1970

# Next-state equation generation for asynchronous sequential circuits - normal mode

Gregory Martin Bednar

# NEXT-STATE EQUATION GENERATION FOR ASYNCHRONOUS SEQUENTIAL CIRCUITS - NORMAL MODE

BY

GREGORY MARTIN BEDNAR, 1944-

A THESIS

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI - ROLLA

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

1970

Approved by

James H. Tracey (Advisor) William H. Tranter

Max Engelhardt

## ABSTRACT

This paper describes the known methods of generating next-state equations for asynchronous sequential circuits operating in normal fundamental mode. First, the methods that have been previously developed by other authors are explained and correlated in a simple and uniform language in order that the subtle differences of these approaches can be seen. This review is then followed by a development of a new method for generating minimal next-state equations which has some advantages over the previous methods.

From the comparison of the previous known methods, it is noted that any one of these methods may be desirable for certain designs since each has some advantages that the others do not have. However, these methods also have limitations in that some methods can only be used with particular types of assignments. Also, as flow tables become larger the amount of work required to use some of these methods becomes excessive and tedious.

The method developed here is a simple and straightforward approach which can be used for any unicode, single transition time assignment and will easily lend itself to computer application. The heart of this method emanates from the role that the Karnaugh map plays in the conventional approach for generating the next-state equations.

The main advantage of this method seems to be its capability and proficiency in handling large flow tables.

## ACKNOWLEDGEMENT

The author wishes to express his appreciation and extend his thanks to Dr. James H. Tracey for his assistance and guidance throughout this project.

The author also wishes to express his gratitude to his family for their interest and constant encouragement rendered during the writing of this paper.

## TABLE OF CONTENTS

## LIST OF ILLUSTRATIONS

# I. INTRODUCTION

Sequential switching circuits are those circuits whose operation and output depend on both the present and previous inputs. These circuits can be further classified into two categories called synchronous and asynchronous sequential circuits. Synchronous circuits are those circuits whose operation is timed or synchronized by clock pulses. Conversely, asynchronous circuits are not timed by clock pulses and offer the advantage of faster operation, being limited only by the speed of the circuit components involved in any particular operation.

In recent years, considerable work has been done in the area of generating next-state equations for asynchronous sequential circuits. Although this work has been accomplished by a number of people, it has been primarily done on an individual basis without knowledge of each other's efforts. The intent of this paper will be twofold. First, the major works of other authors in this area will be explained in a simple and uniform language so that the subtle differences of these approaches can be seen and correlated. Although the developments of the methods presented here may differ somewhat from the original presentations, the basic ideas of the credited authors are used. Second, a development of a new method for generating next-state equations will be given which has advantages over the other methods.

The next section of this paper will review the basic concepts of asynchronous sequential circuit theory. A reader who is qualified in this area may skip this section.

# II. BASIC CONCEPTS AND TERMINOLOGY

## A. Description of Asynchronous Sequential Circuits

Asynchronous sequential circuits are usually represented by a block diagram of the type shown in Figure 1.

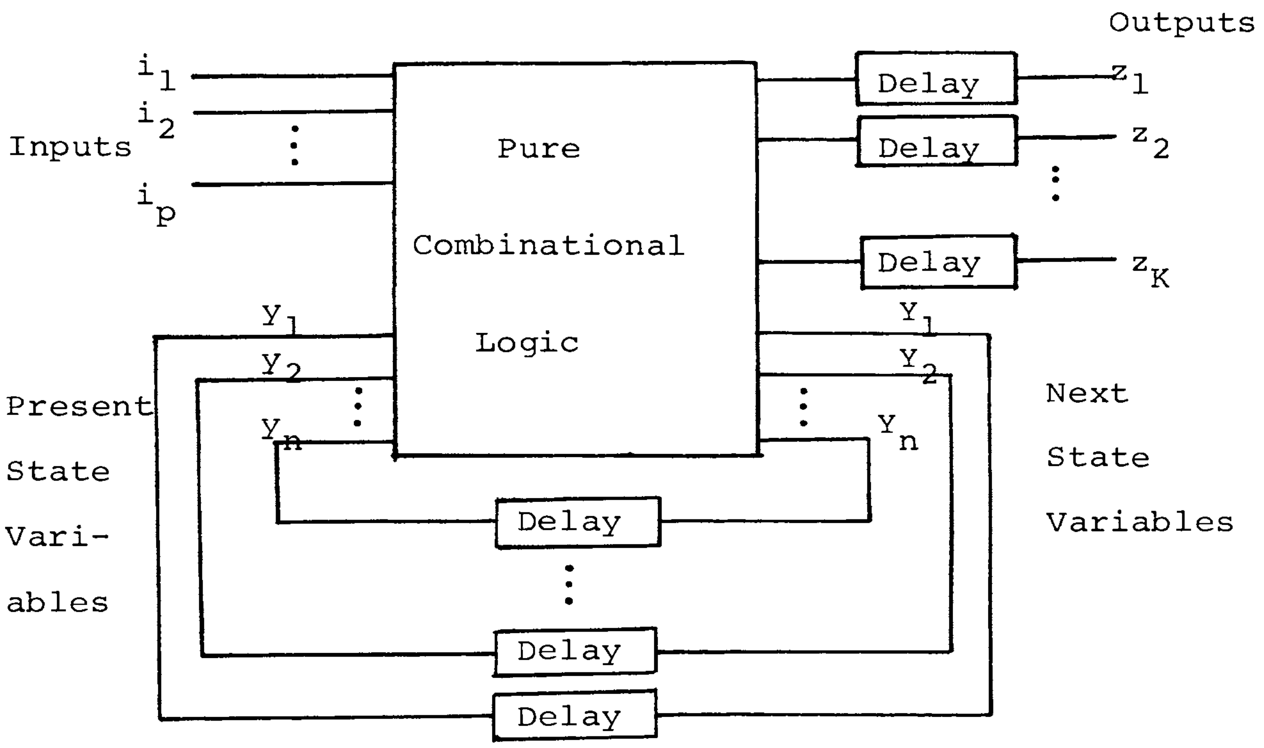


Figure 1. Model of an Asynchronous Sequential Circuit

The delay shown in Figure 1 is the time required for the signals to propagate through the combinational logic and is inherent in the physical circuit components. Since pure combinational logic (i.e., logic with no delay) can be represented mathematically by Boolean algebra in the form of output and next-state equations, the delay is considered to be removed from the logic and lumped separately as shown. In a simple mathematical model of an asynchronous circuit, the delay associated with the output logic is ignored while the delay of the next-state logic must be kept, since it is part of a required feedback loop. It is this lumped delay in the feedback loop that provides the interpretation or physical distinction between the next-state and present-state variables. The value of the next-state variable will become the present state after some delay in time.

Attention will now be directed toward obtaining the next-state equations for the model shown in Figure 1. If there are n internal-state variables and m input states for such a model, the general form for the next state equations will be:

$$
\begin{aligned}
Y_1 &= f_{11}(y_1,y_2,\cdots,y_n)I_1 + f_{12}(y_1,y_2,\cdots,y_n)I_2 + \cdots \\
&\quad + f_{1m}(y_1,y_2,\cdots,y_n)I_m \\
Y_2 &= f_{21}(y_1,y_2,\cdots,y_n)I_1 + f_{22}(y_1,y_2,\cdots,y_n)I_2 + \cdots \\
&\quad + f_{2m}(y_1,y_2,\cdots,y_n)I_m \\
&\vdots \\
Y_n &= f_{n1}(y_1,y_2,\cdots,y_n)I_1 + f_{n2}(y_1,y_2,\cdots,y_n)I_2 + \cdots \\
&\quad + f_{nm}(y_1,y_2,\cdots,y_n)I_m \qquad (1)
\end{aligned}
$$

where $y_1, y_2, \cdots, y_n$ are the present state variables; $Y_1, Y_2, \cdots, Y_n$ are the next-state variables; $I_1, I_2, \cdots, I_m$ are the input states; and $f_{11}, f_{12}, \cdots, f_{nm}$ are functions of the internal-state variables. From hereafter the next-state equations will be written in this form. In this paper, minimization of these equations will be done with respect to the function of the internal-state variables only and will not consider codings of the input states.

General information on this class of circuits can be found in references [1] and [2].

## 1. Flow Table

The flow table is one of the principal means of describing the operation of an asynchronous sequential circuit. As shown in Figure 2, it is a two-dimensional array consisting of next-state entries, with its columns representing the input states and its rows representing the internal states of the circuit. (The flow table usually shows the output states, too, but in this paper the output states are not relevant and therefore will not be shown in the flow table.) The row in which the circuit is currently operating is often referred to as the present internal state or just the present state. For example, if the present state of the circuit described by Figure 2 is c and then an input of $I_2$ is applied, the next state or state that the circuit will go to is e.

| | | $I_1$ | $I_2$ | Input States |
|---|---|---|---|---|
| | a | c | ⓐ | |
| Internal | b | c | ⓑ | |
| States | c | ⓒ | e | |
| | d | ⓓ | a | |
| | e | d | ⓔ | |

Figure 2. Flow Table for an Asynchronous Sequential Circuit

If a next-state entry is found to be the same as the internal state representing that row, then the internal state is said to be stable with respect to that input column and is denoted by a circled next-state entry. Similarly, uncircled entries denote unstable internal states.

## 2. Fundamental Mode

An asynchronous sequential circuit is said to be operating in fundamental mode if the inputs are never changed unless the circuit is in a stable state.

## 3. Normal Mode

An asynchronous sequential circuit is said to be operating in normal mode if each unstable state leads directly to a stable state.

4. Internal-State Assignment

An internal-state assignment is a binary coding for the internal states of a sequential circuit. For an asynchronous circuit is must be constructed in a manner such that the circuit will function according to flow table specifications, independent of variations in transmission delays within the circuit.

5. Transition

The change of values of state variables from the internal-state code associated with the present state to the code associated with the next state is said to be a transition from the present state to the next state.

6. Direct Transition

A transition whereby all state variables that are to undergo a change of state are simultaneously excited is direct transition.

7. Critical Race

A critical race is an undesirable feature of an internal-state assignment which occurs when the binary code of the next internal state differs from the code of the present state in two or more bit positions, and there is a possibility that unequal transmission delays may cause the circuit to reach a stable state other than the one intended.

### 8. Uni-code Single Transition Time Assignment

A state assignment is called a single transition time (STT) assignment when all transitions are direct transitions without critical races. Further, if only a single coding is associated with each state, it is called a uni-code single transition time (USTT) assignment.

### 9. State Table

A state table differs from a flow table in that a state table shows all of the internal states that a sequential circuit can assume along with corresponding next-state entries, whereas a flow table indicates only the initial and final states. For example, a flow table that has seven internal states and is coded with a four-variable internal-state assignment may have a corresponding state table with sixteen internal states. Of these sixteen states, those which are not involved in any transitions in the state table are referred to as the don't-care or unspecified states.

### 10. Transition Table

A transition table has the same form as the state table except that the next-state entries are replaced with their respective codes.

## B. Conventional Approach of Generating Next-State Equations

Assume that the flow table with the USTT assignment shown in Figure 3 describes the operation of a normal fundamental mode asynchronous sequential circuit.

| $y_1$ | $y_2$ | $y_3$ | | $I_1$ | $I_2$ | $I_3$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | a | ⓐ | b | ⓐ |
| 1 | 1 | 0 | b | c | ⓑ | ⓑ |
| 1 | 0 | 0 | c | ⓒ | b | a |
| 0 | 0 | 1 | d | a | ⓓ | e |
| 0 | 1 | 1 | e | ⓔ | d | ⓔ |

Figure 3. Flow Table

In using the conventional approach of generating the next-state equations [1] it is necessary to first construct the state and transition tables as shown in Figure 4. As previously mentioned, when constructing these tables all possible internal-state codings must be listed along with their corresponding next-state entries. Also since all transitions must be direct and hence carried out in a single transition time, it is necessary to insure that no critical races can occur. Therefore, to prevent a critical race during the transition from states a to b under input $I_2$, the intermediate state coded as 010 in the state table must have state b as its next-state entry under $I_2$.

| $y_1$ | $y_2$ | $y_3$ | | $I_1$ | $I_2$ | $I_3$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | a | $\textcircled{a}$ | b | $\textcircled{a}$ |
| 1 | 1 | 0 | b | c | $\textcircled{b}$ | $\textcircled{b}$ |
| 1 | 0 | 0 | c | $\textcircled{c}$ | b | a |
| 0 | 0 | 1 | d | a | $\textcircled{d}$ | e |
| 0 | 1 | 1 | e | $\textcircled{e}$ | d | $\textcircled{e}$ |
| 0 | 1 | 0 | - | - | b | - |
| 1 | 0 | 1 | - | - | - | - |
| 1 | 1 | 1 | - | - | - | - |

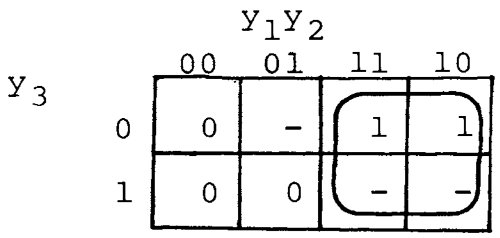
(a)

| $y_1$ | $y_2$ | $y_3$ | | $I_1$ | $I_2$ | $I_3$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | a | $\boxed{000}$ | 110 | $\boxed{000}$ |
| 1 | 1 | 0 | b | 100 | $\boxed{110}$ | $\boxed{110}$ |
| 1 | 0 | 0 | c | $\boxed{100}$ | 110 | 000 |
| 0 | 0 | 1 | d | 000 | $\boxed{001}$ | 011 |
| 0 | 1 | 1 | e | $\boxed{011}$ | 001 | $\boxed{011}$ |
| 0 | 1 | 0 | - | - | 110 | - |
| 1 | 0 | 1 | - | - | - | - |
| 1 | 1 | 1 | - | - | - | - |

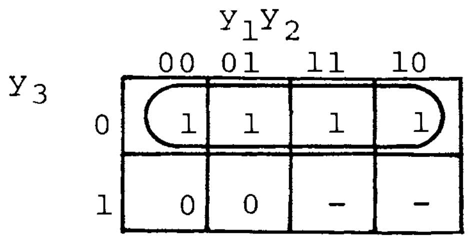
(b)

Figure 4. (a) State Table, (b) Transition Table

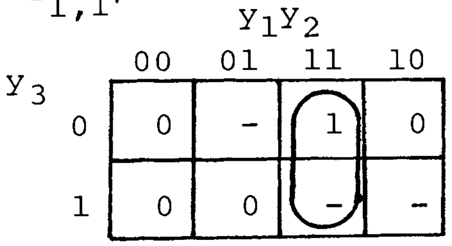
The next step is to construct Karnaugh maps from the transition table and then derive the next-state equations, $Y_i$'s, where i represents the ith state variable. The Karnaugh maps for $Y_1$ are shown in Figure 5.



$Y_1$ under $I_1$
(denoted as $Y_{1,1}$)



$Y_1$ under $I_2$
(denoted as $Y_{1,2}$)



$Y_1$ under $I_3$
(denoted as $Y_{1,3}$)

Figure 5. Karnaugh Maps for $Y_1$

By grouping the ones in the Karnaugh maps the sum-of-products form of a Boolean expression for $Y_1$ can be derived:

$$Y_1 = Y_{1,1} + Y_{1,2} + Y_{1,3}$$

$$Y_{1,1} = y_1I_1$$

$$Y_{1,2} = \bar{y}_3I_2$$

$$Y_{1,3} = y_1y_2I_3$$

Therefore,

$$Y_1 = y_1I_1 + \bar{y}_3I_2 + y_1y_2I_3$$

Similarly, the equations for $Y_2$ and $Y_3$ are:

$$Y_2 = \bar{y}_1 y_2 I_1 + \bar{y}_3 I_2 + (y_2 + y_3) I_3$$

$$Y_3 = \bar{y}_1 y_2 I_1 + y_3 I_2 + y_3 I_3$$

Definition: A y-variable expression is said to cover a set of states if for those states the expression is true (logical 1) and for all other states the expression is false (logical 0).

