9 Event driven circuits

9.1 Introduction

Some sequential circuits are driven by events rather than by a train of clock pulses. For example, a digital alarm will be activated by the event that raised the alarm. In this example it is the event that drives the logic, and since the events are frequently irregular occurrences, such a circuit is referred to as an asynchronous sequential circuit or, perhaps more meaningfully, as an event driven circuit.

Asynchronous circuits are also called fundamental mode circuits. The main characteristic of this type of circuit is that only one input is allowed to change at any given instant. Simultaneous changes are forbidden as, indeed, are changes that may take place before the circuit reaches a stable condition after the preceding change. This is clearly different from the behaviour of a synchronous sequential circuit, where inputs changing at arbitrary times are allowed and state changes are activated by the repetitive clock signal.

There are two conditions in which an asynchronous circuit may exist, namely stable and unstable. The total state of the circuit at a given time is defined by the logical values of the inputs and the present state of the circuit. If the next state is the same as the present one the circuit is in a stable condition. If, however, an input changes, the circuit may move to an unstable condition and at some later time the state variables will have taken on their new values such that the next state has become the present state, and stability has been restored.

When designing asynchronous circuits, the designer has to eliminate the possibility of the occurrence of static hazards, dynamic hazards, essential hazards and races, in order to avoid circuit malfunction. These problems, with the exception of static hazards, do not exist in synchronous circuits since they are always designed to reach a steady-state condition before the next clock pulse arrives. Bearing in mind the design difficulties, perhaps the main advantage of asynchronous circuits is that they can work at their own speed and are not constrained to work within the time limits imposed on them by a repetitive clock signal.

9.2 Design procedure for asynchronous sequential circuits

The design procedure for asynchronous sequential circuits is similar in many respects to that developed for synchronous circuits in Chapter 8. The aim of the design is to produce hazard-free next state equations and output functions. The steps in the design procedure are summarised below:

1. *Problem definition*: An unambiguous statement is required by the designer.
2. *Basic state table and internal state diagram*: A basic state table should be constructed from the information given in step 1 above. In many cases the designer

may find it helpful to produce a state diagram first, and then develop the basic state table from the information provided on the state diagram.

3. *Reduction of the basic state table*: If possible by using Caldwell's merging rules or a merging diagram, reduce the number of rows in the table, thus reducing the number of states. In some cases it may be necessary to use an implication chart to reduce the number of states.
4. *State assignment*: Secondary variables are assigned to the states, care being taken to avoid races.
5. *Equations for the state variables*: The equations for the variables assigned to the states can be obtained using a sequential equation, such as $Q^{t+\delta t} = (S + \bar{R}Q)^t$, as developed in Chapter 6. This will lead to a gate implementation of the equations and steps should be taken to ensure that they are hazard-free. Alternatively, the equations can be implemented using latches and the next state equations for their inputs may be determined from the reduced state table.

9.3 Stable and unstable states

The equation of the SR latch developed in Chapter 6 can be written as follows:

$$Q^{t+\delta t} = (S + \bar{R}q)^t$$

where $Q^{t+\delta t}$ is the next state while q^t is its present state. The gate circuit for the latch is shown in Figure 9.1(a), and if the feedback path is removed it can be regarded as a purely combinational circuit in which the condition $S = R = 1$ is not allowed. A K-map for those combinations of the variables that are allowed is plotted in Figure 9.1(b).

For the condition $SRq = 000, q = Q = 0$, and if the feedback path is reconnected, the state of the latch will remain unchanged. This is a stable state which is indicated by ringing the entry on the K-map.

For the condition $SRq = 011, q = 1$ and $Q = 0$. In this case, if the feedback path is reconnected, there will be a change of state to $SRq = 010, q = 0$ being the next present state. $SRq = 011$ is an unstable state and is not ringed on the K-map.

On the K-map in Figure 9.1(b) all the stable states have been ringed leaving the remaining unstable states not ringed. In Figure 9.1(c) the states have been defined numerically and the unstable states are given the same number as the adjacent stable state having the same values of S and R. If the latch is in state 2 and R makes the

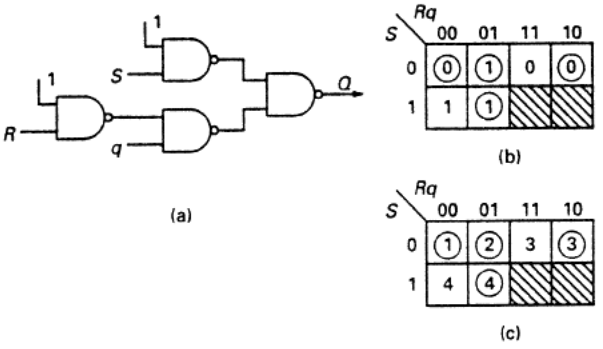


Figure 9.1 *(a) SR latch gate circuit (b) next state map for Q (c) stable and unstable states*

transition 0 → 1 it enters the unstable state 3 before finally settling in stable state 3. These changes can be summarised as follows:

SRq	001	→	011	→	010
	Stable	→	Unstable	→	Stable
	2	→	3	→	3

9.4 Design of a lamp switching circuit

Step 1: Problem definition

An asynchronous sequential circuit is to be designed to ensure that a correct manual switching procedure is carried out by the operator for part of an electrical key operating mechanism. If switch X is made, followed by switch Y, then a red lamp L_R is to be turned on, indicating that the incorrect switching procedure has been followed. If switch Y is made followed by switch X, then a green lamp L_G is to be illuminated, indicating that the correct switching procedure has been adopted.

A block diagram for the problem is shown in Figure 9.2(a). The two inputs X and Y generated when the switches are made are referred to as the primary or input variables. The circuit has two outputs, one of which drives the red lamp and the other, the green lamp.

Step 2: Internal state diagram and basic state table

The internal state diagram for the problem is shown in Figure 9.2(b). S_0 may be regarded as the quiescent state in that it represents the condition that both switches are off. The path taken through the state diagram for the correct switching procedure is $S_0 \rightarrow S_3 \rightarrow S_4$, and for the incorrect procedure the path is $S_0 \rightarrow S_1 \rightarrow S_2$. A number of options are available for the switching procedure in the reverse direction. The ones available for this design are:

(a) From green light on, either $S_4 \rightarrow S_3 \rightarrow S_0$ or $S_4 \rightarrow S_1 \rightarrow S_0$
(b) From red light on, either $S_2 \rightarrow S_1 \rightarrow S_0$ or $S_2 \rightarrow S_3 \rightarrow S_0$

A basic state table can now be drawn up from the information appearing on the state diagram (see Figure 9.2(b)). The table has five rows, one for each possible present state, and four columns, one for each of the possible combinations of the input variables.

The entry in the top left-hand cell is $\textcircled{S_0}$ and is ringed, indicating a stable state. The entry in the top right-hand cell is S_1, indicating an unstable state. This implies that if, in the quiescent state S_0 where $XY = 00$, X changes from 0 to 1, then a new unstable state S_1 is defined by the total state S_0XY before the circuit settles into a new stable state defined by S_1XY. This condition defines the fourth cell on the second row where the entry is $\textcircled{S_1}$.

It will be noted that there is only one stable state per row, and that each unstable state is preceded and succeeded by a stable state. A further examination of the basic state table shows that some cells do not have an entry at all. For example, in the first row where the present state is S_0, there is no entry in the cell corresponding to the input combination $XY = 11$. The basic state table shows that the state S_0 is entered from either S_1 or S_3 with an input signal $XY = 00$, and to enter the cell on the first row where $XY = 11$ would now require a simultaneous change of the input variables. Such a change is not allowable for a circuit operating in the fundamental mode. Cells with no entries in them are marked with a '–'.

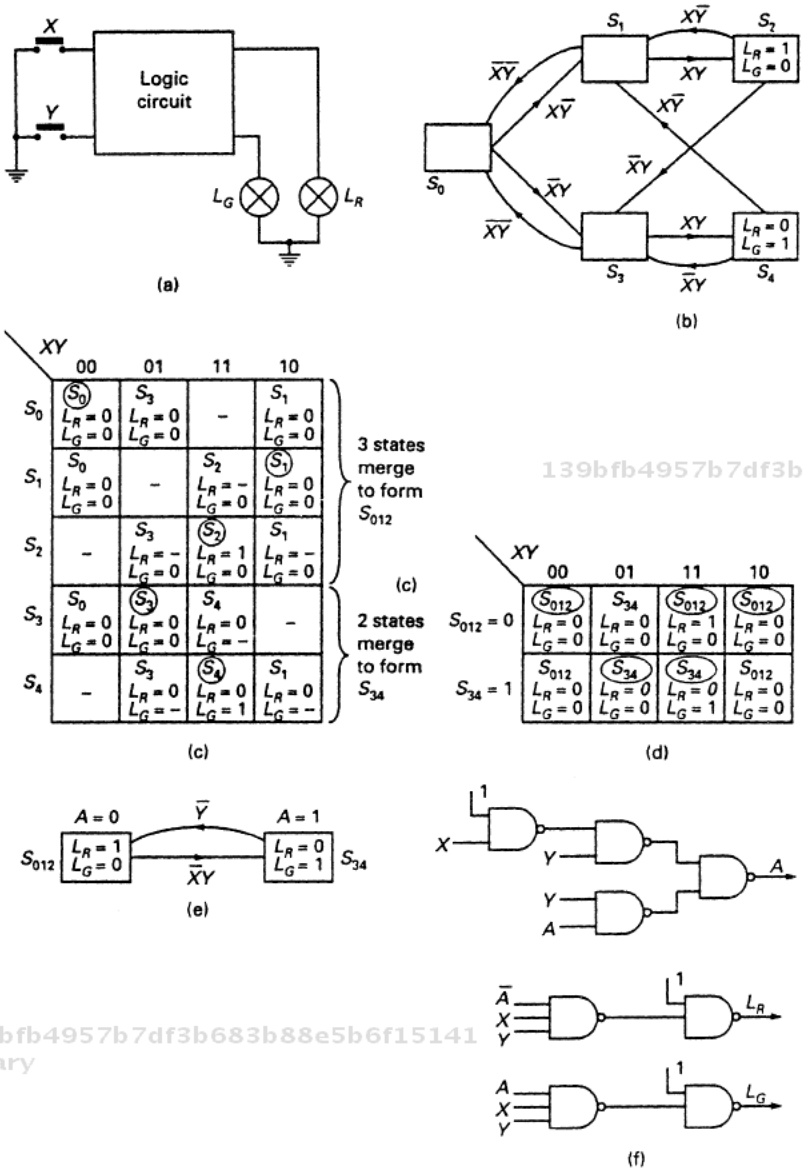


Figure 9.2 *(a) Block diagram for lamp switching circuit (b) the internal state diagram (c) primitive state table (d) reduced state table (e) reduced state diagram (f) circuit implementation*

State tables which contain cells marked with a '–' are referred to as *incompletely specified tables*. In the table shown in Figure 9.2(c) the cells marked with a '–' correspond to forbidden input combinations. These cells can be regarded as 'can't happen' conditions, and may enable a simplification of the table which did not at first sight seem possible. The justification of the allocation of a 'can't happen' condition in the state table is the same as for 'can't happen' conditions in combinational logic problems. If an event cannot happen, the designer 'doesn't care' what the circuit would do in response to it.

Entries are made in each cell of the table for the outputs, and two possible situations can arise:

1. The output entries are identical in the states immediately preceding and succeeding an unstable state. For this situation, the output entries in an unstable state should be identical to those in the immediately preceding and succeeding states.
2. The output entries in the immediately preceding and succeeding states are different. For this situation, the entries in the intervening unstable state can be '–', indicating a 'don't care' entry which can be used in the simplification of the output function.

Step 3: Reduction of the basic state table

When simplifying an incompletely specified table, it is possible to assign a next state and output to a cell containing a '–' in such a way as to make the row in which the '–' occurs identical to a second row. The states at the head of these two rows are then identical, since all the next state entries and outputs in corresponding cells on the two rows are the same, and the rows can be merged.

An examination of the table in Figure 9.2(c) shows that the rows headed S_0, S_1 and S_2, and those headed S_3 and S_4 are identical and can be merged using Caldwell's merging rules. S_0, S_1 and S_2 are merged to form a new state S_{012} and states S_3 and S_4 are merged to form a new state S_{34}. The reduced state table is shown in Figure 9.2(d) and the reduced state diagram in Figure 9.2(e) is constructed from the information in the reduced state table.

Step 4: State assignment

Since there are only two states in the reduced state diagram, just one state variable A is required to define them. For the state S_{012}, $A = 0$, and for the state S_{34}, $A = 1$. As there is only one state variable in this case, the problem of races does not arise.

Step 5: Equations for the state variable and the outputs

The equation for the state variable can now be obtained with the aid of the NAND sequential equation $Q^{t+\delta t} = (S + \bar{R}Q)^t$, where S is defined as the turn-on condition for Q and R is defined as the turn-off condition for Q. The turn-on and the turn-off conditions for the secondary variable A can be obtained directly from the reduced state diagram:

Turn-on condition for $A = \bar{X}Y$
Turn-off condition for $A = \bar{Y}$
Hence $A^{t+\delta t} = (\bar{X}Y + YA)^t$

And the outputs may be written

$L_G = AXY$ and $L_R = \bar{A}XY$

The implementation of the circuit is shown in Figure 9.2(f).

9.5 Races

When the state variables were allocated to the internal states of a clock-driven sequential circuit, the criterion for the allocation was that it should lead to a minimum

hardware implementation. It was pointed out in the previous chapter that there is no known method for the allocation of the state variables that will lead to minimum hardware implementation, although guidelines were presented which, when used, lead to a simple, if not the simplest, circuit. The criterion for the allocation of state variables in event-driven circuits is somewhat different, and in this section those factors which govern this allocation will be examined.

An alternative state diagram for the light-switching problem is shown in Figure 9.3(a). An extra state S_5 has been introduced to allow an extra return path from the 'light-on' states S_2 and S_4 to the quiescent state S_0. Using the techniques described in section 9.4, the reduced state table and state diagram have been obtained. Four combinations of two state variables A and B have been arbitrarily allocated, one to each of the four states, and it will be noticed that when a transition is made from S_5 to S_0 on the signal XY, both of the state variables have to change. There are three possible cases to consider:

1. A and B change simultaneously: a direct transition is made from S_5 to S_0.
2. A changes before B: the circuit makes a transition to S_0 via the route $S_5 \rightarrow S_{34} \rightarrow S_0$.
3. B changes before A: the circuit makes2 a transition to S_0 via the route $S_5 \rightarrow S_{12} \rightarrow S_0$.

