

EC 8392 – DIGITAL ELECTRONICS

UNIT – IV : ASYNCHRONOUS SEQUENTIAL CIRCUITS

INTRODUCTION

A sequential circuit is specified by a time sequence of inputs, outputs and internal states. In synchronous sequential circuits, the change of internal state occurs in response to the synchronized clock pulses. Asynchronous sequential circuits do not use clock pulses. The change of internal state occurs when there is a change in the input variables. The memory elements in synchronous sequential circuits are clocked flip-flops. The memory elements in asynchronous sequential circuits are either unclocked flip-flops or time -delay elements. The memory capability of a time-delay device depends on the finite amount of time it takes for the signal to propagate through digital gates. An asynchronous sequential circuit quite often resembles a combinational circuit with feedback.

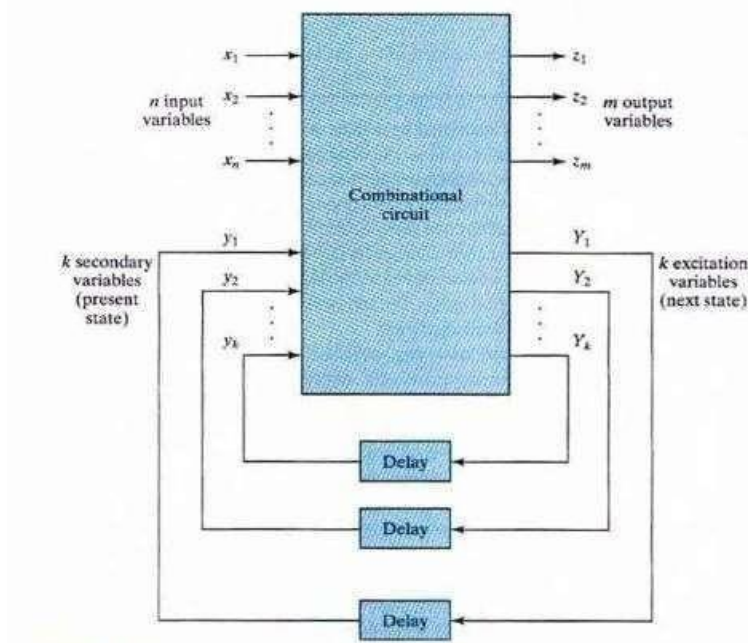


Fig 4.1 - block diagram of an asynchronous sequential circuit

Image source from Digital Design by Moris Mano (Page No. 416)

The above figure shows the block diagram of an asynchronous sequential circuit that consists of a combinational circuit and delay elements connected to form feedback loops. There are n input variables, m output variables and k internal states. The delay elements can be visualized as providing short-term memory for the sequential circuit. In a gate-type circuit, the propagation delay that exists in the combinational circuit path from input to output provides sufficient delay along the feedback loop so that no specific delay elements are actually inserted into the feedback path. The present-state and next-state variables in asynchronous sequential circuits are customarily called secondary variables and excitation variables, respectively. The excitation variables should not be confused with the excitable table used in the design of clocked sequential circuits.

ANALYSIS PROCEDURE

The analysis of asynchronous sequential circuits consists of obtaining a table or a diagram that describes the sequence of internal states and outputs as a function of changes in the Input variables. A logic diagram manifests the behavior of an asynchronous sequential circuit if it has one or more feedback loops or if it includes unclocked flip-flops.

TRANSITION TABLE

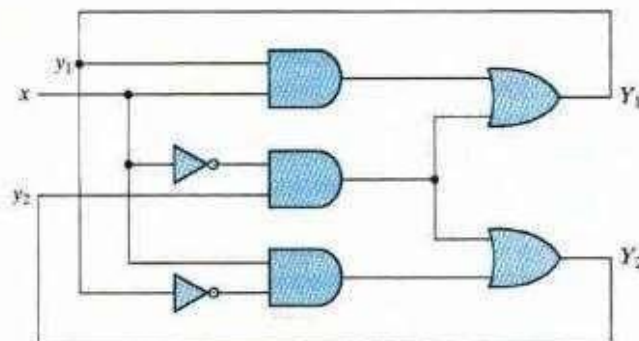


Fig 4.2 – Example of an asynchronous sequential circuit

Image source from Digital Design by Moris Mano (Page No. 418)

The diagram clearly shows two feedback loops from the OR gate outputs back to the AND gate inputs. The circuit consists of one input variable x and two internal states. The internal states have two excitation variables, Y_1 , and Y_2 and two secondary variables, y_1 and y_2 . The delay associated with each feedback loop is obtained from the propagation delay between each Y input and its corresponding output. Each logic gate in the path introduces a propagation delay of about 2 to 10 ns. The wires that conduct electrical signals introduce approximately a 1-ns delay for each foot of wire. Thus, no additional external delay elements are necessary when the combinational circuit and the wires in the feedback path provide sufficient delay.