This definition implies that the above next-state equations cover those states whose corresponding next-state variable, $Y_i$, has a value of one in some input column.

It would now be good to analyze the foregoing procedure to determine what has actually taken place and why it works. The construction of the state and transition tables introduce the unspecified (or don't-care) states which are later used in the Karnaugh maps to obtain a reduced form of the next-state equations. Another main point that should be noted from these tables is that all states which lead to the same stable state under a particular input have identical next-state entries. This observation plays a prominent role in some of the other procedures to be discussed later. Next, it is observed that by using the Karnaugh map two important feats are accomplished:

1) It permits the grouping of all "1" next-state variable entries into subcubes which can easily be covered by subsets of internal-state variables. These subcubes are the largest possible subcubes which can be selected, such that they do not contain any "0" next-state variable entries. (However, they may contain don't-care entries.) Intuitively speaking, these groupings represent the states involved in those transitions in which the corresponding y-variable of the stable state is one.

2) It provides a minimal y-variable expression that covers those states which have a "1" for their $Y_i$ entry. This expression is formed from the internal-state variables that cover the subcubes obtained in 1) above.

Keeping these basic ideas in mind will provide insight in the development of the more sophisticated methods to be discussed later.

The main disadvantage of using the conventional method to generate the next-state equations is the amount of time needed to construct the state table, transition table, and Karnaugh maps, especially for large flow tables.

# III. REVIEW OF THE KNOWN METHODS FOR GENERATING NEXT-STATE EQUATIONS

## A. Method # 1 - G. K. Maki, J. H. Tracey, and R. J. Smith

Maki, Tracey and Smith [3] jointly developed a means to obtain the next-state equations directly from the flow table and the internal state assignment. The advantage of this method over the conventional approach is that the state table, transition table, and Karnaugh maps do not have to be explicitly formed.

Definition: A destination set of a flow table column is the set of all unstable states leading to the same stable state, together with that stable state.

This definition implies that a destination set is a collection of all those states under a particular input that have the same next-state entry. The destination sets for the flow table in Figure 3 are:

$$\begin{array}{ll} \underline{D_{I_1}} & \underline{D_{I_2}} \\ \underline{a}\ d & a\ \underline{b}\ c \\ b\ \underline{c} & \quad \underline{d}\ e \\ \quad \underline{e} & \end{array}$$

where $D_{I_i}$ deontes the set of destination sets under input $I_i$ and the stable state of each destination set is underlined.

Definition: Each pair of states consisting of one unstable state and the stable state of the destination set is called a *transition pair*, since there is a transition from the unstable state to the stable state.

Note that a destination set may contain one or more transition pairs. For example, the destination set $\underline{a}$ d under $I_1$ only contains one transition pair while the destination set a $\underline{b}$ c under $I_2$ consists of two transition pairs, a $\underline{b}$ and c $\underline{b}$.

Definition: An internal state $s_r$ is said to be an *intermediate state* between states $s_i$ and $s_j$ of a transition pair if the state variables may assume the value associated with the internal state $s_r$ during the transition from $s_i$ to $s_j$.

Definition: A *transition pair subspace* is a portion of the total state space having a span that consists of all possible internal states, both terminal and intermediate, which can be assumed by the circuit during a transition. This subspace is represented by a product function of the internal state variables.

Consider the transition pair a $\underline{b}$ from the destination set a $\underline{b}$ c under input $I_2$. The internal state coding for states a and b are 0 0 0 and 1 1 0 respectively. During the transition between states a and b, any of the internal states - - 0, where the dashes represent all combinations of 1's and 0's, could be assumed momentarily due to possible unequal transmission delays. To insure that the circuit reaches the proper terminal state, all the states represented by - - 0 must have the next-state entry of 1 1 0. It can be seen that this is the case in the transition table of Figure 4(b). These states form the transition pair subspace which is expressed as $\overline{y}_3$.

The method developed by Maki, Tracey and Smith [3] for representing the transition pair subspace as a product of internal-state variables is as follows:

1) List the codes assigned to the states of the transition pair.

2) The function that will represent the transition pair subspace will be a product of the internal-state variables. If the internal-state variable $y_j$ is a 1 for both states of the transition pair, it will appear uncomplemented in the product function. If the internal-state variable $y_j$ appears as a 0 in both of the states of the transition pair, its complement will appear in the product function. If the internal-state variable $y_j$ appears as both a 1 and a 0 in the

states of the transition pair, it is considered a don't-care variable and does not appear in the product expression.

Therefore, using these rules, - - 0 is expressed as $\bar{y}_3$.

The transition pair subspace is equivalent to the subcube spanned by the transition pair in a Karnaugh map. From the Karnaugh map of Figure 6, it is seen that the transition a to b will take place within the subcube covered by $\bar{y}_3$, where states 0 1 0 and 1 0 0 could be intermediate states of the transition. This agrees with the result obtained earlier for the transition pair subspace.

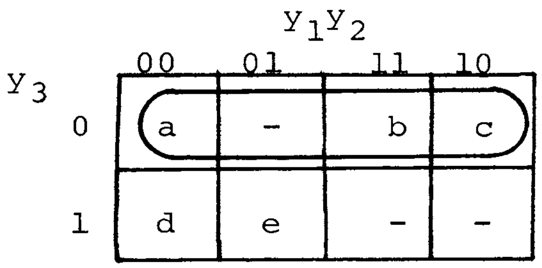


Figure 6. Karnaugh Map Showing a Transition Pair Subspace

Definition: A <u>destination</u> <u>set</u> <u>subspace</u> is that portion of the total state space consisting of all possible internal-states that the circuit could assume during transitions between states within the destination set.

The expression covering a destination set subspace is equal to the sum of the transition pair subspaces, where each transition pair subspace is represented as a product of the internal-state variables. Using the destination set a $\underline{b}$ c under input $I_2$, Figure 7 shows the steps in finding the destination set subspaces. Whereas the destination set itself represented all terminal states of a flow table

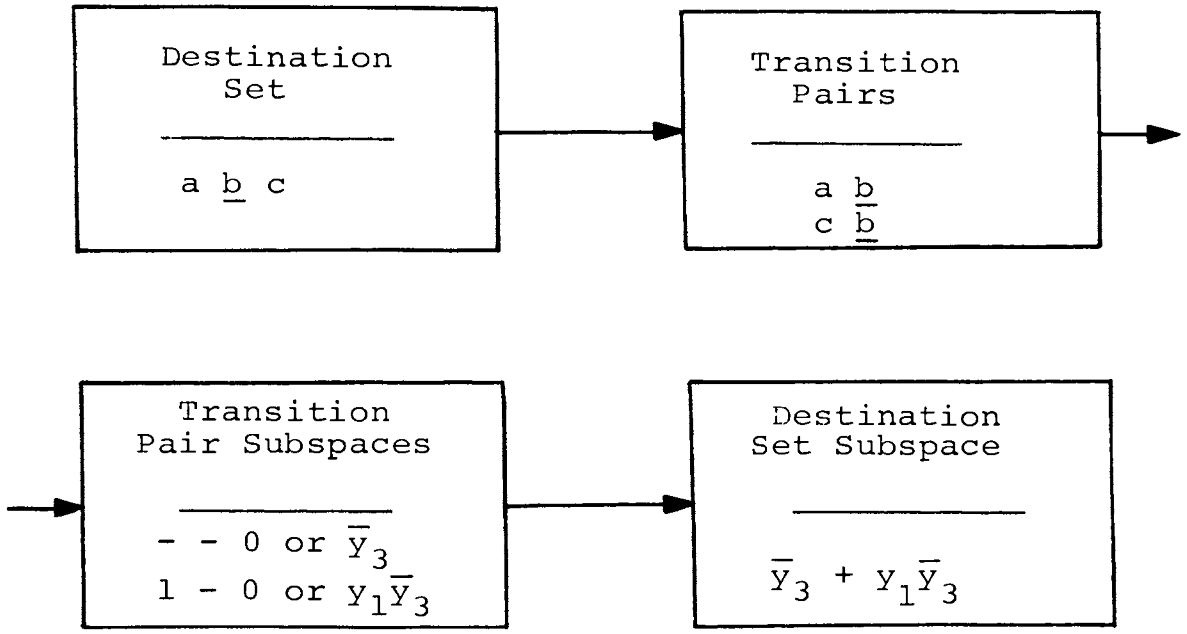


Figure 7. Calculating Destination Set Subspaces

column that have the same next-state entry, the destination set subspaces represent all specified internal states of a particular input column, both terminal and intermediate states, that have the same next-state entry. Therefore, destination set subspaces perform one of the functions of the state table in the conventional approach since the state table also designates all specified internal states. Another main function of the state table, which will now be considered, is the designation of all the unspecified states (don't-cares) of the circuit. In order to obtain the reduced form of the next-state equations, it is necessary to use the unspecified states.

A sum-of-products expression which logically represents all the specified states of an input column can be obtained by summing the expressions for the destination set subspaces of that column. It then follows that the unspecified states of that column would simply be the logical complement of this expression. The simplified expression for the unspecified states under input column $I_2$ is shown in Figure 8.

| $D_{I_2}$ | Destination Set Subspaces | Specified States under $I_2$ | Unspecified States under $I_2$ |
|---|---|---|---|
| a $\underline{b}$ c | $\bar{y}_3 + y_1\bar{y}_3$ | $\bar{y}_3 + y_1\bar{y}_3 + \bar{y}_1 y_3$ | $y_1 y_3$ |
| $\underline{d}$ e | $\bar{y}_1 y_3$ | | |

Figure 8. Calculating the Unspecified States

The essential information contained in the state table of the conventional approach is now available in the form of logical y-variable expressions.

To find the next-state equation $Y_i$, only those groupings of internal states which have a "1" next-state entry with respect to $Y_i$ are considered. These groupings of internal states correspond just to those destination set subspaces in which the $y_i$ variable in the binary coding for the stable state is a 1. Again. this is true because all next-state entries of the internal states represented by the destination set subspaces are the same, namely that of the stable state. So for example, if $y_1$ in the stable state is 1, then all entries of the next-state variable $Y_1$ will be 1 for those states represented by the respective destination set subspaces.

Definition: A destination set will be called a _1-destination set_ with respect to variable $y_i$ of $y_i = 1$ for the stable state and a _0-destination set_ if $y_i = 0$ for the stable state.

To complete the construction of the next-state equations it is only necessary to sum the logical expressions representing the subspaces of those 1-destination sets that are in the same input column along with the expression representing the unspecified states of that column. This will insure that the resulting next-state expression representing that column will be in reduced form after a simplification procedure is applied. Retaining the identity of the input column with the simplified expression, the process is then repeated for the remaining input columns at which time the final equation will be complete.

The process of combining the expressions representing the subspaces of the 1-destination sets with the expressions representing the unspecified states of each column is analogous to the function of the Karnaugh maps in the conventional approach. With the Karnaugh map, those states which had 1-entries for $Y_i$ were grouped with unspecified states in order to obtain the most minimal form of the next-state expression.

To help provide a better understanding, a more succinct picture of the foregoing method will be given by first summarizing the steps of the procedure and then following with an example.

| | Maki-Smith-Tracey <u>Procedure</u> | <u>Meaning</u> |
|---|---|---|
| 1. | List the destination sets for each input column. | Represents groupings of internal states of flow table that have same next-state entries in a column. |
| 2. | Find y-variable expressions for the transition pair subspaces of each destination set. | Equivalent to the subcube of the Karnaugh map in which the transition takes place. |
| 3. | Form the y-variable expressions representing the destination set subspaces by summing the expressions representing the transition pair subspaces of each destination set. | Represents the groupings of the specified internal states in a state table which have the same next-state entries. |

| | | |
|---|---|---|
| 4. | Find a y-variable expression for the specified states of each column by summing the expressions for all destination sets subspaces in a column. | Steps 4 and 5 provide the same information that is found in the state table of the conventional approach. |
| 5. | Find a y-variable expression for the unspecified states of each input column by taking the logical complement of the expression representing the specified states of that column. | |
| 6. | List the 1-destination sets and their subspaces for each input column. | Analogous to information found in the transition table and the Karnaugh maps. |
| 7. | Combine the expressions representing the subspaces of the 1-destination sets with the expression representing the unspecified states for each input column to find the minimal | Equivalent to using Karnaugh maps to find minimal next-state expressions by selecting groupings with maximum 1-entries and don't cares. |

form of next-state expressions. Retain the identity of each input state with the resulting expression for that column.

Example Problem

| $y_1$ | $y_2$ | $y_3$ | | $I_1$ | $I_2$ | $I_3$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | a | $\textcircled{a}$ | b | $\textcircled{a}$ |
| 1 | 1 | 0 | b | c | $\textcircled{b}$ | $\textcircled{b}$ |
| 1 | 0 | 0 | c | $\textcircled{c}$ | b | a |
| 0 | 0 | 1 | d | a | $\textcircled{d}$ | e |
| 0 | 1 | 1 | e | $\textcircled{e}$ | d | $\textcircled{e}$ |

The results of this method should be the same as the results obtained in the conventional approach.

Following the steps of the above procedure:

1. The destination sets are:

| $D_{I_1}$ | $D_{I_2}$ | $D_{I_3}$ |
|---|---|---|
| $\underline{a}$ d | a $\underline{b}$ c | $\underline{a}$ c |
| b $\underline{c}$ | $\underline{d}$ e | $\underline{b}$ |
| $\underline{e}$ | | d $\underline{e}$ |

2. The transition pair subspaces are:

| Input | Destination Sets | Transition Pairs | Transition Pair Subspaces |
|---|---|---|---|
| $I_1$ | $\underline{a}$ d | a d | 0 0 - or $\bar{y}_1\bar{y}_2$ |
| | $\underline{b}$ c | b c | 1 - 0 or $y_1\bar{y}_3$ |
| | $\underline{e}$ | e | 0 1 1 or $\bar{y}_1y_2y_3$ |
| $I_2$ | a $\underline{b}$ c | a b | - - 0 or $\bar{y}_3$ |
| | | b c | 1 - 0 or $y_1\bar{y}_3$ |
| | $\underline{d}$ e | d e | 0 - 1 or $\bar{y}_1y_3$ |
| $I_3$ | $\underline{a}$ c | a c | - 0 0 or $\bar{y}_2\bar{y}_3$ |
| | $\underline{b}$ | b | 1 1 0 or $y_1y_2\bar{y}_3$ |
| | d $\underline{e}$ | d e | 0 - 1 or $\bar{y}_1y_3$ |

3. The destination set subspaces are:

| Input | Destination Sets | Destination Set Subspaces |
|---|---|---|
| $I_1$ | $\underline{a}$ d | $\bar{y}_1\bar{y}_2$ |
| | $\underline{b}$ c | $y_1\bar{y}_3$ |
| | $\underline{e}$ | $\bar{y}_1y_2y_3$ |
| $I_2$ | a $\underline{b}$ c | $\bar{y}_3 + y_1\bar{y}_3$ |
| | $\underline{d}$ e | $\bar{y}_1y_3$ |
| $I_3$ | $\underline{a}$ c | $\bar{y}_2\bar{y}_3$ |
| | $\underline{b}$ | $y_1y_2\bar{y}_3$ |
| | d $\underline{e}$ | $\bar{y}_1y_3$ |

4. Expressions for specified states of each input column are:

| Input | Specified States |
|---|---|
| $I_1$ | $\bar{y}_1\bar{y}_2 + y_1\bar{y}_3 + \bar{y}_1y_2y_3$ |
| $I_2$ | $\bar{y}_3 + y_1\bar{y}_3 + \bar{y}_1y_3$ |
| $I_3$ | $\bar{y}_2\bar{y}_3 + y_1y_2\bar{y}_3 + \bar{y}_1y_3$ |

Note: The above expressions are not simplified.

