CS 8351 – DIGITAL PRINCIPLES AND SYSTEM DESIGN

UNIT – IV : ASYNCHRONOUS SEQUENTIAL CIRCUITS

REDUCTION OF STATE AND FLOW TABLES

State Table to Demonstrate Equivalent States

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

Image source from Digital Design by Moris Mano

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 $x = 0$	Next State $x = 1$	Output $x = 0$	Output $x = 1$
a	d	b	0	0
b	e	a	0	0
c	g	f	0	1
d	a	d	1	0
e	a	d	1	0
f	c	b	0	0
g	a	e	1	0

Fig 4.8 – State Table

Image source from Digital Design by Moris Mano

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 stares, 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 (J). 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 slate, 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.

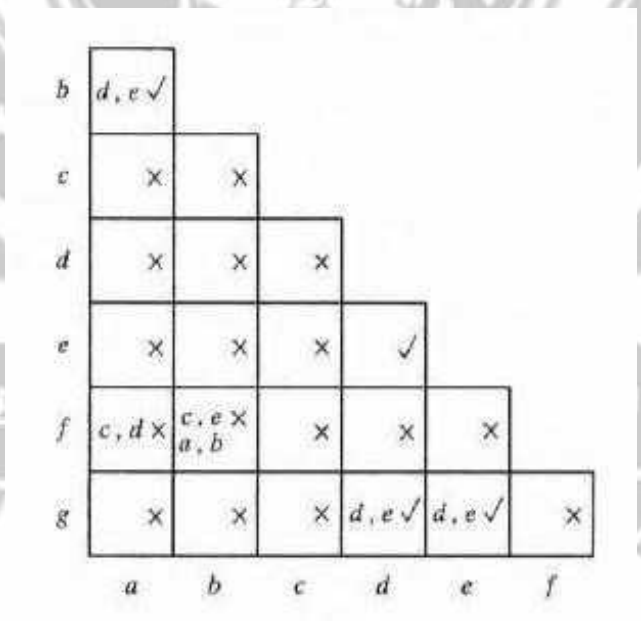


Fig : 4.8 - Implication Table

Image source from Digital Design by Moris Mano

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 stales. The last tthree 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

Reduced State 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 4.9 – Reduced Flow Table

Image source from Digital Design by Moris Mano

The process that must be applied in order to find a suitable group of compatibles (or the purpose of merging a flow tab le can be di vided 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, row s 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.

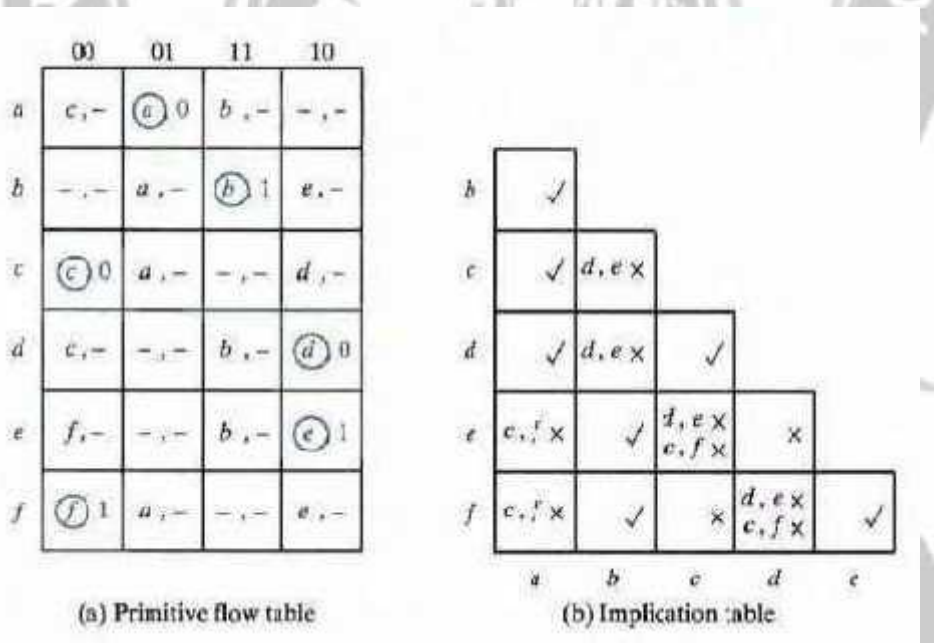


Fig 4.10 – a. Primitive Flow Table, b). Implication table Image source from Digital Design by Moris Mano

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 diagonal s 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)