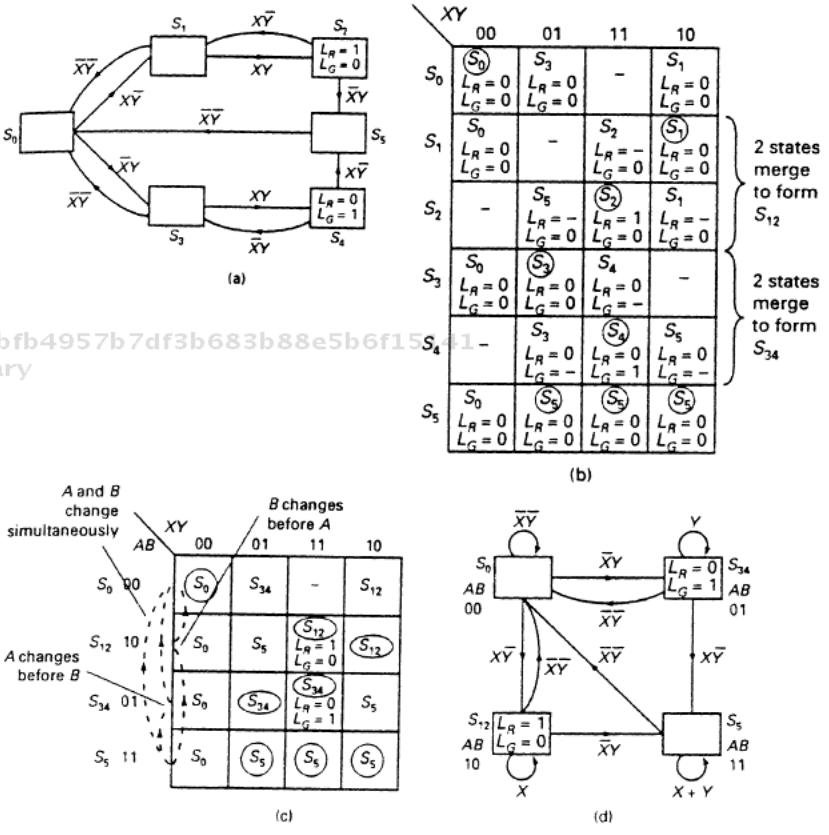


Figure 9.3 *(a) Modified light switching state diagram (b) primitive state table (c) reduced state table (d) reduced state diagram exhibiting a non-critical race*

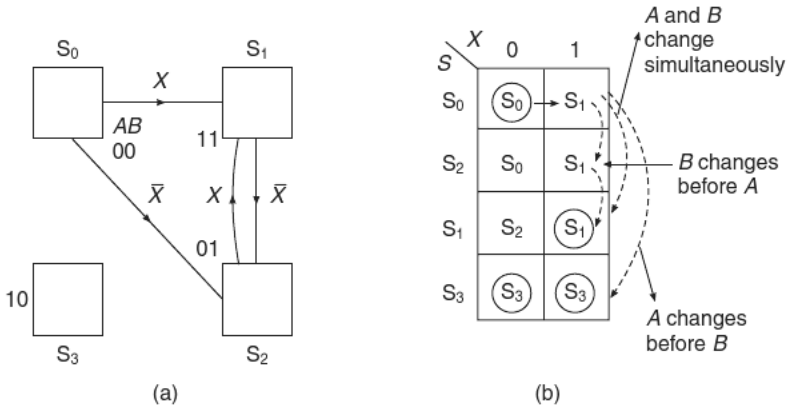


Figure 9.4 *(a) State diagram for a circuit exhibiting critical races (b) State table illustrating a critical race*

In all three cases the circuit enters a stable state S_0 and remains there until a further change of input variables occurs. The various transitions between states for the three conditions described above are illustrated in the reduced state table shown in Figure 9.3(c).

From the foregoing remarks it may be concluded that whenever two state variables change in response to a change in an input variable, a race condition exists. The condition has its origin in the different delays when the A and B signals are generated. In the case described above the races identified are both non-critical races since, irrespective of the transition made, the circuit always ends up in the same stable state.

However, there are races that can occur in event-driven circuits in which the final state reached depends upon the order in which the state variables change. Such races are termed critical races. For example, the internal state diagram of a state machine and its corresponding state table are shown in Figure 9.4. It will be assumed that the machine is in the state defined by $AB = 00$, and $X = 0$. If the input X is now changed to 1, the machine will make a direct transition to the stable state defined by $X = 1$ and $AB = 11$ (S_1) providing A and B change simultaneously. Alternatively, if A changes before B, the machine will make a transition to the state defined by $X = 1$ and $AB = 10$. Since this is a stable state, the circuit will remain there, and in fact a quick glance at the state diagram shows that the circuit remains locked in that state indefinitely because of the absence of an output path from the state. However, B may change before A and then the circuit will make a transition to the state defined by $X = 1$ and $AB = 01$. This state is unstable and the circuit makes a further transition to the state defined by $X = 1$ and $AB = 11$. The transitions described can clearly lead to faulty circuit operation. Critical races occur in this circuit because it is possible to end up in one of two stable states, depending on the order in which the state variables change. The various transitions which can take place in this circuit are indicated on the state table shown in Figure 9.4(b).

9.6 Race free assignments

If critical races are to be avoided, it is necessary to provide a race-free assignment of the state variables on the state diagram. In effect, this means that when a transition is made from one state to the next, only one state variable should be allowed to change.

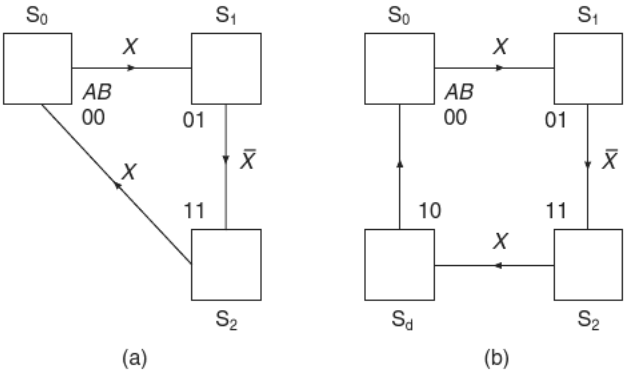


Figure 9.5 *(a) State machine requiring race-free secondary assignment (b) Inclusion of a dummy state to give race-free assignment*

In some cases it is not possible to satisfy this requirement without making modifications to the state diagram. For example, the three-state diagram shown in Figure 9.5(a) requires two state variables to define the three states. An arbitrary state assignment has been made on the diagram, but inspection reveals that on making a transition from S_2 to S_0 both secondary variables must change. Unfortunately, it is impossible to find a race-free assignment for a three-state diagram if transitions are required between each pair of states.

Furthermore, two state variables can define four states, which implies that for the three-state diagram of Figure 9.5(a) there is one unused state which has been omitted from the diagram. If there is no exit from the unused state it can become a 'lock-in' state as described in section 9.5.

These two problems are overcome by incorporating the unused state $AB = 10$ (S_d) in the modified state diagram shown in Figure 9.5(b). This modification allows the circuit to return unconditionally from this dummy state to state S_0.

The four-state diagram in Figure 9.6 is structured in such a way that there are no race problems providing adjacent states are allocated state variables that differ in one variable only. If, however, the state diagram for the machine includes transitions between two states that are not adjacent, for example $S_3 \rightarrow S_1$ in the state diagram shown in Figure 9.7(a), then a race-free assignment is not possible with two state variables. The state diagram reveals that with the same state assignment as the one shown in Figure 9.6 there is a double change in state variables when the transition $S_3 \rightarrow S_1$ is made. No matter how the state variables are allocated, there will always be at least one transition which will result in a double change of the state variables, and this implies that a race-free assignment can only be achieved by using three state variables.

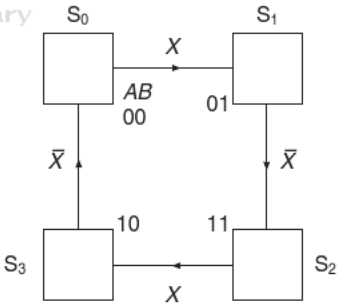


Figure 9.6 *State diagram for a four-state machine with transitions between adjacent states*

A race-free assignment can most easily be obtained from a K-map of the three state variables, as shown in Figure 9.7(b). It is a property of the K-map that adjacent cells differ in one digit position only, and consequently two states allocated to adjacent cells will have state assignments that differ in one digit place.

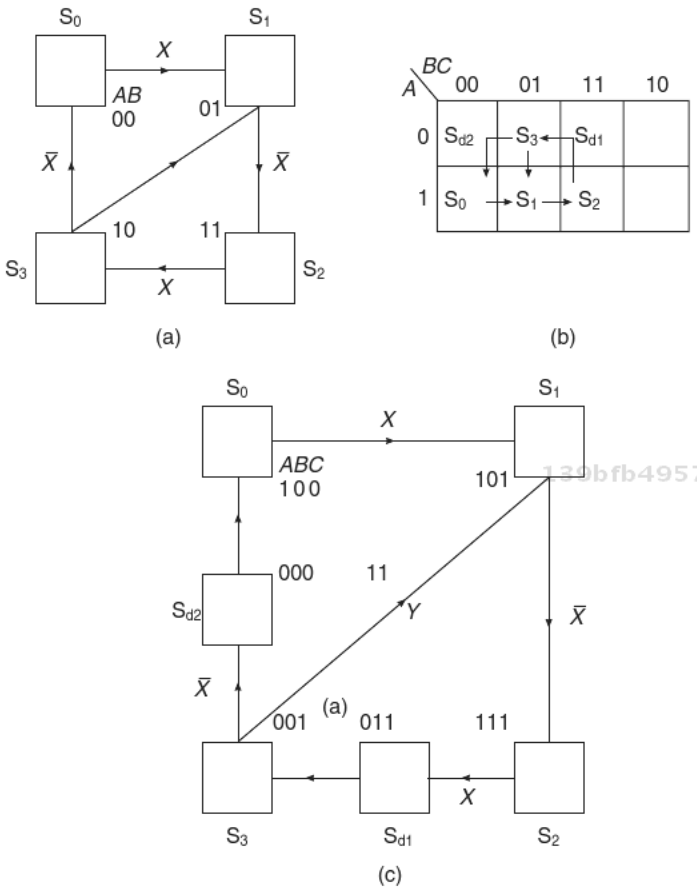


Figure 9.7 *(a) State diagram for a four-state machine with one diagonal transition (b) K-map for determining a race-free assignment (c) Race-free state diagram for a four-state machine having a diagonal transition*

Four of the states have been allocated to cells such that S_0 is adjacent to S_1, S_1 to S_2, and S_1 to S_3. However, for a race-free assignment, S_2 should be adjacent to S_3, and so should S_0. The K-map shows that with three state variables such adjacencies are impossible, and the transitions $S_2 \rightarrow S_3$ and $S_3 \rightarrow S_0$ have been made via the dummy states S_{d1} and S_{d2} respectively. The modified state diagram consists of six states, two of which are dummies, as shown in Figure 9.7(c). Each transition on this diagram has only one change of state variable, and hence the assignment is race-free. An event driven circuit will now be designed which requires the inclusion of a dummy state.

9.7 The pump problem

Step 1: Problem definition

Water is pumped into a water tank by two pumps, p_1 and p_2. Both pumps are to turn on when the water goes below level 1 and they are to remain on until the water reaches level 2, when pump p_1 turns off and remains off until the water is below level 1 again. Pump p_2 remains on until level 3 is reached when it also turns off and remains off until

the water falls below level 1 again. Level sensors are used to provide level detection signals as follows:

Signal $a = 1$ when the water is at or above level 1, otherwise $a = 0$
Signal $b = 1$ when the water is at or above level 2, otherwise $b = 0$
Signal $c = 1$ when the water is at or above level 3, otherwise $c = 0$

The aim is to develop an event driven circuit to control pumps p_1 and p_2 according to the specification given above.

A schematic diagram of the water tower is shown in Figure 9.8(a), and a block diagram of the proposed circuit is shown in Figure 9.8(b).

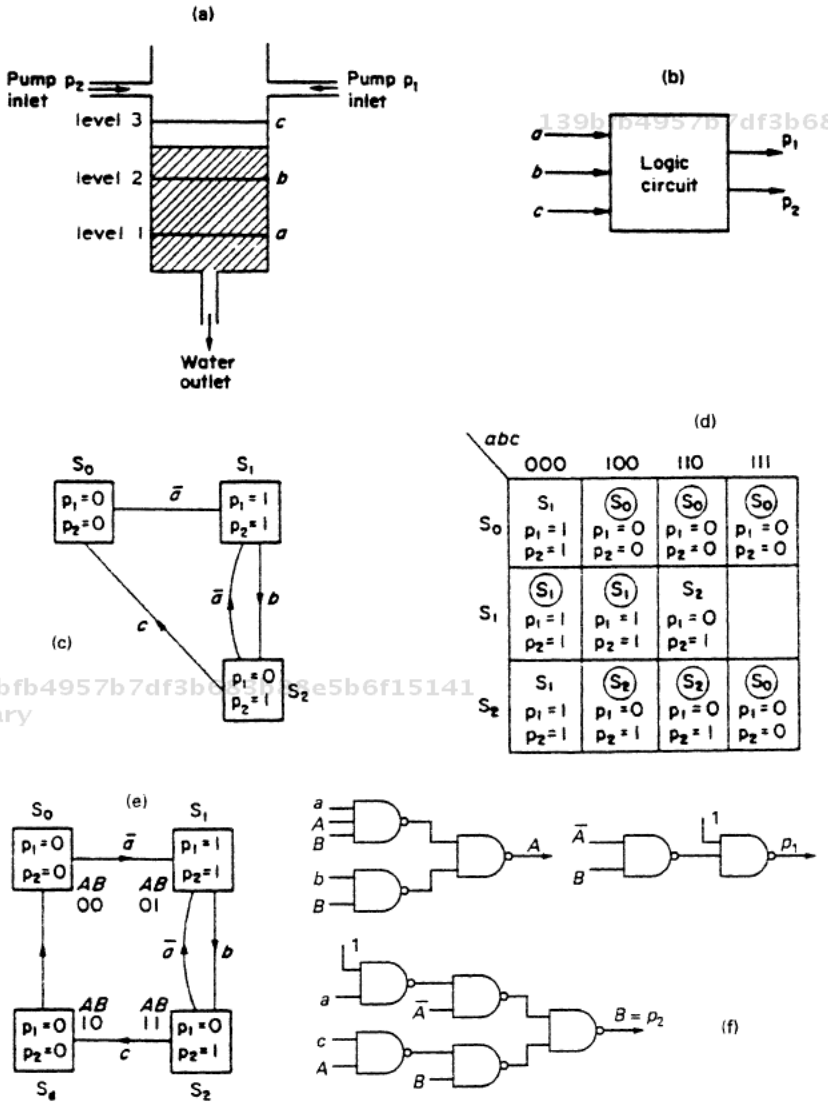


Figure 9.8 *(a) Diagram of the water pump problem (b) Block diagram of pump controller (c) Basic internal state diagram for the pump problem (d) State table (e) Modified state diagram (f) Circuit implementation of the pump controller*

Step 2: The state diagram

A suitable state diagram is shown in Figure 9.8(c), in which the state S_0 is related to the condition when the water is above level 3 and both pumps are off. As the tank empties, the water level falls until it is below level 1 and a transition is then made to S_1, since $\bar{a} = 1$. In state S_1, both pumps are on. If the water continues to rise and reaches level 2, a transition is made to S_2 and pump p_1 is then turned off. In state S_2, two options are available. If the water level falls below level 1 again a transition will be made back to S_1 on the signal $\bar{a} = 1$. Alternatively, if the water continues to rise, when level 3 is reached a transition is made to S_0 and both pumps are turned off.