5. Expressions for unspecified states of each column after simplification are:

| Input | Unspecified States |
|---|---|
| $I_1$ | $y_1y_3 + \bar{y}_1y_2\bar{y}_3$ |
| $I_2$ | $y_1y_3$ |
| $I_3$ | $y_1y_3 + \bar{y}_1y_2\bar{y}_3$ |

6. The 1-destination sets and their subspaces are:

| y-variable | Input | 1-Destination Sets | Subspaces |
|---|---|---|---|
| $y_1$ | $I_1$ | b $\underline{c}$ | $y_1\bar{y}_3$ |
| | $I_2$ | a $\underline{b}$ c | $\bar{y}_3 + y_1\bar{y}_3$ |
| | $I_3$ | $\underline{b}$ | $y_1y_2y_3$ |

| y-variable | Input | 1-Destination Sets | Subspaces |
|---|---|---|---|
| $y_2$ | $I_1$ | $\underline{e}$ | $\bar{y}_1y_2y_3$ |
| | $I_2$ | a $\underline{b}$ c | $\bar{y}_3 + \bar{y}_1\bar{y}_3$ |
| | $I_3$ | $\underline{b}$ | $y_1y_2\bar{y}_3$ |
| | | d $\underline{e}$ | $\bar{y}_1y_3$ |
| $y_3$ | $I_1$ | $\underline{e}$ | $\bar{y}_1y_2y_3$ |
| | $I_2$ | $\underline{d}$ e | $\bar{y}_1y_3$ |
| | $I_3$ | d $\underline{e}$ | $\bar{y}_1y_3$ |

7. Combining the subspaces for the 1-destination sets with the expressions for the unspecified states yields the following next-state equations:

$$Y_1 = [y_1\bar{y}_3 + d(y_1y_3 + \bar{y}_1y_2\bar{y}_3)]I_1$$
$$+ [\bar{y}_3 + y_1\bar{y}_3 + d(y_1y_3)]I_2$$
$$+ [y_1y_2\bar{y}_3 + d(y_1y_3 + \bar{y}_1y_2\bar{y}_3)]I_3$$

$$Y_2 = [\bar{y}_1y_2y_3 + d(y_1y_3 + \bar{y}_1y_2\bar{y}_3)]I_1$$
$$+ [\bar{y}_3 + y_1\bar{y}_3 + d(y_1y_3)]I_2$$
$$+ [\bar{y}_1y_3 + y_1y_2\bar{y}_3 + d(y_1y_3 + \bar{y}_1y_2\bar{y}_3)]I_3$$

$$Y_3 = [\bar{y}_1 y_2 y_3 + d(y_1 y_3 + \bar{y}_1 y_2 \bar{y}_3)]I_1$$

$$+ [\bar{y}_1 y_3 + d(y_1 y_3)]I_2$$

$$+ [\bar{y}_1 y_3 + d(y_1 y_3 + \bar{y}_1 y_2 \bar{y}_3)]I_3$$

where d( ) represents the subspaces of the unspecified states. After simplifying:

$$Y_1 = y_1 I_1 + \bar{y}_3 I_2 + y_1 y_2 I_3$$

$$Y_2 = \bar{y}_1 y_2 I_1 + \bar{y}_3 I_2 + (y_2 + y_3) I_3$$

$$Y_3 = \bar{y}_1 y_2 I_1 + y_3 I_2 + y_3 I_3$$

When simplifying, only those don't-care states are used which help simplify the next-state expressions. It is seen that the equations obtained with this method are identical to those of the conventional approach.

As mentioned earlier, the Maki, Tracey, and Smith method generates a minimal set of next-state equations without explicitly constructing the state and transition tables. It sould also be pointed out that this method will work for any satisfactory USTT assignment, i.e., no matter whether the assignment has been developed from a transition pair basis or a destination set basis. This is true because transition pairs make up destination sets, and this method uses transition pairs for its basic building blocks as opposed to other methods that use destination sets. Also, a computer program implementing this method has been developed by Smith et al. [4].

## B. Method # 2 - D. P. Burton and D. R. Noaks

Burton and Noaks [5] recognized that for large flow tables the derivation of next-state equations is a major difficulty. The motivation for their method was to establish a systematic procedure of obtaining a satisfactory USTT assignment for a normal-fundamental mode flow table that would lead to relatively simple next-state equations. The technique used to accomplish this goal involves generating the next-state equations in a semi-parallel fashion with the construction of the USTT assignment. The next-state equations have the characteristics that no product terms contain more than one state variable and no product terms contain any complemented state variables.

Definition: A product term in the next-state equation is said to be a simple product term if it contains only one internal-state variable.

Again, for correlation purposes, the same flow table that was used in the previously discussed procedures will be considered here and is repeated in Figure 9 for convenience. In this case, though, the USTT assignment is unknown.

| | $I_1$ | $I_2$ | $I_3$ |
|---|---|---|---|
| a | $\textcircled{a}$ | b | $\textcircled{a}$ |
| b | c | $\textcircled{b}$ | $\textcircled{b}$ |
| c | $\textcircled{c}$ | b | a |
| d | a | $\textcircled{d}$ | e |
| e | $\textcircled{e}$ | d | $\textcircled{e}$ |

Figure 9. Flow Table Without a State Assignment

As in the Maki, Tracey and Smith method, the first step in this procedure is to list the destination sets for each input column. In addition, associate with each destination set an internal-state variable, $y_i$, such that $y_i = 1$ for those states in the destination set and $y_i = 0$ for all other states. To help clarify the following discussion, the destination sets associated with a $y_i$ state variable will sometimes be referred to as the $y_i$ destination set. The destination sets and associated y-variables for the flow table in Figure 9 are:

| $\underline{D_{I_1}}$ | $\underline{D_{I_2}}$ | $\underline{D_{I_3}}$ |
|---|---|---|
| $y_1 \rightarrow \{\underline{a}\ d\}$ | $y_4 \rightarrow \{a\ \underline{b}\ c\}$ | $y_6 \rightarrow \{\underline{a}\ c\}$ |
| $y_2 \rightarrow \{b\ \underline{c}\}$ | $y_5 \rightarrow \{\underline{d}\ e\}$ | $y_7 \rightarrow \{\underline{b}\}$ |
| $y_3 \rightarrow \{\underline{e}\}$ | | $y_8 \rightarrow \{d\ \underline{e}\}$ |

This association of the state variables with the destination sets provides the following initial USTT assignment for the flow table:

| $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | $y_7$ | $y_8$ | Internal States |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | a |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | b |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | c |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | d |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | e |

In the previously discussed methods, the next-state equations consisted of y-variable expressions covering those states in which the next-state entries were 1. The same is true in Burton and Noaks' method. From the above assignment the next-state equations could be derived by constructing the Karnaugh maps as in the conventional approach, or by finding the expressions for the subspaces of the 1-destination sets as in the Maki, Tracey and Smith approach. But that would defeat the purpose of making a larger assignment in the manner shown, since it is supposed to make the construction of the next-state equations easier than in the previous methods.

It was stated previously that in Burton and Noaks' method each $y_i$ state variable will be a 1 for only those states in its associated $y_i$ destination set. Since all next-state entries for the states in a destination set are the same and equal to the stable state, the states of those destination sets whose stable states are contained in the $y_i$ destination will have $y_i = 1$ for their next-state entry. Therfore, to derive the next-state equation for some $Y_i$, the stable states of all destination sets are compared with the states in the $y_i$ destination set. If the stable state of some destination set, say the $y_j$ destination set, is also a member of the $y_i$ destination set, then the $Y_i$ equation must include a simple produce term $y_jI_m$, where $I_m$ is the input under which the $y_j$ destination set is located. Of course, if more than one destination set under the same input have their stable states contained in the $y_i$ destination set, the resulting equation for $Y_i$ will contain more than one simple product term with the same input. For instance, the $Y_i$ equation may contain two simple product terms like $y_jI_m + y_kI_m$ which equals $(y_j + y_k)I_m$.

To illustrate the above ideas, the next-state equation for $y_1$ will be derived. Since the $y_1$ destination set is $\{\underline{a}\ d\}$, $Y_1$ will equal one for all states in destination sets whose <u>stable</u> states are either "a" or "d". The destination sets under each input column which satisfy this condition are:

| $\underline{I_1}$ | $\underline{I_2}$ | $\underline{I_3}$ |
|---|---|---|
| $y_1 \rightarrow \{\underline{a}\ d\}$ | $y_5 \rightarrow \{\underline{d}\ e\}$ | $y_6 \rightarrow \{\underline{a}\ c\}$ |

Therefore, the next-state equation for $Y_1$ can easily be written as

$$Y_1 = y_1I_1 + y_5I_2 + y_6I_3$$

As previously noted, this same expression could be obtained using the conventional approach by finding the largest groupings of 1-entries and don't cares in a Karnaugh map, and with the Maki, Tracey and Smith method by combining the expressions of the subspaces for the 1-destination sets with the expressions for the unspecified states. But with such a large assignment these methods would require an excessive amount of work. For example, consider the derivation of the term $y_5I_2$ of the above equation, when using the conventional approach. The Karnaugh map for $Y_1$ and input column $I_2$, which would be constructed from the transition table, is shown in Figure 10. By circling the largest possible subcubes of the map which contains only "1" and don't-care entries, an expression for $Y_1$ under input $I_2$ can be obtained. As illustrated in this case, all 1-entries can be grouped into one subcube which is covered by the state variable $y_5$. Therefore, from the Karnaugh map we can write the expression $y_5I_2$ which agrees with the result obtained above. This example also illustrates that the use of don't-care states in the derivation of the y-variable expressions is inherent in Burton and Noaks' method.

$y_1y_2y_3y_4$

| $y_5\ y_6\ y_7\ y_8$ | 0000 | 0001 | 0011 | 0010 | 0110 | 0111 | 0101 | 0100 | 1100 | 1101 | 1111 | 1110 | 1010 | 1011 | 1001 | 1000 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 0 0 0 | – | 0 | – | – | – | – | 0 | – | – | 0 | – | – | – | – | 0 | – |
| 0 0 0 1 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 0 0 1 1 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 0 0 1 0 | – | 0 | – | – | – | – | 0 | – | – | 0 | – | – | – | – | 0 | – |
| 0 1 1 0 | – | 0 | – | – | – | – | 0 | – | – | 0 | – | – | – | – | 0 | – |
| 0 1 1 1 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 0 1 0 1 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 0 1 0 0 | – | 0 | – | – | – | – | 0 | – | – | 0 | – | – | – | – | 0 | – |
| 1 1 0 0 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 1 1 0 1 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 1 1 1 1 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 1 1 1 0 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 1 0 1 0 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 1 0 1 1 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |
| 1 0 0 1 | 1 | – | – | 1 | – | – | – | – | – | – | – | – | 1 | – | – | 1 |
| 1 0 0 0 | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – | – |

Figure 10. Karnaugh Map for $Y_1$ under $I_2$

The foregoing discussion explains the heart of Burton and Noaks' method. The remaining portions of their method deal with minimization techniques for eliminating redundant y-variables from the initial assignment and for simplifying the corresponding next-state equations. The description of these techniques will be stated in a narrative fashion with an informal plausible proof given. A more rigorous mathematical presentation and proof can be found in the reference cited [5].

Simplification Test I:

The first test which is applied to check for redundancy involves examining all terms in the next-state equations of the form $(y_i + y_j + \cdots)I_m$. If the union of the destination sets associated with the y-variables of this term is equal to:

1) some other $y_s$ destination set, then the term $(y_i + y_j + \cdots)I_m$ is replaced by $y_s I_m$.
2) the total state set (i.e., the set of all internal states in the flow table), then the term $(y_i + y_j + \cdots)I_m$ is replaced by $1 \cdot I_m$ or just $I_m$.

The reasoning behind these rules can be explained in the following manner:

1) If there exists the term $(y_i + y_j)I_m$ in a next-state equation and if this term is true

(logical 1), then either $y_i$ or $y_j$ is true. This implies that the circuit is operating in a state of the $y_i$ destination set or in a state in the $y_j$ destination set. If the union of the $y_i$ and $y_j$ destination sets equal the $y_s$ destination set, then all the states of $y_i$ and $y_j$ destination sets are in the $y_s$ destination set too. Therefore, whenever $y_i$ or $y_j$ is true, $y_s$ will also be true and the term $(y_i + y_j)I_m$ can be replaced by $y_sI_m$.

2) If there exists the term $(y_i + y_j)I_m$ and the union of the $y_i$ and $y_j$ destination sets equal the total state set, then the circuit will always be operating in a state of the $y_i$ destination set or in a state of the $y_j$ destination set. This implies that either $y_i$ or $y_j$ will always be true and therefore the expression $(y_i + y_j)$ can be replaced by the constant 1. (Note: in the term $(y_i + y_j)I_m$ both $y_i$ and $y_j$ could not be true at the same time since this would imply that two destination sets under the same input shared a common state which is the condition of a critical race.) These ideas can be extended to any term in the next-state equations of the form $(y_i + y_j + y_k + \cdots)I_m$.

An example of Test I can be given by considering the next-state equation for $y_4$ of the preceding assignment.

Remembering the method of derivation, the equation for $Y_4$ can be written as:

$$Y_4 = (y_1 + y_2)I_1 + y_4I_2 + (y_6 + y_7)I_3$$

It is observed that there are two compound terms in the equation. First, looking at the term $(y_i + y_2)I_1$, the union of the associated destination sets yields the set {a b c d} which is not the same as any one destination set. Therefore, the expression $(y_1 + y_2)$ cannot be reduced. Now looking at the other term $(y_6 + y_7)I_3$, the union of the $y_6$ and $y_7$ destination sets yield the set {a b c} which corresponds to the $y_4$ destination set. Therefore, by Test I, the expression $(y_6 + y_7)$ can be replaced by $y_4$. The reduced equation for $Y_4$ is now:

$$Y_4 = (y_1 + y_2)I_1 + y_4I_2 + y_4I_3$$

Simplification Test II:

If two or more destination sets contain identical states, whether or not they have the same stable state, then these destination sets are referred to as being equivalent and the same y-variable can be assigned to both sets. The reasoning behind this rule follows directly along the same lines of Test I. If two different y-variables represent equivalent destination sets, then both y-variables will be true at the same time. Therefore, one y-variable is redundant and can be eliminated. As an example, the destination sets for the flow table in Figure 9 will be considered. Under input $I_2$, $y_5$ was assigned to destination set ($\underline{d}$ e) and under input $I_3$, $y_8$ was assigned to destination set (d $\underline{e}$). Since both of these destination sets are equivalent (irrespective of which states are stable), the $y_8$ variable can be eliminated and the $y_5$ variable can be assigned to both destination sets. This eliminates one y-variable of the initial assignment and consequently one next-state equation.

Test II can also be restated in another way. If the right hand side of two or more next-state equations are identical, then only one of the y-variables associated with the left hand side of these equations is necessary for the assignment. This results from the fact that equivalent destination sets will always give the same next-state equations.

The next test in the minimization process is less specific and clear-cut, but more important than the previous tests. This test requires the construction of a dependency diagram which is used to select a minimum set of y-variables, from the remaining y-variables of the initial assignment, that will provide a satisfactory USTT assignment for the given flow table. Before listing the steps of this test, it is necessary that a few fundamental definitions and conditions for a satisfactory USTT assignment be stated.

Definition: A partition $\pi$ on a set of states is a grouping of the states into disjoint subsets called blocks, such that every state belongs to exactly one block.

