

8.7 ASYNCHRONOUS SEQUENTIAL CIRCUITS

The two types of sequential circuits were introduced in Section 7.1. The design of clocked sequential circuits have been discussed in section 8.6. Another important class of sequential circuits, i.e. asynchronous sequential circuits have been discussed here.

8.7.1 Asynchronous versus Synchronous Sequential Circuits

- In a clocked sequential circuit a change of state occurs only in response to a synchronizing clock pulse. All the FLIP-FLOPs are clocked simultaneously by a common clock pulse. In an asynchronous sequential circuit, the state of the circuit can change immediately when an input change occurs. It does not use a clock.
- In clocked sequential circuits input changes are assumed to occur between clock pulses. The circuit must be in the stable state before next clock pulse arrives.

In asynchronous sequential circuits input changes should occur only when the circuit is in a stable state.

- In clocked sequential circuits, the speed of operation depends on the maximum allowed clock frequency.
Asynchronous sequential circuits do not require clock pulses and they can change state with the input change. Therefore, in general the asynchronous sequential circuits are faster than the synchronous sequential circuits.
- In clocked sequential circuits, the memory elements are clocked FLIP-FLOPs.
In asynchronous sequential circuits, the memory elements are either unclocked FLIP-FLOPs (latches) or gate circuits with feedback producing the effect of latch operation.
- In clocked sequential circuits, any number of inputs can change simultaneously (during the absence of the clock).

In asynchronous sequential circuits only one input is allowed to change at a time in the case of the level inputs and only one pulse input is allowed to be present in the case of the pulse inputs. If more than one level inputs change simultaneously or more than one pulse input is present, the circuit makes erroneous state transitions due to different delay paths for each input variable.

8.7.2 Applications of Asynchronous Sequential Circuits

- An asynchronous circuit is preferred over synchronous circuit when high speed of operation is required since asynchronous sequential circuits respond immediately whenever there is a change in any input variable without having to wait for a clock pulse.
- Asynchronous sequential circuits are useful in applications in which input signals may change at any time.
- Asynchronous sequential circuits cost less than the sequential circuits, therefore, for economical reasons, they find useful applications.

8.7.3 Asynchronous Sequential Machine Modes

There are two modes of operations of asynchronous sequential machines depending upon the type of input signals. These are:

- Fundamental Mode
- Pulse Mode

Fundamental Mode

In a fundamental mode circuit, all of the input signals are considered to be levels. Fundamental mode operation assumes that the input signals will be changed only when the circuit is in a stable state and that only one variable can change at a given time.

Pulse Mode

In pulse mode circuits, the inputs are pulses rather than levels. In this mode of operation the width of the input pulses is critical to the circuit operation. The input pulse must be long enough for the circuit to respond to the input but it must not be so long as to be present even after new state is reached. In such a situation the state of the circuit may make another transition.

The minimum pulse width requirement is based on the propagation delay through the next-state logic (Fig 7.1). The maximum pulse width is determined by the total propagation delay through the next state logic and the memory elements.

Both fundamental and pulse mode asynchronous sequential circuits use unclocked S-R FLIP-FLOPs or latches.

In pulse-mode operation, only one input is allowed to have pulse present at any time. This means that when pulse occurs on any one input, while the circuit is in stable state, pulse must not arrive at any other input. Figure 8.40 illustrates unacceptable and acceptable input pulse change. X_1 and X_2 are the two inputs to a pulse mode circuit. In Fig. 8.64a at time t_3 pulse at input X_2 arrives.

While this pulse is still present, another pulse at X_1 input arrives at t_4 . Therefore, this kind of the presence of pulse inputs is not allowed.

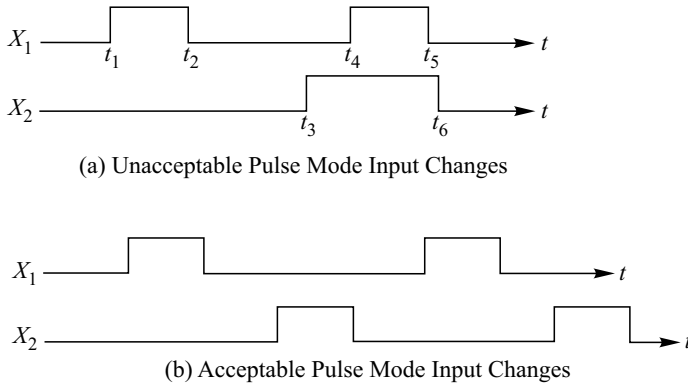


Fig. 8.64

8.7.4 Analysis of Asynchronous Sequential Machines

Analysis of asynchronous sequential circuits operation in fundamental mode and pulse mode will help in clearly understanding the asynchronous sequential circuits.

Fundamental Mode Circuits

Fundamental mode circuits are of two types:

- Circuits without latches
- Circuits with latches

Circuits Without Latches Consider a fundamental mode circuit shown in Fig. 8.65. This circuit has only gates and no explicit memory elements are present. There are two feedback paths from Q_1 and Q_2 to the next-state logic circuit. This feedback creates the latching effect due to delays, necessary to produce a sequential circuit. It may be noted that a memory element latch is created due to feedback in gate circuit (Fig. 7.4).

The first step in the analysis is to identify the *states* and the *state variables*. The combination of level signals from external sources X_1, X_2 is referred to as the *input state* and X_1, X_2 are the *input state variables*.

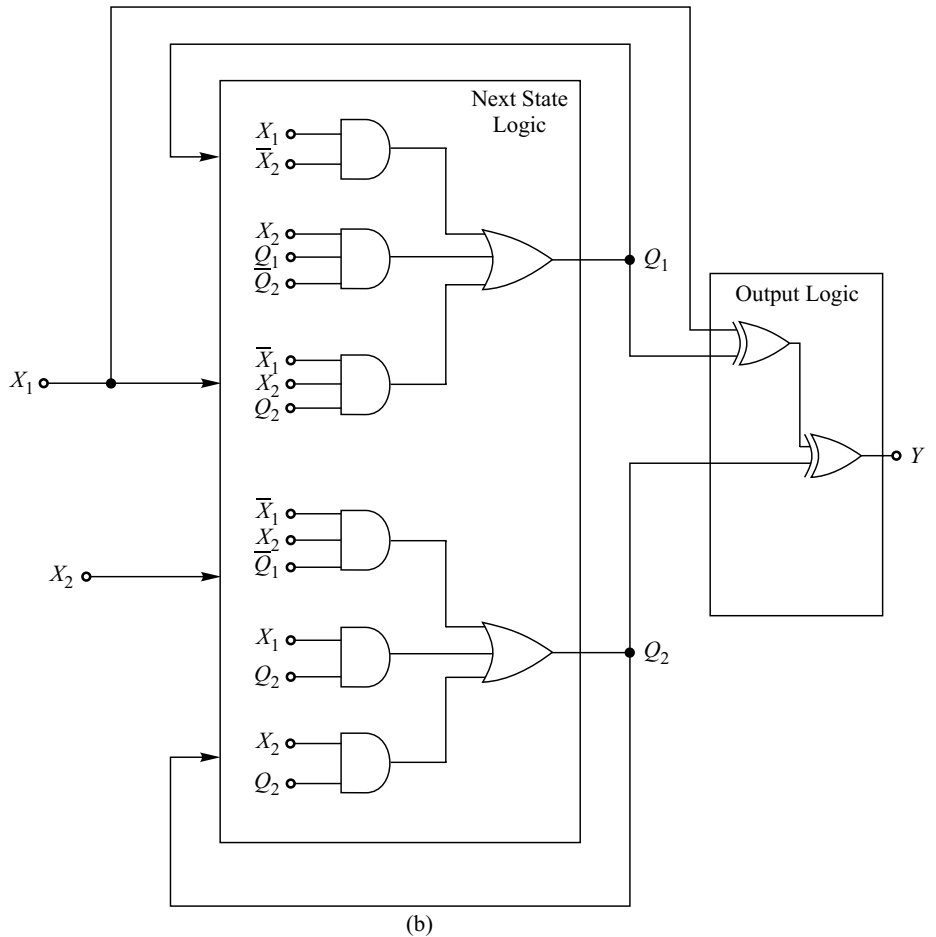
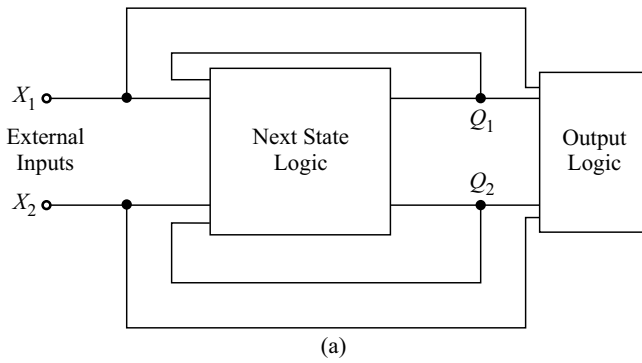