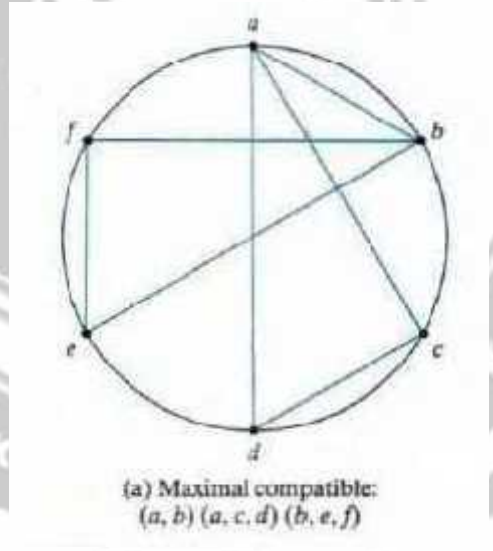


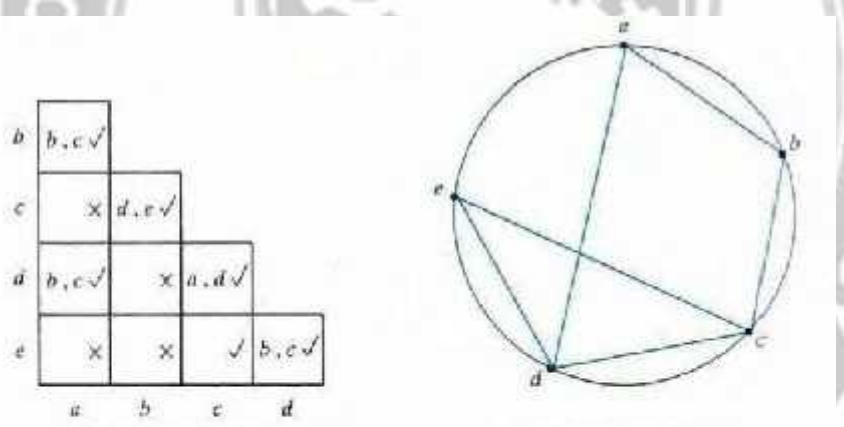
Fig 4.11 – Merger Diagram

Image source from Digital Design by Moris Mano

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 ofcompatibles 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 10 three rows by merging rows a and d, b and c, and c, d and e.



Compatibles	(a, b)	(a, d)	(b, c)	(c, d, e)
Implied states	(b, c)	(b, c)	(d, e)	(a, d) (b, c)

(c) Closure table

Fig 4.12 – a). Implication Table, b).Merger Diagram, c). Closure Table Image source from Digital Design by Moris Mano

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 slates 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 transition s from the other two rows are represented by directed lines in the diagram, which isa pictorial representation of all required transitions between rows.

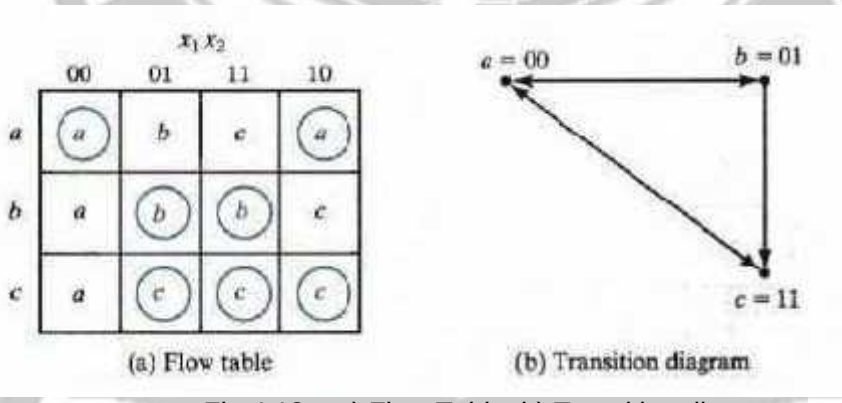


Fig 4.13 – a).Flow Table, b).Transition diagram

Image source from Digital Design by Moris Mano

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 tran sition from a to c may occur directly or pass through b. Note that the tran sition from c to a also ca uses a race condition, but it is noncritical because the transition does not pass through other states.