Step 3: The state table

The state table for the pump problem is shown in Figure 9.8(d). It should be observed that input conditions $abc = 001$, 010, 011 and 101 are missing from the table since these combinations can only exist under fault conditions.

Two state variables A and B are required to define three states. Because there are transitions between each pair of states, a race-free assignment of the state variables is not possible. To overcome this problem an additional dummy state S_d is added to the state diagram. The modified state diagram is shown in Figure 9.8(e).

Step 4: Development of the circuit equations

Turn-on condition for $A = bB$

Turn-off condition for $A = \bar{B} + B\bar{a} = \bar{B} + \bar{a}$

Turn-on condition for $B = \bar{a}\bar{A}$

Turn-off condition for $B = cA$

$$\text{Hence } A^{t+\delta t} = [bB + \overline{(\bar{B}+\bar{a})}A]^t = [bB + aAB]^t$$

$$\text{and } B^{t+\delta t} = [\bar{a}\bar{A} + \overline{(c\bar{A})}B]^t = [\bar{a}\bar{A} + (\bar{c}+\bar{A})B]^t$$

$$\text{Also } p_1 = \bar{A}B$$

$$\text{and } p_2 = \bar{A}B + AB = B$$

Step 5: Circuit implementation

The circuit implementation of the pump controller is shown in Figure 9.8(f).

9.8 Design of a sequence detector

In this section a further example of the design of an event driven circuit will be studied to emphasise some of the problems faced by the designer when developing this type of circuit. The opportunity will also be taken to look at various methods of implementation and the construction of an ASM chart for this problem. The design to be studied concerns a sequence detector which has two inputs X_1 and X_2, and one output Z, as shown in Figure 9.9(a), and which is required to give an output $Z = 1$ when the sequence of input signals $X_1X_2 = 00$, 10, 11 has occurred.

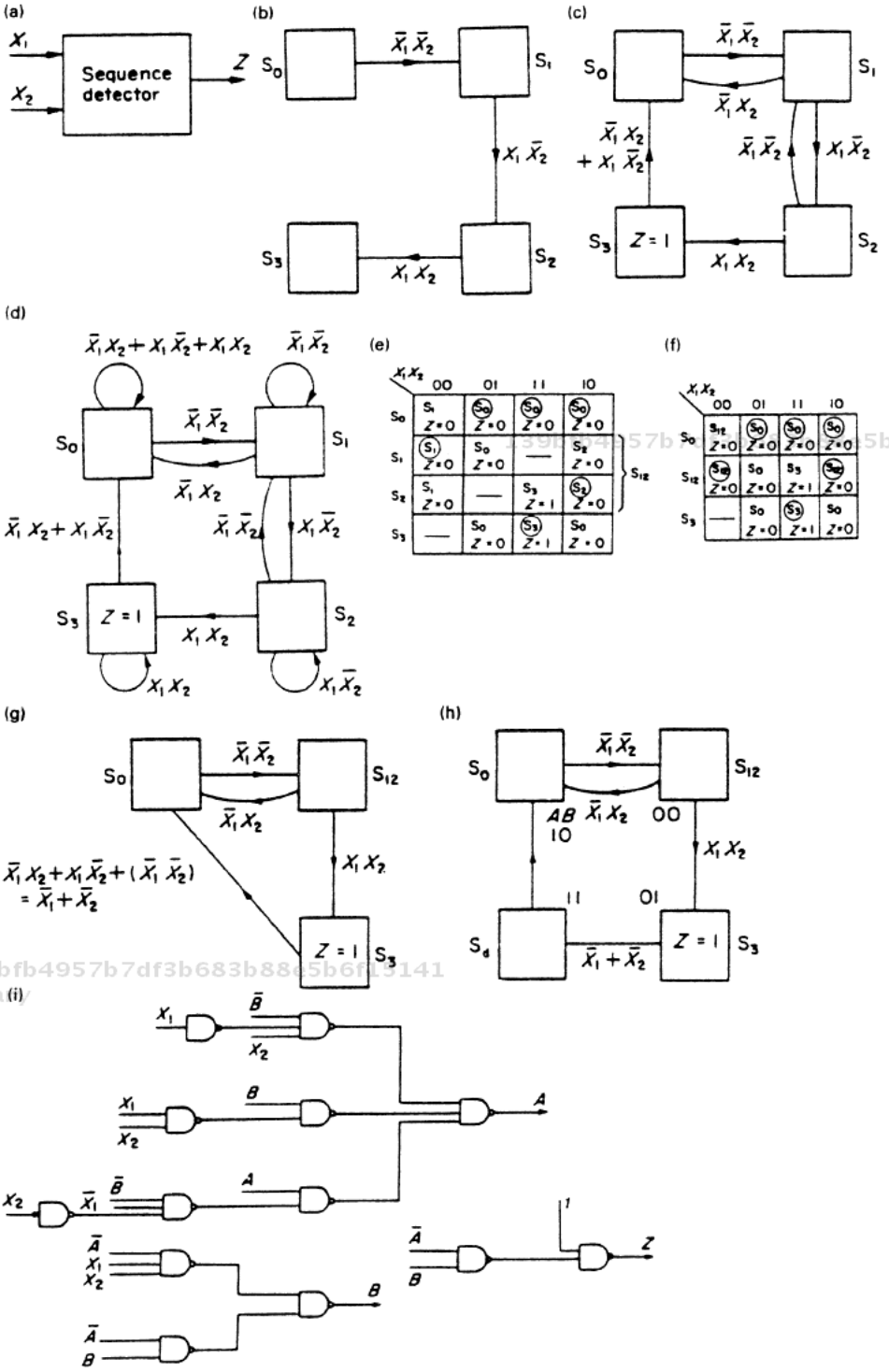


Figure 9.9 *(a) Block diagram of a sequence detector (b) Basic elements of the internal state diagram (c) Internal state diagram for the sequence detector (d) Complete internal state diagram including slings (e) State table (f) Reduced state table (g) Reduced state diagram (h) State diagram including dummy state and secondary assignment (i) Implementation of the sequence detector*

One method of approach open to the designer is to develop the state diagram. In this type of problem a good beginning to the state diagram is to insert the required sequence, as shown in Figure 9.9(b). This requires four states, connected via three transitions, initiated by the transition signals $\bar{X}_1\bar{X}_2$, $X_1\bar{X}_2$ and X_1X_2 respectively.

To complete the state diagram it is now necessary to insert the additional transition paths that may originate at each of the states. For example, the machine enters state S_1 on the transition signal $\bar{X}_1\bar{X}_2$. Since the machine to be designed will be operating in the fundamental mode, there cannot be a simultaneous change in the input variables when state S_1 is entered and the state can only be left on the transition signals $X_1\bar{X}_2$ or $\bar{X}_1X_2$. The transition signal $X_1\bar{X}_2$ represents the second combination of the input signals in the required sequence, and initiates the transition from S_1 to S_2. Alternatively, a change in X_2 from $0 \rightarrow 1$ results in an input signal $\bar{X}_1X_2$ and the machine should be designed to return to the state S_0 to await the arrival of the first signal in the sequence, $\bar{X}_1\bar{X}_2$. The completed state diagram is shown in Figure 9.9(c). In this diagram, the output $Z = 1$ appears in state S_3 at the completion of the required sequence. If, when in this state, the input signals $\bar{X}_1X_2$ or $X_1\bar{X}_2$ are received, the machine will return to S_0, where it will await the next occurrence of the signal $\bar{X}_1\bar{X}_2$, the first combination of the required sequence.

Some designers insert slings (arrows indicating a "transition" to the same state) on the state diagram, and an example of the use of slings has already appeared in Figure 9.3(d). In this problem, if the machine enters state S_1 on the signal $\bar{X}_1\bar{X}_2$, it will stay there as long as this signal still exists, and this can be indicated by a sling originating from and terminating on S_1, as shown in Figure 9.9(d). This diagram includes all possible slings, and it will be observed that when in S_0 the sling signal is $\bar{X}_1X_2 + X_1\bar{X}_2 + X_1X_2$. This means that if the machine entered S_0 on either of the signals $\bar{X}_1X_2$ or $X_1\bar{X}_2$ it would be possible to get a change of input signal from either $\bar{X}_1X_2$ to X_1X_2 or, alternatively, from $X_1\bar{X}_2$ to X_1X_2. If such a sequence of events occurs, the machine will remain in state S_0 and will only leave the state if the input signals X_1 and X_2 change in either of the following two sequences:

1. $11 \rightarrow 01 \rightarrow 00$
2. $11 \rightarrow 10 \rightarrow 00$

The state table is constructed from the information given on the state diagram, and is shown in Figure 9.9(e). Examination of the table shows that rows S_1 and S_2 are mergeable and the table can be reduced to three rows. At first sight this may appear to be a disadvantage for the following two reasons. First, it leads to the presence of an unused state, and second, since the state diagram will now only consist of three states, a race-free assignment is not possible. However, the unused state can be reintroduced as a dummy state having an unconditional transition to the next state. The presence on the state diagram of an unconditional transition will lead to simpler turn-on and turn-off conditions and a simpler logic implementation.

The reduced state table is shown in Figure 9.9(f) and it will be noticed that there is one unoccupied cell on this diagram on the row headed S_3. This is effectively a 'can't happen' condition. If the present state is S_3, then a transition signal $\bar{X}_1\bar{X}_2$ is forbidden. Since this signal cannot occur when the machine is in state S_3 it may be used as an optional term added into the Boolean equation for the $S_3 \rightarrow S_0$ transition, as shown in the reduced state diagram of Figure 9.9(g). In this case the optional term leads to a simplification of the transition signal.

The state diagram, including the dummy state and with a suitable state assignment, is shown in Figure 9.9(h). The turn-on and turn-off equations are taken directly from this diagram:

Turn-on condition for $A = \bar{B}\bar{X}_1X_2 + B(\bar{X}_1 + \bar{X}_2)$
Turn-off condition for $A = \bar{B}\bar{X}_1\bar{X}_2$

$$A^{t+\delta t} = [\bar{B}\bar{X}_1X_2 + B(\bar{X}_1 + \bar{X}_2) + (\overline{\bar{B}\bar{X}_1\bar{X}_2})A]^t$$
$$= [\bar{B}\bar{X}_1X_2 + B(\bar{X}_1 + \bar{X}_2) + (B + X_1 + X_2)A]^t$$

Turn-on condition for $B = \bar{A}X_1X_2$
Turn-off condition for $B = A$
$$B^{t+\delta t} = (\bar{A}X_1X_2 + \bar{A}B)^t$$

The output Z is given by $Z = S_3 = \bar{A}B$, and the machine implementation is shown in Figure 9.10(i).

The simplest form of the equations for the next state of the state variables $A^{t+\delta t}$ and $B^{t+\delta t}$ can be obtained directly from a pair of K-map plots. The state table compiled from the information given in the state diagram of Figure 9.9(h) is shown in Figure 9.10(a) and in this diagram, assignment of A^t and B^t has been placed alongside the states. However, it is more convenient to rearrange the rows of this table so that the secondary variables appear in normal K-map order. At the same time the state entries in the cells in Figure 9.10(a) are replaced by the state variables that define them, as shown in Figure 9.10(b). This table can be regarded as a plot of the next states of the state variables, $A^{t+\delta t}$ and $B^{t+\delta t}$, for every possible combination of the total present state $(X_1X_2\ AB)^t$.

The K-maps for $A^{t+\delta t}$ and $B^{t+\delta t}$ have been separated out in Figure 9.10(c). Using the normal simplification techniques gives the following equations:

$$A^{t+\delta t} = (\bar{X}_1X_2 + B\bar{X}_2 + AX_1)^t$$
$$B^{t+\delta t} = (\bar{A}X_1X_2 + \bar{A}B)^t$$

and

$$Z = S_3 = \bar{A}B$$

The implementation of these three functions is shown in Figure 9.10(d).

An ASM chart for the sequence detector is given in Figure 9.10(e). The chart has been constructed from the information extracted from the state table in Figure 9.10(a). For example, the decision box immediately below the state box S_0 contains the Boolean term $\bar{X}_1\bar{X}_2$. If $\bar{X}_1\bar{X}_2 = 0$ the machine remains in S_0, but if $\bar{X}_1\bar{X}_2 = 1$ a transition is made to the state S_{12}. The condition for the machine to remain in state S_{12} is $\bar{X}_1\bar{X}_2 + X_1\bar{X}_2 = \bar{X}_2 = 1$, i.e. $X_2 = 0$. On leaving state box S_{12} the first decision box contains the variable X_2, and if $X_2 = 0$ the path from this decision box returns the machine to state S_{12}. If, however, $X_1 = 0$ and $X_2 = 1$ the machine takes the path back to S_0 while if $X_1 = 1$ and $X_2 = 1$ the path on the chart leads to state box S_3. The remainder of the chart is constructed using the information obtained from the last two rows of the state table.