Fig. 8.65

Fundamental Mode Asynchronous Sequential Circuit Without Latch
(a) Block Diagram (b) Circuit Diagram

The combination of the outputs of memory elements are known as *secondary*, or *internal states* and these variables are known as *internal* or *secondary state variables*. Here, Q_1 and Q_2 are the internal variables since no explicit elements are present. The combination of both, input state and the secondary state (Q_1, Q_2, X_1, X_2) is known as the *total state*. Y is the output variable.

The next secondary state and output logic equations are derived from the logic circuit in the next-state logic block. The next secondary state variables are denoted by Q_1^+ and Q_2^+ these are given by

$$Q_1^+ = X_1 \bar{X}_2 + \bar{X}_1 X_2 Q_2 + X_2 Q_1 \bar{Q}_2 \quad (8.6)$$

$$Q_2^+ = \bar{X}_1 X_2 \bar{Q}_1 + X_1 Q_2 + X_2 Q_2 \quad (8.7)$$

$$Y = X_1 \oplus Q_1 \oplus Q_2 \quad (8.8)$$

Here, Q_1 and Q_2 are the present secondary state variables when $X_1 X_2$ input-state variables occur, the circuit goes to next secondary state. A state table shown in Table 8.28 is constructed using these logic equations. If the resulting next secondary state is same as the present state, i.e. $Q_1^+ = Q_1$ and $Q_2^+ = Q_2$, the total state $Q_1 Q_2 X_1 X_2$ is said to be *stable*. Otherwise it is unstable. The stability of the next total state is also shown in Table 8.28.

Transition Table A state table can be represented in another form known as *transition table*. The transition table for the state table of Table 8.28 is shown in Fig. 8.66.

In a transition table, columns represent input states (one column for each input state) and rows represent secondary states (one row for each secondary state). The next secondary state values are written into the squares, each indicating a total state. The stable states are circled. For any given present secondary state ($Q_1 Q_2$), the next secondary state is located in the square corresponding to row for the present secondary state and the column for the input state ($X_1 X_2$). For example, for $Q_1 Q_2 = 11$ and $X_1 X_2 = 00$, the next secondary state is 00 (third row, first column) which is an unstable state.

Table 8.28 *State Table*

Present total state				Next total state				Stable total state	Output
Q_1	Q_2	X_1	X_2	Q_1^+	Q_2^+	X_1	X_2	Yes/No	Y
0	0	0	0	0	0	0	0	Yes	0
0	0	0	1	0	1	0	1	No	0
0	0	1	1	0	0	1	1	Yes	1
0	0	1	0	1	0	1	0	No	1
0	1	0	0	0	0	0	0	No	1
0	1	0	1	1	1	0	1	No	1
0	1	1	1	0	1	1	1	Yes	0
0	1	1	0	1	1	1	0	No	0
1	1	0	0	0	0	0	0	No	0
1	1	0	1	1	1	0	1	Yes	0
1	1	1	1	0	1	1	1	No	1
1	1	1	0	1	1	1	0	Yes	1
1	0	0	0	0	0	0	0	No	1
1	0	0	1	1	0	0	1	Yes	1
1	0	1	1	1	0	1	1	Yes	0
1	0	1	0	1	0	1	0	Yes	0

Present internal state $Q_1 Q_2$	Input state $X_1 X_2$				Next state $Q_1^+ Q_2^+$
	00	01	11	10	
00	00	01	00	10	
01	00	11	01	11	
11	00	11	01	11	
10	00	10	10	10	

Fig. 8.66 Transition Table for Table 8.28

For a given input sequence, the total state sequence can be determined from the transition table.

Example 8.28

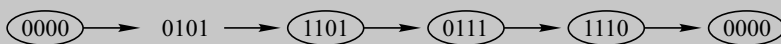
For the transition table shown in Fig 8.66, the initial total state is $Q_1 Q_2 X_1 X_2 = 0000$. Find the total state sequence for an input sequence $X_1 X_2 = 00, 01, 11, 10, 00$.

Solution

For a given internal state of the circuit, a change in the value of the circuit input causes a *horizontal* move in the transition table to the column corresponding to the new input value. A change in the internal state of the circuit is reflected by a *vertical* move. Since a change in the input can occur only when the circuit is in a stable state, a horizontal move can emanate only from a circled entry.

The initial total state is 0000 (first row, first column) which is a stable state. When the input state changes from 00 to 01, the circuit makes a transition (horizontal move) from the present total state to the next total state 0101 (first row, second column) which is unstable. Next, the circuit makes another transition from 0101 to 1101 (vertical move) (second row, second column) which is also an unstable state. Finally in the next transition (vertical move) it comes to stable state 1101 (third row, second column). All these transitions are indicated by arrows. Thus we see that a single input change produces two secondary state changes before a stable total state is reached. If the input is next changed to 11 the circuit goes to total state 0111 (horizontal move) which is unstable and then to stable total state 0111 (vertical move). Similarly, the next input change to 10 will take the circuit to unstable total state 1110 (horizontal move) and finally to stable total state 1110 (vertical move). A change in input state from 10 to 00 causes a transition to unstable total state 0000 (horizontal move) and then to stable total state 0000 (vertical move), completing the state transitions for the input sequence. All the state transitions are indicated by arrows.

The total state sequence is



From the above discussions, we see that from the logic diagram of an asynchronous sequential circuit, logic equations, state table, and transition table can be determined. Similarly, from the transition table, logic equations can be written and the logic circuit can be designed.

From a flow table, transition table can be constructed by assigning binary values to each state and from the transition table logic circuit can be designed by constructing K -maps for Q_1^+ and Q_2^+ .

$$\begin{aligned} Q^+ &= \overline{R + Q} = \overline{R + (S + Q)} \\ &= \overline{R} \cdot (S + Q) \\ &= S\overline{R} + \overline{R}Q \end{aligned}$$

Since, $S = R = 1$ is not allowed, which means $SR = 0$, therefore,

$$S\bar{R} = S\bar{R} + SR = S(\bar{R} + R) = S$$

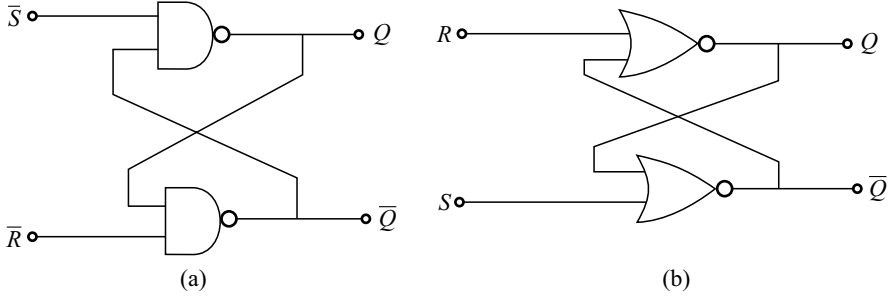


Fig. 8.68 (a) $\bar{S}$ - $\bar{R}$ Latch Using **NAND** Gates (b) S - R Latch Using **NOR** Gates

which gives,

$$Q^+ = S + \bar{R}Q$$

It is same as Eq. (8.9)

The transition table of S - R latch is shown in Fig. 8.69.

