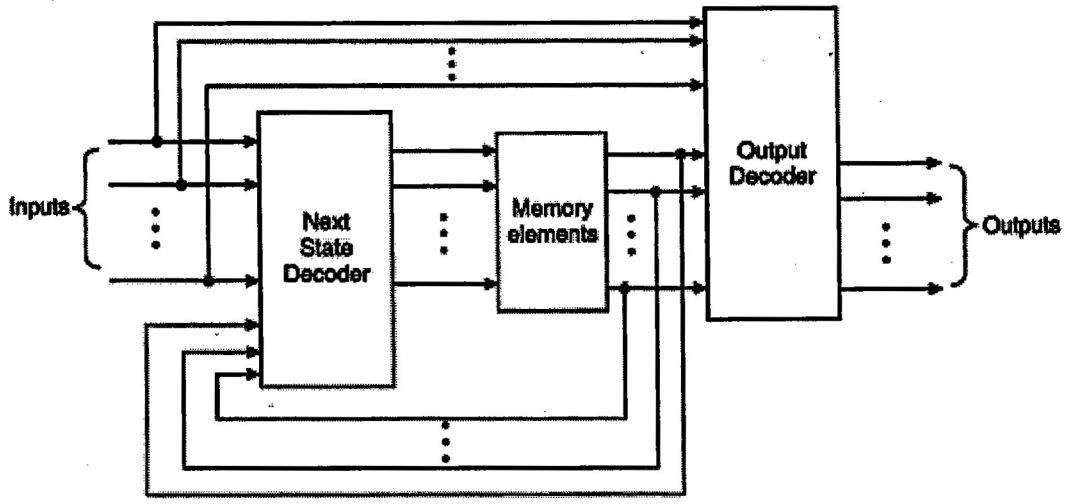


Figure 14.4 Mealy circuit model.

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 14.5 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.

$$T_1 = y_2 x$$
$$T_2 = x$$
$$z = y_1 y_2$$

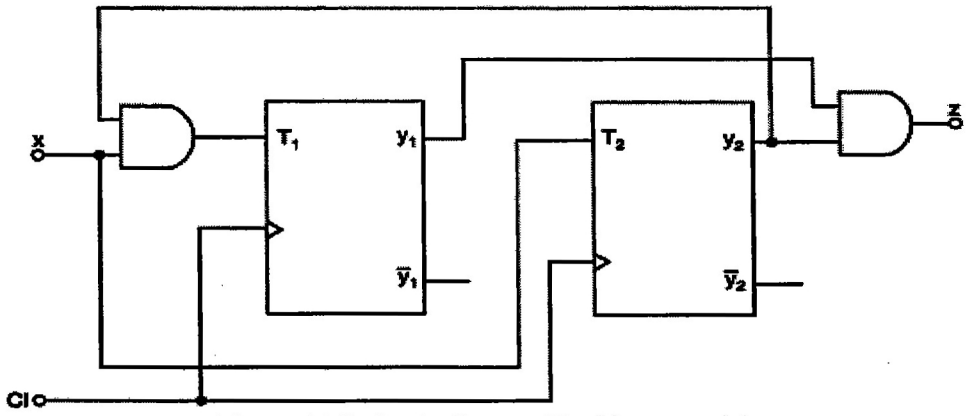


Figure 14.5 Logic diagram of a Moore model.

The characteristic equation of a T flip-flop is

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

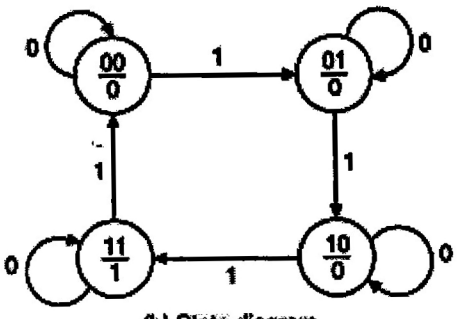
The values for the next state can be derived from the state equations by substituting T_1 and T_2 in the characteristic equation yielding

$$y_1(t+1) = Y_1 = (y_2 x) \oplus y_1 = (\overline{y_2 x})y_1 + (y_2 x)\overline{y}_1$$
$$= y_1 \overline{y}_2 + y_1 \overline{x} + \overline{y}_1 y_2 x$$
$$y_2(t+1) = x \oplus y_2 = x\overline{y}_2 + \overline{x}y_2$$

The state table of the Moore model based on the above state equations and output equation is shown in Figure 14.6a. The state diagram based on the state table is shown in Figure 14.6b.

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 14.6 Moore model.

In general form, the Moore circuit can be represented with its block schematic as shown in Figure 14.7. Figure 14.8 shows the Moore circuit model with an output decoder.

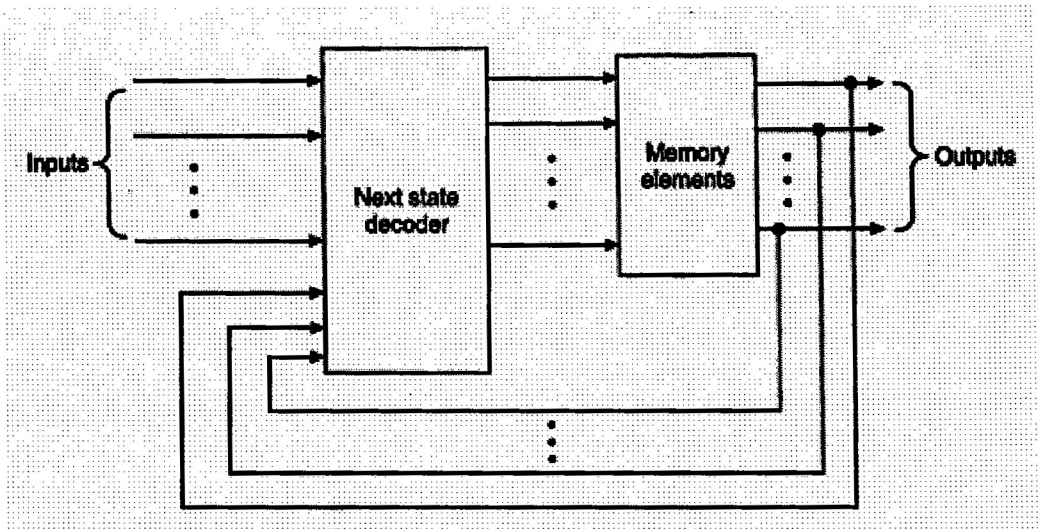


Figure 14.7 Moore circuit model.

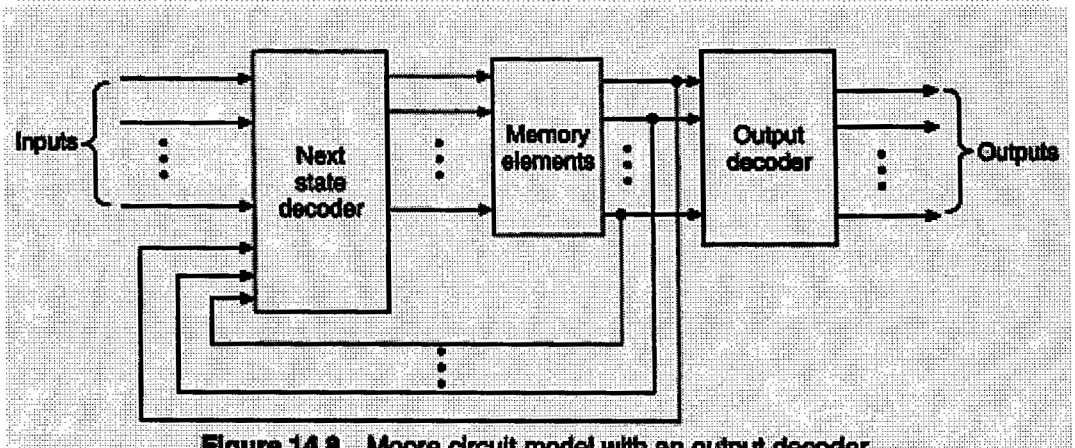
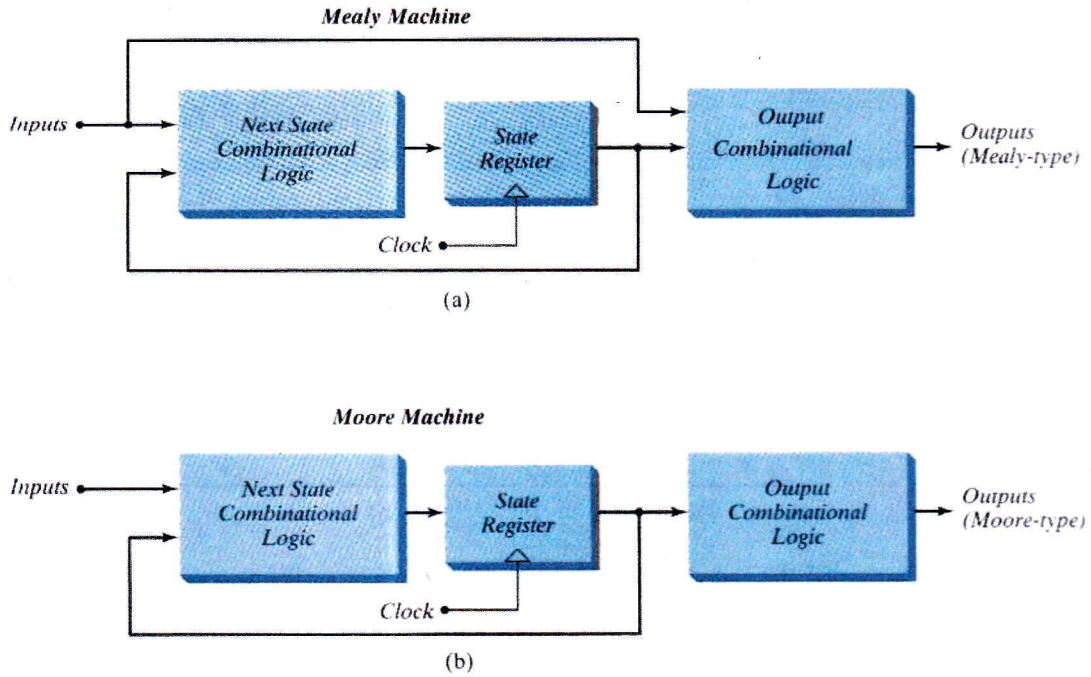


Figure 14.8 Moore circuit model with an output decoder.

Mealy and Moore Models of Finite State Machines

There are two models of sequential circuits: The Mealy model and the Moore model shown below. The only difference is in the generation of output.



Block diagrams of Mealy and Moore state machines

In the Mealy model, the output is a function of both the present state and the input.

In the Moore model, the output is a function of only the present state.

The two models of a sequential circuit are referred to as a Finite State Machine (FSM). They are referred to as Mealy FSM (or Mealy machine), and Moore FSM (or Moore machine).

An example of a Mealy model is given in Fig. 5.15 of the book (the example with two D FFs. Here, the output y is a function of both input x and the present states of A and B.