An alternative method of implementing the design of the sequence detector would be to use SR latches and combinational logic. This requires the development of the next state equations for the two latches, A and B. The tabulation of the next state functions,

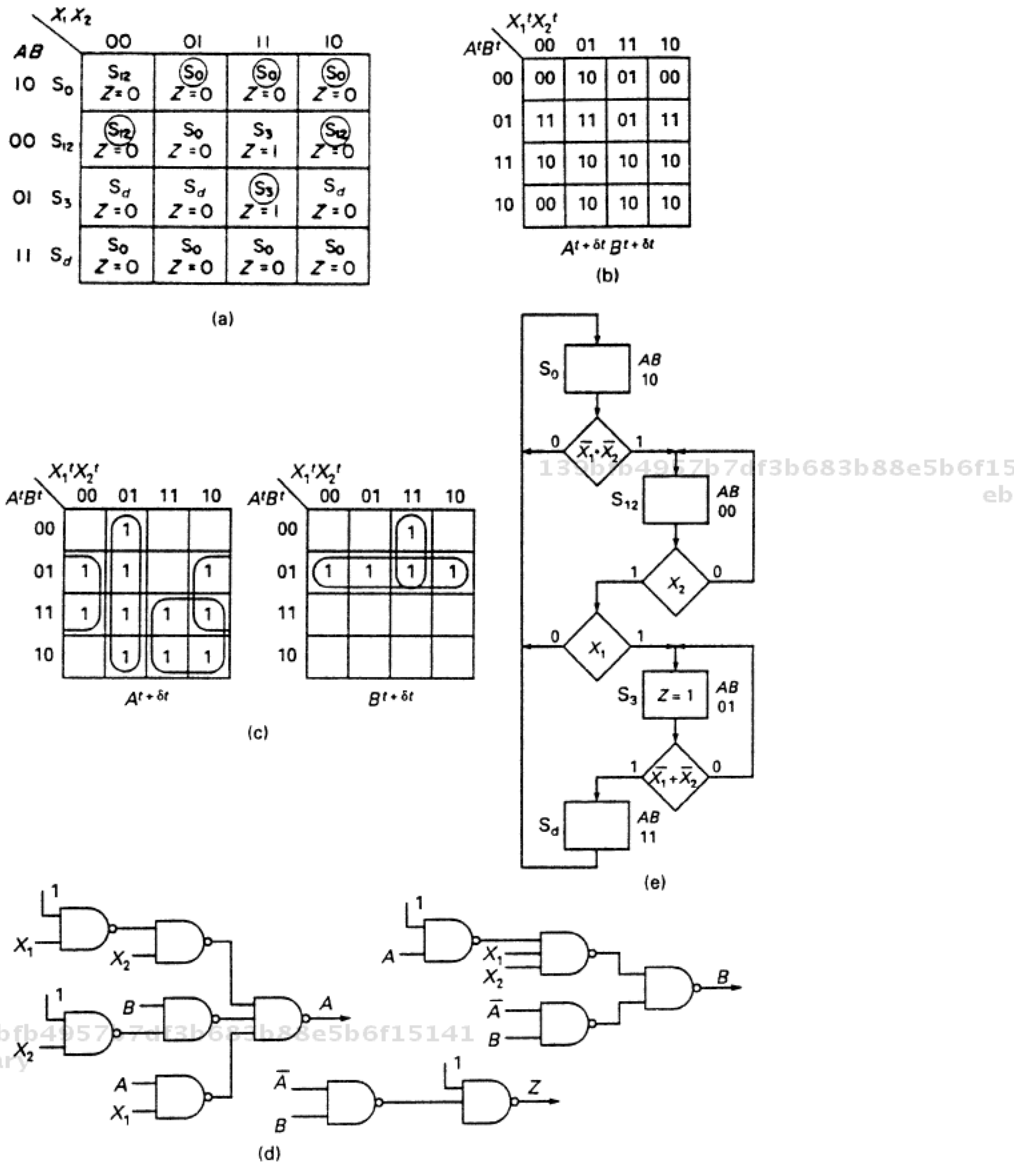


Figure 9.10 *(a) State table for sequence detector (b) Tabulation of next state functions $A^{t+\delta t}$ and $B^{t+\delta t}$ (c) K-maps for the next state functions (d) Alternative implementation of sequence detector (e) The ASM chart for the sequence detector*

repeated again for convenience in Figure 9.11(a), may be regarded as the next state map for the two latches which, in conjunction with the steering table for the SR latch shown in Figure 9.11(b), enables the designer to obtain the K-maps for the S and R inputs to both latches. For example, when the present total state of the machine is $ABX_1X_2 = 0001$ and the input combination $X_1X_2 = 01$ is received, the next total state is $ABX_1X_2 = 1001$. Latch A has made a $0 \rightarrow 1$ transition which requires $S_A = 1$ and $R_A = 0$, while latch B has made a $0 \rightarrow 0$ transition which requires $S_B = 0$ and $R_B = \text{X}$.

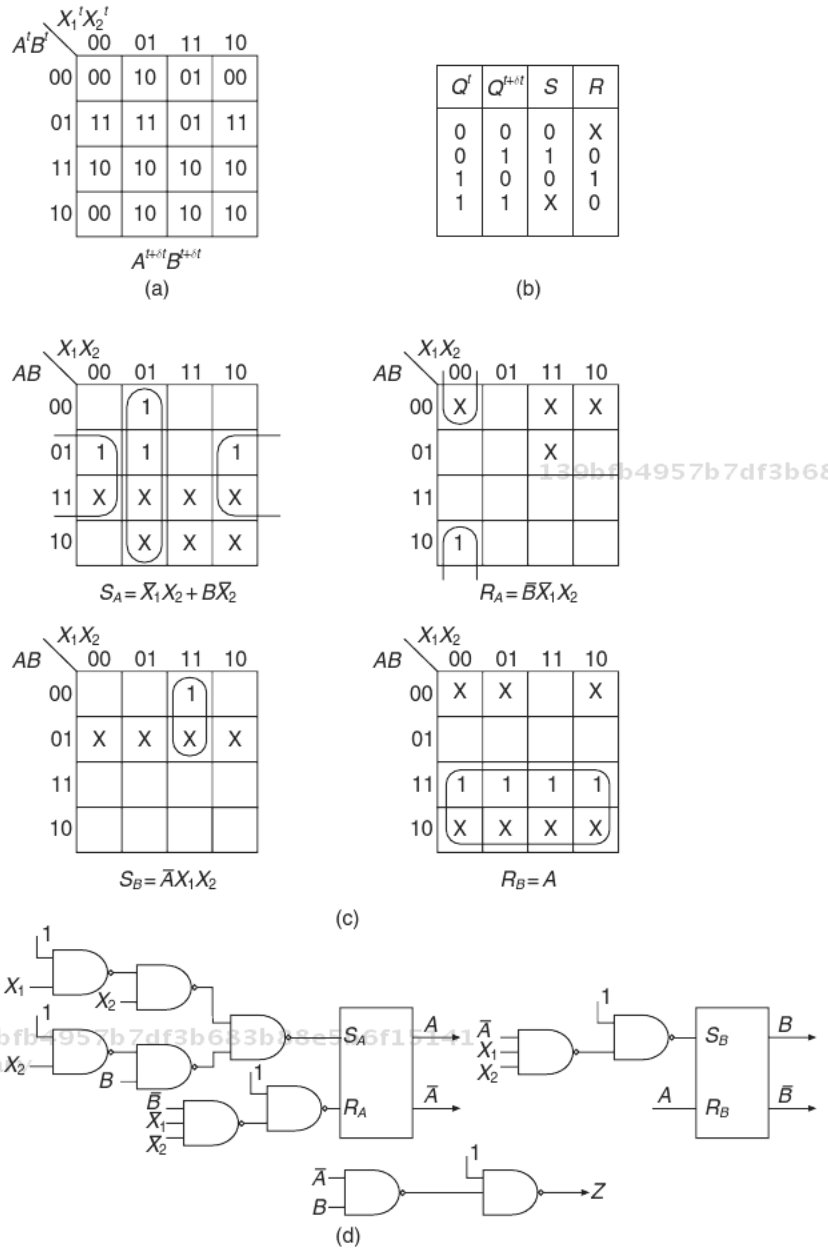


Figure 9.11 *(a) Latch excitation table (b) Steering table for an SR Latch (c) K-maps for the latch input signals (d) Implementation of sequence detector using SR latches*

The K-maps for the latch input signals are shown in Figure 9.11(c) and, after simplification, the following equations are obtained for the set and reset signals:

$$S_A = \bar{X}_1 X_2 + B\bar{X}_2 \qquad S_B = \bar{A} X_1 X_2$$
$$R_A = \bar{B}\bar{X}_1 \bar{X}_2 \qquad R_B = A$$

The output is given by $Z = \mathrm{S}_3 = \bar{A}B$ and the machine implementation is shown in Figure 9.11(d).

9.9 State reduction for incompletely specified machines

In a completely specified machine there is an entry for the next state and output in every cell of the state table. For incompletely specified machines the outputs and the next states may not be specified for some combinations of the present states and inputs. Unspecified next states and outputs can be regarded as 'can't happen' conditions and can be specified in any way the designer may choose. Because of this freedom of choice it is possible to have more than one state reduction for an incompletely specified machine. A state table for an incompletely specified machine and two possible state reductions for the machine are shown in Figure 9.12.

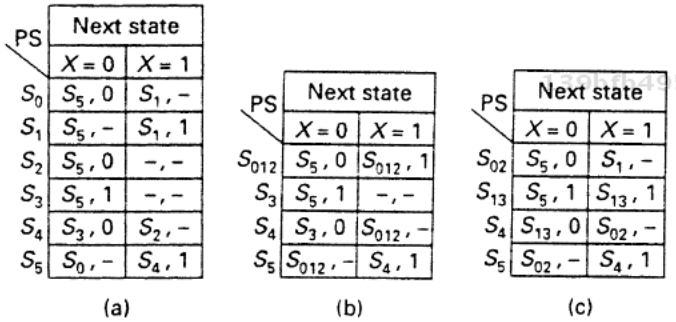


Figure 9.12 *(a) State table for an incompletely specified machine (b) and (c) Two possible state reductions*

9.10 Compatibility

In the previous chapter, when dealing with completely specified state tables, state reduction was achieved by combining equivalent states to form a single state. Equivalent states were defined as those states where next-state entries and outputs were identical for each input condition. When dealing with incompletely specified tables such as the one shown in Figure 9.12, state reduction is achieved by finding compatible states.

As an example of compatibility, consider the two states S_2 and S_3 tabulated below which appear in the state table of an incompletely specified machine:

PS	NS	
	$X = 0$	$X = 1$
S_2	$S_5, 0$	$-, -$
S_3	$-, -$	$S_4, 1$

The unspecified next state and outputs are regarded as 'can't happen' conditions, and entries in the tabulation can be made where the –'s occur which will enable the two states to be combined to form a new state S_{23} where:

PS	NS	
	$X = 0$	$X = 1$
S_{23}	$S_5, 0$	$S_4, 1$

The two states have formed a *compatible pair*. If the output conditions for S_2 and S_3 had been conflicting for either of the two input conditions they would then have been incompatible. Alternatively, if the next state entries for either of the two input conditions had been different, the two states would have been incompatible.

As a further example of compatibility, the two rows tabulated below have been taken from an incompletely specified table:

PS	NS	
	$X = 0$	$X = 1$
S_0	$S_1, 0$	$S_3, -$
S_1	$S_1, -$	$S_6, 1$

The two rows can be made output consistent by inserting a 1 in place of the '–' on the first row, and a 0 in place of the '–' on the second. However, after these two insertions, S_0 and S_1 can only form a compatible pair providing S_3 and S_6 are also compatible.

Additionally, the compatibility relationship is not transitive. It is possible for S_i to be compatible with S_j and S_j may also be compatible with S_k, but it does not follow that S_i will be compatible with S_k. This point is demonstrated by the following example:

PS	NS	
	$X = 0$	$X = 1$
S_1	$-, 0$	$S_6, 0$
S_2	$S_5, 0$	$-, -$
S_3	$-, -$	$S_4, 1$

Clearly S_1 and S_2 form a compatible pair. Similarly, S_2 and S_3 are compatible. However, S_1 and S_3 are incompatible since they are not output consistent.

Summarising, the conditions for the compatibility of two states S_i and S_j are:

1. The outputs on the rows headed by S_i and S_j must be identical for each possible input condition.
2. The next-state entries of S_i and S_j must be compatible when both are specified for each possible input.

A set of output consistent states in which every pair within the set is compatible is called a *compatibility class* while a *maximal compatibility class* is defined as a set of compatible states which are output consistent but are not a subset of any other class. For example, if (S_1, S_5) is a compatibility class, it is not a maximal compatibility class if it is a subset of a maximal compatibility class (S_1, S_4, S_5, S_7).

9.11 Determination of compatible pairs

To determine the compatible pairs for the incompletely specified machine whose state table is shown in Figure 9.13(a), the implication chart described in the previous chapter is used. For an incompletely specified machine, each cell in the chart represents the testing ground for the compatibility of a state pair. The top left-hand cell of the chart is the testing ground for the compatibility of states A and B. In order that the two states should be output consistent, all the output '–'s in the two rows must be replaced by 1's. To satisfy the second condition, states A and C must be compatible, and this implication is entered in the cell.

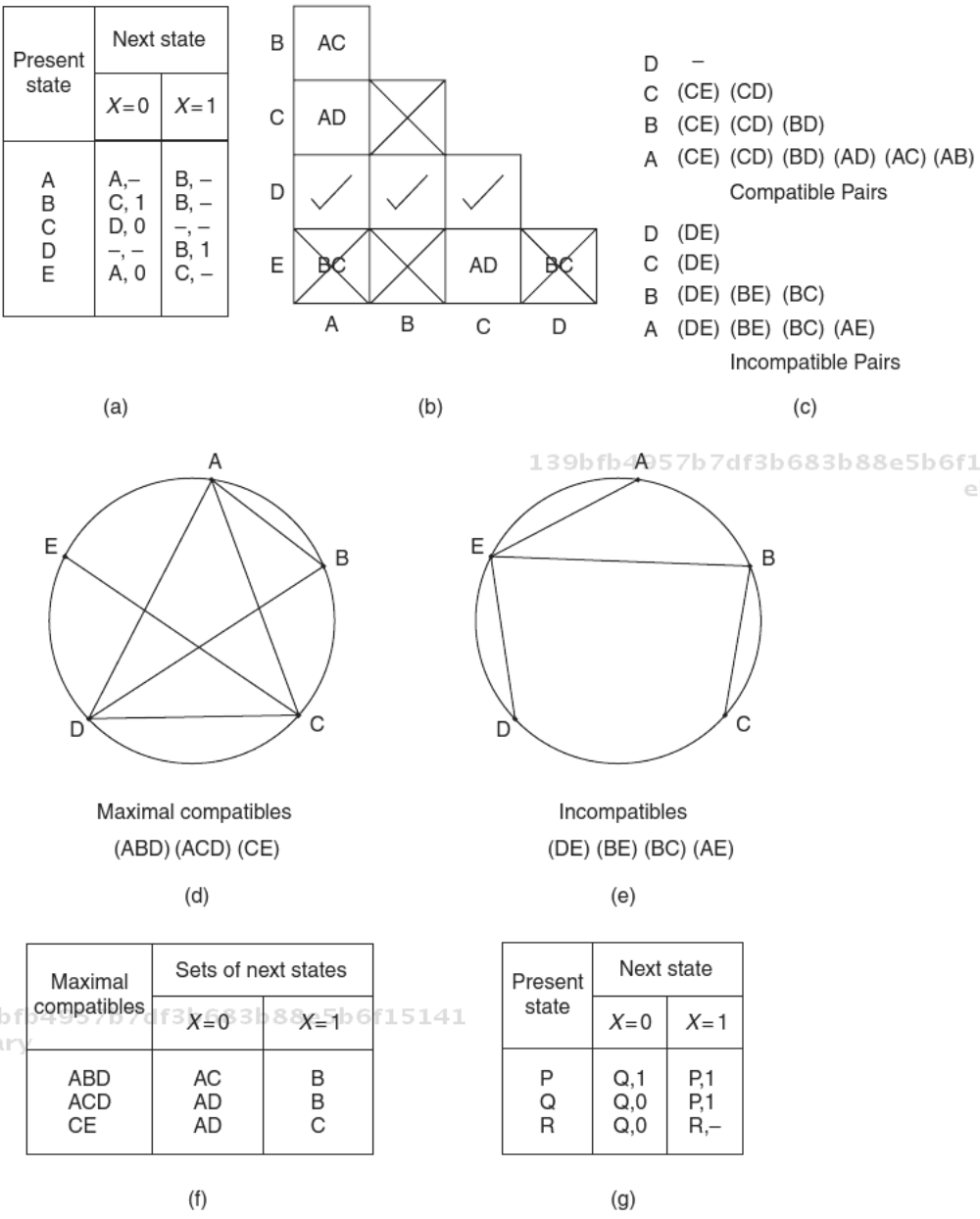


Figure 9.13 *(a) State table for incompletely specified machine (b) implication table (c) compatible and incompatible pairs (d) and (e) merger diagrams (f) closure table (g) reduced state table*