For example, a two-block partition may be defined on the state set {a b c d e} as $\pi = \{(a\ b\ c),(d\ e)\}$.

Definition: The smallest partition of a set of states is denoted as $\pi(0)$ and corresponds to the partition in which each block consists of a single state.

From the preceding example, $\pi(0) = \{(a),(b),(c),(d),(e)\}$.

Definition: The blocks of a partition $\pi_g$ corresponding to the greater lower bound (g.l.b.) of partitions $\pi_1$ and $\pi_2$, written as $\pi_1 \cdot \pi_2$, consist of all the non-empty intersections that can be formed by intersecting a block of $\pi_1$ with a block of $\pi_2$. The g.l.b. operation is both commutative and associative.

For example, if

$$\pi_1 = \{(a\ b\ c),(d\ e)\} \text{ and } \pi_2 = \{(a\ b\ d),(c\ e)\}$$

then

$$\pi_g = \pi_1 \cdot \pi_2 = \{(a\ b),(c),(d),(e)\}\ .$$

Tracey [6] developed USTT assignment methods from the idea that each binary valued state variable may be thought of as inducing a two-block partition on the states of a machine. Tracey [6] also proved in his Assignment

Method #2, that an USTT assignment will contain no critical races if it has been made such that:

1) for every destination set $D_i$, if $D_j$ is another destination set in the same input column, then at least one y-variable partitions the destination sets $D_i$ and $D_j$ into separate blocks.
2) all internal states are distinguished from one another by being in separate blocks of at least one y-variable partition.

Since these conditions lead to the formation of the well-known Liu type assignment [7], they will be referred to hereafter as the Liu assignment conditions. Using this terminology will also help distinguish these conditions from the conditions of the well-known Tracey assignment [6] which will be discussed later.

It should be obvious that the initial assignment of Burton and Noaks will satisfy the Liu assignment conditions since a unique y-variable is assigned to each destination set. A dependency diagram is then used to show that a smaller set of the initial y-variable may exist that also satisfies the Liu conditions which would, therefore, result in a smaller USTT assignment.

Simplification Test III:

This test involves the simplification of the state assignment with the use of a dependency diagram. The rules for the construction of dependency diagrams are:

1) Let each y-variable of the assignment represent a node of the diagram.
2) Check each y-variable's next-state equation. If the next-state variable, $Y_i$, is dependent on other y-variables draw arrows from the nodes of the independent y-variables into the node of the dependent next-state variable $Y_i$.

No formal procedure is known for selecting the minimum set of state variables from the dependency diagram, and the trial and error method suggested here differs somewhat from the method presented by Burton and Noaks [5].

It is necessary to select a set of state variables which have next-state equations independent of all other state variables. This set can be found by drawing boundary lines through the dependency diagram. These lines must be constructed such that all arrows that pass through them are in the same direction. Although the boundary lines will separate sets of state variables which are independent of the remaining variables, this does not, however, imply that each set of independent variables will necessarily result in a satisfactory assignment. To determine whether the set of independent state variables selected do form a satisfactory assignment, the two-block partitions induced by these variables must satisfy the Liu assignment conditions mentioned earlier. First, it is necessary to insure that all destination sets under the same input are in separate blocks of at least one y-variable partition, and second, it is necessary to insure that all states are in separate blocks of at least one y-variable partition. The second condition will be satisfied if the g.l.b. of the y-variable partisions is equal to $\pi(0)$. If the Liu assignment conditions do not hold, the next larger set of independent y-variables is selected and tested. This process is continued until a set of y-variables are selected which will satisfy the Liu conditions. This set of variables will result in the minimal Burton and Noaks' assignment with corresponding minimal next-state equations.

This concludes the description of the Burton and Noaks' method. A brief summary of the procedure is given by the following steps:

1) List all destination sets of the flow table and assign to each set a unique state variable.
2) Construct the next-state equations for the initial state variables by using the strategy of simple product terms.
3) Apply minimization techniques.
    a) Test I - Replace compound terms with simpler terms.
    b) Test II - Eliminate redundant y-variables that represent the same destination set. (Note: This test may be applied concurrently with Step 1.)

c) Test III - Construct and use the dependency diagram.

The complete method will now be illustrated for the flow table in Figure 9.

Step 1.

The destination sets and associated y-variables are listed for each input column.

| $D_{I_1}$ | $D_{I_2}$ | $D_{I_3}$ |
|---|---|---|
| $y_1 \rightarrow \{\underline{a}\ d\}$ | $y_4 \rightarrow \{a\ \underline{b}\ c\}$ | $y_6 \rightarrow \{\underline{a}\ c\}$ |
| $y_2 \rightarrow \{b\ \underline{c}\}$ | $y_5 \rightarrow \{\underline{d}\ e\}$ | $y_7 \rightarrow \{\underline{b}\}$ |
| $y_3 \rightarrow \{\underline{e}\}$ | | $y_5 \rightarrow \{d\ \underline{e}\}$ |

Minimization Test II was applied concurrently with this step by assigning $y_5$ to two equal destination sets.

Step 2.

The set of next-state equations, written in matrix form for clarity, is:

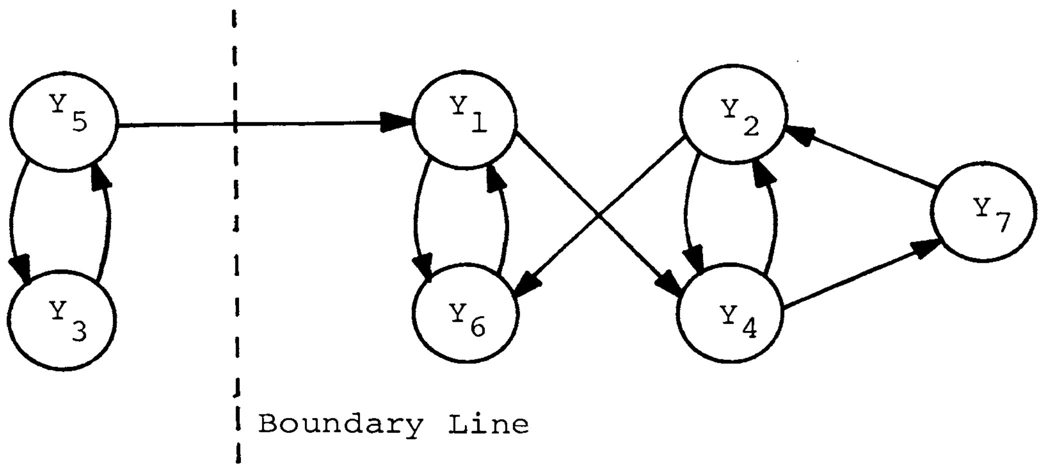
$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{bmatrix} = \begin{bmatrix} y_1 & y_5 & y_6 \\ y_2 & y_4 & y_7 \\ y_3 & 0 & y_5 \\ (y_1+y_2) & y_4 & (y_6+y_7) \\ y_3 & y_5 & y_5 \\ (y_1+y_2) & 0 & y_6 \\ 0 & y_4 & y_7 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix}$$

Step 3.

With minimization test I, only the expression $(y_6+y_7)$ is replaced and the resulting equations are:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{bmatrix} = \begin{bmatrix} y_1 & y_5 & y_6 \\ y_2 & y_4 & y_7 \\ y_3 & 0 & y_5 \\ (y_1+y_2) & y_4 & y_4 \\ y_3 & y_5 & y_5 \\ (y_1+y_2) & 0 & y_6 \\ 0 & y_4 & y_7 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix}$$

Since test II was utilized during Step 1, test III will now be applied. The dependency diagram for the above set of next-state equations is:



Only one boundary line exists for this diagram. The set $\{y_3, y_5\}$ is the independent set because these variables are not dependent on any of the other variables in the diagram. Conversely, the set $\{y_1, y_2, y_4, y_6, y_7\}$ is dependent on the set $\{y_5, y_3\}$ because $y_1$ depends on $y_5$ and $y_5$ in turn depends on $y_3$. The direction of the arrows crossing the boundary line will indicate which set of variables is the independent set.

It is obvious that the set $\{y_3, y_5\}$ is not sufficient to identify the five states of the flow table, since two binary valued variables can only code a maximum of four distinct states. The next largest independent set of y-variables to be tested is in this case the total state set itself. Of course, this set of variables will form a satisfactory assignment, since it is equivalent to the initial assignment given in page 30 (the reader should be convinced that the Liu assignment conditions hold for this assignment). Therefore, the minimum Burton and Noaks' assignment and simplified next-state equations for the flow table of this example are:

Assignment

| $y_1$ | $y_2$ | $y_3$ | $y_4$ | $y_5$ | $y_6$ | $y_7$ | | Internal States |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | - | a |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | - | b |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | - | c |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | - | d |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | - | e |

Next-State Equations

$$Y_1 = y_1I_1 + y_5I_2 + y_6I_3$$

$$Y_2 = y_2I_1 + y_4I_2 + y_7I_3$$

$$Y_3 = y_3I_1 + y_5I_3$$

$$Y_4 = (y_1+y_2)I_1 + y_4I_2 + y_4I_3$$

$$Y_5 = y_3I_1 + y_5I_2 + y_5I_3$$

$$Y_6 = (y_1+y_2)I_1 + y_6I_3$$

$$Y_7 = y_4I_2 + y_7I_3$$

As seen in the example problem, this method required only a seven-variable assignment while the previous methods only used a three-variable assignment for the same flow table. Although this method generally realizes a larger than minimal assignment, the property that no complemented variables are contained in the next-state equations may be a desirable feature with respect to the design and fabrication of the circuit.

The next method to be discussed will follow closely with some of the ideas of Burton and Noaks.

## C. Method # 3 - C. J. Tan

Tan [8] developed an iterative state assignment procedure for normal fundamental-mode asynchronous machines. Although one of Tan's goals was to try to realize low cost machines in terms of the number of logic gate inputs required in the realization of the next-state equations, the main intent here is to present the basic procedure used in the derivation of the next-state equations rather than presenting a cost study for fabricating the machine. As in the method of Burton and Noaks, Tan [8] derives his state assignment and next-state equations in a parallel fashion.

Before proceeding with the derivation and explanation of Tan's specific procedure, a brief review of the theory behind the iterative approach will be presented. It should be pointed out, that although the following presentation is based on the concept of transition pairs, since this is the most fundamental approach, it can easily be extended to the concept of destination sets.

## 1. Basic Theory of Tan's Procedure

Definition: An unordered pair of disjoint subsets of the states of the machine is referred to as a *dichotomy*.

The dichotomy is equivalent to a two-block partition in which all the states of the total state set are not necessarily specified. For example, if two transitions under the same input are $a \rightarrow b$ and $c \rightarrow d$, the dichotomy associated with these transitions is $(a\underline{b}, c\underline{d})$.

Definition: A state variable $y_i$ is said to *cover* a dichotomy if $y_i = 0$ for all states in one block of the dichotomy and $y_i = 1$ for all states in the other block.

For example, $y_i$ is said to cover the dichotomy $(a\underline{b}, c\underline{d})$ if $y_i = 0$ in the binary coding of states a and b, and $y_i = 1$ in states c and d or vice versa.

Definition: A dichotomy consisting of two transition pairs is said to be *relevant* to a state variable $y_i$, if $y_i = 0$ for the stable state of one transition pair and $y_i = 1$ for the stable state of the other transition pair.

For example, the dichotomy of transition pairs $(a\underline{b}, c\underline{d})$ is relevant to $y_i$, if $y_i = 0$ for b and $y_i = 1$ for d or vice versa. It should be noted that relevancy is independent of the value of $y_i$ for the unstable states a and c.

Definition: A transition $a \rightarrow b$ in some column of a flow table will be called a <u>1-transition</u> with respect to state variable $y_i$, if $y_i = 1$ for the stable state b and a <u>0-transition</u> if $y_i = 0$ for b.

A relevant dichotomy is therefore a dichotomy containing transition pairs involved in a 1-transition and a 0-transition with respect to some $y_i$. This definition also implies that all 1-transitions with respect to $y_i$ result from those transition pairs contained in the 1-destination sets of that $y_i$.

Since the assignment and corresponding next-state equations are going to be derived simultaneously, the rules for the construction of both have to be observed.

Again, in this procedure, the Tracey [6] conditions must hold in order to have a satisfactory USTT assignment. Since transition pairs rather than destination sets are being dealt with here, the conditions differ slightly from those used in the Burton and Noaks method. From Tracey's fundamental theorem [6] the necessary conditions for a USTT assignment are:

1) If $(S_i, S_j)$ and $(S_m, S_n)$ are transitions in the same flow table column, then at least one y-variable must partition $(S_i, S_j)$ and $(S_m, S_n)$ into separate blocks.

2) If $(S_i, S_j)$ is a transition and $S_k$ a lone stable state in the same column, then at least

one y-variable must partition ($S_i$, $S_j$) and $S_k$ into separate blocks.

3) For $i \neq j$, $S_i$ and $S_j$ must be in separate blocks of at least one y-variable partition.

Notice that the Tracey conditions stated in the Burton and Noaks method are really an extension of these conditions. If a y-variable partitions two destination sets into separate blocks, then this same y-variable will partition the transition pairs contained in these destination sets into separate blocks. However, the opposite may not be true in cases where destination sets contain more than one transition pair. Therefore, the above conditions dealing with transition pairs have been accepted as the fundamental theorem and all USTT assignments must satisfy it. To restate this theorem more succinctly and in terms used in Tan's procedure, a USTT assignment exists if:

1) The dichotomies associated with every pair of transitions (including lone stable states) occurring in each column of the flow table should be covered by some state variable.

2) Every state has a unique coding.

Condition 1 stated above is the same as conditions 1 and 2 of Tracey's Theorem. If a y-variable covers a dichotomy of transition pairs, this implies that the transition pairs will be in separate blocks of that y-variable partition.

The conditions for the construction of the next-state equations using the concept of transition pairs will now be discussed. As in the case of the state assignment, the previous methods have primarily dealt with the next-state equations from a destination set point of view.

In the conventional approach, the next-state equations resulted from circling the largest groups of 1-entries and don't-cares in a Karnaugh map. These 1-entries in the Karnaugh map came from the transition table and were the next-state entries for those states in 1-destination sets. Or it could be said that they were the 1-entries for those states of the transition pairs contained in the 1-destination sets. This is also the case for the Maki, Tracey, and Smith method, since it is the transition pairs subspaces which make up the subspaces for the 1-destination sets. Therefore, the expression resulting from a Karnaugh map is actually a minimum variable cover which separates all states involved in 1-transitions of a flow table from those states involved in 0-transitions. If the expression is true or false then the circuit is involved in a 1-transition or 0-transition respectively.

The preceding ideas will help explain the theory of Tan's iterative approach. From the flow table the dichotomies of all transition pairs in each input column are listed. The first state variable, $y_1$, with its corresponding induced partition is then selected. Some guidelines for selecting good initial state variables will be given later. All dichotomies are now examined and those found to be relevant to $y_1$ are so designated. These relevant dichotomies separate the 1-transition pairs from the 0-transition pairs with respect to $y_1$. This is what circling the 1-entries in the Karnaugh map essentially did. Now it is necessary to find the cover for the groups of 1-entries or, in Tan's case, for the relevant dichotomies. This cover will insure that the 1-transitions remain separated from the 0-transitions. To find this cover, additional y-variables are chosen until all relevant dichotomies have been covered. From the set of state variables that covers all dichotomies relevant to $y_1$, a simplified sum-of-products expression for $Y_1$ can be written. This procedure is repeated for each y-variable selected. When enough y-variables have been selected to cover all dichotomies, a satisfactory USTT assignment with corresponding next-state equations will exist.