$\begin{matrix} SR \\ Q \end{matrix}$		00	01	11	10
Q	0	①	①	①	1
	1	①	0	0	①

$$Q^+ = \bar{S}\bar{R} + \bar{R}Q = S + \bar{R}Q$$

Fig. 8.69 Transition Table of S - R Latch

From the transition table of S - R FLIP-FLOP, we observe that when SR changes from 11 to 00 the circuit will attain either the stable state ① (first row, first column) or ① (second row, first column) depending upon whether S goes to 0 first or R goes to 0 first respectively. Therefore, $S = R = 1$ must not be applied.

Consider an asynchronous sequential circuit with latches shown in Fig. 8.70. For FF-1, $R_1 = 0$ and the excitation equation for S_1 is

$$S_1 = X_1 \bar{X}_2 \bar{Q}_2 + \bar{X}_1 Q_2 \quad (8.10)$$

The next-state equation is

$$Q_1^+ = S_1 + \bar{R}_1 Q_1 \quad (8.11)$$

Substituting the value of S_1 from Eq. (8.10) in Eq. (8.11) we obtain,

$$Q_1^+ = X_1 \bar{X}_2 \bar{Q}_2 + \bar{X}_1 Q_2 + Q_1 \quad (8.12)$$

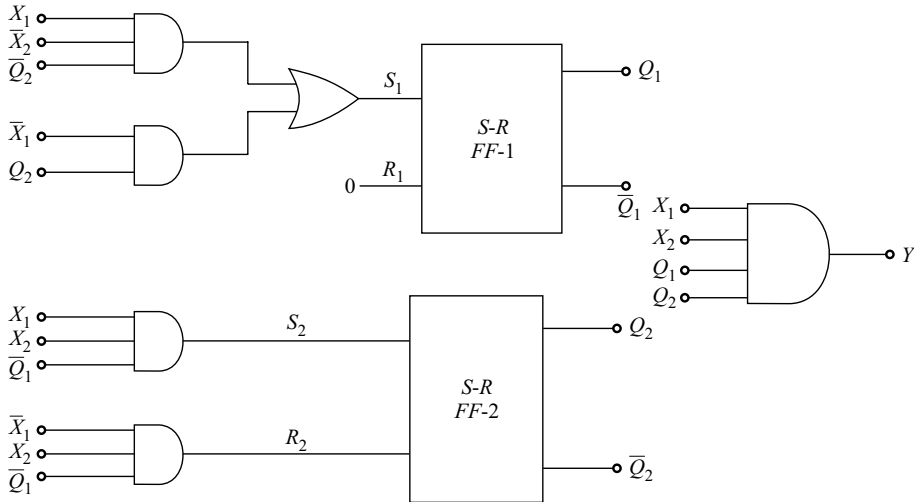


Fig. 8.70 Asynchronous Sequential Circuit with Latches

Similarly, the excitation equations for FF-2 are

$$S_2 = X_1 X_2 \bar{Q}_1, R_2 = \bar{X}_1 X_2 \bar{Q}_1 \quad (8.13)$$

The next-state equation is

$$\begin{aligned} Q_2^+ &= S_2 + \bar{R}_2 Q_2 \\ &= X_1 X_2 \bar{Q}_1 + \overline{\bar{X}_1 X_2 \bar{Q}_1} \cdot Q_2 \end{aligned} \quad (8.14)$$

Using Eqs. (8.12) and (8.14), transition table is obtained as shown in Fig. 8.71.

The output function is

$$Y = X_1 X_2 Q_1 Q_2 \quad (8.15)$$

$X_1 X_2$		$Q_1^+ Q_2^+$			
		00	01	11	10
$Q_1 Q_2$	00	⓪⓪	⓪⓪	10	10
	01	11	11	⓪1	⓪1
	11	⓪1	10	⓪1	⓪1
	10	⓪0	⓪0	⓪0	⓪0

Fig. 8.71 Transition Table for the Circuit of Fig. 8.70

Its flow table is shown in Fig. 8.72.

$X_1 X_2$		$Q_1^+ Q_2^+$			
		00	01	11	10
$Q_1 Q_2$	a	$\textcircled{a}, 0$	$\textcircled{a}, 0$	$b, 0$	$d, 0$
	b	$c, 0$	$c, 0$	$\textcircled{b}, 0$	$\textcircled{b}, 0$
	c	$\textcircled{c}, 0$	$d, 0$	$\textcircled{c}, 1$	$\textcircled{c}, 0$
	d	$\textcircled{d}, 0$	$\textcircled{d}, 0$	$\textcircled{d}, 0$	$\textcircled{d}, 0$

Fig. 8.72 Flow Table for the Circuit of Fig. 8.70

From a flow table, transition table can be obtained by assigning binary values to the states. From the transition table, logic equations can be obtained by constructing K -maps for S and R inputs of every latch. For this, the excitation table of S - R latch will be used. Logic circuit can then be designed using the logic equation for S , R inputs of every latch.

Example 8.29

Design logic circuit using S - R latches for the transition table of Fig. 8.66.

Solution

Since, there are two internal states Q_1 and Q_2 , therefore, two S - R latches are required for the design of logic circuit. Let the two latches be L_1 and L_2 . The inputs and outputs of these latches are given below.

Latch	Inputs	Outputs
L_1	S_1, R_1	$Q_1, \overline{Q}_1$
L_2	S_2, R_2	$Q_2, \overline{Q}_2$

The excitation table of an S - R latch is given in Table 8.29. This is same as Table 7.1 for S - R FLIP-FLOP.

Table 8.29 Excitation Table of S - R Latch

Present state Q	Next state Q^+	Inputs	
		S	R
0	0	0	$\times$
0	1	1	0
1	0	0	1
1	1	$\times$	0

To determine S_1 and R_1 for different values of $X_1 X_2$, we make use of Q_1 and Q_1^+ values for every square of transition table. For example, the square in the first row and first column gives $Q_1 = 0$ and $Q_1^+ = 0$. This means, for the present state 0 the circuit gives next state as 0 for Q_1 . Corresponding to this we find the value of S_1 and R_1 using the Table 8.29, which are

$$S_1 = 0 \quad \text{and} \quad R_1 = X$$

Thus the entry in the cell corresponding to $X_1 X_2 = 00$ and $Q_1 Q_2 = 00$ for K -map of S_1 will be 0 and for K -map of R_1 it will be X . Similarly, K -map entries are determined for S_1 and R_1 .

Following similar procedure, K -maps for S_2 and R_2 are constructed. The K -maps are given in Fig. 8.73.