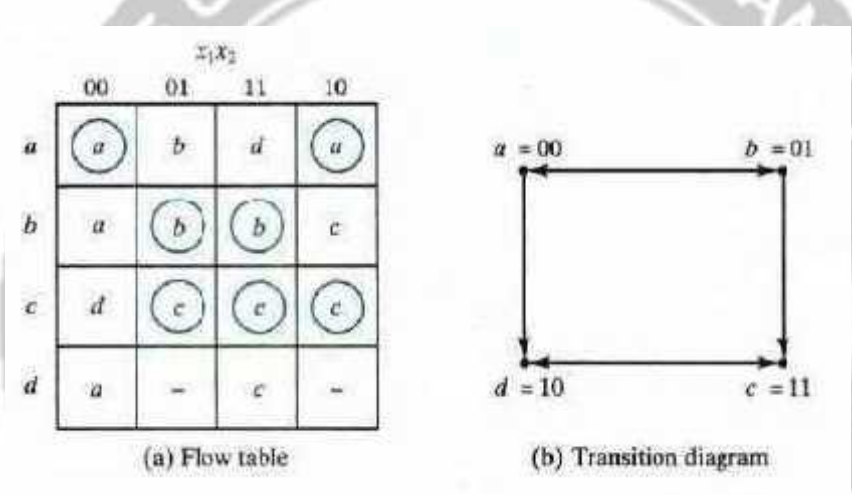


Fig 4.14 – a).Flow Table, b).Transition diagram

Image source from Digital Design by Moris Mano

FOUR-ROW-FLOW TABLE EXAMPLE

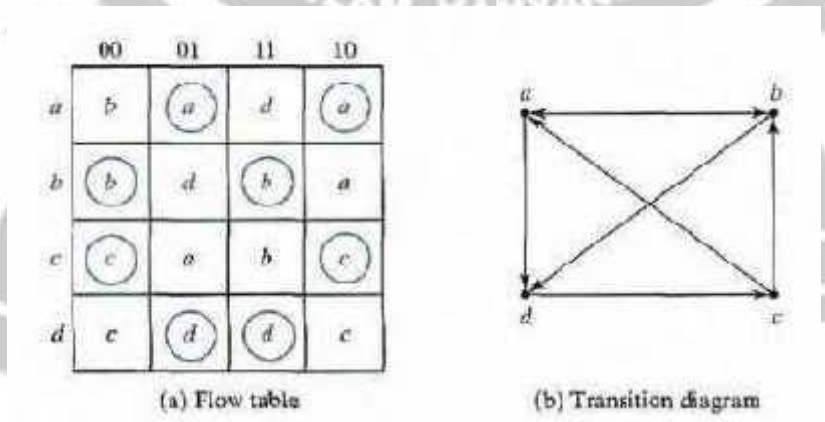


Fig 4.15 – a).Flow Table, b).Transition diagram

Image source from Digital Design by Moris Mano

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.

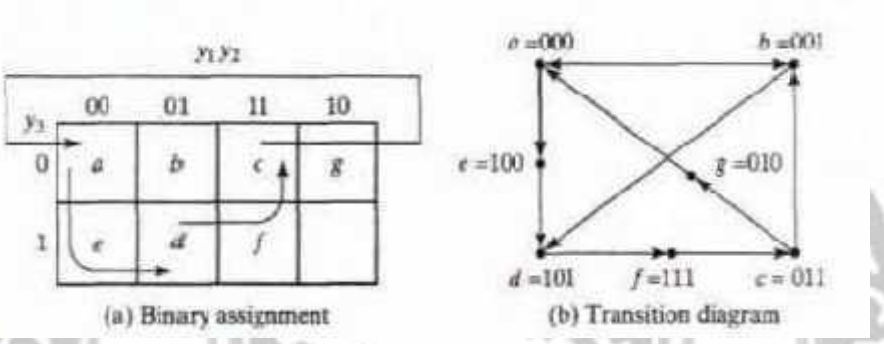


Fig 4.16 – a).Binary Assignment, b).Transition diagram Image source from Digital Design by Moris Mano

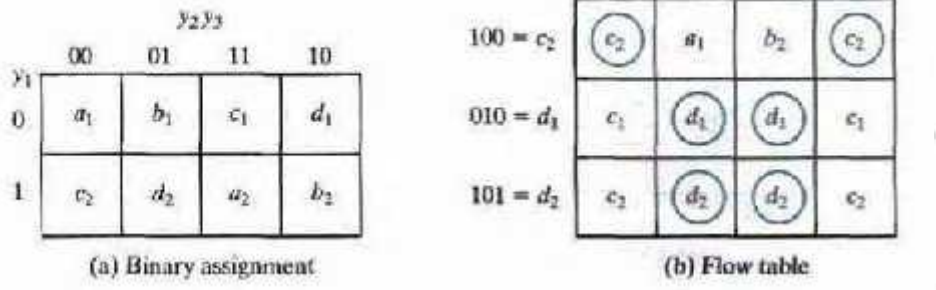
	00	01	11	10
000 = a	b	(a)	e	(a)
001 = b	(b)	d	(b)	a
011 = c	(c)	g	b	(c)
010 = g	–	a	–	–
110 –	–	–	–	–
111 = f	c	–	–	c
101 = d	f	(d)	(d)	f
100 = e	–	–	d	–

Fig 4.17 – Flow Table

Image source from Digital Design by Moris Mano

MULTIPLE-ROW METHOD

The method for making race-free stale 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.



(a) Binary assignment

y_1 \ y_2y_3	00	01	11	10
0	a_1	b_1	c_1	d_1
1	c_2	d_2	a_2	b_2

(b) Flow table

	00	01	11	10
$000 = a_1$	b_1	(a_1)	d_1	(a_1)
$111 = a_2$	b_2	(a_2)	d_2	(a_2)
$001 = b_1$	(b_1)	d_2	(b_1)	a_1
$110 = b_2$	(b_2)	d_1	(b_2)	a_2
$011 = c_1$	(c_1)	a_2	b_1	(c_1)
$100 = c_2$	(c_2)	a_1	b_2	(c_2)
$010 = d_1$	c_1	(d_1)	(d_1)	c_1
$101 = d_2$	c_2	(d_2)	(d_2)	c_2

Fig 4.18 – a).Binary Assignment, b).Flow Table

Image source from Digital Design by Moris Mano