In Figure 11, a columnar Karnaugh map is used to help illustrate the concepts of relevancy and cover by considering the dichotomy ($\underline{a}d$, $\underline{e}$), under input $I_1$ of the flow table in Figure 12. Since $y_1$ codes both stable states a and e with the same value (i.e., a value of 0), the dichotomy ($\underline{a}d$, $\underline{e}$) is not relevant to $y_1$. Therefore, $y_1$ is not useful for separating 1-transitions from 0-transitions as can be verified by noting that both transition pairs of this dichotomy have 0-entries for $Y_1$. Now looking at $y_2$ it is seen that it codes stable states a and e of the dichotomy (ad, e) differently. Hence the dichotomy is relevant to $y_2$ and it does distinguish 1-transitions from 0-transitions as is evident from the next-state entries, $Y_2 = 0$ for states a and d and $Y_2 = 1$ for e. A cover for this relevant dichotomy must now be found. Since $y_2$ itself is a 0 for states a and d and 1 for e, $y_2$ covers the dichotomy (ad, e). Finally, it is seen that the dichotomy (ad, e) is also relevant to $y_3$ but $y_3$ is not a cover for this dichotomy, because $y_3$ does not code states a and d the same. Again, $y_2$ is the cover for this case. The same procedure can be used to find and verify the relevant dichotomies and covers for the remaining dichotomies of transition pairs contained in the flow table of Figure 12. The results for all dichotomies of this flow table are tabulated in Figure 14.

As a final point, it should be noted that the covers selected for the relevant dichotomies do implicitly make use of don't-care states, in that the subcubes of the Karnaugh map represented by the covers may contain some don't-care states.

The sum of products expressions for the next-state equations can be formed by following the implicit rules of Karnaugh mapping. These rules put into words are:

1) If the y-variable that covers a dichotomy is a 1 for the 1-transition pair of the dichotomy, it will appear uncomplemented in a product term

| $y_1$ | $y_2$ | $y_3$ | | $Y_1$ | $Y_2$ | $Y_3$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | – a | 0 | 0 | 0 |
| 0 | 0 | 1 | – d | 0 | 0 | 0 |
| 0 | 1 | 1 | – e | 0 | 1 | 1 |
| 0 | 1 | 0 | – | – | – | – |
| 1 | 1 | 0 | – b | 1 | 0 | 0 |
| 1 | 1 | 1 | – | – | – | – |
| 1 | 0 | 1 | – | – | – | – |
| 1 | 0 | 0 | – c | 1 | 0 | 0 |

Figure 11. Columnar Karnaugh Map Showing Next-State Entries Under Input $I_1$ of the Flow Table in Figure 12

of the next-state expression. Conversely, if it is a 0 for the 1-transition pair, it will appear complemented in the product term.

2) If only one relevant dichotomy appears under a particular input, or if more than one relevant dichotomy appears but are covered by the same y-variable, then a simple product term of the form $\underline{y}_i I_m$ will occur in the next-state equation where $\underline{y}_i$ is the respective cover for the dichotomies. (Note: $\underline{y}_i$ implies either $y_i$ or $\overline{y}_i$).

3) If more than one relevant dichotomy appears under a particular input, and each has a

different 1-transition pair then a product term of the form $(\underline{y}_i + \underline{y}_j + \cdots + \underline{y}_n)I_m$ will occur in the next-state equation, where the state variables in the term represent the respective covers for the relevant dichotomies.

4) If the same 1-transition pair appears in more than one relevant dichotomy under a particular input, then a product term of the form $\underline{y}_i\underline{y}_j \cdots \underline{y}_n I_m$ will occur in the next-state equation. Conversely, if no dichotomies under a particular input are relevant to $y_i$, then no product term for this input will occur in the equation for $Y_i$.

As an example of Tan's theory, the flow table which was used in the previous methods will again be considered here and is repeated in Figure 12.

| | $I_1$ | $I_2$ | $I_3$ |
|---|---|---|---|
| a | ⓐ | b | ⓐ |
| b | c | ⓑ | ⓑ |
| c | ⓒ | b | a |
| d | a | ⓓ | e |
| e | ⓔ | d | ⓔ |

Figure 12. Flow Table

The first step would be to list all the destination sets under each input.

| $D_{I_1}$ | $D_{I_2}$ | $D_{I_3}$ |
|---|---|---|
| $\underline{a}$ d | a $\underline{b}$ c | $\underline{a}$ c |
| b $\underline{c}$ | $\underline{d}$ e | $\underline{b}$ |
| $\underline{e}$ | | d $\underline{e}$ |

From the destination sets the dichotomies of transition pairs that can be formed are:

| $I_1$ | $I_2$ | $I_3$ |
|---|---|---|
| ($\underline{a}$ d, b $\underline{c}$) | (a $\underline{b}$, $\underline{d}$ e) | ($\underline{a}$ c, $\underline{b}$) |
| ($\underline{a}$ d, $\underline{e}$) | ($\underline{b}$ c, $\underline{d}$ e) | ($\underline{a}$ c, d $\underline{e}$) |
| (b $\underline{c}$, $\underline{e}$) | | ($\underline{b}$, d $\underline{e}$) |

For the selection of the initial state variable, let $y_1$ induce the partition {(a d e),(b c)} such that states in block (a d e) are coded with a 0 and states in block (b c) are coded with a 1. A searching procedure is now initiated, as shown in Figure 13, to find those dichotomies which are relevant to $y_1$.

It is observed in Figure 13 that two relevant dichotomies are not covered. To cover these dichotomies the state variables $y_2$ and $y_3$, which induce the partitions {(a c d),(b e)} and {(a b c),(d e)} respectively, are chosen. Therefore, $Y_1 = f_1(I_m, \sigma_1)$ where $\sigma_1 = \{y_1, y_2, y_3\}$. The table in Figure 13 is extended to include $y_2$ and $y_3$ in Figure 14. Notice that all dichotomies of the flow table are now covered and each state has a unique coding. This means that the Tracey conditions are satisfied and

| Partial Assignment $y_1$ | | States |
|---|---|---|
| 0 | - | a |
| 1 | - | b |
| 1 | - | c |
| 0 | - | d |
| 0 | - | e |

| Input | Dichotomies | $y_1$ |
|---|---|---|
| $I_1$ | $(\underline{a}d, b\underline{c})$ | X |
| | $(\underline{a}d, \underline{e})$ | - |
| | $(b\underline{c}, \underline{e})$ | X |
| $I_2$ | $(a\underline{b}, \underline{d}e)$ | √ |
| | $(\underline{b}c, \underline{d}e)$ | X |
| $I_3$ | $(\underline{a}c, \underline{b})$ | √ |
| | $(\underline{a}c, d\underline{e})$ | - |
| | $(\underline{b}, d\underline{e})$ | X |

Legend

- means not relevant

√ means relevant to $y_i$ but not covered by $y_i$

X means relevant to $y_i$ and covered by $y_i$

Figure 13. Determination of Relevant Dichotomies

the USTT assgnment will also consist of the set $\{y_1, y_2, y_3\}$. The equations for $Y_1$, $Y_2$, and $Y_3$ can be written directly from Figure 14 using the rules previously stated.

These equations are:

$$Y_1 = y_1I_1 + \bar{y}_3I_2 + y_1y_2I_3$$

$$Y_2 = \bar{y}_1y_2I_1 + \bar{y}_3I_2 + (y_2 + y_3)I_3$$

$$Y_3 = \bar{y}_1y_2I_1 + y_3I_2 + y_3I_3$$

In some cases where two different variables cover the same dichotomy either variable may be used in the product term.

Partial Assignment

| $y_1$ | $y_2$ | $y_3$ | States |
|---|---|---|---|
| 0 | 0 | 0 | a |
| 1 | 1 | 0 | b |
| 1 | 0 | 0 | c |
| 0 | 0 | 1 | d |
| 0 | 1 | 1 | e |

| Input | Dichotomies | $y_1$ | $y_2$ | $y_3$ |
|---|---|---|---|---|
| $I_1$ | $(\underline{a}d, b\underline{c})$ | X | - | - |
| | $(\underline{a}d, \underline{e})$ | - | X | ✓ |
| | $(b\underline{c}, \underline{e})$ | X | ✓ | X |
| $I_2$ | $(a\underline{b}, \underline{d}e)$ | ✓ | ✓ | X |
| | $(\underline{b}c, \underline{d}e)$ | X | ✓ | X |
| $I_3$ | $(\underline{a}c, \underline{b})$ | ✓ | X | - |
| | $(\underline{a}c, d\underline{e})$ | - | ✓ | X |
| | $(\underline{b}, d\underline{e})$ | X | - | X |

Figure 14. Complete Cover of Relevant Dichotomies

This is equivalent to having more than one choice in which the 1-entries of a Karnaugh map can be circled.

It should now be pointed out that in the above example the selection of the state variables was predetermined. The state variables were chosen so that the resulting assignment would be the same as the one used in the conventional approach and in the Maki, Tracey and Smith method. This was done to show that the same next-state equations could be obtained with Tan's method as with these other approaches.

This concludes the review of the fundamental theory of Tan's iterative approach. A specific iterative procedure developed by Tan [8] will now be discussed. This procedure results in a Liu type assignment and so is only concerned with destination sets and not with transition pairs. Hence, the theory given above will be somewhat extended to a destination set point of view.

## 2. Tan's Specific Iterative Procedure

In Tan's specific iterative procedure [8], the derivation of the next-state equations is very similar to the method used by Burton and Noaks. One difference in the methods is that Tan does not start with a complete initial assignment by assigning all destination sets a unique y-variable. Instead, an initial partial-state assignment is made by discreetly assigning only a few y-variables to those destination sets which help minimize the number of additional variables still needed for a satisfactory USTT assignment.

Some guidelines for selecting this initial partial-state assignment are listed in the following priority:

1) Select those y-variables that will partition the destination sets of more than one input column in the same manner. Each additional input column partitioned will result in a savings of one y-variable in the final assignment. One way in which these y-variables may be found is by assigning a y-variable to a destination set which appears in more than one input column, because y-variables assigned to the same destination sets will induce identical

partitions. This rule is essentially the same as the Simplification Test II of Burton and Noaks, which said that y-variables assigned to equal destination sets are redundant and only one y-variable is needed for all such sets.

2) Select those y-variables which take on the binary value 0 for all the stable states in a particular input column. These y-variables can be determined by finding those induced partitions in which all of the stable states of an input column will appear in the 0 coded block of the partition. If $y_i = 0$ for all stable states in an input column then no product term for this input column will appear in the equation for $Y_i$.

3) Select those y-variables such that in their respective next-state equations, most of the product terms also appear in some other equations. Hence, the cost of these equations will likely be small.

The third selection criterion is the least definite of the three and would probably only be used when criteria one and two failed to yield a sufficient number of state variables for an initial partial-state assignment. In many cases this criterion will yield more variables than what is actually needed for a satisfactory partial-state assignment. Therefore, the set of variables obtained from this selection technique should be examined and just those variables that are needed for a partial-state assignment should be selected. An example of a partial-state assignment would be an assignment which only satisfies two columns of a four-column flow table.

The flow table shown in Figure 15 will now be considered.

|   | $I_1$ | $I_2$ | $I_3$ |
|---|---|---|---|
| 1 | ① | 2 | ① |
| 2 | ② | ② | 3 |
| 3 | 4 | ③ | ③ |
| 4 | ④ | 5 | 1 |
| 5 | 2 | ⑤ | 3 |

Figure 15. Flow Table

The destination sets for this table are:

| $D_{I_1}$ | $D_{I_2}$ | $D_{I_3}$ |
|---|---|---|
| $\underline{1}$ | $1\ \underline{2}$ | $\underline{1}\ 4$ |
| $\underline{2}\ 5$ | $\underline{3}$ | $2\ \underline{3}\ 5$ |
| $3\ \underline{4}$ | $4\ \underline{5}$ | |

In examining these destination sets, it is seen that no y-variable can be selected which will induce identical partitions on the destination sets of more than one input column. Hence, criterion one of the selection technique given above cannot be satisfied. However, criterion two is satisfied by three state variables, in which the first two, $y_1$ and $y_2$, induce partitions on the destination sets under input $I_2$ and the third, $y_3$, induces a partition on the destination sets under $I_3$. These variables with their respective partitions are:

$$y_1 \rightarrow \{(1\ 2\ 3),(4\ 5)\}$$
$$y_2 \rightarrow \{(1\ 2\ 4\ 5),(3)\}$$
$$y_3 \rightarrow \{(2\ 3\ 5),(1\ 4)\}$$

Following the convention that the states in the first block of the partition are coded with a 0 binary value with respect to $y_i$ and in the second block a 1 binary value, notice that for each partition $y_i = 0$ for all stable states of at least one input column, thus satisfying the condition of criterion two. Also, these three state variables do form a partial-state assignment since they separate the destination sets under input columns $I_2$ and $I_3$.

Once the initial partial-state assignment has been determined the next-state equations for these variables are then derived. To expedite the derivation of these equations, essentially the same theory that was used in Burton and Noaks' method is used in Tan's method. As previously noted in Tan's method, each y-variable takes on the value 1 for all states in the second block (1-block) of its respective partition, while in Burton and Noaks' method each variable takes on a value 1 for the states in its associated destination set. (The associated destination set is actually a 1-block of its corresponding induced y-partition.) Even though in Tan's method a 1-block of a y-partition may contain more than one destination set, the derivation of the next-state equations is performed in the same fashion, i.e., with the 1-blocks of the respective partitions being treated in the same manner as the associated destination sets in Burton and Noaks' method.

In Burton and Noaks' method, the equations for $Y_i$ consisted of product terms representing covers for those destination sets having their stable states contained in the associated $y_i$ destination set. Since each destination set was initially associated with a unique y-variable, all product terms could be immediately written in the equation. However, this is not the case in Tan's method, since a partial assignment does not separate all destination sets in the flow table. Therefore, there may exist destination sets having stable states which are contained in the 1-blocks of the initial y-variable partitions (some 1-destination sets), but have not yet been separated by at least one y-variable partition from the other destination sets (0-destination sets), in their corresponding input column. (Note: Insuring the separation of the 1-destination sets from the 0-destination sets is the same as covering the relevant dichotomies of destination sets or transition pairs as discussed in Tan's basic theory.) It should now be clear that if some of the covers for the 1-destination sets with respect to $y_i$ are not initially known, then some of the product terms in the equation for $Y_i$ cannot be immediately written. To determine those destination sets which still lack covers and to provide assistance in the selection of additional y-variables, a symbolic representation of the next-state equations can be given by:

$$Y_i \rightarrow [\ a\ b\ \cdots\ q] = \cdots + I_n[d\ e\ \cdots\ r] + \cdots + I_n[f\ g\ \cdots\ s]^* \qquad (2)$$

where $y_i = 1$ for states $[a\ b\ \cdots\ q]$; states $[d\ e\ \cdots\ r]$ are those contained in 1-destination sets with respect to $y_i$ under some input $I_n$, and are covered by at least one of the y-variables of the initial set; $[f\ g\ \cdots\ s]^*$ are those states contained in 1-destination sets under input $I_n$, but are not covered by any y-variable that has already been chosen. Equations of this form will be referred to as state transition equations. The state transition equations for the initial assignment found above will be:

$$Y_1 \rightarrow [4\ 5] = I_1[3\ 4]^* + I_2[4\ 5]$$
$$Y_2 \rightarrow [3] = I_2[3] + I_3[2\ 3\ 5]$$
$$Y_3 \rightarrow [1\ 4] = I_1[1,\ 3\ 4]^* + I_3[1\ 4]$$

Examining the above equations it is seen that the terms $I_1[3\ 4]^*$ and $I_1[1,\ 3\ 4]^*$ are not covered by any one of the variables of the initial assignment. The next step is to assign new y-variables to those destination sets that are contained in the starred product terms of the state transition equations. In the example here, assigning $y_4$ to $I_1[3\ 4]^*$ and $y_5$ to $I_1[1,\ 3\ 4]^*$ will yield the following partitions:

$$y_4 \rightarrow \{(1\ 2\ 5),(3\ 4)\}$$
$$y_5 \rightarrow \{(2\ 5),(1\ 3\ 4)\}$$

and

$$Y_4 \rightarrow [3\ 4] = I_1[3\ 4] + I_2[3] + I_3[2\ 3\ 5]$$
$$Y_5 \rightarrow [1\ 3\ 4] = I_1[i,\ 3\ 4] + I_2[3] + I_3$$

Since no more starred products were introduced in the state transition equations for $Y_4$ and $Y_5$, it would appear that our assignment is complete. To be sure, the assignment should be checked to see if the Liu conditions of separating all destination sets in an input column and distinguishing each state uniquely are satisfied. In this example these conditions are satisfied and therefore the set $\{y_1, y_2, y_3, y_4, y_5\}$ forms a satisfactory USTT Liu type assignment [7].