$Q_1 Q_2 \backslash X_1 X_2$		$X_1 X_2$			
		00	01	11	10
$Q_1 Q_2$	00	0	0	0	1
	01	0	1	0	1
	11	0	×	0	×
	10	0	×	×	×

(a) K -Map for S_1

$Q_1 Q_2 \backslash X_1 X_2$		00	01	11	10
		00	01	11	10
00	×	×	×	0	
01	×	0	×	0	
11	1	0	1	0	
10	1	0	0	0	

(b) K -Map for R_1

$Q_1 Q_2 \backslash X_1 X_2$		$X_1 X_2$			
		00	01	11	10
$Q_1 Q_2$	00	0	1	0	0
	01	0	×	×	×
	11	0	×	×	×
	10	0	0	0	0

(c) K -Map for S_2

$Q_1 Q_2 \backslash X_1 X_2$		00	01	11	10
		00	01	11	10
00	×	0	×	×	
01	1	0	0	0	
11	1	0	0	0	
10	×	×	×	×	

(d) K -Map for R_2

Fig. 8.73

From the K -map of Fig. 8.73, we obtain logic equations for S_1 , R_1 , S_2 , and R_2 .

$$S_1 = X_1 \bar{X}_2 + \bar{X}_1 X_2 Q_2$$

$$R_1 = \bar{X}_1 \bar{X}_2 + X_1 X_2 Q_2$$

$$S_2 = \bar{X}_1 X_2 \bar{Q}_1$$

$$R_2 = \bar{X}_1 \bar{X}_2$$

The logic circuit is shown in Fig. 8.74.

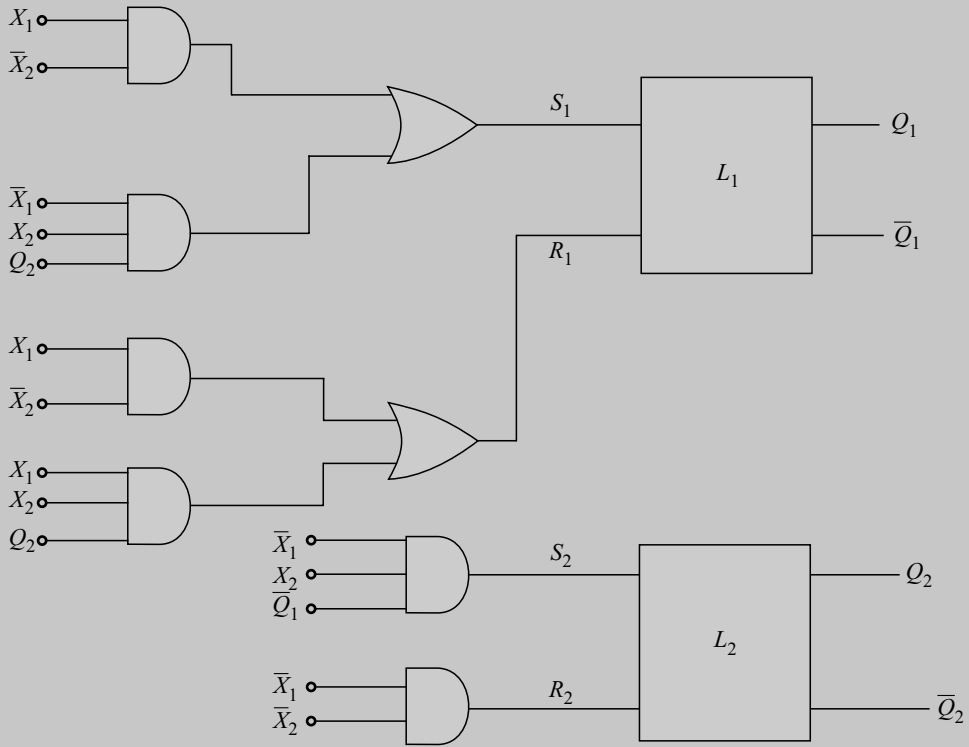


Fig. 8.74 **Logic Circuit for Ex. 8.29**

Races and Cycles

A *race* condition exists in an asynchronous sequential circuit when more than one state variable change value in response to a change in an input variable. This is caused because of unequal propagation delays in the path of different secondary variables in any practical electronic circuit. Consider a transition table shown in Fig. 8.75. When both the inputs X_1 and X_2 are 0 and the present state is $Q_1 Q_2 = 00$, the resulting next-state $Q_1^+ Q_2^+$ will have $Q_1^+ = 1$ and $Q_2^+ = 1$ simultaneously if the propagation delays in the paths of Q_1 and Q_2 are equal.

Since Q_1 and Q_2 both are to change and in general the propagation delays in the paths of Q_1 and Q_2 are not same, therefore, either Q_1 or Q_2 may change first instead of both changing simultaneously. As a consequence of this the circuit will go to either state 01 or to state 10.

$X_1 X_2$		$Q_1^+ Q_2^+$			
		00	01	11	10
$Q_1 Q_2$	00	11	⓪0	10	01
	01	11	00	11	⓪1
	11	⓪1	00	10	⓪1
	10	11	⓪0	⓪0	11

Fig. 8.75

If Q_2^+ changes faster than Q_1^+ , the next state will be 01, then 11 (first column, second row) and then the stable state 11 (first column, third row) will be reached. On the other hand, if Q_1^+ changes faster than Q_2^+ , the next state will be 10, then 11 (first column, fourth row) and then to the stable state 11 (first column, third row) will be reached. In both the situations, the circuit goes to the same final stable state 11. This situation, where a change of more than one secondary variable is required is known as a *race*. There are two types of races: *noncritical race* and *critical race*. In the case of noncritical race, the final stable state in which the circuit goes does not depend on the sequence in which the variables change. The race discussed above is a noncritical race. In the case of critical race, the final stable state reached by the circuit depends on the sequence in which the secondary variables change. Since the critical race results in different stable states depending on the sequence in which the secondary states change, therefore, it must be avoided.

Example 8.30

In the transition table of Fig. 8.75, consider the circuit in stable total state 1100. Will there be any race, if the input state changes to 01? If yes, find the type of race.

Solution

When the circuit is in stable total state, $X_1 X_2 = 00$. Now X_2 changes to 1 while $X_1 = 0$. From Fig. 8.75 we see that the required transition is to state 00. If Q_1^+ and Q_2^+ become 00 simultaneously, then the transition will be

$$11 \rightarrow 00 \rightarrow 00$$

These transitions are shown by solid arrows in Fig. 8.76. If Q_2^+ becomes 0 faster than Q_1^+ , the circuit will go to the state 10 and then to 10, which is a stable state. The transition is

$$11 \rightarrow 10 \rightarrow 10$$

On the other hand, if Q_1^+ becomes 0 faster than Q_2^+ , the transition will be

$$11 \rightarrow 01 \rightarrow 00 \rightarrow 00$$

It is shown by dotted arrow in Fig. 8.76. Thus, we see that the circuit attains different stable states 00 or 10 depending up on the sequence in which the secondary variables change. Therefore, the race condition exists in this circuit and it is critical race.

$X_1 X_2$		$Q_1^+ Q_2^+$			
		00	01	11	10
$Q_1 Q_2$	00		00		
	01		00		
	11	11	00		
	10		10		

Fig. 8.76

Races can be avoided by making a proper binary assignment to the state variables in a flow table. The state variables must be assigned binary numbers in such a way so that only one state variable can change at any one time when a state transition occurs in the flow table. The state transition is directed through a unique sequence of unstable state variable change. This is referred to as a *cycle*. This unique sequence must terminate in a stable state, otherwise the circuit will go from one unstable state to another unstable state making the entire circuit unstable.

Example 8.31

In the state transition table of Fig. 8.75, if $X_1X_2 = 10$ and the circuit is in stable state ⑩, find the cycle when X_2 is changed to 1 while X_1 remaining 1.

Solution

The circuit is in stable state ⑩ (fourth column, second row). When X_2 changes to 1, the circuit will go to the state 11 (third column, second row), then to state 10 (third column, third row) and finally to the stable state ⑩ (third column, fourth row). Thus the cycle is

$$\textcircled{01} \rightarrow 11 \rightarrow 10 \rightarrow \textcircled{10}$$

Pulse-Mode Circuits

In a pulse mode asynchronous sequential circuit, an input pulse is permitted to occur only when the circuit is in stable state and there is no pulse present on any other input. When an input pulse arrives, it triggers the circuit and causes a transition from one stable state to another stable state so as to enable the circuit to receive another input pulse. In this mode of operation critical race can not occur. To keep the circuit stable between two pulses, FLIP-FLOPs whose outputs are levels must be used as memory elements.