An example of a Moore model is given in Fig. 5.18 of the book (the example with two JK FFs. Here, the output is a function of the present states only.

Another example of a Moore model is the sequential circuit with two T FFs shown in Fig. 5.20 of the book. Here, the output depends only on the FFs output values which are functions of the present state only.

In a Moore model, the outputs of the sequential circuit are synchronized with the clock, because they depend only on FF outputs that are synchronized with the clock. In a Mealy model, the outputs may change if the inputs change during the clock. This causes momentary false values at the outputs because of the delay between the input change and the FF output change. In order to synchronize a Mealy type circuit, the inputs of the sequential circuit must be synchronized with the clock and the outputs must be sampled immediately before the clock edge. The inputs are changed at the inactive edge of the clock to ensure that the inputs of the FFs stabilize before the active edge of the clock edge occurs. Thus, the output of the Mealy machine is the value that is present immediatly before the active edge of the clock.

In the state diagram of a Mealy model circuit outputs are marked on the arcs in the form of "input/output". In the state diagram of a Moore circuit outputs are marked in the circles in the form of "State/output".

ANALYSIS OF CLOCKED SEQUENTIAL CIRCUITS

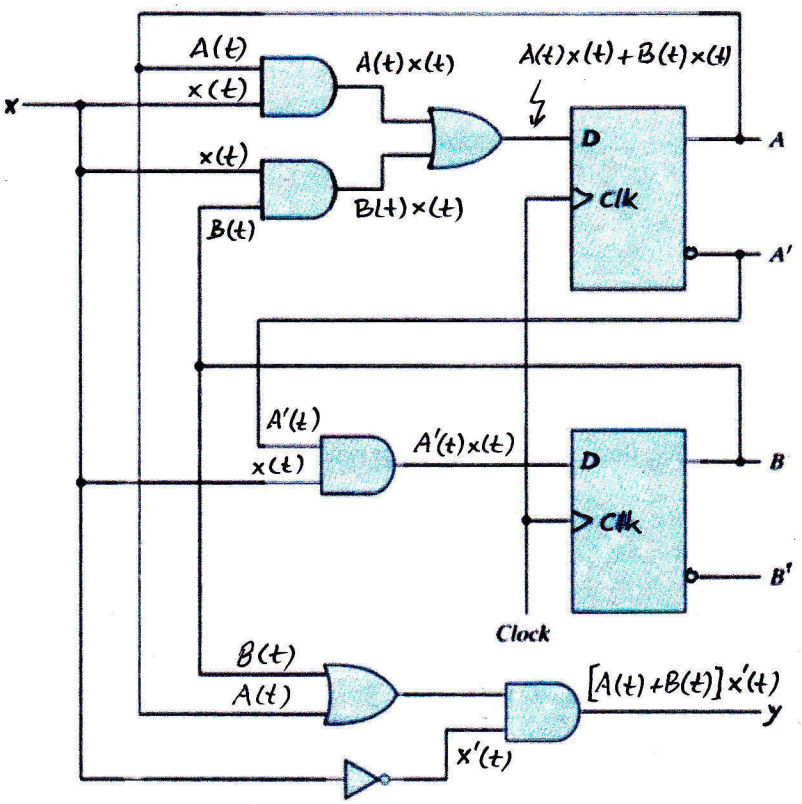
Analysis of a clocked sequential circuit describes the behavior of the circuit from the inputs, outputs and the state of its FFs.

The outputs and the next state are both functions of the inputs and the present state.

The analysis of a sequential circuit consists of obtaining the state equations, and then the state table or state diagram.

State Equations : A state equation (or transition equation) is an algebraic equation that specifies the next state as a function of the present state and inputs.

Example: Consider the sequential circuit shown below. It consists of two D FFs (A and B), and an input x and an output y.



For a D FF, the next state $Q(t+1)$ is equal to the D input after the clock transition. Therefore, we can write the following state equations:

$$A(t+1) = A(t)x(t) + B(t)x(t)$$

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

The left side of the state equation, with $(t+1)$ denotes the next state of the FFs one clock edge later. The right side of the state equation is a Boolean expression that specifies the present state and the input conditions that make the next state equal to 1.

Since all the variables in the Boolean expressions are a function of the present state, we can ommit the designation (t) after each variable for convenience and can express the state equations in the more compact form

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

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

$$y(t) = [A(t) + B(t)]x'(t)$$

By removing the symbol (t) for the present state, we obtain the output Boolean equation:

$$y = (A + B)x'$$

State Table : The time sequence of inputs, outputs and FF states can be shown in a state table (or transition table) in one of the following two forms.

State Table for the Circuit

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

Second Form of the State Table

Present State		Next State				Output	
		x = 0		x = 1		x = 0	x = 1
A	B	A	B	A	B	y	y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

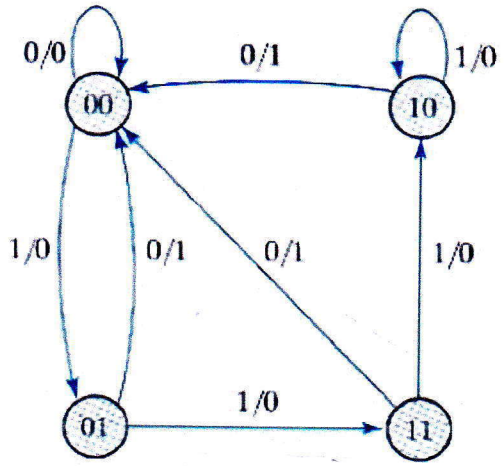
In making a state table, we have to list all possible combinations of the presen states and inputs (A, B, and x for this example).

The next state values are then determined from the state equations (or from the logic diagram).

In general, a squential circuit with m FFs and n inputs needs 2^{m+n} rows in the state table. The "next state" section has m colomns, one for each FF.

State Diagram : Represents the information available in a state table graphically. In a state diagram,
- A state is represented by a circle,
- (Clock triggered) transitions between states are indicated by directed lines connecting the circles.

The state diagram of the squential circuit given in the example is shown below.



State diagram

The state diagram provides the same information as the state table.
- The binary numbers inside each circle identifies the states of the FFs.
- The directed lines are labeled with two binary numbers separated by a slash (Present value of the input / Corresponding output value) Note that the output occurs during the present state and with the indicated input, and has nothing to do with the transition to the next state.

For example, the directed line from state 00 to 01 is labeled 1/0, meaning that when the sequential circuit is in the present state 00 and the input is 1, the output is 0. After the next clock cycle, the circuit goes to the next state 01. If the input changes to 0,

then the output becomes 1, and the FFs shifts to 00 state; if the input changes to 1, then the output becomes 0, and the FFs goes to state 11.

A directed line connecting a circle with itself indicates that no change of state occurs.

There is no difference between a state table and a state diagram, except in the manner of representation.

- The state table is easier to derive from a given logic diagram and the state equations.
- The state diagram gives pictorial view of state transitions and is in the form more suitable for human interpretation of the circuit's operation.

Flip-Flop Input Equations

The Boolean functions that describe the inputs of the FFs are called FF input equations (or, sometimes, excitation equations).

The Boolean functions describing the part of the combinational circuit that generates the external outputs are called output equations.

For example, the flip-flop input equation

$$D_Q = x + y$$

means that, a D flip-flop whose output is labeled with the symbol Q, and whose input is obtained from an OR gate with inputs x and y.

For the previous example, consisting of two D FFs A and B, an input x, and an output y, the FF input equations and the output equation:

$$D_A = Ax + Bx$$
$$D_B = A'x$$
$$y = (A + B)x'$$

The FF input equations imply the type of the FF from the letter symbol, and they fully specify the combinational circuit that drives the FFs.

Note that the input equation of a D FF is identical to the state equation. This is because the characteristic equation that equates the next state to the value of the D input: $Q(t+1) = D_Q$

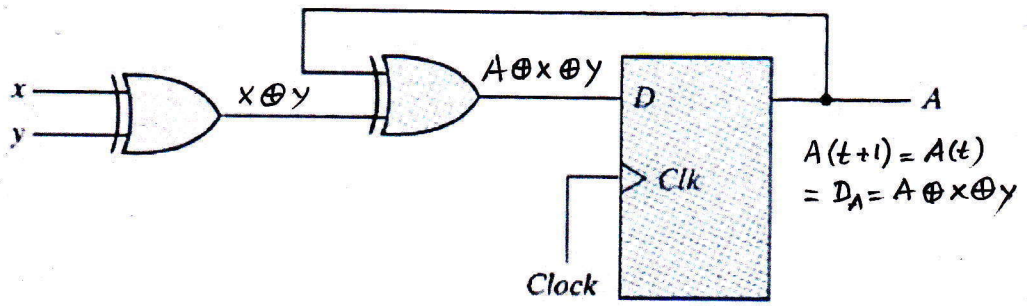
Analysis with D Flip-Flops

Suppose that the circuit we want to analize is described by the input equation

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

The D_A symbol implies a D FF with output A. The x and y variables are the inputs to the circuit. No output equations are given, which implies that the output comes from the output of the FF.

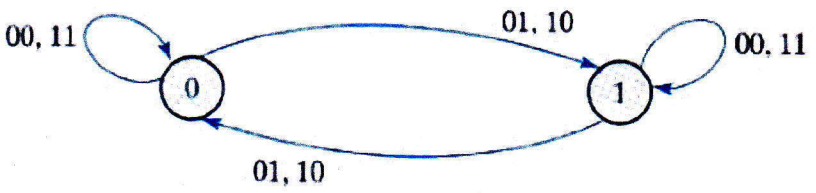
The logic diagram is obtained from the input equation as follows:



(a) Circuit diagram

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

(b) State table



(c) State diagram

Squential circuit with D FF

The state table has one column for the present state of A FF, two columns for the two inputs (x and y), one column for the next state of A. Axy are listed from 000 through 111. The next state values are obtained from the state eduation

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

The expression specifies an odd function and is equal to 1 when only one variable is 1 or when all three variables are 1. This is indicated in the column for the next state of A.

The circuit has one FF and two states. The state diagram consists of two circles, one for each state. The present state and the output can be 0 or 1, as indicated by the number inside the circles. A slash on the directed lines is not needed, because there is no output for the combinational circuit. The two inputs can have four possible combinations for each state. Two input combinations during each state transition are separated by a comma to simplify the notation.