It is clear from an examination of the state table that states B and C cannot be compatible since they are not output consistent and the cell at the intersection of these two states is marked with an X. On the other hand, states A and D can be made output consistent by replacing the output '–'s, on the rows headed by A and D, with 1's. The second condition is satisfied if the next state '–' for $X = 0$ in the row headed D is replaced by A. The two states are then compatible, and the cell that identifies them is marked with a $\surd$. Every cell on the chart is examined in this way, and the

appropriate entry is made in each cell, as shown in Figure 9.13(b). Finally, the chart must be examined systematically to see if any of the implications involve a pair of states that have already been found to be incompatible. For example, the entry BC in the cell at the intersection of D and E is an incompatibility, and this cell must be marked with an X. Similarly, the entry BC at the intersection of A and E is an incompatibility and must also be marked with an X. The states are now listed in reverse order as shown in Figure 9.13(c), and the implication table is examined column by column from right to left to determine first the compatible and then the incompatible pairs. The compatible pairs are:

(CE)(CD)(BD)(AD)(AC)(AB)

and the incompatible pairs are:

(AE)(BE)(BC)(DE)

9.12 The merger diagram

The next step in the state reduction process is to find the maximal compatibles, and this process can be assisted by the construction of a merger diagram. In this diagram the original states of the machine can be represented by dots equally spaced round a circle as shown in Figure 9.13(d). A line is then used to connect each of the compatible pairs. The maximal sets of compatible states can be obtained from the merger diagram by noting those sets of states in which every state is connected to every other state by a line. A typical example of a maximal compatible in Figure 9.13(d) is (ACD) and it will be observed that it is impossible to add any other state to this triangular grouping. The remaining maximal compatibles on the merger diagram are (ABD) and (CE). The maximal incompatibles can also be found on the incompatible merger diagram shown in Figure 9.13(e). They are (AE), (BC), (BE) and (DE).

9.13 The state reduction procedure

The maximal compatibles are now selected to provide a reduced state table which will represent the behaviour of the incompletely specified machine. When making the selection, three conditions must be satisfied:

1. *Completeness*: The chosen set of maximal compatibles must contain all the states in the original machine.
2. *Consistency*: The set of chosen maximal compatibles must be closed. This condition is satisfied if the implied next states of each selected maximal compatibility is contained by another maximal compatibility within the selected set.
3. *Minimality*: The smallest number of maximal compatibles required for a minimal realisation.

The process of selecting a set of maximal compatibles to represent the machine, whose incompletely specified state table is shown in Figure 9.13(a), is one of trial and error. In this problem, all three maximal compatibles will be selected in order to satisfy completeness and consistency. Hence the reduced state table will consist of three states, P = ABD, Q = ACD and R = CE. The reduced state table is shown in Figure 9.13(g).

9.14 Circuit hazards

One cause of malfunction in combinational and sequential circuits can be traced to the presence of race hazards. The designer should have a clear understanding of the mechanism that produces such hazards and should also be aware of their effects on circuit performance.

There are four types of hazard which can occur in digital systems:

1. Static hazards
2. Dynamic hazards
3. Function hazards
4. Essential hazards

Static hazards are due to a momentary change in output caused by an input change that does not affect the steady-state output. They may be present in both combinational circuits and gate-implemented asynchronous circuits. Dynamic hazards occur when, due to a single output change, the output changes several times before reaching its steady state value. Function hazards occur when more than one input variable change takes place at the same time, while essential hazards are peculiar to fundamental mode sequential circuits and they cannot be eliminated without controlling the delays in the circuit.

9.15 Gate delays

If a two-input NAND gate is used as an inverter in a combinational network, as illustrated in Figure 9.14, there will be a finite time delay t_g before any change at the input to the gate produces the required change at the output. This delay is demonstrated in the timing diagrams, where the change in A from 0 to 1 is followed by a change in $\bar{A}$ from 1 to 0, t_g seconds later. Similarly, when A changes from 1 to 0, the corresponding change in $\bar{A}$ from 0 to 1 also occurs later (see Chapter 4).

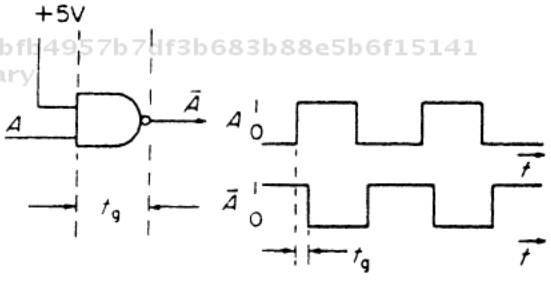


Figure 9.14 *The effect of gate delays when inverting a signal A*

9.16 The generation of spikes

If the signal A and its complement $\bar{A}$, generated by the NAND gate shown in Figure 9.14, are both fed to the inputs of a two-input AND gate as shown in Figure 9.15, then according to the laws of Boolean algebra the output of the gate should be $A \cdot \bar{A} = 0$ at all times. However, it will be observed from an examination of the timing diagrams that in the time periods that have been shaded, A and $\bar{A}$ are simultaneously equal to 1, so that

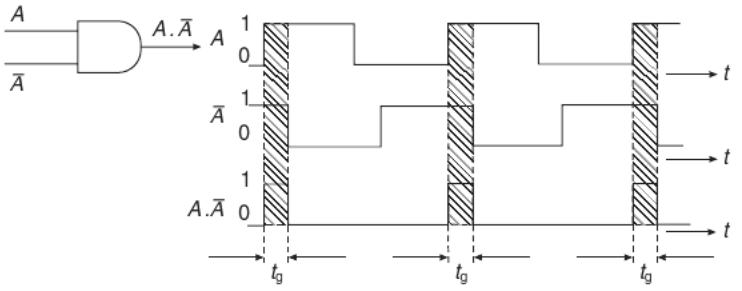


Figure 9.15 *Generation of spikes by an AND gate*

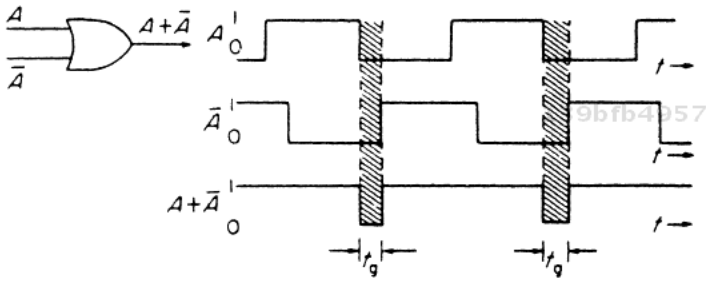


Figure 9.16 *Generation of spikes by an OR gate*

during these periods the gate output is $A \cdot \bar{A} = 1$. The output of the gate, $A\bar{A}$, consists of a series of positive going spikes which are initiated when A is changing from 0 to 1, each of time duration t_g, the gate delay of the inverter shown in Figure 9.14. The circuit used to generate the signal $A \cdot \bar{A}$ is said to exhibit a *static 0-hazard* because the output signal, which should be permanently 0, goes to 1 for a short transient period.

Alternatively, if the signals A and $\bar{A}$ are applied to the inputs of a two-input OR gate as shown in Figure 9.16 then the output of the gate is $A + \bar{A}$, which, according to the laws of Boolean algebra, should be 1 at all instants of time. The waveforms of A and $\bar{A}$ (see Figure 9.16) show that during the shaded time periods, they are both simultaneously equal to 0. In these shaded time periods, which are of short time duration, the output goes to 0. The circuit is said to exhibit a *static 1-hazard* because its output, which is normally 1, goes to 0 for short time periods. It will be observed that for the OR gate, the negative-going spikes are initiated at the instant when A is changing from 1 to 0.

The generation of spikes by NAND and NOR gates is illustrated in Figure 9.17. Negative-going spikes are generated by a NAND gate at the instant when A is

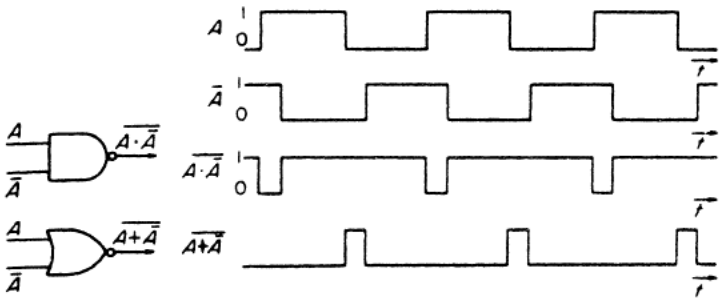


Figure 9.17 *Generation of spikes by NAND and NOR gates*

changing from 0 to 1. The circuit exhibits a static 1-hazard. In the NOR circuit, positive-going spikes are generated at the instant when A is changing from 1 to 0. This circuit exhibits a static 0-hazard.

9.17 The generation of static hazards in combinational networks

When an input to a combinational network is changing, spikes may be generated at the output of the circuit. The spikes, when they occur, are due to different path lengths in the network which introduce different time delays. For example, the Boolean function

$$f = AB + \bar{A}C$$

may be implemented by NAND gates, as shown in Figure 9.18. There are two paths through the circuit, the first via g_1, g_2 and g_3 and the second via g_4 and g_3. If it is assumed that all gates have exactly the same time delay, then it is apparent that the delay through the first path is greater than the delay through the second path.

The changes taking place in the circuit are illustrated in Figure 9.18 for the circuit condition $B = 1$, $C = 1$ and A, changing from 1 to 0. For this change in A, the output of g_4 changes from 0 to 1 and produces a change in the output of g_3 from 1 to 0. For the other path through the circuit, the output of g_1 first changes from 0 to 1, followed by the output of g_2 changing from 1 to 0, thus producing a change in the output of g_3 from 0 to 1. Because the g_4, g_3 path has the shorter time delay, it is clear that the change in output propagated along this path occurs earlier in time than the change propagated along the alternative path.

Since it has been assumed that $B = C = 1$, the network equation reduces to $f = A + \bar{A}$. When a circuit equation, under certain specified input conditions, reduces to this form, a static 1-hazard will be generated. In the example chosen here, the timing diagrams shown in Figure 9.18 reveal that due to the inverter delay, for a short period of time both A and $\bar{A}$ are equal to 0, and $A + \bar{A} = 0$. Providing the condition $B = C = 1$ is maintained and the input signal consists of a train of positive-going pulses, a series of negative-going spikes will be generated. The presence of the negative-going spikes confirms the earlier deduction, made by following the signal changes through the circuit diagram, that the output changes are $1 \rightarrow 0 \rightarrow 1$.

The dual function of $f = AB + \bar{A}C$ is:

$$f_d = (A + B)(\bar{A} + C)$$

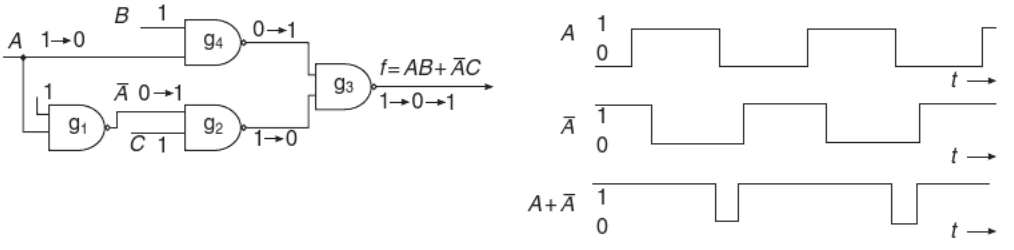


Figure 9.18 *The production of a static 1-hazard in a combinational network*

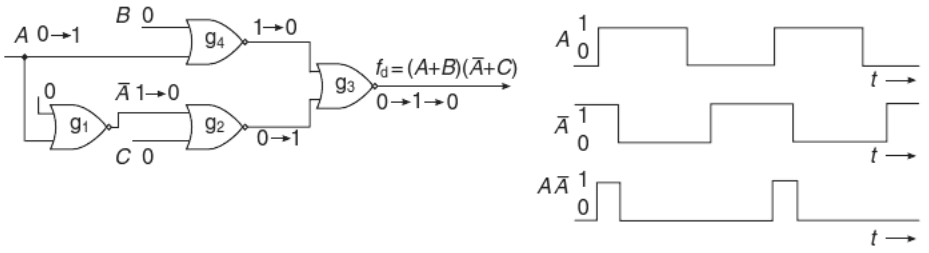


Figure 9.19 *The production of a static 0-hazard in a combinational network*

The implementation of this function using NOR gates is shown in Figure 9.19. When $B = C = 0$ the circuit equation reduces to $f_d = A \cdot \bar{A}$. Under the conditions specified a static 0-hazard will be generated when A is changing from 0 to 1. The production of the static 0-hazard is illustrated in Figure 9.19. Immediately after A changes from 0 to 1 both A and $\bar{A}$ are simultaneously 1, hence $A \cdot \bar{A} = 1$. The output remains at this value until $\bar{A}$ falls to 0 when $A \cdot \bar{A}$ resumes its value of 0 again.

Signal changes are also illustrated in Figure 9.19, where it has been assumed that $B = C = 0$ and that A consists of a stream of positive-going pulses. If all the gates have the same time delay, then path g_4, g_3 has the shortest time delay and the change in output due to A changing from 0 to 1 will propagate along this path faster than along path g_1, g_2, g_3. This results in the output changing from 0 to 1. When the corresponding change arrives at the output along the alternative path, the output changes back to zero again.