For the analysis of pulse-mode circuits, the model used for the fundamental-mode circuits is not valid since the circuit is stable when there are no inputs and the absence of a pulse conveys no information. For this a model similar to the one used for synchronous sequential circuits will be convenient to use.

In pulse-mode asynchronous circuits the number of columns in the next-state table is equal to the number of input terminals.

Consider a pulse-mode circuit logic diagram shown in Fig. 8.77. In this circuit there are four input variables X_1, X_2, X_3 , and X_4 , and Y is the output variable. It has two states Q_1 and Q_2 .

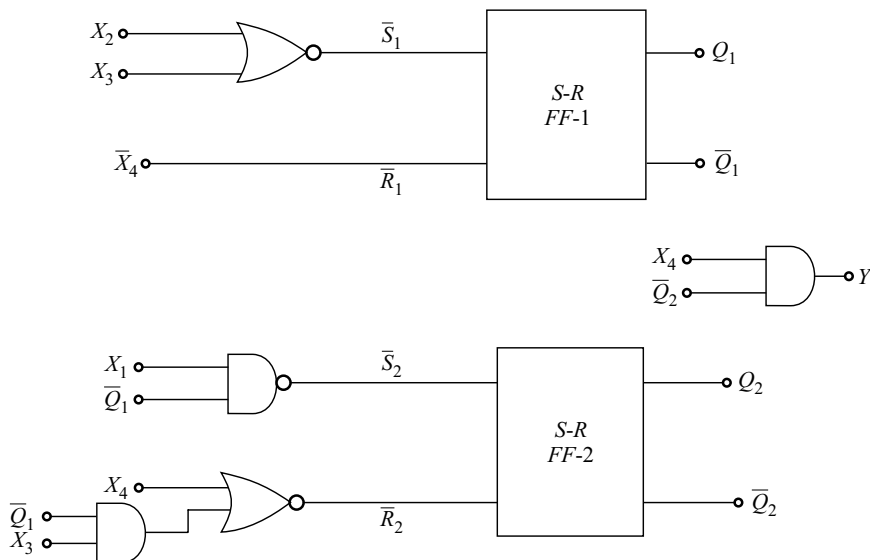


Fig. 8.77

A Pulse-Mode Asynchronous Circuit

The excitation and output equations are:

$$\bar{S}_1 = \overline{(X_2 + X_3)} \quad \text{or} \quad S_1 = X_2 + X_3 \quad (8.16a)$$

$$\bar{R}_1 = \bar{X}_4 \quad \text{or} \quad R_1 = X_4 \quad (8.16b)$$

$$\bar{S}_2 = \overline{\bar{Q}_1 X_1} \quad \text{or} \quad S_2 = \bar{Q}_1 X_1 \quad (8.16c)$$

$$\bar{R}_2 = \overline{(X_4 + \bar{Q}_1 X_3)} \quad \text{or} \quad R_2 = X_4 + \bar{Q}_1 X_3 \quad (8.16d)$$

$$Y = X_4 \bar{Q}_2 \quad (8.17)$$

The next-state equations are obtained by using the excitation equations (Eqs. (8.16)) and the characteristic equation of latch (Eq. (8.9)). These are:

$$\begin{aligned} Q_1^+ &= S_1 + \bar{R}_1 Q_1 \\ &= X_2 + X_3 + \bar{X}_4 Q_1 \end{aligned} \quad (8.18)$$

and

$$\begin{aligned} Q_2^+ &= S_2 + \bar{R}_2 Q_2 \\ &= \bar{Q}_1 X_1 + \overline{(X_4 + \bar{Q}_1 X_3)} \cdot Q_2 \\ &= \bar{Q}_1 X_1 + \bar{X}_4 \cdot \overline{(\bar{Q}_1 X_3)} \cdot Q_2 \\ &= \bar{Q}_1 X_1 + \bar{X}_4 \cdot (Q_1 + \bar{X}_3) \cdot Q_2 \\ &= \bar{Q}_1 X_1 + Q_1 Q_2 \bar{X}_4 + Q_2 \bar{X}_3 \bar{X}_4 \end{aligned} \quad (8.19)$$

The transition table is constructed by evaluating the next-state and output for each present state and input value using Eqs. (8.17), (8.18), and (8.19). The transition table is shown in Fig. 8.78. It has four rows (one row for each combination of state variables) and four columns (one column for each input variable). Since in pulse-mode circuits only one input variable is permitted to be present at a time, therefore, the columns are for each input variable only and not for the combinations of input variables.

Flow table can be constructed from the transition table and is shown in Fig. 8.79. Here, S_0 , S_1 , S_2 , and S_3 are the four state variables.

From a flow table a transition table can be constructed by assigning binary values to the states. From a transition table next-state equations can be obtained and the logic diagram can then be obtained.

Present state	Input variables				Next-state value
	$Q_1 Q_2$	X_1	X_2	X_3	X_4
00	01, 0	10, 0	10, 0	00, 1	Output value
	01, 0	11, 0	10, 0	00, 0	
	11, 0	11, 0	11, 0	00, 0	
	10, 0	10, 0	10, 0	10, 0	

Fig. 8.78

Transition Table of Fig. 8.77

Present state	Input variables			
	$Q_1 Q_2$	X_1	X_2	X_3
S_0	$S_1, 0$	$S_3, 0$	$S_3, 0$	$S_0, 1$
S_1	$S_1, 0$	$S_2, 0$	$S_3, 0$	$S_0, 0$
S_2	$S_2, 0$	$S_2, 0$	$S_2, 0$	$S_0, 0$
S_3	$S_3, 0$	$S_3, 0$	$S_3, 0$	$S_0, 0$

Fig. 8.79

Flow Table of Pulse-Mode Circuit

8.7.5 Asynchronous Sequential Circuit Design

Design of asynchronous sequential circuits is more difficult than that of synchronous sequential circuits because of the timing problems involved in these circuits. Designing an asynchronous sequential circuit requires obtaining logic diagram for the given design specifications. Usually the design problem is specified in the form of statements of the desired circuit performance precisely specifying the circuit operation for every applicable input sequence.

Design Steps

1. Primitive flow table is obtained from the design specifications. When setting up a primitive flow table it is not necessary to be concerned about adding states which may ultimately turn out to be redundant. A sufficient number of states are to be included to completely specify the circuit performance for every allowable input sequence. Outputs are specified only for stable states.
2. Reduce the primitive flow table by eliminating the redundant states, which are likely to be present. These redundant states are eliminated by merging the states. *Merger diagram* is used for this purpose.
3. Binary numbers are assigned to the states in the reduced flow table. The binary state assignment must be made to ensure that the circuit will be free of critical races. The output values are to be chosen for the unstable states with unspecified output entries. These must be chosen in such a way so that momentary false outputs do not occur when the circuit switches from one stable state to another stable state.
4. Transition table is obtained next.
5. From the transition table logic diagram is designed by using the combinational design methods. The logic circuit may be a combinational circuit with feedback or a circuit with *S-R* latches.

The above design steps will be illustrated through an example.

Example 8.32

The output (Y) of an asynchronous sequential circuit must remain 0 as long as one of its two inputs X_1 is 0. While $X_1 = 1$, the occurrence of first change in another input X_2 , should give $Y = 1$ as long as $X_1 = 1$ and becomes 0 where X_1 returns to 0. Construct a primitive flow table.

Solution

This circuit has two inputs X_1 , X_2 and one output Y . For the construction of flow table, the next-state and output are required to be obtained. The flow table is shown in Fig. 8.80.