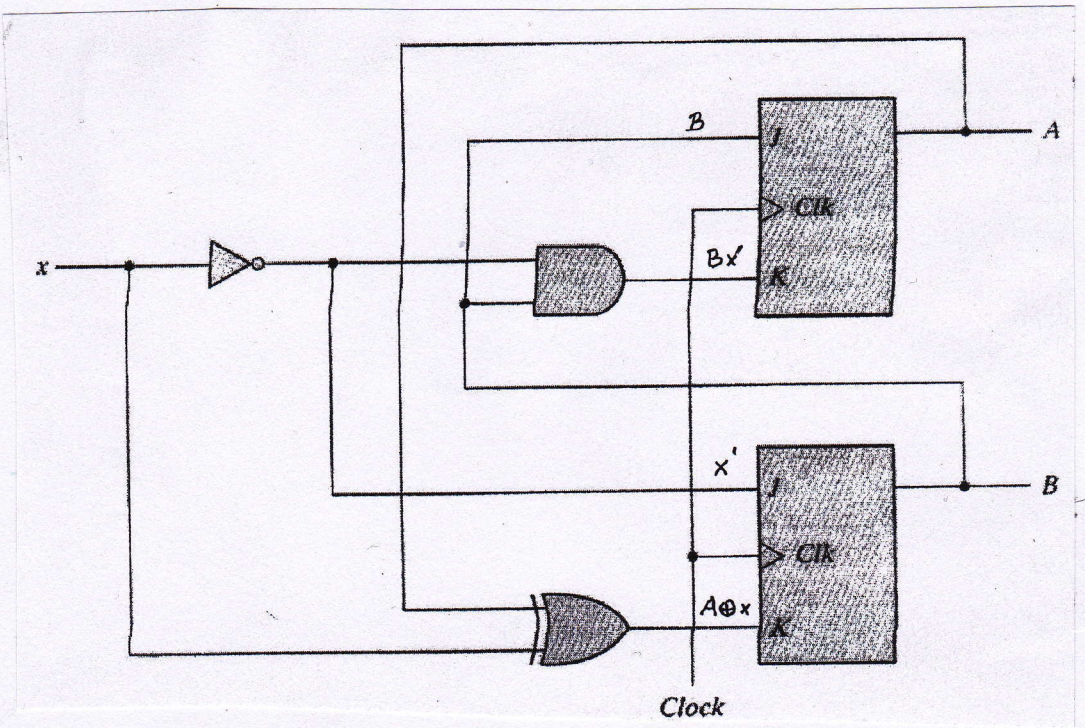
Analysis with JK Flip-Flops

For a D FF the state equation is the same as the input equation. When a FF other than the D type is used, such as JK or T, it is necessary to use the characteristic equation to obtain the next state values.

The next state values of a sequential circuit that uses JK or T type FFs can be derived as follows.

1. Determine the FF input equations in terms of the present state and input values.
2. List binary values for each input equation.
3. Use the characteristic table (or the state equations) to determine the next state value in the state table.

Example : Obtain the state table and the state diagram for the sequential circuit shown below.



The input equations of the circuit :

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

The state table of the sequential circuit:

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

The input equations are not part of the state table, but are needed for the evaluation of the next state (as specified in step 2 of the procedure). These columns are not needed when the state equations are used.

The binary values FF inputs (J_A, K_A, J_B, and K_B) are for the combinations of the present states A and B, and the input x.

The next state of each FF is obtained from the corresponding JK inputs and the characteristic table of the JK FF given below.

Characteristic Table of the JK FF

J	K	Q(t+1)	
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

The next state values can also be obtained by evaluating the state equations from the characteristic equation as follows:

Characteristic equation of a JK FF: $Q(t+1) = JQ'(t) + K'Q(t)$

Substiting A and B instead of Q(t),

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

Substituting the values of J_A, K_A, J_B, and K_B in the above equations we obtain the <u>state equations</u>:

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

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

$$= B'x' + [(A'x)' \cdot (Ax')']B$$

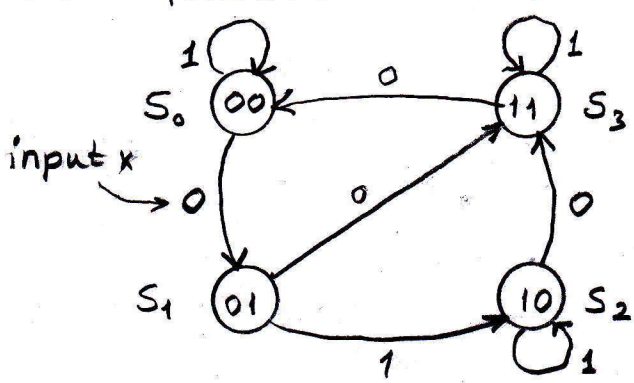
$$= B'x' + [(A+x') \cdot (A'+x)]B$$

$$= B'x' + (\underbrace{AA'}_{0} + Ax + A'x' + \underbrace{xx'}_{0})B$$

$$= B'x + ABx + A'Bx'$$

These state equations provides the bit values for the column headed "Next State" in the state table. Note that the columns headed "Flip-flop Inputs" are not needed when when state equations are used.

The <u>state diagram</u> of the sequential circuit is obtained from the state table as follows:



Note that, since the circuit has no outputs, the directed lines out of the circles are marked with the value of x only.

Analysis with T Flip-Flops

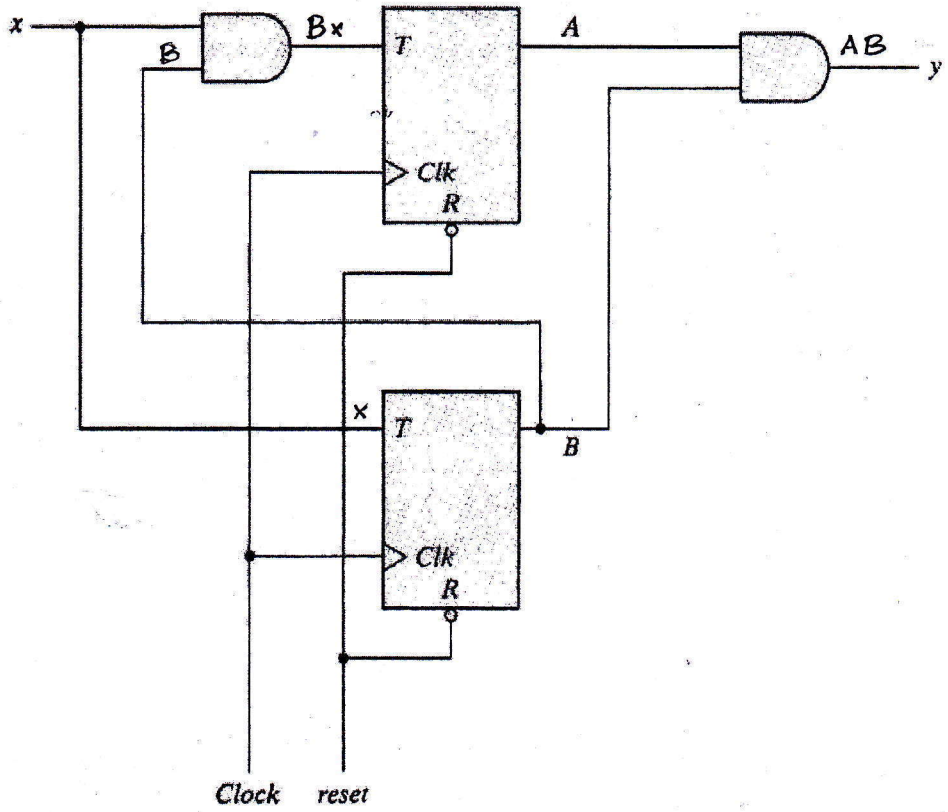
The analysis of a sequential circuit with T FFs is similar to that of JK FFs. The next state values in the state table can be obtained by using either the characteristic table listed below or the characteristic equation.

The characteristic table of a T flip-flop

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

The characteristic equation of a T FF : $Q(t+1) = T \oplus Q = T'Q + TQ'$

Example : Obtain the state table and state diagram of the sequential circuit shown below.



The sequential circuit has two FFs A and B, one input x, and one output y.

The input and output equations:

$$T_A = Bx$$
$$T_B = x$$
$$y = AB$$

State equations are obtained by substituting the input equations T_A and T_B in the characteristic equations, yielding

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

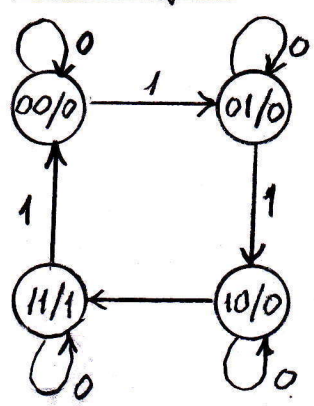
$$B(t+1) = x \oplus B$$

The next state values of A and B are obtained from these state equations and inserted in the following state table.

State Table for Sequential Circuit with T Flip-Flops

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

The state diagram:



As long as x is equal to 1 the circuit behaves as a binary counter with a sequence of states 00, 01, 10, 11, and back to 00.

Note that since the circuit has no outputs, the directed lines out of the circles are marked with the value of the input x only. The values of the output y are shown in the circles after the slash because the output y depends only on the present states A and B; it is independent of the input x.

DESIGN PROCEDURE

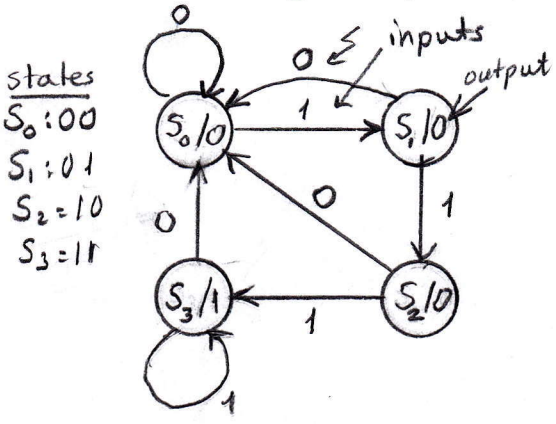
The design of a clocked sequential circuit starts with a set of specifications and ends up with a logic diagram or a list of Boolean functions from which the logic diagram can be obtained.

Design Steps of a Synchronous Sequential Circuits

1. From the word description and specifications of the desired operation, derive the state diagram of the circuit.
2. Reduce the number of states if necessary.
3. Assign binary values to the states.
4. Obtain the binary coded state table
5. Chose the type of the FFs to be used.
6. Derive the simplified input equations and output equations.
7. Draw the logic diagram.

Example:
Design a circuit that detects a sequence of three or more consecutive 1's in a string of bits coming through an input line. (the input is a serial bit stream).

The state diagram for this circuit is derived by starting with S_0, the reset state.



If input = 0, the circuit remains at state S_0
If input = 1, the circuit goes to state S_1 to indicate that 1 is detected.
If the next input = 1, state goes to S_2 to indicate the arrival of two consecutive 1 bits.
If the next input = 0, state goes to S_0.

- The third consecutive 1 sends the circuit to state S_2. If more 1's are detected, the circuit stays in S_3. Any 0 input sends the circuit back to S_0.

This is a Moore model sequential circuit, since the output is 1 when the circuit is in state S_3 and is 0 otherwise. That is, the output is a function of the present state only, but not a function of the input.

To design the circuit by hand, we need to assing binary codes to the states and list the state table (An alternative way is to use HDL - Hardware Description Language for the state diagram).

State Table for Sequence Detector

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

} y = 1 for the present state AB = 11

Let's choose D FFs to represent the four states, and label their outputs A and B. There is only one input x and one output y.