Converting the state-transition equations into next-state equations can be done in a relatively easy and straightforward manner. The destination sets contained in the product terms of the state-transition equations will be replaced by the y-variables (either $y_i$ or $\overline{y}_i$) of the assignment, whose partitions have a block equal to these destination sets. Following this rule, the next-state equations can be written as:

$$Y_1 = I_1y_4 + I_2y_1$$
$$Y_2 = I_2y_2 + I_3\overline{y}_3$$
$$Y_3 = I_1y_5 + I_3y_3$$

$$Y_4 = I_1y_4 + I_2y_2 + I_3\bar{y}_3$$

$$Y_5 = I_1y_5 + I_2y_2 + I_3$$

Although the final assignment obtained may not be a minimal assignment, the resulting next-state equations are minimal and consist of only simple product terms.

Tan [8] also showed that some of the simple product terms can be replaced by compound product terms if the overall effect was to reduce the total number of gate inputs required to realize the next-state equations. For example, the term $I_1y_5$ in $Y_3$ could be replaced by $I_1(y_2+y_3)$ since the sum of the 1-blocks of the $y_2$ and $y_3$ partitions equals (1 3 4), the 1-block of the $y_5$ partition. However, it can be shown that this replacement would not result in a lower number of total gate inputs and therefore would not be made (even though the replacement would eliminate the need for $y_5$ in the assignment).

Reduced cost in terms of the number of gate inputs required to logically realize the next-state equations is a point that Tan [8] pursued very meticulously and rigorously, by integrating a cost analysis with his derivation of the next-state equations. With today's technology in circuit packaging, especially in the areas of integrated circuitry and module construction, it is debatable whether the detailed cost study is worth increasing the complexity of his method.

This completes the description of Tan's specific procedure. A summary of the steps of this method is:

1) Using the criteria of the selection technique, select a set of state variables for the initial partial-state assignment.

2) Derive the state-transition equations for the variables in the initial assignment.

3) From the equations derived in step 2, generate new variables for the assignment.

4) After a sufficient number of state variables have been generated for a valid USTT Liu type assignment, form the next-state equations for these variables.

As an assistance in comparing this method with the other methods discussed, a final example will be given for the same flow table used in the previous methods.

Given a machine represented by the following flow table:

| | | $I_1$ | $I_2$ | $I_3$ |
|---|---|---|---|---|
| | a | $\textcircled{a}$ | b | $\textcircled{a}$ |
| Internal | b | c | $\textcircled{b}$ | $\textcircled{b}$ |
| States | c | $\textcircled{c}$ | b | a |
| | d | a | $\textcircled{d}$ | e |
| | e | $\textcircled{e}$ | d | $\textcircled{e}$ |

The destination sets for this flow table are:

| $D_{I_1}$ | $D_{I_2}$ | $D_{I_3}$ |
|---|---|---|
| $\underline{a}$ d | a $\underline{b}$ c | $\underline{a}$ c |
| b $\underline{c}$ | $\underline{d}$ e | $\underline{b}$ |
| $\underline{e}$ | | d $\underline{e}$ |

By criterion one of the initial assignment selection technique:

$$y_1 \rightarrow \{(a\ b\ c),(d\ e)\}$$

By criterion 2:

$$y_2 \rightarrow \{(a\ b\ c\ d),(e)\}$$

$$y_3 \rightarrow \{(a\ c\ d\ e),(b)\}$$

Since variables $y_1$ and $y_3$ will separate the destination sets under inputs $I_2$ and $I_3$, let the initial partial-state assignment consist of $(y_1, y_3)$.

Next, the state-transition equations will be constructed for the variables in the partial-state assignment.

$$Y_1 \rightarrow [d\ e] = I_1[e]^* + I_2[d\ e] + I_3[d\ e]$$

$$Y_3 \rightarrow [b] = I_2[a\ b\ c] + I_3[b]$$

Now select a y-variable to cover the term $I_1[e]^*$. Note that this term will be covered by the partition $\{(a\ b\ c\ d),(e)\}$ which is equal to the $y_2$ partition. Therefore, add $y_2$ to the assignment and derive its state-transition equation.

$$Y_2 \rightarrow [e] = I_1[e] + I_3[d\ e]$$

Since no more starred products have occurred, check to see if the conditions for a valid Liu type assignment are satisfied. In checking these conditions, it is found that destination sets {a d} and {b c} under $I_1$ have not yet been separated. Therefore assign:

$$y_4 \rightarrow \{(a\ d\ e),(b\ c)\}$$

and

$$Y_4 \rightarrow [b\ c] = I_1[b\ c] + I_2[a\ b\ c] + I_3[b]$$

Again, no further starred products have been generated and the conditions for a valid USTT assignment are now satisfied. Therefore, the assignment consists of $\{y_1, y_2, y_3, y_4\}$ and the corresponding next-state equations generated from the state-transition equations are:

$$Y_1 = y_2I_1 + y_1I_2 + y_1I_3$$

$$Y_2 = y_2I_1 + y_1I_3$$

$$Y_3 = \bar{y}_1I_2 + y_3I_3$$

$$Y_4 = y_4I_1 + \bar{y}_1I_2 + y_3I_3$$

Although the results will vary for each flow table, the above example points out some interesting highlights of Tan's method. The resulting assignment has three less variables than the assignment obtained using Burton and Noaks' method. Since both methods use the destination set approach, the difference in their results is primarily attributed to the use of complemented variables in Tan's method while none are permitted in Burton and Noaks' method. Also note that even though the number of y-variables in Tan's assignment is one greater than the number of variables in the minimal assignment used in the conventional approach, Tan's next-state equations require less gate inputs to implement than do the equations of the conventional approach.

Depending on the specifications of the design, Tan's specific procedure may or may not be desirable. However, it is conceivable that his basic theory could be used to develop other specific iterative procedures or it could be used on conjunction with other USTT assignment methods, which may better meet the specifications of the design.

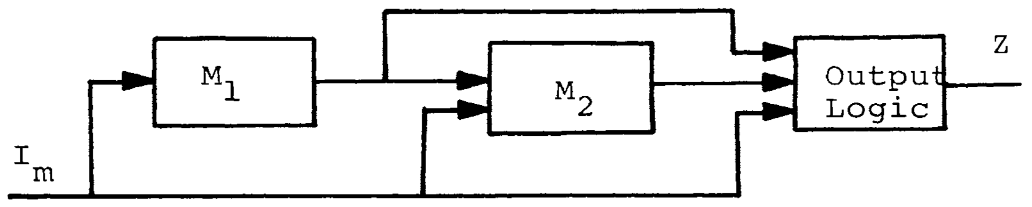
For example, Tan, Menon and Friedman [9,2] jointly developed a method for generating a USTT assignment and next-state equations by parallel and serial decompositions of asynchronous sequential circuits. The method used for generating the next-state equations is the same as that discussed in Tan's basic theory. Therefore, only a brief description of the decomposition method will be given here, since the main goal of this paper is to present the different known methods of generating next-state equations and not necessarily that of finding the state assignment itself.

Definition: Two partitions $\pi$ and $\pi'$ on the set of states of a sequential machine M are a <u>partition pair</u> denoted by $P(\pi, \pi')$ if for all states in the same block of $\pi$, their next-states to which the machine goes, when any input $I_m$ is applied, are in the same block of $\pi'$.

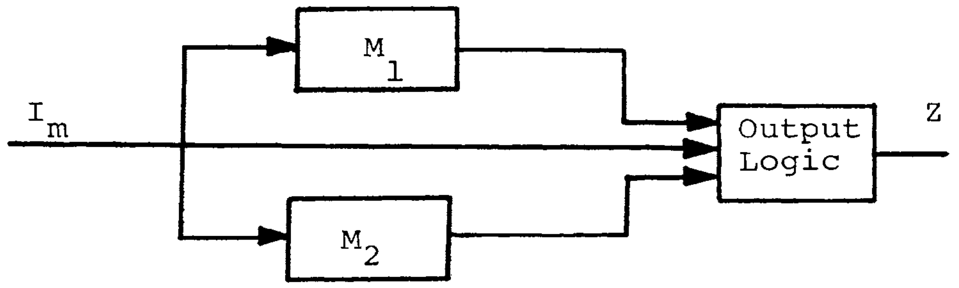
Definition: A partition $\pi$ on the states of a sequential machine M is called a preserved partition if for all states in the same block of $\pi$, their next-states to which the machine goes, when any input $I_m$ is applied, are also in a common block of $\pi$.

For a machine having the set of internal states $\{a, b, c, d\}$, the partitions, $\pi(0) = \{(a), (b), (c), (d)\}$ and $\pi(1) = \{(a,b,c,d)\}$ always form preserved partitions and are referred to as the trivial cases. Also from the above definitions it should be apparent that a preserved partition will form a partition pair with itself, i.e., $P(\pi, \pi)$.

Each component machine of a decomposition is defined by a partition pair $P(\pi, \pi')$, where the blocks of $\pi'$ correspond to the states of the submachine and the blocks of $\pi$ correspond to the internal information required to calculate the next-state of the submachine. In order to obtain a decomposition resulting in a USTT assignment for the component machines and therefore for the composite machine, it is necessary that at least one non-trivial preserved partition exists for the given flow table. This preserved partition will form a partition pair with itself which will define the first component machine of a serial decomposition. The model of a serial decomposition is shown in Figure 16(a) below.



(a)



(b)

Figure 16. (a) Serial Decomposition (b) Parallel Decomposition

The product of the partitions induced by the y-variables of the assignment for submachine $M_1$ will equal the preserved partition. Since $M_1$ feeds $M_2$, the preserved partition of $M_1$ is an input partition to $M_2$. Therefore, the partition pair that defines $M_2$ will be a combination of the preserved partition of $M_1$ and the y-variable partitions that are needed to cover the dichotomies of transition pairs of the given flow table that were not covered by the assignment of $M_1$. These y-variables will form the state assignment for $M_2$.

If two or more non-trivial preserved partitions exist for the given flow table such that their product equals $\pi(0)$, then these partitions will define submachines of a parallel decomposition as shown in Figure 16(b). The output logic recognizes the overall internal state of the composite machine as a function of the internal states of submachines $M_1$ and $M_2$.

For flow tables that do not have any preserved partitions, methods have been developed for decompositions leading to multicode STT assignments and multiple transition time assignments, but these methods will not be discussed here [9].

The procedure terminates when all component machines have a valid assignment which implies that the product of all y-variable partitions equals $\pi(0)$. It has been shown that the assignments and next-state equations of the component machines will combine to yield a satisfactory non-critical race assignment and next-state equations for the composite or given machine [9,2].

To help clarify the above description, an example will be given for the following flow table:

| | $I_1$ | $I_2$ | $I_3$ | $I_4$ |
|---|---|---|---|---|
| 1 | $\textcircled{1}$ | $\textcircled{1}$ | 4 | 2 |
| 2 | $\textcircled{2}$ | 3 | $\textcircled{2}$ | $\textcircled{2}$ |
| 3 | $\textcircled{3}$ | $\textcircled{3}$ | 4 | - |
| 4 | $\textcircled{4}$ | - | $\textcircled{4}$ | 5 |
| 5 | $\textcircled{5}$ | 1 | - | $\textcircled{5}$ |

Two preserved partitions for this flow table are:

$$\pi_1 = \{(13),(245)\}$$

$$\pi_2 = \{(1),(2),(34),(5)\}$$

and

$$\pi_1 \cdot \pi_2 = \{(1),(2),(3),(4),(5)\} = \pi(0)$$

Therefore, the preserved partitions $\pi_1$ and $\pi_2$ will form partition pairs $P(\pi_1,\pi_1)$ and $P(\pi_2,\pi_2)$ which will define component machines $M_1$ and $M_2$ of a parallel decomposition.

The flow tables representing the submachines $M_1$ and $M_2$ are:

Submachine $M_1$

| Blocks of $\pi_1$ | | Internal States | $I_1$ | $I_2$ | $I_3$ | $I_4$ |
|---|---|---|---|---|---|---|
| (13) | - | a | $\textcircled{a}$ | $\textcircled{a}$ | b | b |
| (245) | - | b | $\textcircled{b}$ | a | $\textcircled{b}$ | $\textcircled{b}$ |

Submachine $M_2$

| Blocks of $\pi_2$ | | Internal States | $I_1$ | $I_2$ | $I_3$ | $I_4$ |
|---|---|---|---|---|---|---|
| (1) | - | A | Ⓐ | Ⓐ | C | B |
| (2) | - | B | Ⓑ | C | Ⓑ | Ⓑ |
| (34) | - | C | Ⓒ | Ⓒ | Ⓒ | D |
| (5) | - | D | Ⓓ | A | - | Ⓓ |

The next step is to derive a USTT assignment and corresponding next-state equations for each of the component flow tables. The most general approach which could be used to accomplish this task is to derive the state assignment for each component flow table using a known state assignment procedure (e.g., the Tracey method [6]), and then use Tan's basic theory to derive the corresponding next-state equations. However, in this case, Tan's specific iterative procedure was used to derive the following state assignment and next-state equations simultaneously for each component flow table.

For submachine $M_1$:

State Assignment

| $y_1$ | | Internal States | | |
|---|---|---|---|---|
| 0 | - | a | - | (13) |
| 1 | - | b | - | (245) |

Next-State Equations

$$Y_1 = y_1 I_1 + I_3 + I_4$$

And for submachine $M_2$:

State Assignment

| $y_2$ | $y_3$ | $y_4$ | Internal States |
|---|---|---|---|
| 0 | 1 | 1 | A - (1) |
| 1 | 1 | 0 | B - (2) |
| 0 | 0 | 0 | C - (34) |
| 0 | 0 | 1 | D - (5) |

Next-State Equations

$$Y_2 = y_2I_1 + y_2I_3 + y_3I_4$$
$$Y_3 = y_3I_1 + y_4I_2 + y_2I_3 + y_3I_4$$
$$Y_4 = y_4I_1 + y_4I_2 + \bar{y}_3I_3$$

Now the assignment and corresponding next-state equations for the composite machine is obtained by combining the results of the component machines. Therefore, the composite state assignment and corresponding next-state equations are:

Composite State Assignment

| $y_1$ | $y_2$ | $y_3$ | $y_4$ | | Internal States |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | - | 1 |
| 1 | 1 | 1 | 0 | - | 2 |
| 0 | 0 | 0 | 0 | - | 3 |
| 1 | 0 | 0 | 0 | - | 4 |
| 1 | 0 | 0 | 1 | - | 5 |

Next-State Equations

$$Y_1 = y_1I_1 + I_3 + I_4$$
$$Y_2 = y_2I_1 + y_2I_3 + y_3I_4$$
$$Y_3 = y_3I_1 + y_4I_2 + y_2I_3 + y_3I_4$$
$$Y_4 = y_4I_1 + y_4I_2 + \bar{y}_3I_3$$

This concludes the review of the presently known methods for generating next-state equations. The remainder of this paper will deal with a new method which was developed from a study of the foregoing methods.