The analysis of the circuit starts with a consideration of the excitation variables as outputs and the secondary variables as inputs. We then derive the Boolean expressions for the excitation variables as a function of the input and secondary variables. These expressions readily obtained from the logic diagram are,

$$Y_1 = xy_1 + x'y_2$$
$$Y_2 = xy_1' + x'y_2$$

The next step is to plot the Y_1 and Y_2 functions in a map. The encoded binary values of the Y variables are used for labeling the rows and the input x variable is used to designate the columns.

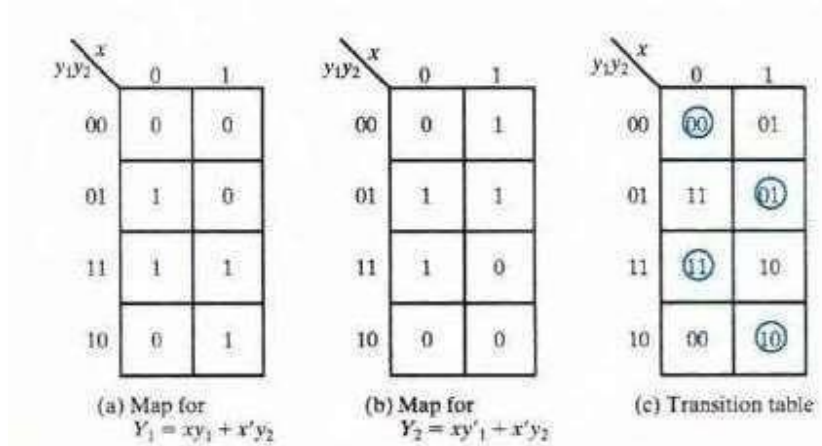


Fig 4.3 – Maps and transition table

Image source from Digital Design by Moris Mano (Page No. 418)

The transition table is obtained from the maps by combining the binary values in corresponding squares. The transition table shows the value of $Y = Y_1Y_2$ inside each square. The first bit of Y is obtained from the value of Y_1 and the second bit is obtained from the value of Y_2 in the same square position. For a state to be stable, the secondary variables must match the excitation variables (i.e the value of Y must be the same as that of $y = y_1y_2$). Those entries in the transition table where $Y = y$ are circled to indicate a stable condition. An un circled entry represents an unstable state.

The procedure for obtaining a transition table from the circuit diagrams of an asynchronous sequential circuit is as follows:

Determine all feedback loops in the circuit.

1. Designate the output of each feedback loop with variable Y ; and its corresponding input with y_i for $i = 1, 2, \dots, k$, where k is the number of feedback loops in the circuit.
2. Derive the Boolean functions of all Y 's as a function of the external inputs and the y 's
3. Plot each Y function in a map using the y variables for the rows and the external inputs for the columns.

4. Combine all the maps into one table showing the value of $Y = Y_1Y_2 \dots Y_i$ inside each square.
5. Circle those values of Y in each square that are equal to the value of $y = y_1y_2 \dots y_i$ in the same row.

FLOW TABLE

During the design of asynchronous sequential circuits, it is more convenient to name the states by letter symbols without making specific reference to their binary values. Such a table is called a flow table and is similar to a transition table except that the internal states are symbolized with letters rather than binary numbers. The flow table also includes the output values of the circuit for each stable state.

www.binils.com

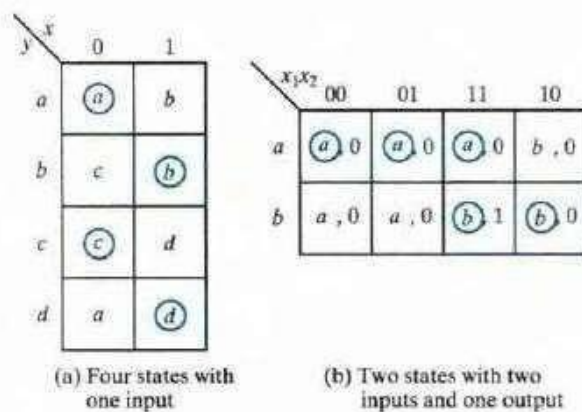


Fig 4.3 – Example of Flow tables

Image source from Digital Design by Moris Mano (Page No. 420)

If we assign the following binary values to the states: $a = 00$, $b = 01$, $c = 11$ and $d = 10$. The table of Figure (a) is called a primitive now table because it has only one stable state in each row. Figure (b) shows a flow table with more than one stable table in the same row. It has two states a and b , two inputs x_1 and x_2 and one output z . The binary value

the output variable is indicated inside the square next to the state symbol and is separated from the state symbol by a comma. From the flow table, we observe the following behavior of the circuit: If $x_1 = 0$, the circuit is in state a. If x_1 goes to 1 while x_2 is 0, the circuit goes to state b. With inputs $x_1x_2 = 11$, the circuit may be either in state a or in state b. If it is in state a, the output is 0, and if it is in state b, the output is 1. State b is maintained if the inputs change from 10 to 11. The circuit stays in state a if the inputs change from 01 to 11. Remember that in fundamental mode two input variables cannot change simultaneously; therefore, we do not allow a change of inputs from 00 to 11.