The characteristic equation of the D FF is $Q(t+1) = D_Q$, which means that the next state values in the state table specify the D input condition of the FF.

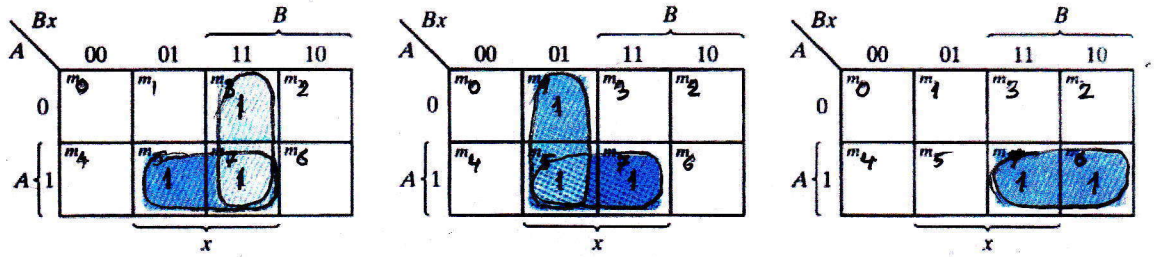
The FF input equations are directly obtained from the next state columns of A and B and expressed in sum-of-minterms form as follows:

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

} input equations of the FFs

$$y(A,B,x) = \Sigma(6,7)$$ } output equation

Let's simplify the input equations by means of the Karnaugh map.



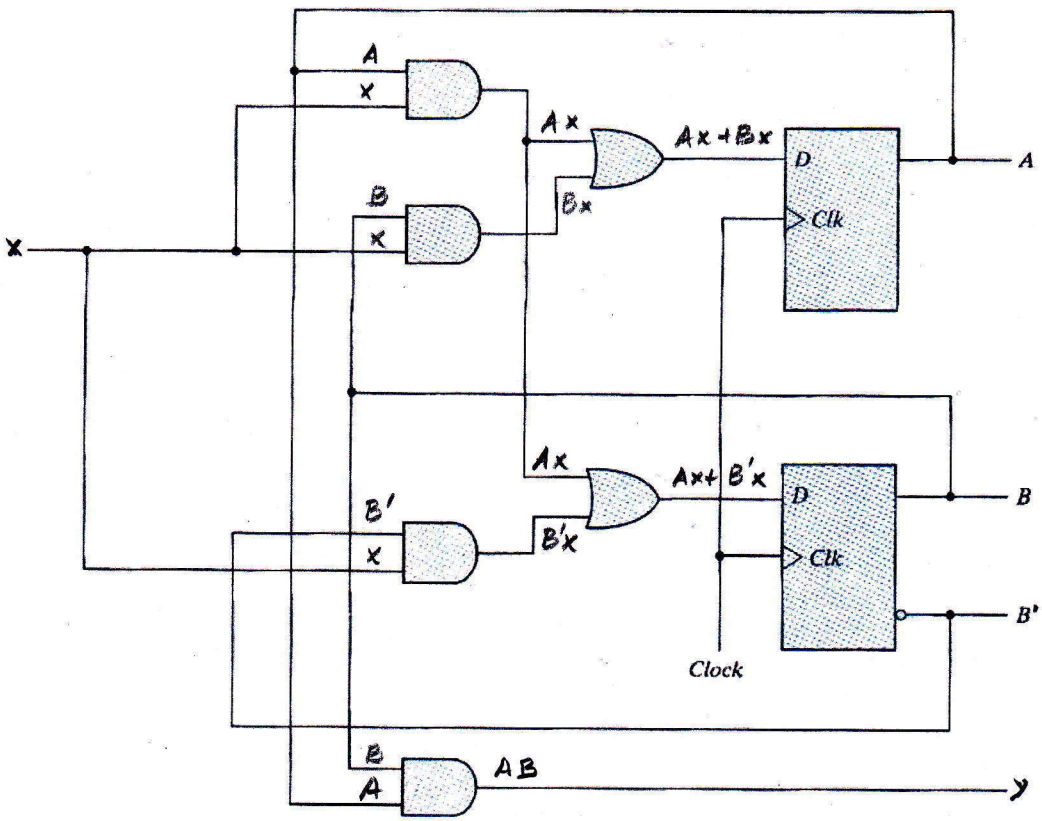
$D_A = Ax + Bx$ $D_B = Ax + B'x$ $y = AB$

The simplified input equations are:

$$D_A = Ax + Bx \;;\; D_B = Ax + B'x \;;\; y = AB$$

The advantage of designing with D FF is that the input equations are directly obtained from the state table

The logic diagram of the sequence detector:

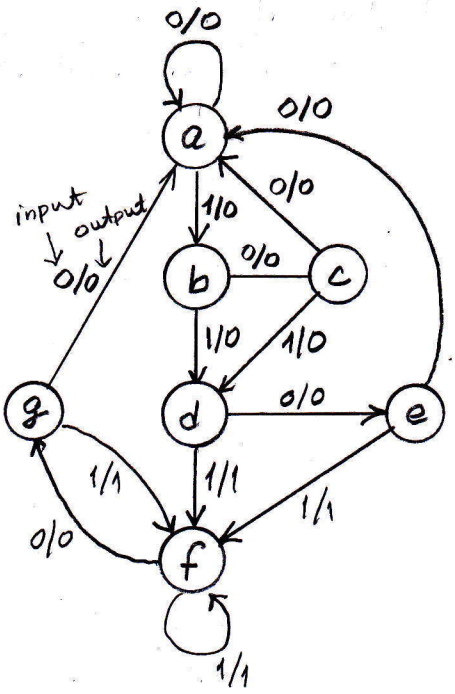


STATE REDUCTION AND ASSIGNMENT

State reduction is referred to as the reduction in the number of FFs in a sequential circuit, while keeping the input-output requirements unchanged. Since m FFs produces 2^m states, a reduction in the number of states may (or may not) result in a reduction in the number of FFs. Reducing the number of FFs sometimes causes an increase in the number of combinational gates in the equivalent circuit. We will illustrate the state-reduction procedure with an example.

Example: For the state diagram given below, consider the input sequence 0 1 0 1 0 1 1 0 1 0 0 starting from the initial state a. Each input of 0 and 1 produces an output of 0 and 1 and causes the circuit to go to the next state.

(In this example, since only the input-output sequences are important; the internal states are used merely to provide the required sequence, states marked inside the circles are denoted by letters instead of binary values).



State diagram

From the state diagram, we obtain the output and sequence for the given input sequence as follows:

With the circuit in initial state a, an input of 0 produces an output of 0 and the circuit remains in state a.
With present state a and input of 1, the output is zero and the next state is b. Continuing this process we find the complete sequence to be as follows:

State	a	a	b	c	d	e	f	f	g	f	g	a
Input	0	1	0	1	0	1	1	0	1	0	0	
Output	0	0	0	0	0	1	1	0	1	0	0	

In each column we have the present state, input and output values. The next state is written on top of the next column.

In this circuit the states are of secondary importance, because we are interested only in the output sequences caused by the input sequences. The problem of state reduction is to find ways of reducing number of states in a sequential circuit without altering the input output relationship.

Since the state table is more convenient to apply state reduction process rather than the use of state diagram, we need to obtain the state table

State Table

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

Two state are said to be equivalent if, for each input they give exactly the same output and send the circuit either to the same state or an equivalent state. When the two states are equivalent one of them can be removed without altering the input output relationship.

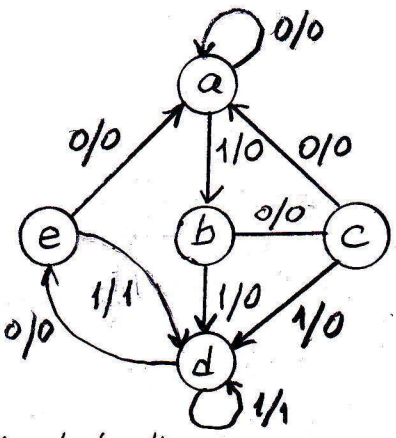
In the state table, we look for two present states that go to the same next state and have the same output for both input combinations.

- Present states g and e are such two cases ($g \equiv e$). Replace g by e each time it occurs in the "next state" column, and remove the row with the present state g.
- Present states f and d are equivalent ($f \equiv d$). Replace f by d each time it occurs in the "next state" column, and remove the row with the present state f.

The reduced state table:

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

The reduced state diagram



The following list can be derived from the reduced state diagram:

State	a	a	b	c	d	e	d	d	e	d	e	a
Input	0	1	0	1	0	1	1	0	1	0	0	
Output	0	0	0	0	0	1	1	0	1	0	0	

It is observed that, for the same input sequence use previously, the same output sequence results, although the state sequence is different.

In this example, the number of states are reduced from 7 to 5. However, this does not guarantee a saving in the number of FFs or the number of gates.

Reduction of state tables (Row matching Technique)

Elimination of redundant states:

present state	Next state		o/p	
	x=0	x=1	x=0	x=1
A	B	C	0	0
B	D	E	0	0
C	F	G	0	0
D	H	I	0	0
E	J	K	0	0
F	L	M	0	0
G	N	P	0	0
H	A	A	0	0
I	A	A	0	0
J	A	A	0	1
K	A	A	0	0
L	A	A	0	1
M	A	A	0	0
N	A	A	0	0
P	A	A	0	0

Note: Two states are said to be equivalent, if & only if, they have same next states for possible I/p's, & have same o/p's.

→ The procedure used to find equivalent states in the above state table is known as 'row matching'.

In general row matching is not sufficient to find all equivalent states, except in special case where the circuit resets to the starting state after receiving a fixed no. of I/p's.

→ From state table, H & I are equivalent, because they have the same next states & o/p's for the possible I/p conditions.

similarly for k,m,n,p; & we have to repeatedly check for equivalent states.

present State	Next State		o/p	
	x=0	x=1	x=0	x=1
A	B	C	0	0
B	D	E	0	0
C	F	G	0	0
D	H	~~I~~ H	0	0
E	J	~~K~~ H	0	0
F	L	~~M~~ H	0	0
G	~~N~~ H	~~P~~ H	0	0
H	A	A	0	0
~~I~~	~~A~~	~~A~~	~~0~~	~~0~~
J	A	A	0	[redacted]
~~K~~	~~A~~	~~A~~	~~0~~	~~0~~
L	A	A	0	1
~~M~~	~~A~~	~~A~~	~~0~~	~~0~~
~~N~~	~~A~~	~~A~~	0	0
~~P~~	~~A~~	~~A~~	~~0~~	~~0~~

~~Remove~~ After finding the equivalent states, Remove all the states & replace the removed state if any in the next state of the state table with any of the equivalent state.

For example H & [redacted] I are equivalent, so, remove any one state (I) & replace the I state if any in the ~~Next~~ next states with H.

→ In above table, replace I, K, M, N, P with H.

I, K, M, N, P → H.