# IV. NEW METHOD OF GENERATING NEXT-STATE EQUATIONS

Burton and Noaks' procedure and Tan's specific procedure both have been directed toward the generation of the next-state equations in parallel with the selection of a USTT assignment. These methods may be desirable for some designs since the Burton and Noaks method yields next-state equations with no complemented variables, while Tan's method may yield equations that require a low number of gate inputs. However, in order to obtain the next-state equations using these methods, their particular type of assignments (Liu type) must also be used for the design.

Since methods have already been developed which provide different types of USTT assignments for a flow table (e.g., the Tracey method for a minimal variable assignment), a simple method for generating minimal next-state equations that could be used for any USTT assignment would be extremely valuable, especially for large flow tables.

It has been shown that the Maki, Tracey and Smith method and the basic theory of Tan can be used to obtain next-state equations for any USTT assignment, but as flow tables become larger the amount of work required to use these methods becomes excessive and tedious. For example, in Maki, Tracey and Smith's method the Boolean expressions representing the specified states must be complemented to obtain the expressions for the unspecified states. These expressions in turn are combined with the expressions of the 1-destination set subspaces to obtain minimal next-state equations. It is obvious that much work and time (computer time, too, if programmed) is required to perform these manipulations and the necessary simplification to obtain minimal equations. Also, when using Tan's basic theory, all dichotomies of transition pairs must be listed and then an exhaustive search for those state variables that minimally cover the relevant dichotomies must be carried out. Only after this is completed can the heuristic rules be used to construct the next-state equations. So again it is apparent that for large flow tables an extreme amount of work and time is required, since there would be many, many dichotomies of transition pairs to examine. Therefore, the main intent here is to develop a simple and straightforward method which will require less work to generate minimal next-state equations for any given USTT assignment and will easily lend itself to computer application.

The heart of the following method of generating next-state equations emanates from the role that the Karnaugh map plays in separating the 1-destination sets from the 0-destination sets. It was believed that if the operations performed with a Karnaugh map could somehow be carried out implicitly in a simple and straightforward manner, a powerful means for generating the next-state equations would result. To show why such a method would be a powerful tool, the flow table used in the previous methods, repeated in Figure 17, will again be considered here.

| $y_1$ | $y_2$ | $y_3$ | | $I_1$ | $I_2$ | $I_3$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | a | $\textcircled{a}$ | b | $\textcircled{a}$ |
| 1 | 1 | 0 | b | c | $\textcircled{b}$ | $\textcircled{b}$ |
| 1 | 0 | 0 | c | $\textcircled{c}$ | b | a |
| 0 | 0 | 1 | d | a | $\textcircled{d}$ | e |
| 0 | 1 | 1 | e | $\textcircled{e}$ | d | $\textcircled{e}$ |

Figure 17. Flow Table

The destination sets for this table are:

| $\underline{I_1}$ | $\underline{I_2}$ | $\underline{I_3}$ |
|---|---|---|
| $\underline{a}$ d | a $\underline{b}$ c | $\underline{a}$ c |
| b $\underline{c}$ | $\underline{d}$ e | $\underline{b}$ |
| $\underline{e}$ | | d $\underline{e}$ |

The Karnaugh map of the USTT assignment for this table is shown in Figure 16.

| $y_3$ \ $y_1y_2$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | a | - | b | c |
| 1 | d | e | - | - |

Figure 18. Karnaugh Map

Now suppose that the next-state equation $Y_1$ is to be calculated. In order to successfully derive the minimal equation for $Y_1$, minimal y-variable covers must be found for each input column which separate the 1-destination sets subspaces from the 0-destination set subspaces. The 1-destination sets with respect to $y_1$ under each input are:

| $I_1$ | $I_2$ | $I_3$ |
|---|---|---|
| b $\underline{c}$ | a $\underline{b}$ c | $\underline{b}$ |

From the Karnaugh map it is seen that the state variable $y_1$ separates the subspace of the 1-destination set (b c) from the remaining 0-destination set subspaces under $I_1$. Therefore, the minimal cover for this separation can be written as the simple product term $y_1I_1$. It should be noted that the don't-care states are again being used implicitly here, since they are in the subcube covered by $y_1$. Similarly, the minimal cover for the subspace of (a b c) under $I_2$ is the simple product term $\bar{y}_3I_2$ and for the subspace of (b) under $I_3$ the minimal cover can be either $y_1y_2I_3$ or $y_2\bar{y}_3I_3$. Therefore, from a knowledge of the 1- and 0-destination sets and with the use of the Karnaugh map, the equation for $Y_1$ can be written by inspection as:

$$Y_1 = y_1I_1 + \bar{y}_3I_2 + y_1y_2I_3$$

The above equation is a minimal next-state equation and agrees with the results obtained in the conventional approach, the Maki, Smith and Tracey method and the approach used in the basic theory of Tan. The reason the above equation is in minimal form is because the y-variable covers were selected in such a manner to separate the largest possible subcubes containing the 1-destination sets (thus inherently making use of don't-care states) from the subcubes of the 0-destination sets.

It should now be apparent that, by using the Karnaugh map in the above manner, the derivation of the next-state equations can be accomplished in a simpler and faster manner than in the previous methods.

When working with the Karnaugh map in the above example, the operations were performed either by hand or were done mentally. The goal of the following method will be to describe these same operations in a definite language that would permit computer application for large flow tables.

Definition: A two block partition $\alpha_{i,j}$ is called the <u>$\alpha$-partition</u> with respect to $Y_i$ and input $I_j$ if the left block contains all the internal states for which $Y_i = 0$ under input $I_j$ and the right block contains all the internal states for which $Y_i = 1$ under input $I_j$.

This definition implies that the left-block of an $\alpha$-partition will contain the states in 0-destination sets while the right block will contain the states in 1-destination sets, all with respect to some $y_i$ and $I_j$.

For an example, the flow table in Figure 17 will again be considered. The 0- and 1-destination sets with respect to $Y_1$ and under input $I_j$ are:

| <u>0-destination Sets</u> | <u>1-destination Sets</u> |
|---|---|
| $\underline{a}$ d | b $\underline{c}$ |
| $\underline{e}$ | |

Therefore, $\alpha_{1,1}$ can be written as:

$$\alpha_{1,1} = \{(a\ d\ e),(b\ c)\}$$

Similarly,

$$\alpha_{1,2} = \{(d\ e),(a\ b\ c)\}$$

$$\alpha_{1,3} = \{(a\ c\ d\ e),(b)\}$$

Comparing with the conventional approach, it is seen that the above $\alpha$-partitions indicate which states will have either 0- or 1-entries in the Karnaugh maps that were derived from the transition table for $Y_1$ (see Figure 5, page 11). Therefore, the $\alpha$-partitions provide part of the information attainable from the Karnaugh maps. Of course, the remaining portion of information that is obtained with the Karnaugh map is the selection of a minimal y-variable cover that separates the 0-destination sets from the 1-destination sets.

Definition: A set A of states is said to cover a set B of states if $B \subseteq A$ (read as set B is contained in or equal to set A).

It should be kept in mind that since the word "cover" takes on several different meanings, any particular meaning must be derived from the context in which it is being used.

It is known from the conditions for a valid USTT assignment that all of the states in 1-destination sets are separated from those in 0-destination sets. These separations are provided by the state variables of the assignment as was shown in the first example by the Karnaugh map in Figure 18. Therefore, it follows that the covers of the $\alpha$-partitions can be obtained from the partitions induced by the y-variables of the assignment.

Definition: A two block partition, $\tau_i$, is called a <u>$\tau$-partition</u> with respect to $y_i$, if the left block contains all of the internal states for which $y_i = 0$ and the right block contains all the internal states for which $y_i = 1$.

The concept of this definition has already been used in the previous methods of Tan and Burton and Noaks. The definition implies that the states in the left block of a $\tau_i$ partition are covered by the y-variable expression $\overline{y}_i$ while the states in the right block are covered by the expression $y_i$. The $\tau$-partitions induced by the state variables in the assignment of the example are:

$$\tau_1 = \{\overset{\bar{y}_1}{(a\ d\ e)}, \overset{y_1}{(b\ c)}\}$$

$$\tau_2 = \{\overset{\bar{y}_2}{(a\ c\ d)}, \overset{y_2}{(b\ e)}\}$$

$$\tau_3 = \{\overset{\bar{y}_3}{(a\ b\ c)}, \overset{y_3}{(d\ e)}\}$$

The y-variable covers for all blocks have been explicitly shown.

The next step is to find which blocks of the $\tau$-partitions separate the blocks of the $\alpha$-partitions. This separation can be accomplished by selecting those blocks of the $\tau$-partitions that together cover the right block ($Y = 1$ block) of the $\alpha$-partition and do not contain any states from the left block of the $\alpha$-partition. (Those blocks satisfying the second condition are referred to as being <u>disjoint</u> from the left block of the $\alpha$-partition.) For example, to separate the blocks of $\alpha_{1,1} = \{(a\ d\ e), (b\ c)\}$ a cover for the block $(b\ c)$ is needed. Examining the $\tau$-partitions, it is observed that the right block of $\tau_1$ would be the most minimal cover because it equals $(b\ c)$ itself. Now, since the block $(b\ c)$ is covered by $y_1$ in the $\tau_1$ partition, it can be concluded that $y_1$ will separate the blocks of $\alpha_{1,1}$ and $Y_{1,1}$ can be written as:

$$Y_{1,1} = y_1 I_1$$

where $Y_{i,j}$ represents the next-state variable $Y_i$ under input $I_j$.

Here again the use of unspecified states in obtaining the y-variable cover is inherent in the $\tau$-partitions. For example, from the Karnaugh map of Figure 18 (page 78), it is seen that $\tau_1$ actually implies the partition

$$\tau_1 = \{\overset{\bar{y}_1}{(ade-)},\overset{y_1}{(bc--)}\},$$

where $y_1$ covers the subcube spanned by specified states b and c and two unspecified states. It is this inherent characteristic of $\tau$-partitions that enables a minimal y-variable next-state expression to be obtained.

Similarly for $\alpha_{1,2} = \{(d\ e),(a\ b\ c)\}$, block (a b c) is covered by the left block of $\tau_3$. Therefore,

$$Y_{1,2} = \bar{y}_3 I_2$$

Now for $\alpha_{1,3} = \{(a\ c\ d\ e),(b)\}$, there is no single block of the $\tau$-partitions that covers block (b) and does not contain any of the states in block (a d c e). In other words, under input $I_3$, there is no single y-variable that separates the 1-destination sets from the 0-destination sets with respect to $y_1$. Therefore, it is necessary to examine the product partitions of the form $\tau_i \cdot \tau_j$. These partitions will be the g.l.b. partition of $\tau_i$ and $\tau_j$. In our example, the product partitions of the form $\tau_i \cdot \tau_j$ are:

$$\tau_1 \cdot \tau_2 = \{\overset{\bar{y}_1\bar{y}_2}{(a\ d)}, \overset{\bar{y}_1 y_2}{(e)}, \overset{y_1\bar{y}_2}{(\ c\ )}, \overset{y_1 y_2}{(\ b\ )}\}$$

$$\tau_1 \cdot \tau_3 = \{\overset{\bar{y}_1\bar{y}_3}{(a)}, \overset{\bar{y}_1 y_3}{(d\ e)}, \overset{y_1\bar{y}_3}{(b\ c)}, \overset{y_1 y_3}{(\ 0\ )}\}$$

$$\tau_2 \cdot \tau_3 = \{\overset{\bar{y}_2\bar{y}_3}{(a\ c)}, \overset{\bar{y}_2 y_3}{(\ d\ )}, \overset{y_2\bar{y}_3}{(\ b\ )}, \overset{y_2 y_3}{(\ e\ )}\}$$

Again the appropriate y-variable cover is associated with each block. As an example, the first block of $\tau_1 \cdot \tau_2$ is obtained by intersecting the $\bar{y}_1$ block of $\tau_1$ with the $\bar{y}_2$ block of $\tau_2$, i.e.:

$$\overset{\bar{y}_1}{(a\ d\ e)} \cap \overset{\bar{y}_2}{(a\ c\ d)} = \overset{\bar{y}_1\bar{y}_2}{(a\ d)} \quad \text{etc.}$$

Now the blocks of the $\tau_i \cdot \tau_j$ partitions are examined to find a cover for block (b) of $\alpha_{1,3}$. It is seen that block (b) can be covered by either block $y_1y_2$ of $\tau_1 \cdot \tau_2$ or block $y_2\bar{y}_3$ of $\tau_2 \cdot \tau_3$. This is the same choice of covers that existed in the preceding example with the Karnaugh map. Choosing $y_1y_2$ as the cover, then:

$$Y_{1,3} = y_1y_2I_3$$

The expressions for $Y_{1,1}$, $Y_{1,2}$, and $Y_{1,3}$ can now be combined to give:

$$Y_1 = Y_{1,1} + Y_{1,2} + Y_{1,3}$$

$$Y_1 = y_1I_1 + \bar{y}_3I_2 + y_1y_2I_3$$

The result is identical to the result obtained with the Karnaugh map and is therefore the minimal next-state equation for $Y_1$. Similarly for $Y_2$ and $Y_3$:

$\alpha_{2,1} = \{(a\ b\ c\ d),(e)\}$ $\qquad$ $\alpha_{3,1} = \{(a\ b\ c\ d),(e)\}$

$\alpha_{2,2} = \{(d\ e),(a\ b\ c)\}$ $\qquad$ $\alpha_{3,2} = \{(a\ b\ c),(d\ e)\}$

$\alpha_{2,3} = \{(a\ c),(b\ d\ e)\}$ $\qquad$ $\alpha_{3,3} = \{(a\ b\ c),(d\ e)\}$

From the $\tau_i$ and $\tau_i \cdot \tau_j$ partitions derived above:

$Y_{2,1} = \bar{y}_1 y_2 I_1$ $\qquad$ $Y_{3,1} = \bar{y}_1 y_2 I_1$

$Y_{2,2} = \bar{y}_3 I_2$ $\qquad$ $Y_{3,2} = y_3 I_2$

$Y_{2,3} = (y_2 + y_3) I_3$ $\qquad$ $Y_{3,3} = y_3 I_3$

Now,

$$Y_i = \sum_{j=1}^{m} Y_{i,j} \tag{3}$$

Therefore,

$$Y_2 = \bar{y}_1 y_2 I_1 + \bar{y}_3 I_2 + (y_2 + y_3) I_3$$

$$Y_3 = \bar{y}_1 y_2 I_1 + y_3 I_2 + y_3 I_3$$

In some cases it is necessary to consider a union of blocks from the $\tau$-partitions in order to find a satisfactory cover for the right block of an $\alpha$-partition. For example, in finding $Y_{2,3}$ it was necessary to take the union of blocks $y_2$ of $\tau_2$ and $y_3$ of $\tau_3$ to cover block (b d e) of $\alpha_{2,3}$.

In order to have minimal next-state equations it is necessary that the covers which are selected for the blocks of the $\alpha$-partitions are the minimal covers for those blocks. For instance, a union of two blocks from some $\tau$-partitions would not be a minimal cover and hence would not be used, if a cover exists which consists of only one block from some other $\tau$-partition. The minimal cover corresponds to the largest possible circling of "1" and "don't-care" entries in the Karnaugh map. This can be seen by comparing the minimal covers found for $Y_1$ in the above example with the circlings in the Karnaugh maps that were used in the conventional approach (Figure 5).