For $X_1 X_2 = 00$, let us take state a . When the circuit has $X_1 X_2 = 00$ the output is 0 (since $X_1 = 0$) and the circuit is in stable state $\textcircled{a}$. The next-state and output are shown in the first column, first row of Fig. 8.80. Since only one input is allowed to change at a time, therefore, the next input may be $X_1 X_2 = 01$ or 10.

If $X_1 X_2 = 01$, let us take another state b , correspondingly the second row of the flow table corresponds to state b . when the inputs change from $X_1 X_2 = 00$ to 01, the circuit is required to go to stable state $\textcircled{b}$ and output is 0 (since $X_1 = 0$). Therefore, the entry in the second column, first row will be $b, 0$ and in the second column, second row will be $\textcircled{b}, 0$. The output corresponding to unstable state b is taken as 0 so that no momentary false outputs occur when the circuit switches between stable states. On the other hand if $X_1 X_2$

		$X_1 X_2$			
		00	01	11	10
Present-state	a	$\textcircled{a}, 0$	$b, 0$	$-, -$	$c, 0$
	b	$a, 0$	$\textcircled{b}, 0$	$d, 0$	$-, -$
	c	$a, 0$	$-, -$	$e, -$	$\textcircled{c}, 0$
	d	$-, -$	$b, 0$	$\textcircled{d}, 0$	$t, -$
	e	$-, -$	$b, -$	$\textcircled{e}, 0$	$t, 1$
	f	$a, -$	$-, -$	$e, 1$	$\textcircled{f}, 1$

Fig. 8.80

Flow Table of Ex. 8.32

= 10, the circuit is required to go to another stable state $\odot$ with output 0. Therefore, the entries in the fourth column, first row and fourth column, third row will be respectively c , 0 and $\odot$, 0.

Since both the inputs cannot change simultaneously, therefore, from stable state $\odot$, the circuit cannot go to any specific state corresponding to $X_1 X_2 = 11$ and accordingly the entry in the third column, first row will be $-$, $-$. The dashes represent the unspecified state, output.

Now consider the stable state $\odot$. The inputs $X_1 X_2$ can change to 00 or 11.

If $X_1 X_2 = 00$, the circuit will go to state a . Therefore, the entry in the first column, second row will be a , 0. From this unstable state the circuit goes to stable state $\odot$. On the other hand if $X_1 X_2 = 11$, then the circuit goes to a new state d . The output corresponding to $X_1 X_2 = 11$ will be 0 since, there is no change in X_2 , which is already 1. Therefore, the entry in the third column, second row will be d , 0. The fourth row corresponds to state d , and the entry in the third column, fourth row, will be $\odot$, 0. From $\odot$, the circuit is not required to go to any specific state and therefore, the entry in the fourth column, second row will be $-$, $-$.

Similarly, now consider stable state $\odot$. The inputs can change to $X_1 X_2 = 11$ or 00. If $X_1 X_2 = 11$, the circuit goes to a new stable state $\odot$ and the output will be 1, since X_2 changes from 0 to 1 while $X_1 = 1$. The entry in the third column, third row will be e , $-$. Output has to change from 0 to 1 from stable state $\odot$ to stable state $\odot$, which may or may not change to 1 for unstable e . The entry in the third column, fifth row will be $\odot$, 1. The entry in the second column third row will be $-$, $-$ and the entry in the first column, third row will be a , 0 (for $X_1 X_2 = 00$).

In the same manner, we consider the stable $\odot$ and obtain the entries f , $-$ (fourth column, fourth row); $\odot$, 1 (fourth column, sixth row); b , 0 (second column, fourth row) and $-$, $-$ (first column, fourth row).

Similar procedure applied to $\odot$ and $\odot$, yields the remaining entries of the flow table (Problem. 8.36).

Since, every row in the flow table of Fig. 8.80 contains only one stable state, therefore, this flow table is a primitive flow table.

Reduction of States

The necessity of reducing the number of states has been discussed in Section 8.6 and the equivalent states have been defined. When asynchronous sequential circuits are designed, the design process starts from the construction of primitive flow table. A primitive flow table is never completely specified. Some states and outputs are not specified in it as shown in Fig. 8.80 by dashes. Therefore, the concept of equivalent states can not be used for reduction of states. However, incompletely specified states can be combined to reduce the number of states in the flow table. Two incompletely specified states can be combined if they are *compatible*.

Two states are *compatible* if and only if, for every possible input sequence both produce the same output sequence whenever both outputs are specified and their next states are compatible whenever they are specified. The unspecified outputs and states shown as dashes in the flow table have no effect for compatible states.

Example 8.33

In the primitive flow table of Fig. 8.80, find whether the states a and b are compatible or not. If compatible, find out the merged state.

Solution

The rows corresponding to the states a and b are shown in Fig. 8.81. Each column of these two states is examined.

Column-1 Both the rows have the same state a and the same output 0. a in first row is stable state and in the second row is unstable state.

Since for the same input both the states a and b have the same specified next-state a and the same specified output 0. Therefore, this input condition satisfies the requirements of compatibility.

Column-2 The input condition $X_1 X_2 = 01$ satisfies the requirements of compatibility as discussed for column-1.

Column-3 The first row has unspecified next-state and output and the second row has specified state and output. The unspecified state and output may be assigned any desired state and output and therefore, for this input condition also the requirements of compatibility are satisfied.

Column-4 The requirements of compatibility are satisfied for the reasons same as applicable to column-3.

Therefore, we conclude that since the next-states and the outputs for all the input combinations are compatible for the two states a and b , the two states are compatible.

The merged state will be as shown in Fig. 8.82. When the merged state entries are determined a circled entry and an uncircled entry results in a circled entry, since the corresponding state must be stable as shown in Fig. 8.82.

		$X_1 X_2$			
		00	01	11	10
Present state	a	@, 0	b , 0	-, -	c , 0
	b	a , 0	(b), 0	d , 0	-, -

Fig. 8.81

		$X_1 X_2$			
		00	01	11	10
Present state	a	@, 0	(b), 0	d , 0	c , 0

Fig. 8.82

Example 8.34

In the primitive flow table of Fig. 8.80 find whether the states a and e are compatible or not. Examine their compatibility if the entries in the fourth column for the states a and e have same output.

Solution

The partial flowtable for states a and e of Fig. 8.80 is shown in Fig. 8.83.

From this we observe the following

- | | |
|----------|--|
| Column-1 | compatible |
| Column-2 | compatible |
| Column-3 | compatible |
| Column-4 | not compatible, since the outputs are different. |

Therefore, the states a and e are not compatible.

In case of same output in column-4, the outputs are said to be *not conflicting* and the states a and e are compatible if and only if the states c and f are compatible. This is referred to as c, f is implied by a, b or a, b implies c, f .

		$X_1 X_2$			
		00	01	11	10
Present state	a	@, 0	b , 0	-, -	c , 0
	e	-, -	b , -	(c), 0	t , 1

Fig. 8.83

Merger Diagram

A *merger diagram (or graph)* is prepared for a primitive flow table to determine all the possible compatible states (maximal compatible states) and from this a minimal collection of compatibles covering all the states.

A merger graph is constructed following the steps outlined below:

- Each state is represented by a vertex, which means it consists of n vertices, each of which corresponds to a state of the circuit for an n -state primitive flow table. Each vertex is labelled with the state name.
- For each pair of compatible states an undirected arc is drawn between the vertices of the two states. No arc is drawn for incompatible states.
- For compatible states implied by other states a broken arc is drawn between the states and the implied pairs are entered in the broken space.

The flow table is required to be examined for all the possible pairs of states. All the pairs are checked and the merger graph is obtained. Thus we see that the merger graph displays all possible pairs of compatible states and their implied pairs. Next it is necessary to check whether the incompatible pair(s) does not invalidate any other implied pair. If any implied pair is invalidated it is neglected. All the remaining valid compatible pairs form a group of *maximal* compatibles.