The Reduced state table

Present state	Next state		output	
	x=0	x=1	x=0	x=1
A	B	C	0	0
B	D	E	0	0
C	F	~~G~~ D	0	0
D	H	H	0	0
E	J	H	0	0
F	~~K~~ J	H	0	0
~~G~~	~~H~~	~~H~~	~~0~~	~~0~~
H	A	A	0	0
J	A	A	0	1
~~L~~	~~A~~	~~A~~	~~0~~	~~1~~

Again from above table D & G states are equivalent and J and L are equivalent, hence.

Reduced state table now,

Present state	Next state		output	
	x=0	x=1	x=0	x=1
A	B	C	0	0
B	D	E	0	0
C	~~F~~ E	D	0	0
D	H	H	0	0
E	J	H	0	0
~~F~~	~~J~~	~~H~~	~~0~~	~~0~~
H	A	A	0	0
J	A	A	0	1

Now from above 'E' and 'F' states are equivalent hence remove 'F' and replace it by 'E' in state table.

Reduced state table : (final)

Present state	Next state		Output	
	x=0	x=1	x=0	x=1
A	B	C	0	0
B	D	E	0	0
C	E	D	0	0
D	H	H	0	0
E	J	H	0	0
H	A	A	0	0
J	A	A	0	1

STATE ASSIGNMENT

In designing a sequential circuit, it is necessary to assign unique coded binary values to the states.

For a circuit with m states, the code must contain n bits, where $2^n \geq m$ (or $n \geq \log_2 m$). For example, with 3 bits, it is possible to assign codes to 8 states. If only 5 states need binary code assignment, the unused 3 states are treated as "don't care" conditions in the design.

Three possible Binary State Assignment

State	Assignment 1 Binary	Assignment 2 Gray Code	Assignment 3 One-Hot	
a	000	000	00001	
b	001	001	00010	
c	010	011	00100	
d	011	010	01000	
e	100	110	10000	(5 states, 5 bits)

In one-hot assignment, at a given time, only one bit is equal to one while all others are 0. This type of assignment uses one FF per state. One-hot encoding usually leads to simpler decoding logic for the next state and output. One-hot machines can be faster than machines with sequential binary encoding.

The binary form of the state table is used to drive the next-state and output-forming combinational logic part of the sequential circuit. The complexity of the combinational circuit depends on the binary state assignment chosen.

ALGORITHMIC STATE MACHINES

15.1 INTRODUCTION

The binary information stored in a digital system can be classified as either data or control information. Data are discrete elements of information that are manipulated to perform arithmetic, logic, shift, and other similar data processing tasks. These operations are implemented with digital components such as adders, multiplexers, counters and shift registers. Control information provides command signals that supervise the various operations in the data section in order to accomplish the desired data processing tasks. The logic design of a digital system can be divided into two distinct parts. One part is concerned with the design of the digital circuits that perform the data processing operations. The other part is concerned with the design of the control circuits that determine the sequence in which the various actions are performed. Figure 15.1 shows the relationship between the control logic and the data processing in a digital system. The data processing path, commonly referred to as the datapath, manipulates data in registers, according to the system's requirements. The control logic initiates a sequence of commands to the datapath. The control logic uses status conditions from the datapath to serve as decision variables for determining the sequence of control signals.

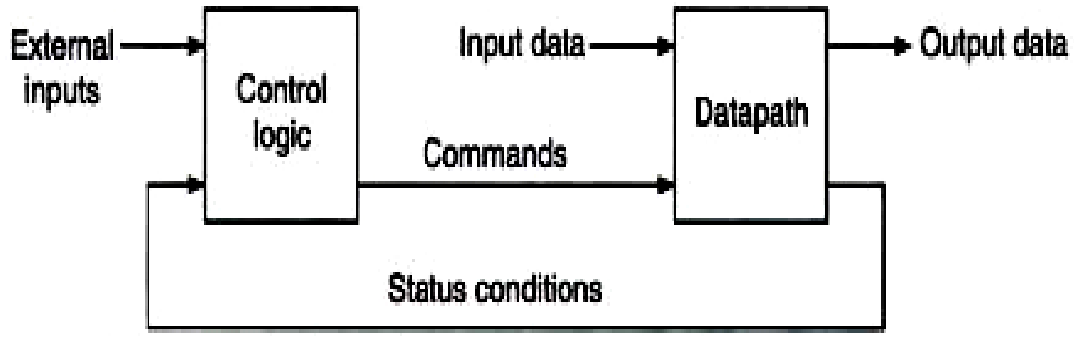


Figure 15.1 Interaction between control logic and datapath.

From the above discussion it is clear that to design a digital system, we have to design two subsystems—datapath subsystem and control subsystem. A datapath subsystem consists of digital circuits which are required to perform specified operations on the data information. A control subsystem is a sequential circuit whose internal states decide the control commands for the subsystem. At any given time, the state of the sequential circuit initiates a prescribed set of commands. The sequential circuit considers the status conditions and the other external inputs to determine the next state to initiate other operations.

The control sequence and datapath tasks of a digital system are represented by means of a hardware algorithm. An algorithm consists of a finite number of procedural steps that specify how to obtain a solution to a problem. A hardware algorithm is a step-by-step procedure to implement the desired tasks with selected circuit components.

A state machine is another term for a sequential circuit which is the basic structure of a digital system. Just as a flow chart serves as a useful aid in writing software programs, algorithmic state machine (ASM) charts help in the hardware design of digital systems. An ASM chart is a special flow chart that has been developed specifically to define digital hardware algorithms. ASM charts provide a conceptually simple and easy to visualize method of implementing algorithms used in hardware design. ASM charts are advantageously used in the design of the control unit of the computer and in general control networks in any digital systems. ASM charts look similar to flow charts but differ in that certain specific rules must be observed in constructing them. An ASM chart is equivalent to a state diagram or a state graph. Normally state diagrams are used for the design of finite state machines. For larger circuits ASM charts are useful. ASM chart is a useful tool which leads directly to hardware realization.

The ASM chart is an alternative for describing the behaviour of finite state machines and is similar to a conventional flow chart used in various engineering designs except for the timing considerations. A conventional flow chart describes the sequence of procedural steps and decision paths for an algorithm without any concern for their time relationship. The ASM chart on the other hand describes the sequence of events as well as the timing relationships between the states of a sequence controller and the events that occur while going from one state to the next state after each clock edge.

An algorithmic state machine (ASM) diagram offers several advantages over state diagrams:

- For larger state diagrams, often are easier to interpret
- May be easily converted to other forms

A key point to remember about ASM charts is that given a state, they do not enumerate all the possible inputs and outputs. Only the inputs that matter and the outputs that are *asserted* are indicated.

COMPONENTS OF ASM CHART:

There are three components of ASM chart:

1. State box,
2. Decision box &
3. Conditional output box.

Out of these, the state box and decision boxes are familiar from use in conventional flow charts. The third element, the conditional box, is unique to the ASM chart.

The ASM Diagram Blocks

An ASM chart has an entry point and is constructed with blocks. A block is constructed with the following type of symbols.

1. **State box:**

A state of a clocked sequential circuit is represented by a rectangle called *state box*. It is equivalent to a node in the state diagram or a row in the state table. The name of the state is written to the left of the box. The binary code assigned to the state is indicated outside on the top right-side of the box. A list of unconditional outputs if any associated with the state are written within the box. They are clearly Moore type outputs. These outputs depend only on the values of the state variables that define the state. Finally a line drawn into the state box indicates the state entry and a line drawn from the state box known as state exit indicates the path to the next state. Figure 15.2 shows a state box.

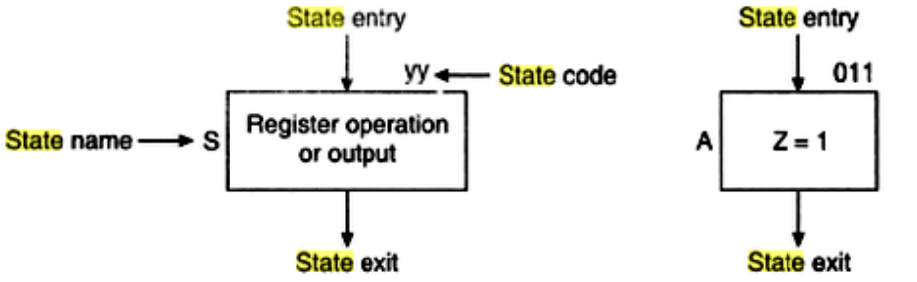
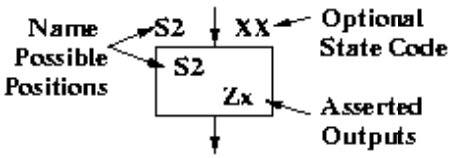


Figure 15.2 State box.

In simple, State box is defined as:

The state box has a name and lists outputs that are asserted when the system is in that state. These outputs are called synchronous or *Moore* type outputs and is shown in figure below.



2. Decision box:

The decision box or condition box is represented by a diamond-shaped symbol with one input and two or more output paths. The output branches are true and false branches. The decision box describes the effect of an input on the control subsystem. A Boolean variable or input or expression written inside the diamond indicates a condition which is evaluated to determine which branch to take. The exit paths lead to the blocks corresponding to the next states of the circuit following the next clock pulse. For example, X written inside this box indicates that the decision is based on the value of X, whereas $\overline{A + B}$ written inside the box indicates that the true path is chosen if $\overline{A + B} = 1$ and the false path is chosen otherwise. Figure 15.3 shows a decision box.

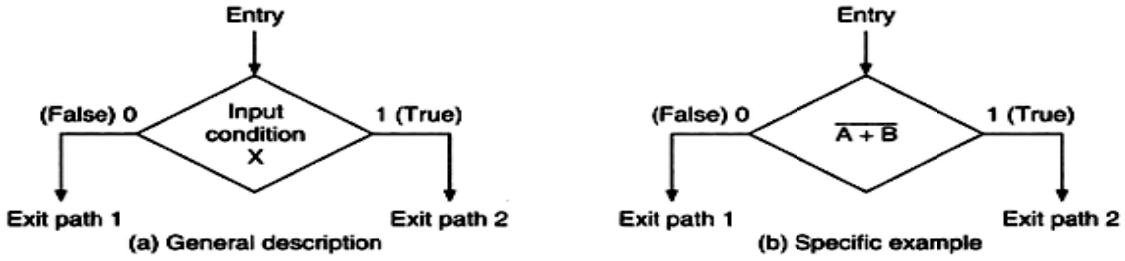
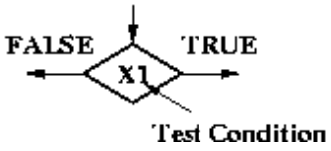


Figure 15.3 Decision box.

In simple, Decision box is defined as:

A decision box may be conditioned on a signal or a test of some kind.



3. Conditional output box:

The conditional output box is represented by a rectangle with rounded corners or by an oval with one input line and one output line. The outputs that depend on both the state of the system and the inputs are indicated inside the box. These are Mealy type outputs. Figure 15.4 shows a conditional output box.