A similar analysis can be carried out for both the AND/OR and OR/AND configurations, and this will show that the AND/OR circuit implementing the function $f = AB + \bar{A}C$ will generate a static 1-hazard. Similarly, for the OR/AND circuit implementing the function $f_d = (A + B)(\bar{A} + C)$, it can be shown that a static 0-hazard will be generated.

9.18 The elimination of static hazards

The equation of the NAND circuit shown in Figure 9.18 is $f = AB + \bar{A}C$. The consensus product for this equation is BC, and this can be added to the original equation without altering its value. Thus:

$$f = AB + \bar{A}C + BC$$

and for the condition $B = C = 1$ the equation reduces to $f = A + \bar{A} + 1$, and even if A and $\bar{A}$ are, for a short period of time, simultaneously equal to 0, the value of the function f remains at 1.

The effect of adding the consensus product can be studied by examining the K-map plot of the function before and after the addition of the consensus product. The original function is shown plotted in Figure 9.20(a) and the plot of the function, after the inclusion of the consensus product, is shown in Figure 9.20(b). Comparison of the two plots shows that before the addition of the consensus product, there are two 1's in adjacent cells not covered by the same prime implicant. On covering these two adjacent 1's by the same prime implicant, as in Figure 9.20(b), the hazard is removed from the circuit.

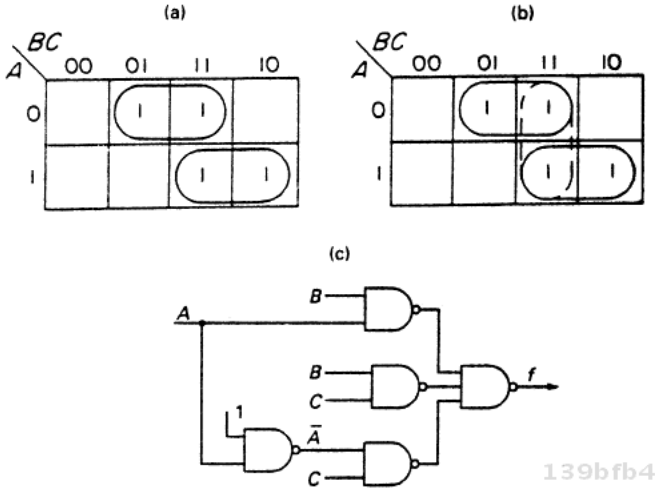


Figure 9.20 *(a) Plot of $f = AB + \bar{A}C$ (b) Plot of $f = AB + \bar{A}C + BC$ (c) Implementation of the hazard-free function $f = AB + \bar{A}C + BC$*

It follows that static 1-hazards can be detected by looking for adjacent 1's on a K-map plot of the function that are not covered by the same prime implicant. They can then be removed at the design stage by including additional prime implicants which cover adjacent 1's not otherwise covered by the same prime implicant.

The hazard-free circuit for the Boolean function $f = AB + \bar{A}C$ is shown in Figure 9.20(c), and it will be observed that an additional NAND gate has been introduced for generating the required consensus product BC.

For the NOR circuit of Figure 9.19, $f_d = (A + B)(\bar{A} + C)$. The consensus term for this equation is $(B + C)$, and this can be included in the above equation without altering its value, so that:

$$f_d = (A + B)(\bar{A} + C)(B + C)$$

If $B = C = 0$ then:

$$f_d = A \cdot \bar{A} \cdot 0$$

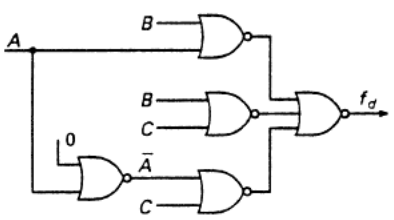


Figure 9.21 *Implementation of the hazard-free function $f_d = (A + B)(\bar{A} + C)(B + C)$*

With the inclusion of the consensus sum, the value of the function is always 0, irrespective of whether A and $\bar{A}$ are simultaneously equal to 1.

The static 0-hazard is eliminated by the inclusion of the consensus term $(B + C)$, and the resulting hazard-free circuit is shown in Figure 9.21. Elimination of the hazard requires the inclusion of an additional gate which generates the inverse of the consensus sum.

When looking for a static 0-hazard, a K-map plot of the function which identifies those combinations of the variables that cause the function value to be 0 is required. To obtain a plot of the 0-terms, the inverse of the function f_d must be plotted. The equation of the circuit is:

$$f_d = (A + B)(\bar{A} + C)$$

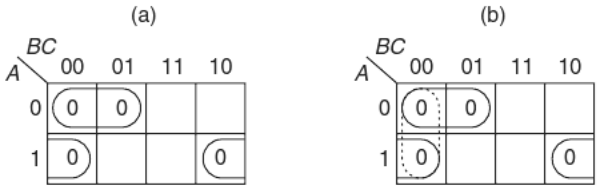


Figure 9.22 *(a) Plot of* $\bar{f} = \bar{A}\bar{B} + A\bar{C}$ *(b) Plot of* $\bar{f}$ *including consensus term for removing the hazard*

Inverting:

$$\overline{f_{\mathrm{d}}} = \bar{A}\bar{B} + A\bar{C}$$

The inverse function is shown plotted in Figure 9.22(a) and it will be noticed that the two 0's in the adjacent cells 000 and 100 are not covered by the same prime implicant. The function containing the additional prime implicant $\bar{B}\bar{C}$ becomes:

$$\overline{f_{\mathrm{d}}} = \bar{A}\bar{B} + A\bar{C} + \bar{B}\bar{C}$$

Inverting, $f_{\mathrm{d}} = (A + B)(\bar{A} + C)(B + C)$ which is the hazard-free function obtained previously by introducing the consensus term to the function equation.

The algorithm for finding static 0-hazards follows:

Step 1: Plot the inverse function.
Step 2: Look for adjacent 0's not covered by the same prime implicant.
Step 3: Insert additional prime implicants to cover all adjacent 0's that are not covered by the same prime implicant.
Step 4: Modify the inverse equation by including the additional prime implicants.
Step 5: Re-invert the equation to obtain the hazard-free form of the function.

9.19 Design of hazard-free combinational networks

In this section the function represented by the equation

$$f = \sum 2, 5, 6, 7, 10, 13, 15$$

will be implemented in hazard-free form using (a) NAND gates, and (b) NOR gates. A fan-in limitation of three will be imposed.

For the NAND implementation, the circuit has to be free of static 1-hazards. The first step in the design is to plot the K-map of the function and simplify in the normal way (see Figure 9.23). The plot is now examined to see if there are any 1's in adjacent cells not covered by the same prime implicant. In this case a pair of such cells are 0111 and 0110, and an additional prime implicant is added to the plot to eliminate the uncovered adjacency. The 1's that constitute the added prime implicant are enclosed by dotted lines on the K-map plot.

Reading from the map, the hazard-free function is:

$$f = BD + \bar{A}C\bar{D} + \bar{B}C\bar{D} + \bar{A}BC$$

To meet the fan-in restriction, the equation can be factorised and then:

$$f = C\bar{D}(\bar{A} + \bar{B}) + BD + \bar{A}BC$$

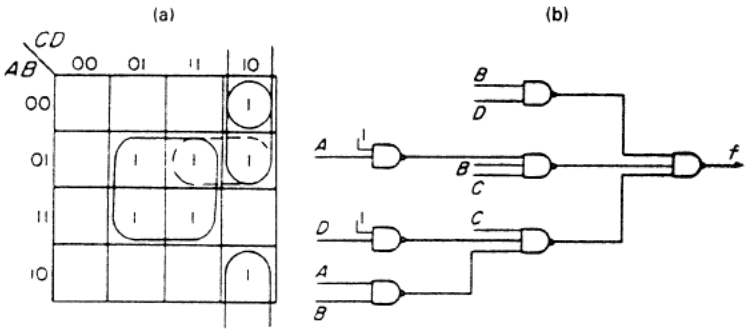


Figure 9.23 *(a) Plot of $f = \Sigma\ 2, 5, 6, 7, 10, 13, 15$ (b) NAND hazard-free implementation*

The factorisation of an equation in this way does not reintroduce hazards. In this problem the hazard would have occurred when $A = 0$, $B = 1$ and $C = 1$, with D changing from 1 to 0. Insertion of these conditions in the factorised equation gives:

$$f = \bar{D}(1 + 0) + D + 1$$
$$= \bar{D} + D + 1$$

which is the required condition for the removal of the hazard. The NAND implementation of the hazard-free function is shown in Figure 9.23(b).

To obtain the hazard-free NOR realisation, the inverse function is plotted and simplified. The inverse plot is derived from Figure 9.23(a) by marking the vacant cells on the map with 0's, as shown in Figure 9.24. The presence of 0's in adjacent cells not covered by the same prime implicant indicates that the simplified function will produce a static 0-hazard under certain prescribed conditions. In this case there are two such pairs of adjacent cells, (a) 0000 and 0001, and (b) 1000 and 1001. The introduction of an additional prime implicant $\bar{B}\bar{C}$, enclosed by dotted lines on the map, covers the uncovered adjacencies and eliminates the static 0-hazard. Reading the inverse function from the map:

$$\bar{f} = \bar{C}\bar{D} + \bar{B}D + \bar{B}\bar{C} + AB\bar{D}$$

and factorising to satisfy the fan-in restriction gives:

$$\bar{f} = \bar{C}\bar{D} + \bar{B}(\bar{C} + D) + AB\bar{D}$$

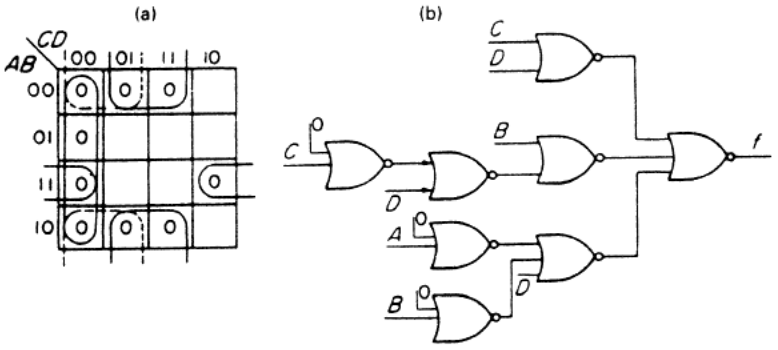


Figure 9.24 *(a) The 0-plot of $f = \Sigma\ 2, 5, 6, 7, 10, 13, 15$ (b) NOR hazard-free implementation*

and re-inverting:

$$f = (C + D)(B + C\bar{D})(\bar{A} + \bar{B} + D)$$

The implementation of this hazard-free function with NOR gates is shown in Figure 9.24(b).

9.20 Detection of hazards in an existing network

The network shown in Figure 9.25 is to be analysed to see if it has any static 0- or static 1-hazards. The equation of the network is :

$$f = AB\bar{C} + (A + B)(\bar{A} + \bar{D})$$

which may be expanded into the following form:

$$f = AB\bar{C} + A\bar{A} + A\bar{D} + \bar{A}B + B\bar{D}$$

This expression contains the term $A\bar{A}$ which, under normal circumstances, would be removed since, by the laws of Boolean algebra, its value is 0. Since the variables A and $\bar{A}$, in combinational networks can be simultaneously 1, they are treated as independent

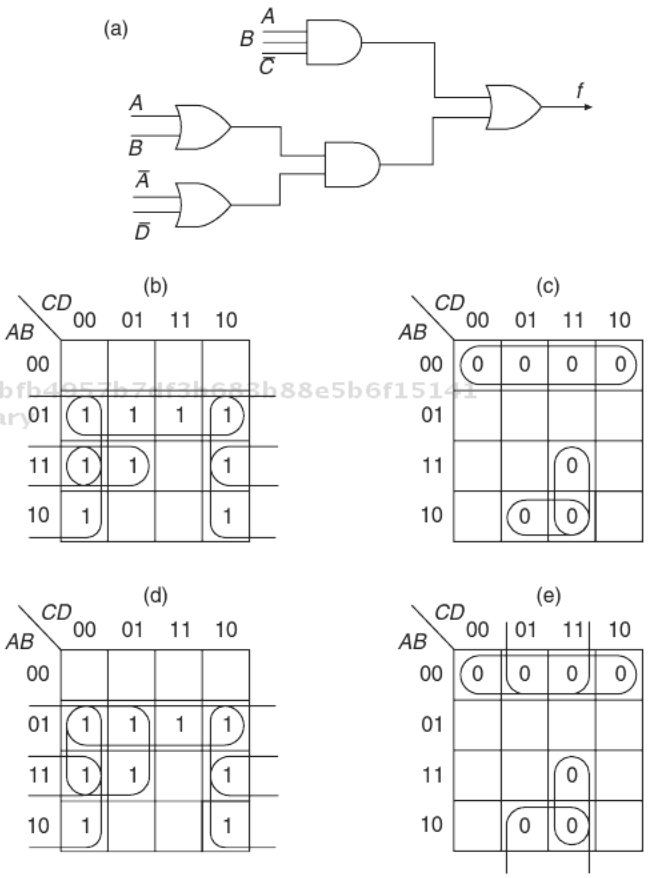


Figure 9.25 *(a) Circuit for the function $f = AB\bar{C} + (A + B)(\bar{A} + \bar{D})$ (b) K-map plot of the function (c) Plot of the inverse function (d) The resimplification of the function (e) Hazard-free plot of the inverse function*

variables in this equation which may be regarded as the equation which holds for transient conditions.

When deriving the transient equation of a circuit, some of the theorems of Boolean algebra may not be used. Those which make use of the identities $A\bar{A} = 0$ and $A + \bar{A} = 1$ may not be used to manipulate the equation into its transient form. For example, the expression $A + \bar{A}B = (A + \bar{A})(A + B) = (A + B)$ cannot be used as the reduction depends upon the identity $(A + \bar{A}) = 1$. Earlier in this chapter, it was shown that A and $\bar{A}$ may be simultaneously equal to zero, and in that case $A + \bar{A} \neq 1$, hence the above reduction is not valid for all instants of time.

The hazards can be detected by examining the expanded equation to see whether it reduces to either of the forms $X\bar{X}$ or $X + \bar{X}$ under defined input conditions, where X and $\bar{X}$ may represent any one of the four variables in the equation. For example, if $B = 0$ and $D = 1$, the equation reduces to $f = A\bar{A}$. Hence for these input conditions, a static 0-hazard occurs when A is changing from 0 to 1. Additionally, if $B = 1, C = 0$ and $D = 1$, the transient equation reduces to $f = A + A\bar{A} + \bar{A}$ and a static 1-hazard occurs when A is changing from 1 to 0. It should be noted that since A is changing from 1 to 0, $A\bar{A} = 0$ since it can only have a value 1 when A is changing from 0 to 1. If, however, $B = 1, C = 0$ and $D = 0$, the transient equation reduces to $f = A + A\bar{A} + A + \bar{A} + 1$. In this case, irrespective of the instantaneous values of A and $\bar{A}$, $f = 1$, and hence there is no static hazard.