In order to obtain the circuit described by a flow table, it is necessary to assign a distinct binary value to each state. Such an assignment converts the flow table into a transition table from which we can derive the logic diagram.

www.binils.com

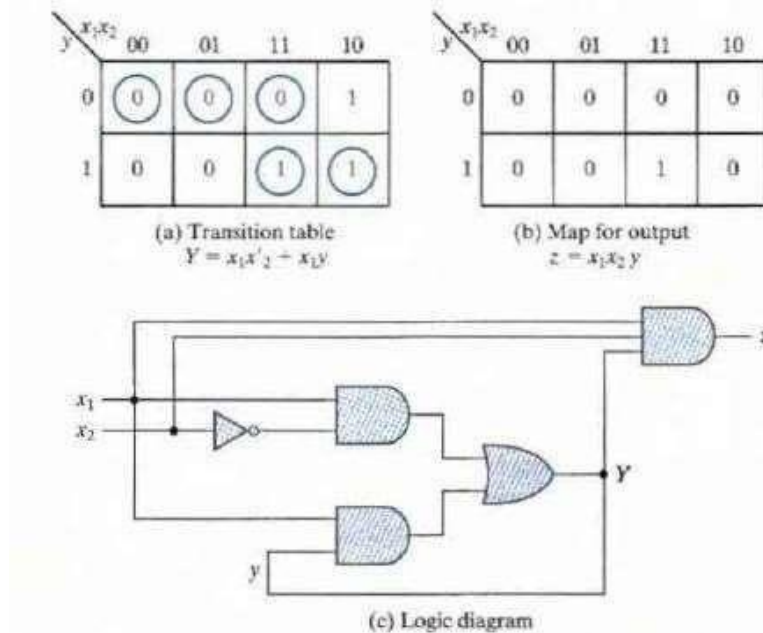


Fig 4.4 – Derivation of a circuit by flow table

Image source from Digital Design by Moris Mano (Page No. 421)

EC 8392 – DIGITAL ELECTRONICS

UNIT – IV : ASYNCHRONOUS SEQUENTIAL CIRCUITS

RACE CONDITIONS

A race condition is said to exist in an asynchronous sequential circuit when two or more binary state variables change value in response to a change in an input variable. When unequal delays are encountered a race condition may cause the state variables to change in an unpredictable manner. For example, if the state variables must change from 00 to 11, the difference in delays may cause the first variable to change sooner than the second, with the result that the state variables change in sequence from 00 to 10 and then to 11. If the second variable changes sooner than the first, the State variables will change from 00 to 01 and then to 11. Thus, the order by which the state variables change may not be known in advance. If the final stable state that the circuit reaches does not depend on the order in which the state variables change, the race is called a noncritical race. If it is possible to end up in two or more different stable states, depending on the order in which the state variables change, then the race is a critical race. For proper operation, critical races must be avoided.

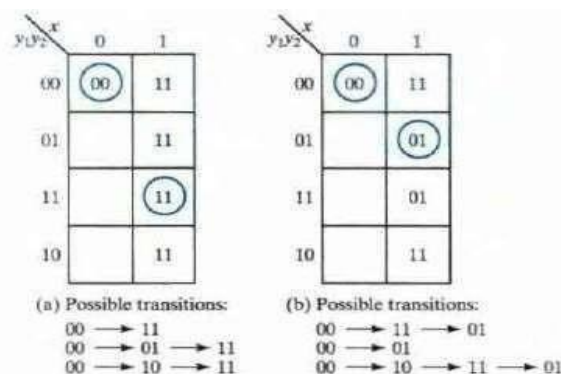


Fig: 4.5 - Examples of noncritical Races

Image source from Digital Design by Moris Mano (Page No. 422)