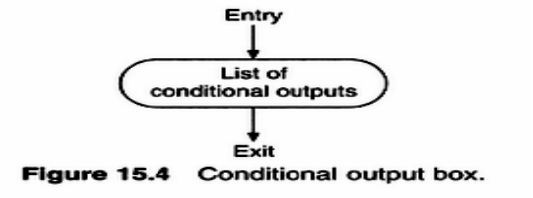


Figure 15.4 Conditional output box.

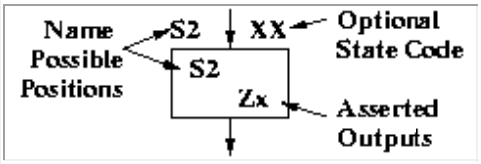
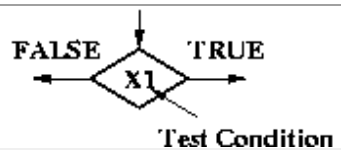
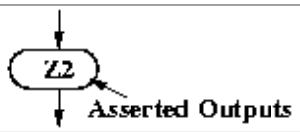
In simple, Conditional output box is defined as:

Such an output box indicates outputs that are conditionally asserted as shown in figure below. These outputs are called asynchronous or *Mealy* outputs.

ASM block: An ASM chart is constructed from one or more interconnected ASM blocks. Each ASM block contains a single state box and any decision and conditional output boxes associated with that state. An ASM block has a single entry path and single or multiple exit paths represented by the structure of the decision boxes. Each ASM block is associated with one specific state; therefore, it describes the finite state machine operation during the time the machine is in that state, i.e. it describes the state of the system under one clock pulse interval.

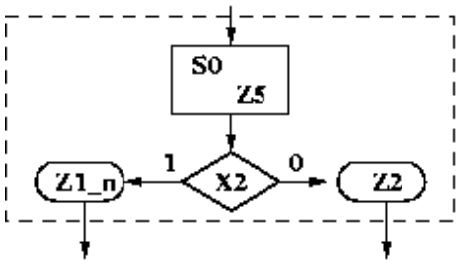
When a synchronous sequential system enters the state included in a given ASM block, the outputs in the output list in the state box are asserted (become true). The conditions in the decision boxes are evaluated to determine which path(s) is followed through the ASM block. When a conditional output box is encountered in such a path, the corresponding conditional outputs become true (asserted). A path through an ASM block from entry to exit is referred to as a link path.

SUMMARY FOR ASM BLOCKS:

	**One state box:** The state box has a name and lists outputs that are asserted when the system is in that state. These outputs are called synchronous or *Moore* type outputs.
	**Optional decision box (es):** A decision box may be conditioned on a signal or a test of some kind.
	**Optional conditional output box (es)** Such an output box indicates outputs that are conditionally asserted. These outputs are called asynchronous or *Mealy* outputs.

There is no rule saying that outputs are exclusively inside an a conditional output box or in a state box. An output written inside a state box is simply independent of the input, while in that state.

The idea is that *flow* passes from ASM block to ASM block, the decision boxes decide the next state and conditional output. Consider the following example of an ASM diagram block. When state S0 is entered, output Z5 is always asserted. Z1_n however is asserted only if X2 is also *high*. Otherwise Z2 is asserted.

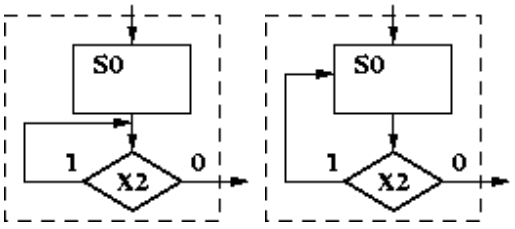


An ASM block

Certain Rules

The drawing of ASM charts must follow certain necessary rules:

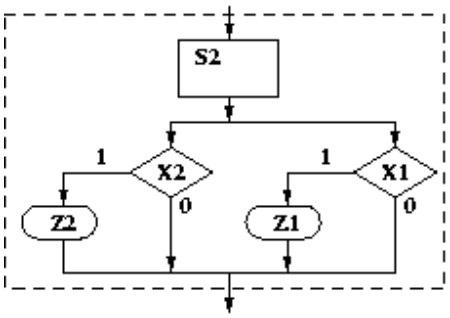
- The entrance paths to an ASM block lead to only one state box
- Of 'N' possible exit paths, for each possible valid input combination, only one exit path can be followed, that is there is only one valid next state.
- No feedback internal to a state box is allowed. The following diagram indicates valid and invalid cases.



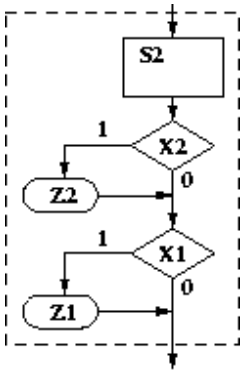
Incorrect **Correct**

Parallel vs. Serial

We can bend the rules, several internal paths can be active, provided that they lead to a single exit path. Regardless of parallel or serial form, all tests are performed concurrently. Usually we have a preference for the serial form. The following two examples are equivalent.



Parallel Form



Serial Form

15.3 SALIENT FEATURES OF ASM CHARTS

The following are the salient features of ASM charts:

1. An ASM chart describes the sequence of events as well as the timing relationship between the states of a sequential controller and the events that occur while going from one state to the next.
2. An ASM chart contains one or more interconnected ASM blocks.
3. Each ASM block contains exactly one state box together with the decision boxes and conditional output boxes associated with that state.
4. Every block in an ASM chart specifies the operations that are to be performed during one common clock pulse.
5. An ASM block has exactly one entrance path and one or more exit paths represented by the structure of the decision boxes.
6. A path through an ASM block from entrance to exit is referred to as a link path.
7. The operations specified within the state and conditional output boxes in the block are performed in the datapath subsystem.
8. Internal feedback within an ASM block is not permitted. Even so, following a decision box or conditional output boxes, the machine may reenter the same state.
9. Each block in the ASM chart describes the state of the system during one clock pulse interval. When a digital system enters the state associated with a given ASM block, the outputs indicated within the state box become true. The conditions associated with the decision boxes are evaluated to determine which path or paths to be followed to enter the next ASM block.

15.4 INTRODUCTORY EXAMPLES OF ASM CHARTS

Example 1 Mod-6 counter: The state diagram of a mod-6 counter is shown in Figure 15.5a. ASM chart equivalent of Figure 15.5a is shown in Figure 15.5b. The states are now represented by state boxes instead of nodes. State transitions are still represented by arrows but the inputs indicated adjacent to arrows in the state diagram are replaced by decision boxes with true and false branches. In the state diagram the outputs are indicated along with inputs using a separator slash or comma.

In the ASM chart Mealy type outputs are now indicated in conditional output boxes and Moore type outputs are indicated within the state box itself.

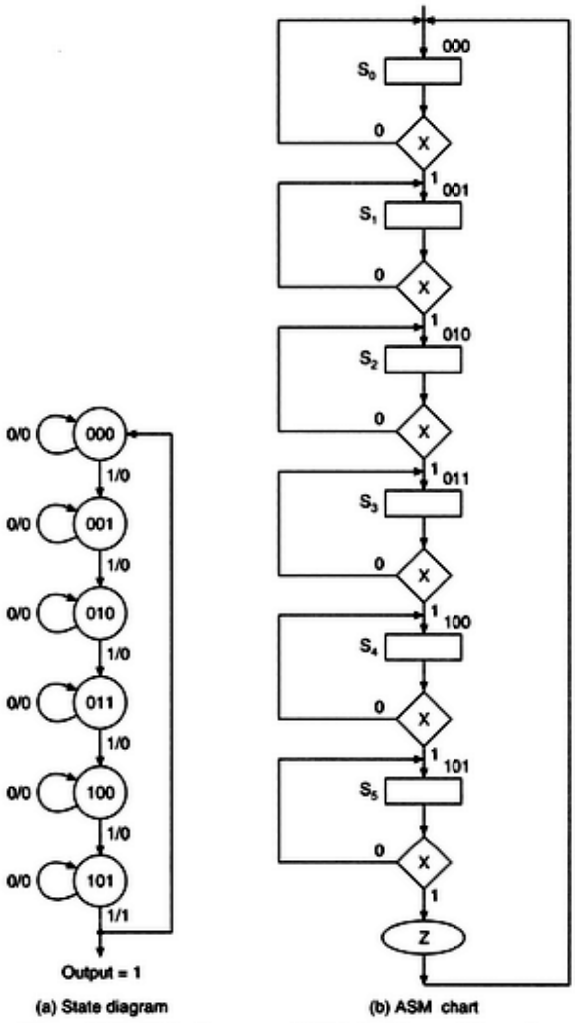


Figure 15.5 State diagram and ASM chart for mod-6 counter.

Look at the ASM chart which has six state boxes named S_0 through S_5. For every clock pulse, X is sensed. If 1, then, the machine goes to the next state: if 0, then, the machine remains in the

Example 3 Serial adder: We discussed earlier that a serial adder adds serial binary numbers. The state diagram of the serial adder is shown in Figure 15.7a. The structure of the serial adder is shown in Figure 15.7b. The ASM chart equivalent to the state diagram describing the same behaviour is shown in Figure 15.8. In the ASM chart observe that there are four exit paths from each state depending on the values of inputs A and B arranged in the form of a binary tree. Also observe that all the paths merge into one entry path for each state. Comparing it with the state diagram, we observe that in the state diagram from each state there is an exit path to the other state and three re-entry paths to itself. This is because there are four different values that AB can assume 00, 01, 10 and 11 and hence there must be four exit paths—three of them happen to be re-entry paths in this example. Observe in the state diagram that there are four incoming paths which merge into one entry path to each state. Correspondingly, there are as many branches in the ASM chart. If the number of inputs is large, the state graph will become quite complex and the ASM chart will become unwieldy. The decision boxes are usually binary as we aim at an algorithmic procedure comprising primitive steps. As we have two inputs we have to show four exit paths from each state.

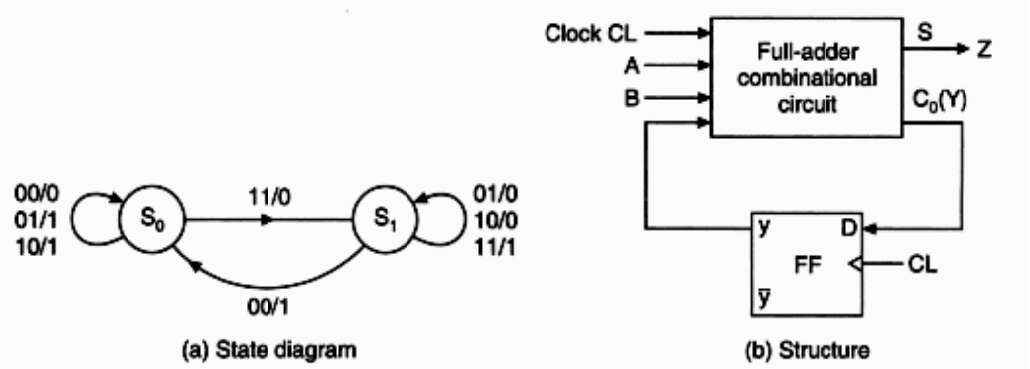


Figure 15.7 State diagram and structure of a serial adder.