Alternatively, the static 1-hazard can be detected by plotting those values of the variables that make the value of the function $f = 1$, as shown in Figure 9.25(b). Examination of this K-map shows that the two 1's in the adjacent cells 1101 and 0101 are not covered by the same prime implicant. The introduction of the prime implicant $B\bar{C}$ will ensure the coverage of these two cells by the same prime implicant and will remove the static 1-hazard.

To detect the possibility of a static 0-hazard, the circuit function has first to be inverted, and then plotted on a K-map. The inverse of the circuit function $f = AB\bar{C} + (A + B)(\bar{A} + \bar{D})$ is:

$$\bar{f} = \bar{A}\bar{B} + A\bar{B}D + ACD + A\bar{A}D$$

Note that the fourth term (the transient term) cannot be represented on the map.

It is clear from an examination of the K-map (Figure 9.25(c)) that the 0s in cells 1001 and 1011 are adjacent to the 0s in the cells 0001 and 0011 and are not covered by the same prime implicant, and a static 0-hazard is present in the circuit shown in Figure 9.24(a). By introducing the prime implicant $\bar{B}D$ to cover these four cells, the static 0-hazard can be removed.

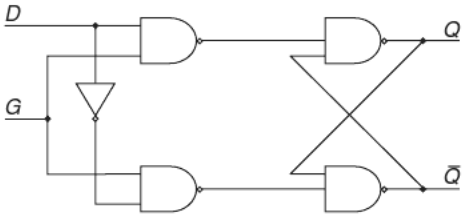
Poorly designed circuits may generate both kinds of static hazard. In practice, it would be a more satisfactory solution to redesign the circuit, shown in Figure 9.25(a), using the K-map plot of Figure 9.25(b) which, for convenience, is repeated in Figure 9.25(d). On this map the function has been simplified in such a way that the function is free of static 1-hazards. The hazard-free function is:

$$f = \bar{A}B + B\bar{D} + A\bar{D} + B\bar{C}$$

If an AND-OR-INVERT configuration is to be used, all that has to be done is to examine the plot of the inverse function for static 0-hazards. The inverted function is:

$$\bar{f} = \bar{A}\bar{B} + \bar{B}D + ACD$$

(a)



(b)

Present state			Next state
G^t	D^t	Q^t	$Q^{t+\delta t}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

(c)

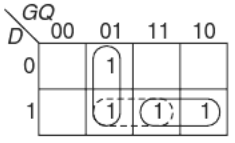


Figure 9.26 *(a) The controlled D-latch (b) state table (c) K-map plot*

and is shown plotted in Figure 9.25(e). Since there are no adjacent 0's not under the same prime implicant there are no static 0-hazards present.

A practical example of the possibility of a static hazard in a controlled D latch was referred to in section 6.5 of Chapter 6. For convenience, the circuit configuration is shown again in Figure 9.26 along with the state table and the K-map plot of the characteristic equation.

The characteristic equation read from the map is:

$$Q^{t+\delta t} = (\bar{G}Q + DG)^t$$

It will be observed that there are two 1's in adjacent cells not covered by the same prime implicant, and consequently a static 1-hazard is present. To eliminate the hazard, an extra prime implicant, DQ, enclosed by dotted lines on the map, is added in Figure 9.26(c), and the modified characteristic equation is:

$$Q^{t+\delta t} = (\bar{G}Q + DG + DQ)^t$$

It is left to the reader to show that the implementation of the latch shown in Figure 9.26(a) does, in fact, correspond to this hazard-free equation.

9.21 Hazard-free asynchronous circuit design

A gate-implemented asynchronous circuit with feedback is, in essence, a group of one or more combinational circuits which, under certain conditions, may generate static hazards. In practice, the designer should examime the design for hazards and then eliminate them using the techniques described earlier in this chapter. To demonstrate the occurrence of hazards in asynchronous circuits, the design of a hazard-free T-type flip-flop will be undertaken.

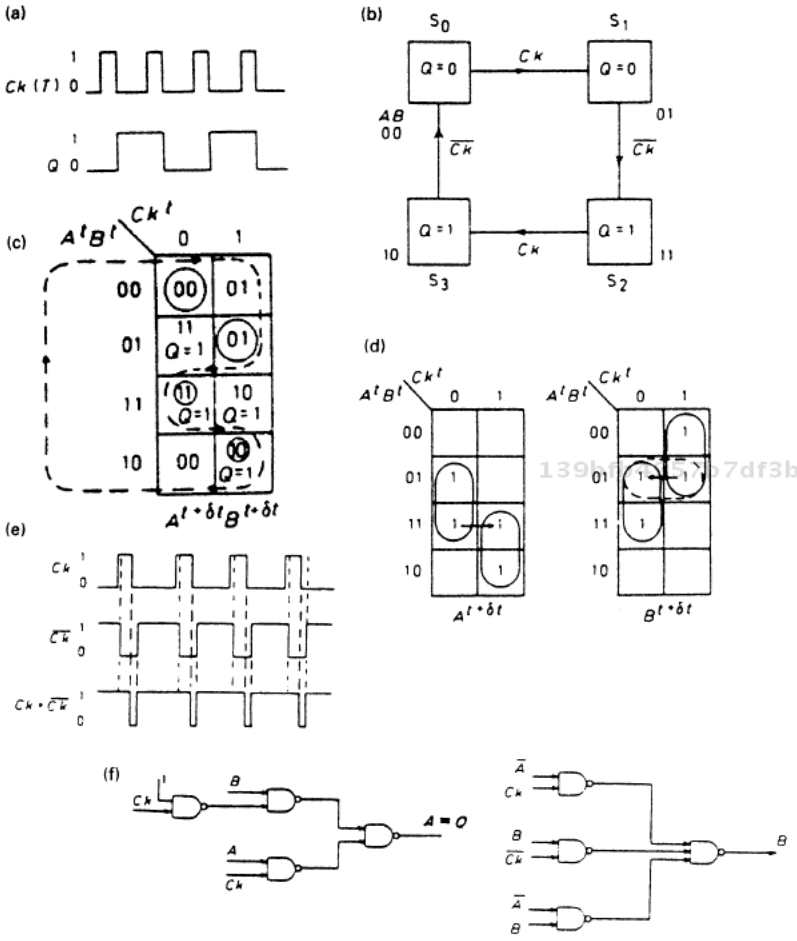


Figure 9.27 *(a) Timing diagram for T-type flip-flop (b) Internal state diagram (c) State table (d) Individual simplified state tables for $A^{t+\delta t}$ and $B^{t+\delta t}$ including hazard-removing prime implicant (e) Timing diagram for the Ck signal (f) Implementation of hazard-free flip-flop*

The timing diagram of a trailing edge triggered TFF is shown in Figure 9.27(a), the output toggling on the trailing edge of successive clock pulses. The state diagram is shown in Figure 9.27(b) and it reveals that the circuit completes a cycle of operation after four changes of the clock signal. It should be noted that in this example the clock transitions can be regarded as events which are able to initiate state transitions.

Since there are four states, two state variables A and B are required, and since this is an asynchronous design, a race-free state assignment has been used. The state table corresponding to the state diagram is shown in Figure 9.27(c), and the path traversed through the state table as one cycle of operation of the flip-flop takes place is illustrated by the dotted line.

In Figure 9.27(d), the state table has been separated into two distinct maps, one for $A^{t+\delta t}$ and one for $B^{t+\delta t}$. After simplification of these two functions it is clear that in both cases two 1's in adjacent cells are not covered by the same prime implicant, and there is a real possibility that a static hazard may be generated in both the A and B circuits. Arrows have been inserted on both maps indicating the direction of the state transitions between the relevant cells.

The equation for $A^{t+\delta t}$ is:

$$A^{t+\delta t} = (B \cdot \overline{Ck} + A \cdot Ck)^t$$

If $A = B = 1$, the equation reduces to:

$$A^{t+\delta t} = (\overline{Ck} + Ck)^t$$

and this condition indicates the possibility of the generation of a static 1-hazard. However, an examination of the timing diagrams for Ck, $\overline{Ck}$ and $(Ck + \overline{Ck})$ in Figure 9.27(e) shows that a static 1-hazard will only occur if Ck is making a $1 \rightarrow 0$ transition. The arrow-head on the $A^{t+\delta t}$ map reveals that the transition concerned is from total state $ABCk = 110$ to $ABCk = 111$; that is, the clock signal is changing from $0 \rightarrow 1$, and it follows that a static 1-hazard can never be generated in the A circuit.

The equation for $B^{t+\delta t}$ is:

$$B^{t+\delta t} = (\bar{A} \cdot Ck + B \cdot \overline{Ck})^t$$

If $\bar{A} = B = 1$ this equation reduces to:

$$B^{t+\delta t} = (Ck + \overline{Ck})^t$$

In this case, the arrow-head on the $B^{t+\delta t}$ map shows that the B circuit makes a transition from total state $ABCk = 011$ to $ABCk = 010$; that is, Ck is making a $1 \rightarrow 0$ transition, and consequently a static 1-hazard will be generated in the B circuit. To eliminate the hazard, the additional prime implicant $\bar{A}B$ is added to the equation for $B^{t+\delta t}$ which now reads:

$$B^{t+\delta t} = (\bar{A} \cdot Ck + B \cdot \overline{Ck} + \bar{A} \cdot B)^t$$

Also from the state diagram:

$$\begin{aligned} Q &= S_3 + S_2 \\ &= AB + A\bar{B} = A \end{aligned}$$

The NAND implementation of the hazard-free T flip-flop is shown in Figure 9.27(f).

9.22 Dynamic hazards

A second type of hazard that can occur in gate networks is referred to as a *dynamic hazard*. The output changes normally expected by the circuit designer are either $0 \rightarrow 1$ or alternatively $1 \rightarrow 0$. If, in practice, the output transitions are $1 \rightarrow 0 \rightarrow 1 \rightarrow 0$ then a dynamic hazard has occurred. Similarly, if an output designed to change from $0 \rightarrow 1$ has the change pattern $0 \rightarrow 1 \rightarrow 0 \rightarrow 1$, then a dynamic hazard is present. In either case there is a minimum of three changes appearing at the output as illustrated in Figure 9.28.

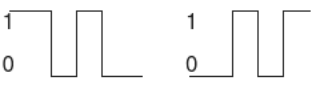


Figure 9.28 *Dynamic hazards*

This type of hazard occurs as a result of the factorisation of a Boolean function, necessary, because of fan-in restrictions, which leads to different path lengths through a circuit. Alternatively, the gates in the circuit configuration may have different time delays, and it is also possible to have differing time delays in the interconnecting leads.

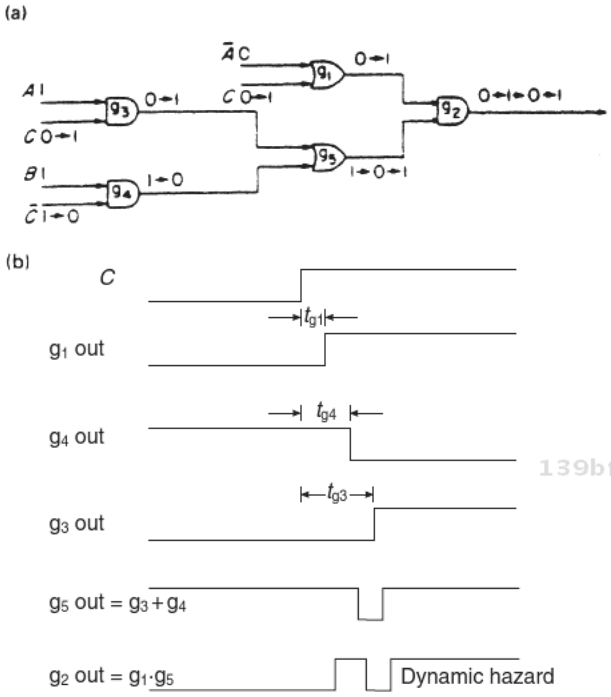


Figure 9.29 *(a) Network with a dynamic hazard (b) Occurrence of a dynamic hazard in the network*

Consider the function:

$$f = (AC + B\bar{C})(\bar{A} + C)$$

implemented with AND and OR gates as shown in Figure 9.29(a). There are three different paths through this network for the variable C and consequently there is a possibility that a dynamic hazard exists in the network. The three paths through the network are

1. via gates g_1 and g_2
2. via gates g_3, g_5 and g_2, and
3. via gates g_4, g_5 and g_2.

There are eight possible starting combinations of the variables A, B, and C. Since, in this circuit, the dynamic hazard is caused by multiple paths taken by the signal representing variable C, in each case only the next change in C need be examined. For the four combinations starting with $A = 0$, the output of OR gate g_1 remains at 1 irrespective of C, and the output of AND gate g_3 remains at 0. Hence, for these combinations, changes in C take the path g_4–g_5–g_2 only and so no dynamic hazards are present.

However, in the case $A = 1$, $B = 1$, and $C = \uparrow$, the upper input of gate g_2 changes from 0 to 1. The other input to g_2 is $AC + B\bar{C} = C + \bar{C}$ in this case, and if $t_{g4} < t_{g3}$ then there will be a static 1-hazard present at the output of gate g_5. Then, if this static hazard itself occurs after the change at the output of g_1, i.e. if $t_{g1} < t_{g4} < t_{g3}$, there will be a dynamic hazard present at the output of gate g_2 as illustrated in Figure 9.29(b).

A similar analysis shows that if $t_{g1} > t_{g4} > t_{g3}$, then there is a dynamic hazard produced in the case $A = 1$, $B = 1$, and $C = \downarrow$. In the two remaining cases,

$A = 1$, $B = 0$, and $C = \text{X}$, there are no further dynamic hazards as signal C takes only two paths through the network. Using the terminology to be introduced in Chapter 13, all three possible paths for C are *sensitised* only when $A = 1$ and $B = 1$.

It is worth noting that providing AND/OR sum-of-products circuits or if OR/AND product-of-sum circuits have been designed such that there are no static hazards present, then these circuits will have no dynamic hazards.

9.23 Function hazards

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	0	1
10	1	0	0	1

Figure 9.30 *K-map plot used for illustrating function hazards*

This type of hazard, which can be either a static 1- or static 0-hazard, occurs when it is specified that two circuit input variables change at the same time. In practice, it is extremely unlikely that two variables will change at precisely the same time but if this should happen to occur it can lead to the presence of a hazard during a transition.