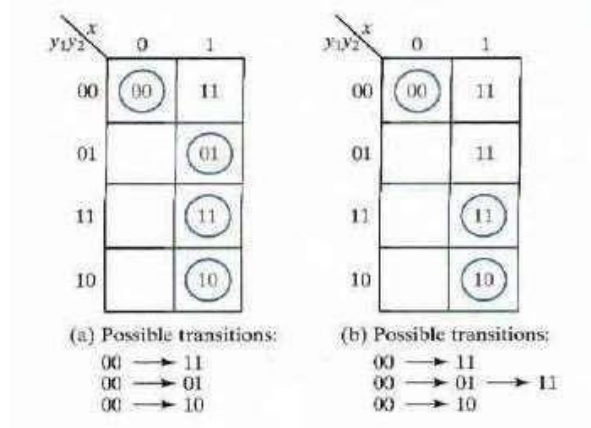


Fig: 4.6 - Examples of Critical Races

Image source from Digital Design by Moris Mano (Page No. 423)

Races may be avoided by making a proper binary assignment to the state variables. The state variables must be assigned binary numbers in such a way that only one state variable can change at any one time when a state transition occurs in the flow table. Races can be avoided by directing the circuit through intermediate unstable states with a unique state-variable change. When a circuit goes through a unique sequence of unstable states, it is said to have a cycle.

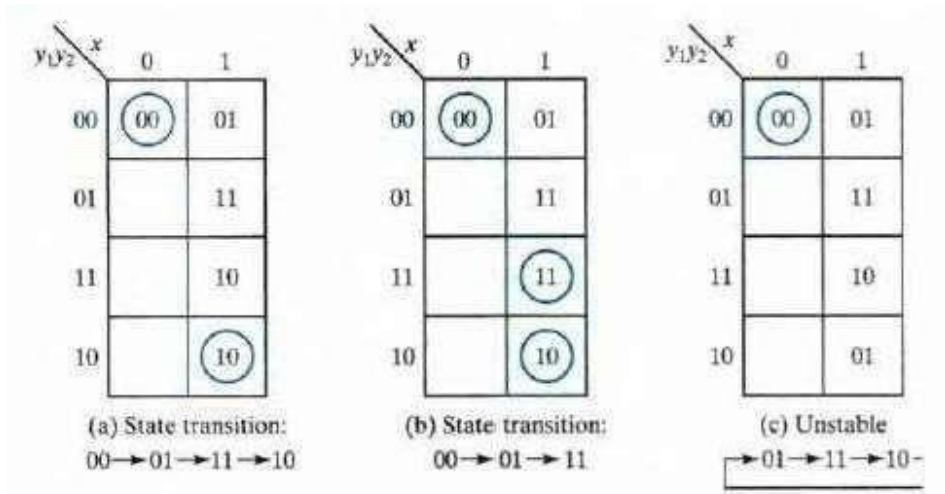


Fig: 4.7 - Examples of cycles

Image source from Digital Design by Moris Mano (Page No. 424)

EC 8392 – DIGITAL ELECTRONICS

UNIT – IV : ASYNCHRONOUS SEQUENTIAL CIRCUITS

REDUCTION OF STATE AND FLOW TABLES

State Table to Demonstrate Equivalent States

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
<i>a</i>	<i>c</i>	<i>b</i>	0	1
<i>b</i>	<i>d</i>	<i>a</i>	0	1
<i>c</i>	<i>a</i>	<i>d</i>	1	0
<i>d</i>	<i>b</i>	<i>d</i>	1	0

Fig 4.16 – state table

Image source from Digital Design by Moris Mano (Page No. 440)

IMPLICATION TABLE AND IMPLIED STATES

The state-reduction procedure for completely specified state tables is based on an algorithm that combines two states in a state table into one, as long as they can be shown to be equivalent. Two states are equivalent if, for each possible input, they give exactly the same output and go to the same next states or to equivalent next states.

State Table to Be Reduced

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
<i>a</i>	<i>d</i>	<i>b</i>	0	0
<i>b</i>	<i>e</i>	<i>a</i>	0	0
<i>c</i>	<i>g</i>	<i>f</i>	0	1
<i>d</i>	<i>a</i>	<i>d</i>	1	0
<i>e</i>	<i>a</i>	<i>d</i>	1	0
<i>f</i>	<i>c</i>	<i>b</i>	0	0
<i>g</i>	<i>a</i>	<i>e</i>	1	0

Fig 4.17 – State Table

Image source from Digital Design by Moris Mano (Page No. 441)

The checking of each pair of states for possible equivalence in a table with a large number of states can be done systematically by means of an implication table, which is a chart that consists of squares, one for every possible pair of states that provide spaces for listing any possible implied states. By judicious use of the table, it is possible to determine all pairs of equivalent states.

On the left side along the vertical are listed all the states defined in the state table except the first, and across the bottom horizontally are listed all the states except the last. The result is a display of all possible combinations of two states, with a square placed in the intersection of a row and a column where the two states can be tested for equivalence.