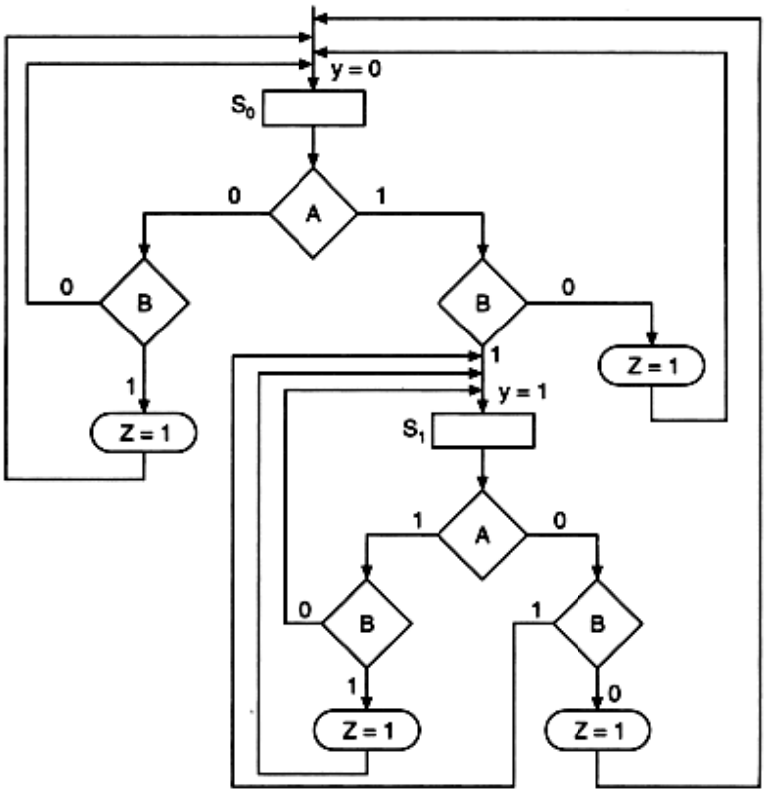


Figure 15.8 ASM chart of serial adder.

Synthesis of the circuit: By inspection of the ASM chart, we note that there are only two states S_0 and S_1 and hence we need to have only one flip-flop. We choose a D flip-flop, because D is easily synthesized looking at the entry paths into each state. For the only flip-flop, D becomes 1 for state S_1. There are four arrows confluent on entry into S_1. Hence, by inspection and tracing the paths indicated by arrows, we write the expression for D given below. Remember that y is the present state variable of the only flip-flop.

$$\begin{aligned} D &= \bar{y}AB + yA\bar{B} + yAB + y\bar{A}B \\ &= AB(y + \bar{y}) + Ay(B + \bar{B}) + By(A + \bar{A}) \\ &= AB + Ay + By \end{aligned}$$

By a similar reasoning, the state S_0 is characterized by $y = 0$. The corresponding excitation is $\bar{D}$. There are four paths confluent on the entry into S_0. By inspection and tracing the paths, we may write

$$\begin{aligned} \bar{D} &= y\bar{A}\bar{B} + \bar{y}\bar{A}\bar{B} + \bar{y}\bar{A}B + \bar{y}A\bar{B} \\ &= \bar{A}\bar{B}(y + \bar{y}) + \bar{A}\,\bar{y}(B + \bar{B}) + \bar{B}\,\bar{y}(A + \bar{A}) \\ &= \bar{A}\bar{B} + \bar{A}\,\bar{y} + \bar{B}\,\bar{y} \end{aligned}$$

See the consistency in the expressions for D and $\bar{D}$. There are four conditional output boxes.

Remember that the output Z is the sum bit to be produced serially. Tracing the corresponding paths it is easy to write the expression for Z.

$$\begin{aligned} Z &= \bar{y}\bar{A}B + \bar{y}A\bar{B} + y\bar{A}\bar{B} + y\,AB \\ &= \bar{y}(\bar{A}B + A\bar{B}) + y(\bar{A}\bar{B} + AB) \\ &= \bar{y}(A \oplus B) + y(A \odot B) \\ &= A \oplus B \oplus y \end{aligned}$$

Notice that the expressions for Z and D are the same as those for sum S and output carry C_0 of a full-adder.

Control subsystem implementation: The control subsystem consists of a sequential circuit. The process to implement the sequential circuit described by ASM chart consists of the following steps:

1. Translate the ASM chart to a state table.
2. Assign the states and convert the state table to a transition and output table.
3. Select the type of flip-flops and obtain the excitation table.
4. Obtain the minimal expressions for circuit outputs and memory element inputs in terms of the present state variables and external inputs.
5. Draw a logic diagram based on those minimal expressions.

The ASM chart of the waveform generator can be expressed as a state table as shown in Figure 15.10b. In the state table, the outputs along with next states are listed in the columns corresponding to each combination of input values and the states are listed in the rows.

In the state table there are three don't cares. When the machine is in state C, for input $X_1X_2 = 11$, the next state and output are don't cares because that waveform is repeated after two clock cycles. Also when the machine is in state D, the next state and output entries for $X_1X_2 = 10$ and 11 are don't cares because these waveforms are repeated after the third clock pulse.

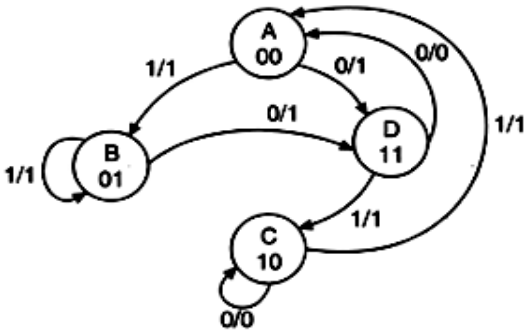
EXAMPLE: Draw the state diagram and ASM chart for the state table given below.

Solution

It is easier to draw the state diagram and the ASM chart from the state table. There are two flip-flops, so two state variables and four states will be there. Let the states be A(00), B(01), C(10) and D(11). The state table for the given circuit is shown in Figure 15.37a. The state diagram based on the state table is shown in Figure 15.37b.

PS		I/P	NS		O/P
y_1	y_0	X	y_1	y_0	Z
0	0	0	1	1	1
0	0	1	0	1	1
0	1	0	1	1	1
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	0	1
1	1	0	0	0	0
1	1	1	1	0	1

(a) State table



(b) State diagram

Figure 15.37 Example 15.9.

From the state diagram, the ASM chart can be drawn as shown in Figure 15.38.

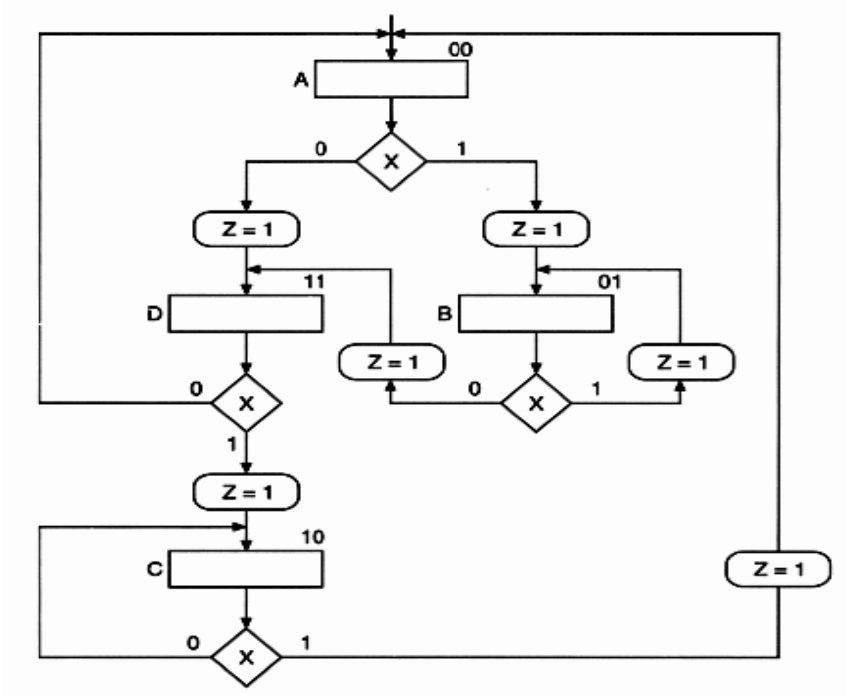
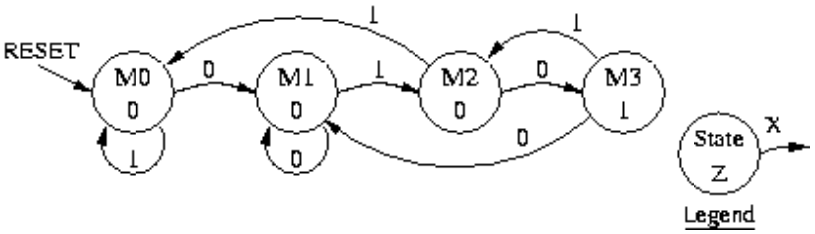


Figure 15.38 Example 15.9: ASM chart.

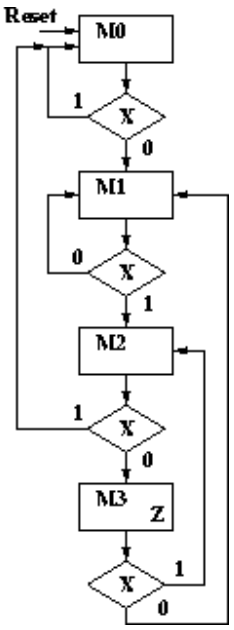
Sequence Detector Example

The use of ASM charts is a trade-off. While the mechanics of ASM charts do reduce clutter in significant designs, its better to use an ordinary state diagrams for simple state machines. Here is an example Moore type state machine with input X and output Z. Once the flag sequence is received, the output is asserted for one clock cycle.

The corresponding ASM chart is to the right. Note that unlike the state diagram which illustrates the output value for each arc, the ASM chart indicates when the output Z only when it is asserted.