The maximal compatible set can be used to construct the reduced flow table by assigning one row to each member of the group. However, the maximal compatibles do not necessarily constitute the set of *minimal* compatibles. The set of minimal compatibles is a smaller collection of compatibles that will satisfy the row merging.

The conditions that must be satisfied for row merging are:

- the set of chosen compatibles must *cover* all the states, and
- the set of chosen compatibles must be *closed*.

The condition of covering requires inclusion of all the states of the primitive flow graph in the set of chosen compatibles. This condition only defines a *lower bound* on the number of states in the minimal set. However, if none of their implied pairs are contained in the set, the set is not sufficient and this is referred to as *closed* condition not being satisfied. Therefore, condition of *closed covering* is essentially required for row merging.

Example 8.35

Construct merger diagram for the primitive flow table of Fig. 8.80. Determine maximal compatibles and the minimal set of compatibles.

Solution

For construction of merger diagram, every row of the primitive flow table is checked with every other row to determine compatibility of states.

Consider row-1 (state *a*)

- a, b* are compatible
- a, c* are compatible
- a, d* are compatible if *c, f* are compatible
- a, e* are compatible if *c, f* are compatible
- a, f* are compatible if *c, f* are compatible

row-2 (state *b*)

- b, c* are compatible if *d, e* are compatible
- b, d* are compatible
- b, e* are not compatible (outputs are conflicting)
- b, f* are not compatible (outputs are conflicting)

row-3 (state *c*)

- c, d* are compatible if *e, d* and *c, f* are compatible
- c, e* are not compatible (outputs are conflicting)
- c, f* are not compatible (outputs are conflicting)

row-4 (state *d*)

- d, e* are not compatible (outputs are conflicting)
- d, f* are not compatible (outputs are conflicting)

row-5 (state e)

e, f are compatible

The primitive flow table has six states therefore, there are six vertices in the merger diagram as shown in Fig 8.84. Solid arcs are drawn between (a, b) , (a, c) , (b, d) , and (f, e) vertices. Corresponding to these states being compatibles. Since (c, f) and (d, e) are not compatible, therefore, there are no implied pairs available.

From the merger diagram, we get the maximal compatibles:

$$(a, b), (a, c), (b, d), (e, f)$$

Since (a, b) is covered by (a, c) and (b, d) , therefore, the minimal set is

$$(a, c), (b, d), (e, f)$$

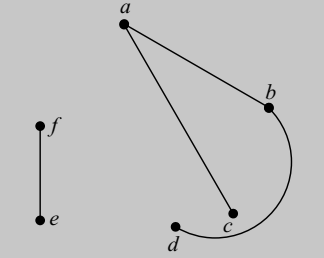


Fig. 8.84

Merger Diagram**Example 8.36**

Determine the reduced flow table of Fig. 8.81.

Solution

From the merger diagram, we have obtained three pairs of compatible states: These compatibles are merged and are represented by

$$a, c : S_0$$

$$b, d : S_1$$

$$e, f : S_2$$

The reduced flow table is shown in Fig. 8.85.

Present state \ $X_1 X_2$	00	01	11	10
S_0	$\textcircled{S_0}, 0$	$S_1, 0$	$S_2, -$	$\textcircled{S_0}, 0$
S_1	$S_0, 0$	$\textcircled{S_1}, 0$	$\textcircled{S_1}, 0$	$S_2, -$
S_2	$S_0, -$	$S_1, -$	$\textcircled{S_2}, 1$	$\textcircled{S_2}, 1$

Fig. 8.85

Reduced Flow Table**Example 8.37**

Assign binary states to the reduced flow table of Fig. 8.85. Avoid critical race.

Solution

Let us assign the following binary states to S_0 , S_1 , and S_2 for the reduced flow table of Fig. 8.85

$$S_0 \rightarrow 00$$

$$S_1 \rightarrow 01$$

$$S_2 \rightarrow 11$$

The transition table will be as shown in Fig. 8.86

In the transition table of Fig. 8.86, we observe that race condition occurs in the following cases:

(i) From stable state $\textcircled{00}$ to unstable state 11 when $X_1 X_2$ changes from 10 to 11.

(ii) From stable state $\textcircled{11}$ to unstable state 00 when $X_1 X_2$ changes from 10 to 00.

To avoid critical race, one unstable state 10 is added with the entries

$$00, -; -, -; 11, -; -, -$$

and the entries in third column, first row is changed from 11, - to 10, - and in first column, third row from 00, - to 10, -.

The modified transition table is given in Fig. 8.87.

$X_1 X_2$		00	01	11	10
$Q_1 Q_2$	00	Ⓐ, 0	01, 0	11, -	Ⓐ, 0
	01	00, 0	Ⓐ, 0	Ⓐ, 0	11, -
	11	00, -	01, -	Ⓐ, 1	Ⓐ, 1
	10				

Fig. 8.86

Transition Table

$X_1 X_2$		00	01	11	10
$Q_1 Q_2$	00	Ⓐ, 0	01, 0	10, -	Ⓐ, 0
	01	00, 0	Ⓐ, 0	Ⓐ, 0	11, -
	11	10, -	01, -	Ⓐ, 1	Ⓐ, 1
	10	00, 1	-, -	11, 1	-, -

Fig. 8.87

Modified Transition Table

Example 8.38

Design logic circuit with feedback for the transition table of Fig. 8.87.

Solution

The K -maps for Q_1^+ , Q_2^+ , and Y determined from the transition table are given in Fig. 8.88.

From the K -maps, we obtain,

$$Q_1^+ = X_1 X_2 \bar{Q}_2 + \bar{X}_2 Q_1 Q_2 + X_1 \bar{X}_2 \bar{Q}_2 + X_1 Q_1$$

$$Q_2^+ = \bar{X}_1 X_2 + X_1 Q_2 + X_1 Q_1$$

$$Y = Q_1$$

Logic circuit using gates can be obtained from the above logic equations.

Thus we see that the design steps outlined above can be used to design an asynchronous sequential circuit.

$X_1 X_2$		00	01	11	10
$Q_1 Q_2$	00	0	0	1	0
	01	0	0	0	1
	11	1	0	1	1
	10	0	×	1	×

(a)

$X_1 X_2$		00	01	11	10
$Q_1 Q_2$	00	0	1	0	0
	01	0	1	1	1
	11	0	1	1	1
	10	0	×	1	×

(b)

$X_1 X_2$		00	01	11	10
$Q_1 Q_2$	00	0	0	×	0
	01	0	0	0	×
	11	×	×	1	1
	10	×	×	×	×

(c)

 Fig. 8.88 K -Maps for (a) Q_1^+ (b) Q_2^+ (c) Y

8.8 HAZARDS IN SEQUENTIAL CIRCUITS

Hazards in combinational circuits have been discussed in Section 5.12. The synchronous sequential circuits operate in synchronism with the clock pulses and the inputs must be stable before the triggering edge arrives. Therefore, even if hazard condition in the combinational part of the synchronous circuit creates erroneous signal i.e., glitch, the operation of the circuit is not affected. However, asynchronous sequential circuits may malfunction because of the occurrence of hazard condition in its combinational part and/or unequal delays along two or more feedback paths that originate from the same input variable change.

In case the hazard condition is caused due to the hazard in the combinational part of an asynchronous sequential circuit, the hazard is either static-1, static-0, or dynamic type of hazard. The hazard caused due to the effects of a single input variable change reaching one feedback path before another feedback path is known as *essential-hazard*.

8.8.1 Hazards in Asynchronous Sequential Circuits Implemented Using Latches

The momentary false signals in the combinational logic part of an asynchronous sequential circuit can cause improper operation. Therefore, the combinational logic portion of an asynchronous sequential circuit should be hazard-free.