Two states having different outputs for the same input are not equivalent. Two states that are not equivalent are marked with a cross [X] in the corresponding square, whereas their equivalence is recorded with a check mark (✓). Some of the squares have entries of implied states that must be investigated further to determine whether they are equivalent. The step-by-step procedure of filling in the squares is as follows: First, we place a cross in any square corresponding to a pair of states whose outputs are not equal for every input. In this case, state c has a different output than any other state, so a cross is placed in the two squares of row c and the four squares of column c. There are nine other squares in this category in the implication table.

b	d, e ✓					
c	x	x				
d	x	x	x			
e	x	x	x	✓		
f	c, d x	c, e x a, b	x	x	x	
g	x	x	x	d, e ✓	d, e ✓	x
	a	b	c	d	e	f

Fig : 4.18 - Implication Table

Image source from Digital Design by Moris Mano (Page No. 441)

Next, we enter in the remaining squares the pairs of states that are implied by the pair of states representing the squares. We do that starting from the top square in the left column and going down and then proceeding with the next column to the right. From the state table, we see that pair (a, b) implies (d, e), so (d, e) is recorded in the square defined by column a and row b. We proceed in this manner until the entire table is completed. Note that states (d, e) are equivalent because they go to the same next state and have the same output. Therefore, a check mark is recorded in the square defined by column d and row e, indicating that the two states are equivalent and independent of any implied pair.

The next step is to make successive passes through the table to determine whether any additional squares should be marked with a cross. A square in the table is crossed out if it contains at least one implied pair that is not equivalent. For example, the square defined by a and f is marked with a cross next to c, d because the pair (c, d)

defines a square that contains a cross. This procedure is repeated until no additional squares can be crossed out. Finally, all the squares that have no crosses are recorded with check marks. These squares define pairs of equivalent states. In this example, the equivalent states are,

(a,b) (d,e) (d,g) (e,g)

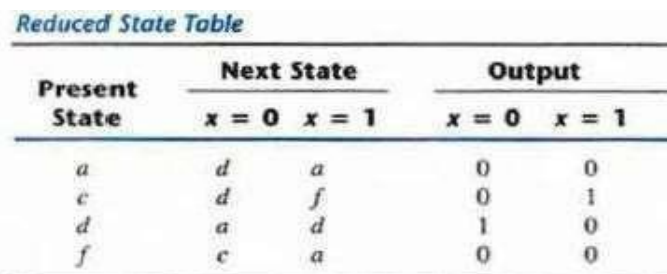
We now combine pairs of states into larger groups of equivalent states. The last three pairs can be combined into a set of three equivalent states (d, e, g) because each one of the states in the group is equivalent to the other two. The final partition of the states consists of the equivalent states found from the implication table, together with all the remaining states in the state table that are not equivalent to any other state.

This group consists

of (a,b) (c) (d,e,g)

(f)

MERGING OF THE FLOW TABLE



Present State	Next State		Output	
	x = 0	x = 1	x = 0	x = 1
a	d	a	0	0
c	d	f	0	1
d	a	d	1	0
f	c	a	0	0

Fig 4.19 – Reduced Flow Table

Image source from Digital Design by Moris Mano (Page No. 442)

The process that must be applied in order to find a suitable group of compatibles (or the purpose of merging a flow table can be divided into three steps:

1. Determine all compatible pairs by using the implication table.
2. Find the maximal compatibles with the use of a merger diagram.
3. Find a minimal collection of compatibles that covers all the states and is closed.

Compatible Pairs

Two states are compatible if, in every column of the corresponding rows in the flow table, there are identical or compatible states and if there is no conflict in the output values. For example, rows a and b in the flow table are found to be compatible, but rows a and f will be compatible only if c and f are compatible. However, rows c and f are not compatible, because they have different outputs in the first column. This information is recorded in the implication table. A check mark designates a square whose pair of states is compatible. Those states which are not compatible are marked with a cross. The remaining squares are recorded with the implied pairs that need further investigation.

	00	01	11	10
a	c,-	a 0	b,-	-, -
b	-, -	a,-	b 1	e,-
c	c 0	a,-	-, -	d,-
d	c,-	-, -	b,-	d 0
e	f,-	-, -	b,-	e 1
f	f 1	a,-	-, -	e,-

(a) Primitive flow table

b	✓				
c	✓	d, e x			
d	✓	d, e x	✓		
e	c, f x	✓	d, e x c, f x	x	
f	c, f x	✓	x	d, e x c, f x	✓
	a	b	c	d	e

(b) Implication table

Fig 4.20 – a. Primitive Flow Table, b). Implication table

Image source from Digital Design by Moris Mano (Page No. 443)

The compatible pairs are, (a,b) (a,c) (a,d) (b,e) (b,f) (c,d) (e,f)