Consider the K-map plot of a 4-variable function shown in Figure 9.30. If the initial condition of the input variables is $ABCD = 1000$ and circuit operation specifies that the variables B and D change simultaneously, one of three possibilities may occur:

1. B and D change simultaneously:
   $ABCD = 1000 \rightarrow 1101$
   $f = 1 \quad \rightarrow 1$
2. B changes before D:
   $ABCD = 1000 \rightarrow 1100 \rightarrow 1101$
   $f = 1 \quad \rightarrow 1 \quad \rightarrow 1$
3. D changes before B:
   $ABCD = 1000 \rightarrow 1001 \rightarrow 1101$
   $f = 1 \quad \rightarrow 0 \quad \rightarrow 1$

If D changes before B a function static 1-hazard is present. Alternatively, if the initial condition of the input variables is $ABCD = 0000$ and a simultaneous change in the variables A and D should occur, one of the following three possibilities may arise:

1. A and D change simultaneously
   $ABCD = 0000 \rightarrow 1001$
   $f = 0 \quad \rightarrow 0$
2. A changes before D
   $ABCD = 0000 \rightarrow 1000 \rightarrow 1001$
   $f = 0 \quad \rightarrow 1 \quad \rightarrow 0$
3. D changes before A
   $ABCD = 0000 \rightarrow 0001 \rightarrow 1001$
   $f = 0 \quad \rightarrow 0 \quad \rightarrow 0$

If A changes before D a function static 0-hazard occurs.

A situation may also arise where it is specified that three variables should change at the same time, and in that case there is the possibility that a *function dynamic hazard*

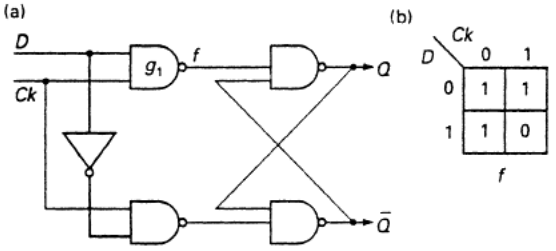


Figure 9.31 *(a) Controlled D-type latch (b) K-map plot for output of gate* g_1

may occur. In practice, function hazards can be avoided at the design stage by ensuring that only one variable can change at any one time.

A function hazard can occur at the input NAND gate of a synchronising latch. It will be assumed that the input D is asynchronous data and that Ck is the synchronising signal, as described in Chapter 8. In this situation there is no way of ensuring that the asynchronous input changes at the same time as the synchronising signal. The K-map for a 2-input NAND gate is shown in Figure 9.31. If the initial condition is $DCk = 10$ and both signals happen to change simultaneously, then the steady-state output of the gate will remain at 1. In practice, they are unlikely to change simultaneously and a spurious output can occur, which in the case of a synchroniser circuit, is referred to as a *runt pulse*. This pulse may not be sufficient to cause the synchroniser to switch from one stable state to another and the latch may enter the metastable state where it will stay for a period which cannot be precisely defined.

9.24 Essential hazards

This type of hazard is peculiar to asynchronous circuits and is caused by a race between an input signal and a state variable. The state diagram for an asynchronous circuit having a race-free state assignment is shown in Figure 9.32. Assuming that the circuit is in state S_0 and a change in the value of X from 0 to 1 occurs, a transition from S_0 to S_1 should take place and, on arriving in S_1, the circuit should remain in that state. However, correct operation of the circuit as described above will depend upon the relative values of the inversion time t_i for the input signal X and and the turn-on time t_t for the state variable B. If the circuit arrives in the state S_1 before the value of $\bar{X}$ has changed from 1 to 0, a further transition to S_2 will be made. Since $X = 1$ when the circuit arrives in state S_2, it follows that a further transition will take place to state S_3, where the circuit will now remain, provided the change in $\bar{X}$ has now occurred. Hence, if $t_i > t_t$, incorrect circuit operation will occur as a consequence of the race between the inversion of the input signal X and the turn-on of the state variable B.

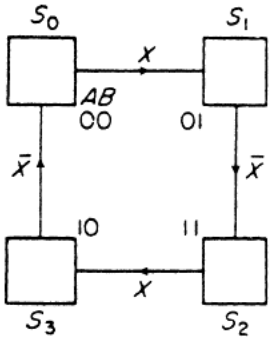


Figure 9.32 *State diagram for a machine which can have an essential hazard*

An examination of the equation for the state variable A reveals more clearly the origin of the hazard. The turn-on condition for $A = B\bar{X}$, the turn-off condition for $A = \bar{B}\bar{X}$, and

$$A^{t+\delta t} = (B\bar{X} + (\overline{\bar{B}\bar{X}})A)^t$$
$$= (B\bar{X} + (B + X)A)^t$$

The first term of this equation provides the turn-on signal for A when the circuit is in state S_1. If B changes to 1 before $\bar{X}$ changes to 0, the value of $B\bar{X} = 1$ and the state variable A is turned on.

The method of dealing with this type of hazard is to insert a delay in the output line of the circuit generating the state variable B. This will ensure that the change in B does not arrive at the input to the circuit generating the state variable A until the value of $\bar{X}$ has changed.

Problems

9.1 A double-sequence detector has two inputs, X_1 and X_2, and one output Z. For an input sequence $X_1X_2 = 00$, 10, 11 the output Z becomes 1, and when the reverse sequence is received the output Z returns to 0. A typical timing diagram for the detector is shown in Figure P9.1. Develop:

(1) A state diagram

(2) An ASM chart

for the detector and obtain a state table. If possible, reduce the state table and implement the design with NAND gates.

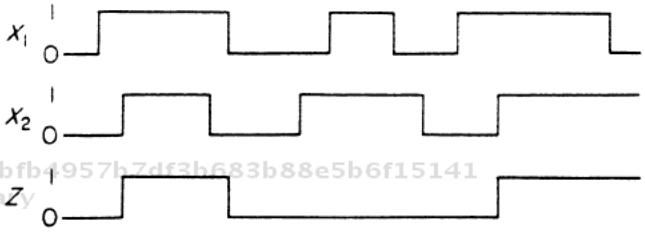


Figure P9.1

9.2 Develop an event-driven circuit to implement a trailing-edge triggered JK flip-flop and draw a timing diagram for the flip-flop.

9.3 X_1 and X_2 are the two inputs to an asynchronous circuit which has two outputs, Z_1 and Z_2. When $X_1X_2 = 00$ the output $Z_1Z_2 = 00$. If a $0 \rightarrow 1$ change in X_1 precedes a $0 \rightarrow 1$ change in X_2, then the output of the circuit is $Z_1Z_2 = 01$. Alternatively, if a $0 \rightarrow 1$ change in X_2 precedes a $0 \rightarrow 1$ change in X_1, then the output of the circuit is $Z_1Z_2 = 10$. In both cases the outputs remain at 01 and 10, respectively, until $X_1X_2 = 00$ again. Draw the state diagram for this system.

9.4 Develop an asynchronous circuit that will give an output clock pulse (Z) after every second data pulse arrives on the X input line. The arrival of the data pulses is purely random and it is to be assumed that the minimum time for a pair of

consecutive data pulses is greater than the periodic time of the clock. A typical timing diagram is shown in Figure P9.4.

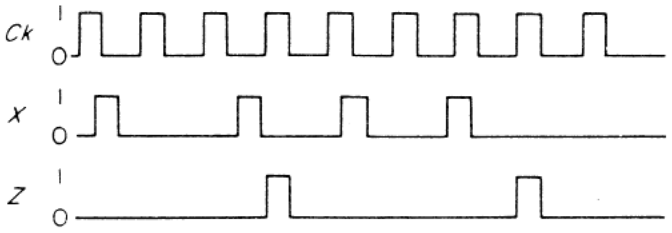


Figure P9.4

9.5 A logic circuit has two asynchronous inputs, X_1 and X_2, and also a synchronous clock signal. The circuit is to be designed so that the first complete clock pulse that occurs after X_1 and X_2 have become 1, in that order, is output on the line marked Z in Figure P9.5. After the output of the clock pulse the circuit must return to its quiescent state when $X_1X_2 = 00$.

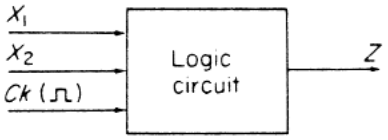


Figure P9.5

Design a circuit that satisfies this specification and implement the design using NAND gates.

9.6 Analyse the fundamental mode circuit shown in Figure P9.6:

(a) Determine the state table.
(b) Determine the state diagram.
(c) Use the state table to determine the output response to the input sequence $X_1X_2 = 00,\ 01,\ 11,\ 10,\ 11,\ 01,\ 00,\ 10,\ 00,\ 01$. Initial conditions $X_1 = X_2 = A = 0$.

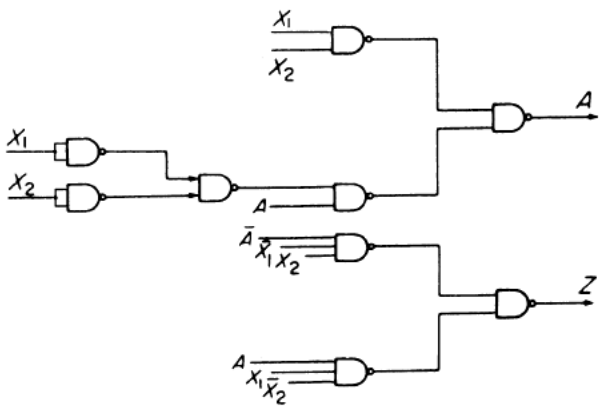


Figure P9.6

9.7 Analyse the circuit shown in Figure P9.7:

(a) Determine the state table.
(b) Determine the state diagram.
(c) Use the state table to determine the output response to the input sequence $X_1X_2 = 00, 01, 11, 10, 00, 01, 11, 01, 11, 10, 00$. Assume the initial conditions are $X_1 = X_2 = 0$ and $A = B = 0$.

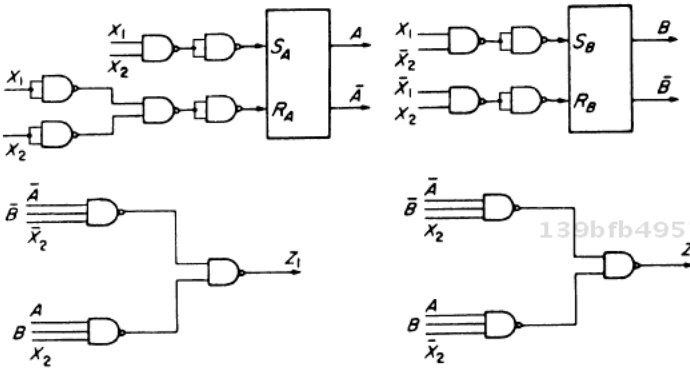


Figure P9.7

9.8 The internal state diagram for a four-state digital machine is shown in Figure P9.8. Construct a state table for the machine and identify all races that will occur if the machine is implemented from the given state diagram, stating whether they are critical or non-critical. For each race, give all the state transitions which may occur.

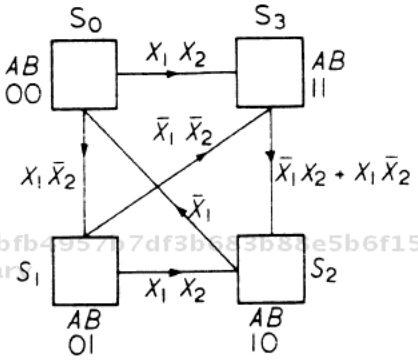


Figure P9.8

9.9 Plot the K-map of the functions

(*a*) $f(A, B, C, D) = \sum 0, 2, 4, 5, 6, 8, 9, 11, 12, 14, 15$, and
(*b*) $f(A, B, C, D) = \sum 3, 4, 5, 6, 11, 12, 13, 14, 15$

and determine hazard-free implementations in both cases, using NAND gates.

9.10 Find all the static hazards in the two networks shown in Figures P9.10(a) and (b). Specify the input conditions that must exist for the hazards to occur and draw the logic diagram for modified networks that are hazard-free.

9.11 Design a hazard-free, D-type flip-flop using asynchronous circuit design techniques. It may be assumed that the output will take on the value of the input on the trailing edge of a clock pulse.

9.12 An incompletely specified table is shown in Figure P9.12. With the aid of an implication chart, find the compatible state pairs. Using a merger diagram obtain the maximal compatibles and construct a reduced state table.

9.13 An electrical system is protected by a fault detector. If a fault occurs within the system a fault signal activates an alarm buzzer. The green light that indicates fault

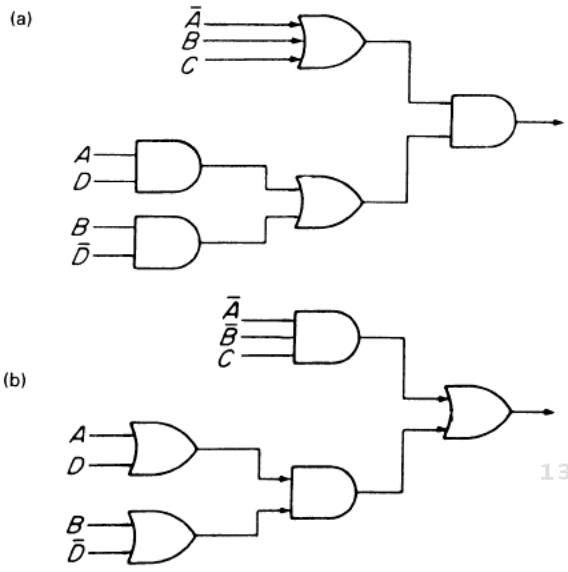


Figure P9.10

	X_1X_2 00	01	11	10
S_0	–,–	S_2,1	S_4,1	S_2,1
S_1	S_4,–	–,–	–,–	–,–
S_2	S_5,–	S_5,1	–,–	–,–
S_3	–,–	–,–	S_1,1	–,–
S_4	–,–	S_5,0	S_0,0	S_3,1
S_5	S_2,0	–,–	S_1,0	S_2,1

Figure P9.12

free operation is switched off by the fault signal and a red light is switched on. When the fault is acknowledged by the system controller the alarm buzzer is turned off. After the fault has been cleared the green light is switched on and the red light is turned off. A test signal is to be provided to check the operation of the fault detector.

Develop an appropriate state diagram and implement your design with the aid of the NAND characteristic equation.

9.14 An asynchronous circuit is to be used to control the gates and a red flashing light at a railway level crossing. The gates are to be closed and the red flashing light is to be turned on when a train enters a defined section of track from either direction. When the train is in a further defined section of track which straddles the crossing the gates must remain closed and the red light must remain flashing. After a train has passed through the crossing the gates are opened and the flashing red light is turned off.

Develop an ASM chart, convert it to a state diagram and implement your design using the NAND sequential equation.

9.15 Design an asynchronous lock operated by five input buttons labelled A, B, C, D and R (the reset button). The unlocking operation can only take place if only one button is activated at a time and in the order B, D, A, C. Draw a state diagram and develop a gate-implemented circuit.

9.16 Using asynchronous circuit design techniques, design a hazard free D-type flip-flop whose output takes up the value of the input on the trailing edge of a clock pulse.