It has been demonstrated in section 5.12.2 that the following types of hazards may occur in 2-level realizations.

- Static-1 hazard in SOP realization i.e. AND–OR or NAND–NAND realization. This type of hazard causes the output of a combinational circuit to go momentarily to 0.
- Static-0 hazard in POS realization i.e. OR–AND or NOR–NOR realization. This type of hazard causes the output of a combinational circuit to go momentarily to 1.

Now let us examine the effect of static-1 and static-0 hazards on the operation of the S - R latches shown in Figs. 8.68*a* and *b*.

- In the case of NOR latch of Fig. 8.68*b*, a momentary 1 signal on either S or R inputs can cause it to set or reset, respectively. However, a momentary 0 signal on S or R inputs has no effect since, when $S = 0$ and $R = 0$, the state of the circuit does not change. Therefore, the combinational logic circuit that precedes the input terminals of an S - R latch is not required to be free of static-1 hazards, but it must be free of static-0 hazards.
- In the case of NAND latch of Fig. 8.68*a*, a momentary 0 signal on either $\bar{S}$ or $\bar{R}$ inputs can cause it to set or reset, respectively. However, a momentary 1 signal on $\bar{S}$ or $\bar{R}$ inputs has no effect since, when $\bar{S} = 1$ and $\bar{R} = 1$, the state of the circuit does not change. Therefore, in this case, the combinational logic circuit that precedes the input terminals of an $\bar{S}$ - $\bar{R}$ latch is not required to be free of static-0 hazards, but it must be free of static-1 hazards.

The hazard-free asynchronous circuits are shown in Fig. 8.89.

8.8.2 Essential-Hazards

In some asynchronous sequential circuits, there is an inherent possibility of making an incorrect transition due to differences in delay through various paths because of change in a single input variable. Such type of hazard is known as essential hazard. It is not possible to modify the circuit to make it free of essential hazard.

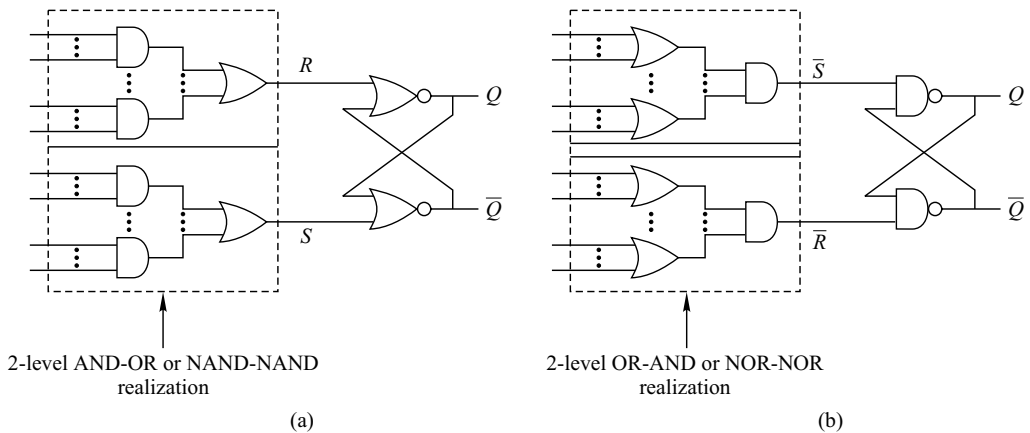


Fig. 8.89 **Hazard Free Asynchronous Circuits Using Latches**

The essential hazard can only be eliminated by introducing additional delay in the feedback path so that incorrect transition does not take place.

Essential hazard in an asynchronous sequential circuit can be detected by observing its flow table. Consider a fundamental mode asynchronous sequential circuit in a total stable state (S_1) . A change in an input variable x causes transition to another stable state (S_2) in an adjacent column, assuming equal delay through all the feedback paths. If the circuit goes to a different stable state (S_3) after three consecutive changes in x due to different delays in the feedback paths, then we say that essential hazard exists in the flow table.

Consider the flow table shown in Fig. 8.90. Let the circuit be in a stable state (A) , changing x input from 0 to 1 will cause transition to another stable state (B) . The state transition is indicated by dotted arrows. This operation corresponds to the situation when there is no problem of occurrence of essential hazard due to unequal delay in different feedback paths. Now let us assume that because of unequal delays in the different feedback paths, the circuit goes to state C (second row, first column), then to state (C) (third row, first column) and finally becomes stable in (C) (third row, second column). This illustrates the occurrence of essential hazard. The operation of the circuit is indicated by solid arrows.

		Next State	
		0	1
Present state	A	(A)	(B)
	B	C	(B)
	C	(C)	(C)
	D	—	—

Fig. 8.90 **Flow Table**

Example 8.39

Determine whether essential hazard exists in the flow tables of Fig. 8.91. Assume initial stable state (A) .

Solution

In Fig. 8.91a when x changes from 0 to 1, the circuit may go to B and then to (B) or it may go to A (second row, first column) and eventually to (B) . This shows that the circuit is free of essential hazard.

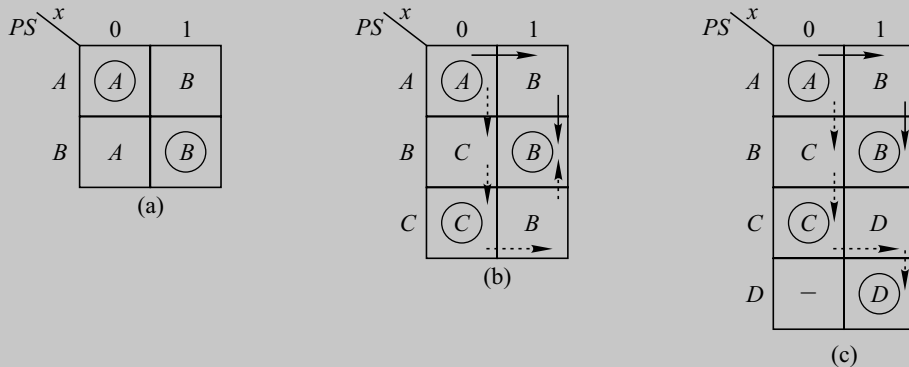


Fig. 8.91 Flow Tables of Ex. 8.39

In Fig. 8.91b. The circuit goes to stable state (B) when x changes from 0 to 1. It may follow either the path shown by solid arrows or the path shown by the dotted arrows.

In Fig. 8.91c. The circuit goes to stable state (B) (see the solid arrows) or to the stable state D following the path shown by the dotted arrows. Therefore, unequal delays in the different feedback paths will cause transition to stable state (D) rather than (B) which shows the existence of essential-hazard.

Essential hazard can be eliminated by introducing additional delay in the feedback path with lesser delay.

SUMMARY

A very important and useful class of digital circuits, sequential circuits, have been discussed in detail in this chapter. Clocked sequential circuits, such as, counters, shift registers, useful in many applications of digital systems have been discussed. Their design using FLIP-FLOPs and standard MSI devices have been included. Design of general clocked sequential circuits using the two models, Mealy and Moore, will be helpful in designing any clocked sequential circuit.

Standard counter ICs are available for counting in normal binary sequence and natural BCD for different modulus. In many applications counters can be designed using these available ICs.

In case any other sequence of count or modulus is required, then the circuit can be designed using FLIP-FLOPs and gates using the methods discussed here.

The analysis and design of asynchronous sequential circuits have been discussed in detail. Although the design of asynchronous sequential circuits is much more difficult than the design of clocked (synchronous) sequential circuits, nonetheless they find widespread applications requiring high speeds, low cost systems in which input signals may change at any time.

There are various problems associated with the asynchronous sequential circuits such as races and cycles, hazards, and essential-hazards. All these cause malfunctioning of the circuit. The causes of existence these problems and the methods used to eliminate them have been discussed.