Maximal Compatibles

The maximal compatible is a group of compatibles that contains all the possible combinations of compatible states. The maximal compatible can be obtained from a merger diagram. The merger diagram is a graph in which each state is represented by a dot placed along the circumference of a circle. Lines are drawn between any two corresponding dots that form a compatible pair. All possible compatibles can be obtained from the merger diagram by observing the geometrical patterns in which states are connected to each other. An isolated dot represents a state that is not compatible with any other state. A line represents a compatible pair. A triangle constitutes a compatible with three states. An n-state compatible is represented in the merger diagram by an n-sided polygon with all its diagonals connected. There are seven straight lines connecting the dots, one for each compatible pair. The lines form a geometrical pattern consisting of two triangles connecting (a,c,d) and (b,e,f) and a line (a,b). The maximal compatibles are (a,b) (a,c,d) (b,e,f)

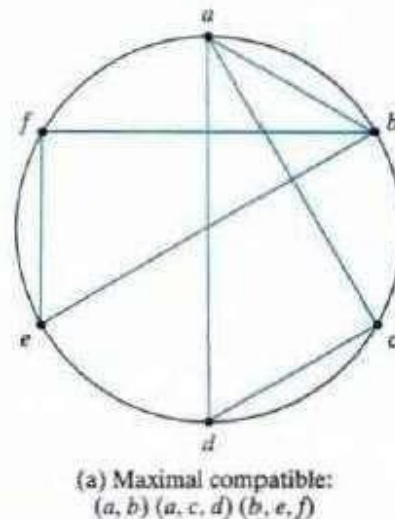


Fig 4.21 – Merger Diagram

Image source from Digital Design by Moris Mano (Page No. 444)

CLOSED-COVERING CONDITION

The condition that must be satisfied for merging rows is that the set of chosen compatibles must cover all the states and must be closed. The set will cover all the states if it includes all the states of the original state table. The closure condition is satisfied if there are no implied states or if the implied states are included within the set. A closed set of compatibles that covers all the states is called a closed covering.

The compatible pairs derived from the below shown implication table are (a,b) (a,d) (b,c) (c,d) (c,e) (d,e). From the merger diagram, the maximal compatibles are, (a,b) (a,d) (b,c) (c,d,e). If we choose the two compatibles (a,b) (c,d,e) then the set will cover all five states of the original table. The closure condition can be checked by means of a closure table. The implied pairs listed for each compatible are taken directly from the implication table. The implied pair of states for (a,b) is (b,c). But (b,c) is not included in the chosen set of (a,b) (c,d,e), so this set of compatibles is not closed. A set of compatibles that will satisfy the closed-covering condition is (a,d) (b,c) (c,d,e). The set is covered because it contains all five states. Note that the same state can be repeated more than once. The closure condition is satisfied because the implied states are (b,c) (d,e) and (a,d), which are included in the set. The original flow table (not shown here) can be reduced from five rows to three rows by merging rows a and d, b and c, and c, d and e.

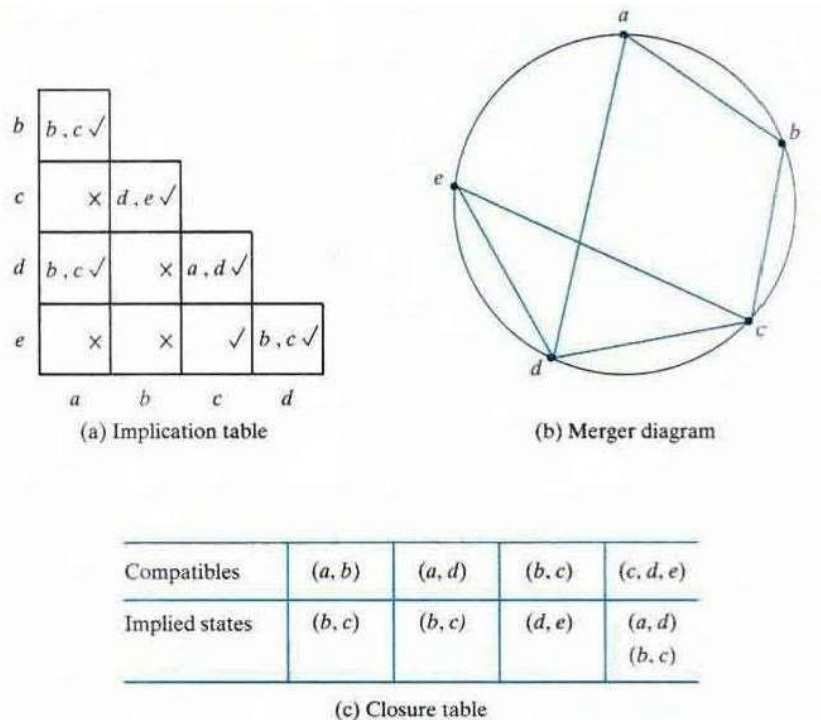


Fig 4.22 – a). Implication Table, b).Merger Diagram, c). Closure Table Image source from Digital Design by Moris Mano (Page No. 446)

RACE - FREE STATE ASSIGNMENT

Once a reduced flow table has been derived for an asynchronous sequential circuit, the next step in the design is to assign binary variables to each stable state. This assignment results in the transformation of the flow table into its equivalent transition table. The primary objective in choosing a proper binary state assignment is the prevention of critical races.