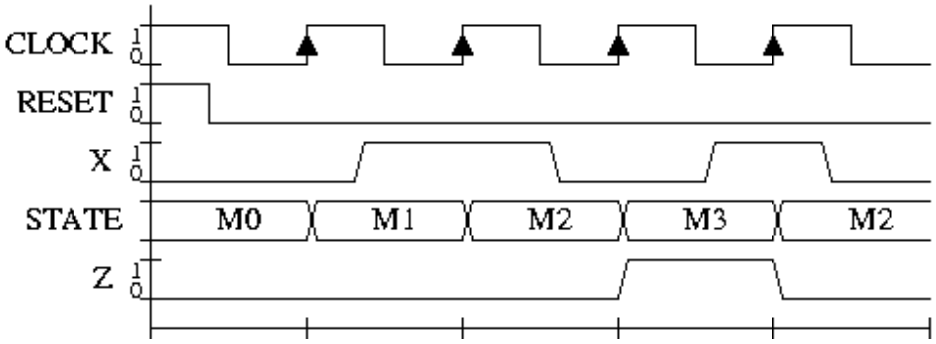


State diagram for sequence detector



ASM chart

The following timing diagram illustrates the detection of the desired sequence. Here it is assumed that the state is updated with a *rising clock edge*. The key concept to observe is that regardless of the input, the output can only be asserted for one entire clock cycle.



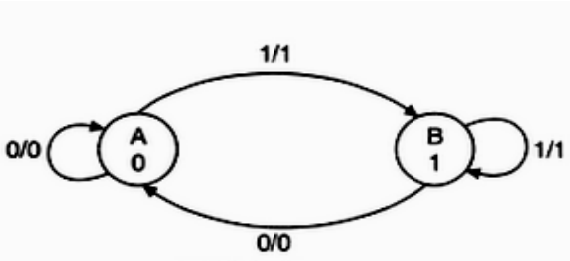
Timing diagram

EXAMPLE 15.10 Draw the state diagram, state table and ASM chart for a D flip-flop.

Solution

A D flip-flop has two states A(0) and B(1). In one state (A), it stores a 0 and in the other state (B) it stores a 1. Its output is same as the present state which is equal to the input after time delay. While in state 0, it may receive the D input as 0 or 1. If the input D = 0, the flip-flop will remain in the same state and outputs a 0. If the input D = 1, the flip-flop will go to state 1 and outputs a 1. While in state 1, it may again receive the input as 0 or 1. If it is 0, it goes to state 0 and outputs a 0. If it is 1, it remains in the same state and outputs a 1.

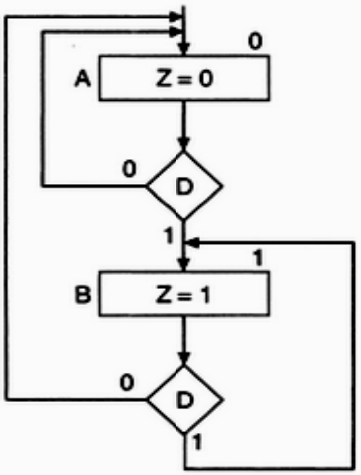
The state diagram, and the state table of the D flip-flop are shown in Figures 15.39a and b respectively. The ASM chart of the D flip-flop is shown in Figure 15.39c. The circuit will have two state boxes for states 0 and 1 and two decision boxes controlled by D input.



(a) State diagram

PS	NS, O/P Input D	
	D = 0	D = 1
A	A, 0	B, 1
B	A, 0	B, 1

(b) State table



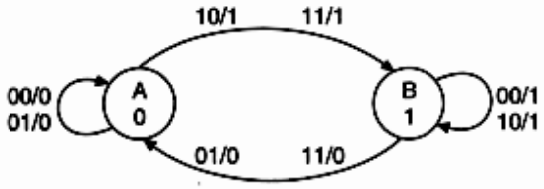
(c) ASM chart

Figure 15.39 Example 15.10.

EXAMPLE 15.11 Draw the state diagram, state table and ASM chart for a J-K flip-flop.

Solution

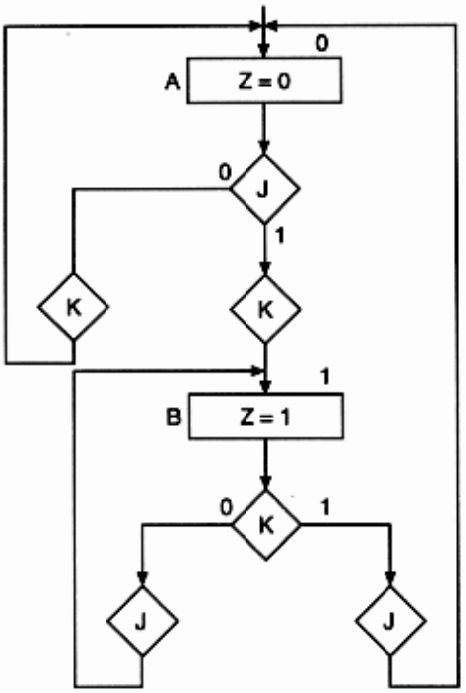
We know that the J-K flip-flop has two states A(0) and B(1). While at A, if J-K = 00 or J-K = 01, it remains in state A itself and outputs a 0. If J-K = 10 or J-K = 11, it goes to state B and outputs a 1. While at B, if J-K = 00 or J-K = 10, it remains in state B itself and outputs a 1. If J-K = 01 or J-K = 11, it goes to state A and outputs a 0. The state diagram and the state table of a J-K flip-flop are shown in Figures 15.40a and b respectively. The ASM chart of the J-K flip-flop is shown in Figure 15.40c.



(a) State diagram

PS	NS, O/P Input J-K			
	00	01	10	11
A	A, 0	A, 0	B, 1	B, 1
B	B, 1	A, 0	B, 1	A, 0

(b) State table



(c) ASM chart

Figure 15.40 Example 15.11.

EXAMPLE 15.17 Obtain the ASM charts for the following state transitions:

(a) If x = 0, control goes from state T_1 to state T_2. If x = 1, generate the conditional operation and go from T_1 to T_2.

(b) If x = 1, control goes from T_1 to T_2 and then to T_3. If x = 0, control goes from T_1 to T_3.

Solution

The ASM charts for the given state transitions are shown in Figure 15.54.

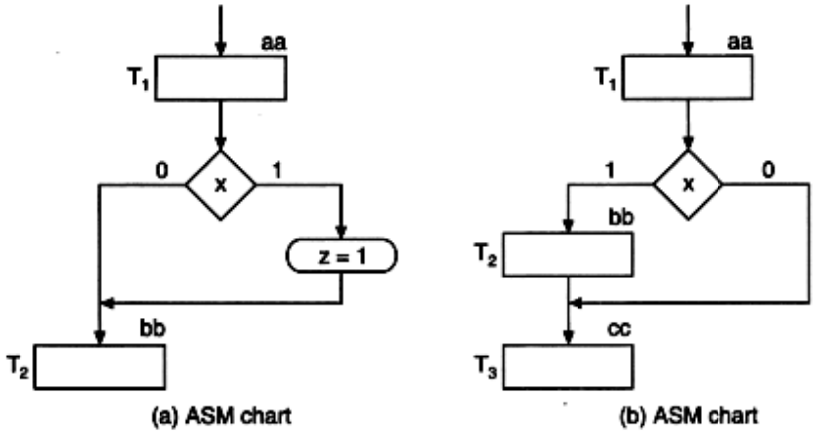


Figure 15.54 Example 15.17: ASM charts.

EXAMPLE 15.18 (a) Draw the state diagram and the state table of the control unit for conditions given below.

(b) Draw the equivalent ASM chart leaving the state box empty.

(c) Design the control unit with the multiplexers for the above problem.

(d) Design the control unit using D flip-flops and a decoder.

(i) From 00 state, if x = 1, it goes to 01 state and if x = 0, it remains in the same state 00.

(ii) From 01 state, if y = 1, it goes to 11 state and if y = 0, it goes to 10 state.

(iii) From 10 state, if x = 1 and y = 0, it remains in the same state 10 and if x = 1 and y = 1, it goes to 11 state, and if x = 0, it goes to 00 state.

(iv) From 11 state, if x = 1, y = 0, it goes to 10 state and if x = 1 and y = 1, it remains in the same state, and if x = 0, it goes to 00 state.

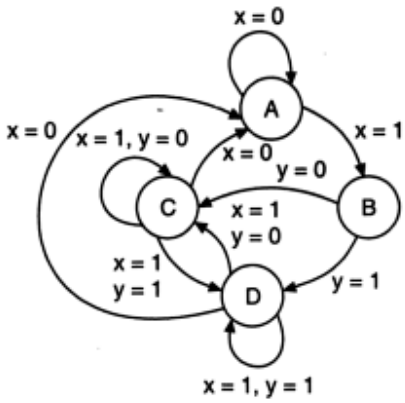
Solution

(a) The state diagram and the state table for the given conditions are shown in Figures 15.55a and b respectively.

(b) The equivalent ASM chart of the control unit leaving the state box empty is shown in Figure 15.56. Let the state assignment be A = 00, B = 01, C = 10, D = 11.

(c) The design of control unit using multiplexers is as follows. From the ASM chart write the transition table. Looking at the conditions of transition in the transition table, write the inputs of the multiplexers as shown in the inputs for multiplexers table. The control circuit using multiplexers is drawn as shown in Figure 15.57.

(d) To design the control unit using D flip-flops and decoder write the state table as shown in Table 15.18.



(a) State diagram

PS	NS Input x y			
	00	01	10	11
A → 00	A	A	B	B
B → 01	C	D	C	D
C → 10	A	A	C	D
D → 11	A	A	C	D

(b) State table

Figure 15.55 Example 15.18.

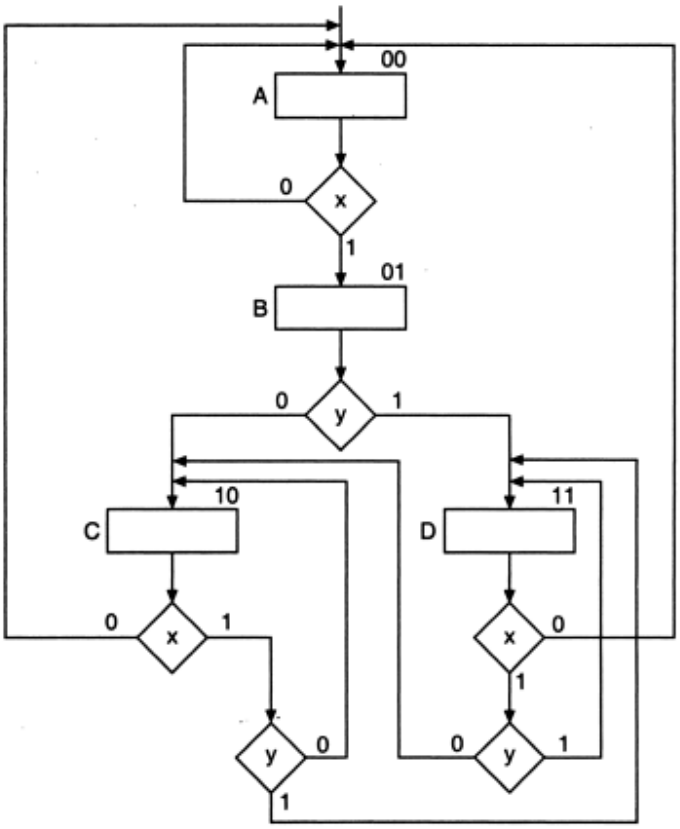


Figure 15.56 Example 15.18: ASM chart.