THREE-ROW-FLOW TABLE EXAMPLE

The assignment of a single binary variable to a flow table with two rows does not impose critical race problems. A flow table with three rows requires an assignment of

two binary variables. The assignment of binary values to the stable states may cause critical races if it is not done properly. Consider, for example, the reduced flow table of Fig.(a). The outputs have been omitted from the table for simplicity. Inspection of row *a* reveals that there is a transition from state *a* to state *b* in column 01 and from state *a* to state *c* in column 11. This information is transferred into a transition diagram, as shown in Fig.(b). The directed lines from *a* to *b* and from *a* to *c* represent the two transitions just mentioned. Similarly, the transitions from the other two rows are represented by directed lines in the diagram, which is a pictorial representation of all required transitions between rows.

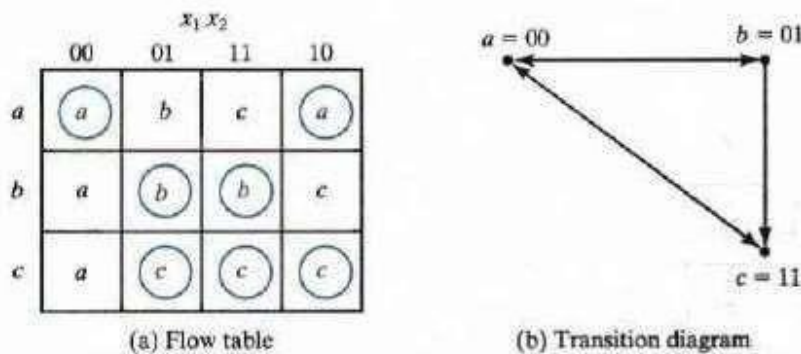


Fig 4.23 – a).Flow Table, b).Transition diagram

Image source from Digital Design by Moris Mano (Page No. 447)

To avoid critical races, we must find a binary state assignment such that only one binary variable changes during each state transition. An attempt to find such an assignment is shown in the transition diagram. State *a* is assigned binary 00, and state *c* is assigned binary

11. This assignment will cause a critical race during the transition from *a* to *c* because there are two changes in the binary state variables and the transition from *a* to *c* may occur directly or pass through *b*. Note that the transition from *c* to *a* also uses a race

condition, but it is noncritical because the transition does not pass through other states.

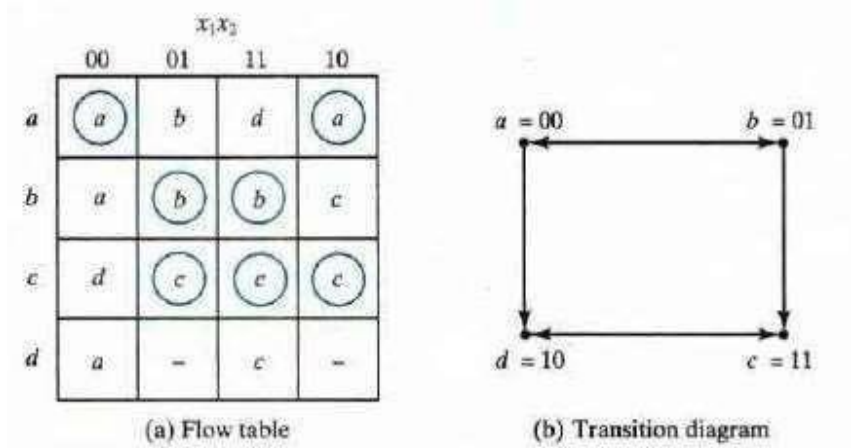


Fig 4.24 – a).Flow Table, b).Transition diagram

Image source from Digital Design by Moris Mano (Page No. 448)

FOUR-ROW-FLOW TABLE EXAMPLE

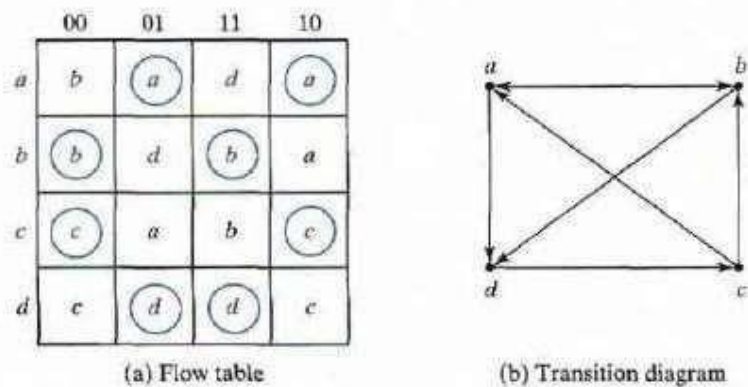


Fig 4.25 – a).Flow Table, b).Transition diagram

Image source from Digital Design by Moris Mano (Page No. 449)

With one or two diagonal transitions, there is no way of assigning two binary variables that satisfy the adjacency requirement. Therefore, at least three binary state variables are needed.

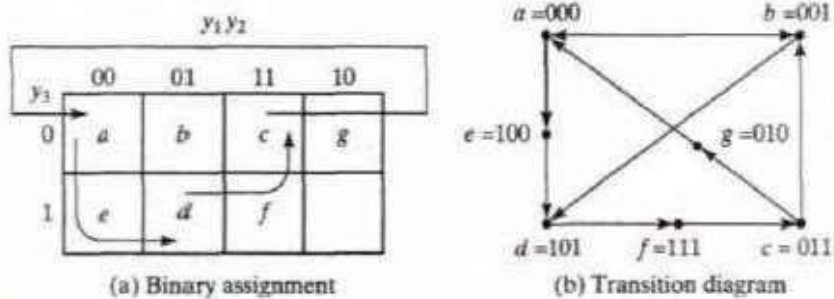


Fig 4.26 – a).Binary Assignment, b).Transition diagram
Image source from Digital Design by Moris Mano (Page No. 450)

www.binils.com

Fig 4.27 – Flow Table

	00	01	11	10
000 = a	b	a	e	a
001 = b	b	d	b	a
011 = c	c	g	b	c
010 = g	-	a	-	-
110 -	-	-	-	-
111 = f	c	-	-	c
101 = d	f	d	d	f
100 = e	-	-	d	-

Image source from Digital Design by Moris Mano (Page No. 450)

MULTIPLE-ROW METHOD

The method for making race-free state assignments by adding extra rows in the flow table, as demonstrated in the previous two examples is sometimes referred to as the shared-row method. A second method, called the multiple-row method, is not as efficient, but is easier to apply. In multiple-row assignment, each state in the original now table is replaced by two or more combinations of slate variables. There are two binary state variables for each stable state, each variable being the logical complement of the other. In the multiple-row assignment, the change from one stable state to another will always cause a change of only one binary state variable. Each stable state has two binary assignments with exactly the same output. At any given time, only one of the assignments is in use.

www.binils.com

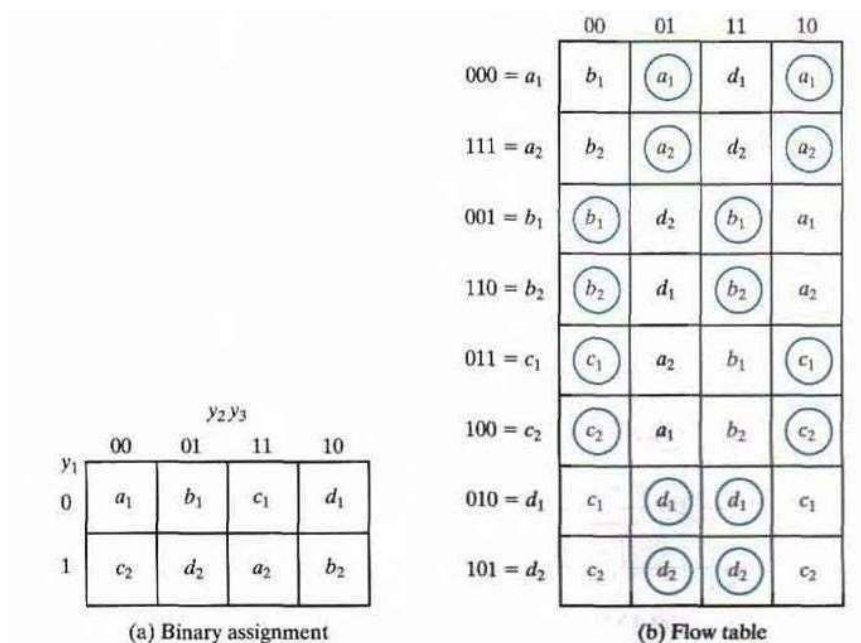


Fig 4.28 – a).Binary Assignment, b).Flow Table

Image source from Digital Design by Moris Mano (Page No. 451)