Rules for selecting the minimal covers in a systematic and orderly fashion can be established. One strategy which could be formalized into such a set of rules is:

1) Check the blocks of the $\alpha$-partition. If its left block or its right block equals the empty set, then the product term corresponding to this partition in the equation for $Y_{i,j}$ will just be $I_j$ or 0 respectively.
2) If neither block of the $\alpha$-partition equals the empty set, then select all blocks from the $\tau$-partitions, that are disjoint from the left block of the $\alpha$-partition. Check these blocks to see if any single block will cover the right block of the $\alpha$-partition. If one of these blocks does cover the right block of the $\alpha$-partition, then the equation for $Y_{i,j}$ can be written as a simple product term. If no single

block exists, check the union of the previously selected disjoint blocks taken two at a time. If still no cover exists, check the union of these blocks taken three at a time, etc., until a cover is found.

3) If none of the blocks of the $\tau$-partitions are disjoint from the left block of the $\alpha$-partition, or if only a partial cover can be obtained with step 2, then form the $\tau_i \cdot \tau_j$ partitions and follow the rules of step 2 until a complete cover is found.

4) If still no complete cover has been obtained, repeat the process with the product partitions $\tau_i \cdot \tau_j \cdot \tau_k$, etc., until a complete cover is found.

Before proceeding with a final example, a summary of the steps of this method is:

1) List all of the destination sets for each input column of the flow table.

2) From the destination sets under each input, form the $\alpha$-partitions with respect to $y_i$ by putting the 0-destination sets in the left block and the 1-destination sets in the right block.

3) Form the $\tau$-partitions that are induced by the state variables of the given assignment.

4) From the blocks of the τ-partitions, find a cover for the right block of the α-partition by following the strategy for the selection of a minimum cover.

5) The next-state equations are then derived from the y-variables representing the selected covers.

The final example will be for a larger flow table in order to show the capability and proficiency of this method.

Suppose that an asynchronous sequential circuit is described by the following flow table having the USTT state assignment as shown.

| $y_1$ | $y_2$ | $y_3$ | $y_4$ | | | $I_1$ | $I_2$ | $I_3$ | $I_4$ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | - | a | d | $\textcircled{a}$ | e | b |
| 0 | 1 | 0 | 1 | - | b | f | $\textcircled{b}$ | h | $\textcircled{b}$ |
| 0 | 1 | 1 | 0 | - | c | l | $\textcircled{c}$ | e | b |
| 1 | 0 | 0 | 0 | - | d | $\textcircled{d}$ | $\textcircled{d}$ | g | b |
| 0 | 0 | 1 | 0 | - | e | i | a | $\textcircled{e}$ | b |
| 1 | 1 | 0 | 1 | - | f | $\textcircled{f}$ | g | $\textcircled{f}$ | b |
| 1 | 0 | 0 | 1 | - | g | d | $\textcircled{g}$ | $\textcircled{g}$ | b |
| 0 | 1 | 1 | 1 | - | h | f | - | $\textcircled{h}$ | b |
| 1 | 0 | 1 | 0 | - | i | $\textcircled{i}$ | j | g | b |
| 1 | 0 | 1 | 1 | - | j | k | $\textcircled{j}$ | g | b |
| 0 | 0 | 1 | 1 | - | k | $\textcircled{k}$ | a | h | b |
| 1 | 1 | 1 | 0 | - | l | $\textcircled{l}$ | c | $\textcircled{l}$ | b |

The destination sets for this flow table are:

| $D_{I_1}$ | $D_{I_2}$ | $D_{I_3}$ | $D_{I_4}$ |
|---|---|---|---|
| a $\underline{d}$ g | $\underline{a}$ e k | a c $\underline{e}$ | a $\underline{b}$ c d e f g h i j k l |
| b $\underline{f}$ h | $\underline{b}$ | $\underline{f}$ | |
| e $\underline{i}$ | $\underline{c}$ l | d $\underline{g}$ i j | |
| j $\underline{k}$ | $\underline{d}$ | b $\underline{h}$ k | |
| c $\underline{l}$ | f $\underline{g}$ | $\underline{l}$ | |
| | i $\underline{j}$ | | |

The $\alpha$-partitions with respect to each $Y_i$ and $I_j$ are:

$\alpha_{1,1} = \{(jk),(abcdefghil)\}$ $\quad \alpha_{2,1} = \{(adegijk),(bcfhl)\}$

$\alpha_{1,2} = \{(abcekl),(dfgij)\}$ $\quad \alpha_{2,2} = \{(adefgijk),(bcl)\}$

$\alpha_{1,3} = \{(abcehk),(dfgijl)\}$ $\quad \alpha_{2,3} = \{(acdegij),(bfhkl)\}$

$\alpha_{1,4} = \{(abcdefghijkl),(\emptyset)\}$ $\quad \alpha_{2,4} = \{(\emptyset),(abcdefghijkl)\}$

$\alpha_{3,1} = \{(abdfgh),(ceijkl)\}$ $\quad \alpha_{4,1} = \{(acdegil),(bfhjk)\}$

$\alpha_{3,2} = \{(abdefgk),(cijl)\}$ $\quad \alpha_{4,2} = \{(acdekl),(bfgij)\}$

$\alpha_{3,3} = \{(dfgij),(abcehkl)\}$ $\quad \alpha_{4,3} = \{(acel),(bdfghijk)\}$

$\alpha_{3,4} = \{(abcdefghijkl),(\emptyset)\}$ $\quad \alpha_{4,4} = \{(\emptyset),(abcdefghijkl)\}$

The $\tau$-partitions induced by the state variables of the assignment are:

$$\tau_1 = \{\overset{\bar{y}_1}{(abcehk)},\overset{y_1}{(dfgijl)}\}$$

$$\tau_2 = \{\overset{\bar{y}_2}{(adegijk)},\overset{y_2}{(bcfhl)}\}$$

$$\tau_3 = \{\overset{\bar{y}_3}{(abdfg)},\overset{y_3}{(cehijkl)}\}$$

$$\tau_4 = \{\overset{\bar{y}_4}{(acdeil)},\overset{y_4}{(bfghjk)}\}$$

From the $\tau$-partitions, the $\tau_i \cdot \tau_j$ partitions will now be formed to have them available if needed.

$$\tau_1 \cdot \tau_2 = \{\overset{\bar{y}_1\bar{y}_2}{(a\ e\ k)},\overset{\bar{y}_1 y_2}{(b\ c\ h)},\overset{y_1\bar{y}_2}{(d\ g\ i\ j)},\overset{y_1 y_2}{(f\ l)}\}$$

$$\tau_1 \cdot \tau_3 = \{\overset{\bar{y}_1\bar{y}_3}{(a\ b)},\overset{\bar{y}_1 y_3}{(c\ e\ h\ k)},\overset{y_1\bar{y}_3}{(d\ f\ g)},\overset{y_1 y_3}{(i\ j\ l)}\}$$

$$\tau_1 \cdot \tau_4 = \{\overset{\bar{y}_1\bar{y}_4}{(a\ c\ e)},\overset{\bar{y}_1 y_4}{(b\ h\ k)},\overset{y_1\bar{y}_4}{(d\ i\ l)},\overset{y_1 y_4}{(f\ g\ j)}\}$$

$$\tau_2 \cdot \tau_3 = \{\overset{\bar{y}_2\bar{y}_3}{(a\ d\ g)},\overset{\bar{y}_2 y_3}{(e\ i\ j\ k)},\overset{y_2\bar{y}_3}{(b\ f)},\overset{y_2 y_3}{(c\ h\ l)}\}$$

$$\tau_2 \cdot \tau_4 = \{\overset{\bar{y}_2\bar{y}_4}{(a\ d\ e\ i)},\overset{\bar{y}_2 y_4}{(g\ j\ k)},\overset{y_2\bar{y}_4}{(c\ l)},\overset{y_2 y_4}{(b\ f\ h)}\}$$

$$\tau_3 \cdot \tau_4 = \{\overset{\bar{y}_3\bar{y}_4}{(a\ d)},\overset{\bar{y}_3 y_4}{(b\ f\ g)},\overset{y_3\bar{y}_4}{(c\ e\ i\ l)},\overset{y_3 y_4}{(h\ j\ k)}\}$$

The next step is to find a minimum cover for the right block of the $\alpha$-partitions. For an example, take the right block of $\alpha_{1,1}$, (abcdefghi). Therefore, find a minimal cover for (abcdefghi) from the block of the $\tau$-partitions such that the cover is disjoint from (jk), the left block of $\alpha_{1,1}$.

Cover of (abcdefghi) $= \overset{y_2}{(bcfhl)} \cup \overset{\bar{y}_3}{(abdfg)} \cup \overset{\bar{y}_4}{(acdeil)}$

Therefore,

$$Y_{1,1} = (y_2 + \bar{y}_3 + \bar{y}_4)I_1$$

Similarly:

$$Y_{1,2} = (y_1\bar{y}_2 + y_1\bar{y}_3)I_2 \qquad Y_{2,1} = y_2 I_1$$

$$Y_{1,3} = y_1 I_3 \qquad Y_{2,2} = (\bar{y}_1 y_2 + y_2 y_3)I_2$$

$$Y_{1,4} = 0 \qquad Y_{2,3} = (y_1 y_2 + \bar{y}_1 y_4)I_3$$

$$Y_{2,4} = I_4$$

$$Y_{3,1} = (\bar{y}_2 y_3 + y_2\bar{y}_4)I_1 \qquad Y_{4,1} = (y_2 y_4 + y_3 y_4)I_1$$

$$Y_{3,2} = (y_1 y_3 + y_2\bar{y}_4)I_2 \qquad Y_{4,2} = (\bar{y}_3 y_4 + y_1\bar{y}_2 y_3)I_2$$

$$Y_{3,3} = (\bar{y}_1 + y_2 y_3)I_3 \qquad Y_{4,3} = (y_4 + y_1\bar{y}_2)I_3$$

$$Y_{3,4} = 0 \qquad Y_{4,4} = I_4$$

Note that in order to find $Y_{4,2}$ it was necessary to form $\tau_1 \cdot \tau_2 \cdot \tau_3$. That is:

$$\tau_1 \cdot \tau_2 \cdot \tau_3 = \{\overset{\bar{y}_1\bar{y}_2\bar{y}_3}{(\ a\ )}, \overset{\bar{y}_1 y_2\bar{y}_3}{(\ b\ )}, \overset{y_1\bar{y}_2\bar{y}_3}{(\ d\ g\ )}, \overset{y_1 y_2\bar{y}_3}{(\ f\ )}, \overset{\bar{y}_1\bar{y}_2 y_3}{(\ e\ k\ )}, \overset{\bar{y}_1 y_2 y_3}{(\ h\ )}, \overset{y_1\bar{y}_2 y_3}{(\ i\ j\ )}, \overset{y_1 y_2 y_3}{(\ l\ )}\}$$

Now the cover of:

$$(bfgij) = \overset{\bar{y}_3 y_4}{(bfg)} \cup \overset{y_1\bar{y}_2 y_3}{(ij)} \quad .$$

The expressions for the covers associated with the $\alpha$-partitions have now been found. The minimal next-state equations can be derived by simply combining these expressions.

$Y_i = Y_{i,1} + Y_{i,2} + Y_{i,3} + Y_{i,4}$

Therefore,

$Y_1 = (y_2 + \bar{y}_3 + \bar{y}_4)I_1 + (y_1\bar{y}_2 + y_1\bar{y}_3)I_2 + y_1I_3$

$Y_2 = y_2I_1 + (\bar{y}_1y_2 + y_2y_3)I_2 + (y_1y_2 + \bar{y}_1y_4)I_3 + I_4$

$Y_3 = (\bar{y}_2y_3 + y_2\bar{y}_4)I_1 + (y_1y_3 + y_2\bar{y}_4)I_2 + (\bar{y}_1 + y_2y_3)I_3$

$Y_4 = (y_2y_4 + y_3y_4)I_1 + (\bar{y}_3y_4 + y_1\bar{y}_2y_3)I_2 + (y_4 + y_1\bar{y}_2)I_3 + I_4$

From the above example the value of this method with respect to large flow tables can be readily seen. Therefore, the method could play a key role in future synthesis applications. It should be made clear that this method is a general method for only generating next-state equations and does not generate the state assignment too.

# V. CONCLUSION

Now that each of the methods for generating the next-state equations for normal-mode asynchronous sequential circuits has been described, a meaningful discussion and comparison of their strengths and weaknesses can be made.

First of all, one major difference that should be remembered is that the state assignment must be derived independently and prior to the use of the Maki, Tracey and Smith method and the new method developed in this paper. On the other hand, Burton and Noaks' method and Tan's method are used to derive both the state assignment and next-state equations in a parallel fashion. (Note: The basic theory of Tan's method can also be used for just deriving the next-state equations for a given state assignment.) As a result of this difference the method of Burton and Noaks and the specific procedure of Tan's method are somewhat limited as general methods for generating next-state equations. In order to obtain the next-state equations using these methods, their particular type of state assignments must also be used. Therefore, these methods could not be utilized in cases where a design specified the use of a state assignment that differed from their particular assignments. However, the Maki, Tracey and Smith method, Tan's basic theory and the new method developed in this paper are general methods which will work for any type of uni-code single transition time (USTT) assignment that is given.

Another interesting comparison is the use of the unspecified states of a flow table as a tool for obtaining minimal next-state equations. The Maki, Tracey and Smith method is the only method that handles the unspecified states in an explicit fashion. The other methods implicitly use the unspecified states. This difference results because, in the Maki, Tracey and Smith method, the y-variable expression that covers a transition pair subspace is selected in such a manner that it covers the smallest subcube of a Karnaugh map that contains that transition pair. Then other y-variable expressions need to be derived to cover the subcubes of the unspecified states. In the other methods, however, the y-variable expressions which cover particular transition pairs (or destination sets) are selected in a manner such that they cover the largest possible subcube of a Karnaugh map that contains the transition pairs (or destination sets) along with any available unspecified states. Therefore, the unspecified states are inherent in these covers.

It is the explicit handling of the unspecified states that leads to the following disadvantages of the Maki, Tracey and Smith method:

1) The amount of time and work required to find the expressions for the unspecified states by taking the logical complement.

2) The amount of time and work required to simplify the expressions to a set of minimal equations.

A recommendation for partially alleviating the above disadvantages without changing the entire strategy can be given. Part of the time and work required above is a result of using transition pairs as the basis of their method. For assignments derived from a transition pair basis, it is necessary that their method also uses the transition pair basis in the derivation of the next-state equations. However, if the given state assignment is known to be one based upon destination sets, then some work and time would be saved if the method could be readily converted to use destination sets as its basis rather than transition pairs. A savings would result because the unsimplified Boolean expressions produced by destination sets would have fewer terms than the equivalent expressions produced by transition pairs. Thus, the manipulations required in the above disadvantage would become less unwieldy.

The best method with respect to the mechanics required in the actual construction of the next-state equations is Burton and Noaks' method. The shortcuts used in their derivation result from the way the initial state assignment is made, i.e., the association of a unique y-variable with each destination set. This state assignment also leads to the unique property of having no complemented variables in the next-state equations, which may be a desirable feature for some designs.

It is recalled that Tan also used a strategy similar to Burton and Noaks' for construction of the next-state equations in his specific iterative procedure. The 1-blocks of the y-variable partitions in Tan's procedure are treated in the same manner as the associated y-variable destination sets in Burton and Noaks' method. However, Tan's resulting next-state equations are not complement free because both the 0-blocks and 1-blocks of the y-variable partitions are used as possible covers in the next-state equation.

One characteristic of Burton and Noaks' method which may be a disadvantage is that a larger assignment and, therefore, more state variables, have to be contended with in the design. This evaluation can be verified by noting that for the same flow table used in the example of each method, the resulting Burton and Noaks' assignment exceeded the assignments of all other methods by at least three state variables. Of course, the effectiveness of this method will vary from circuit to circuit. And there may be cases where a minimal variable assignment could be obtained with this method, but generally the method will realize a larger assignment. This can be considered as the price paid for complement free next-state equations.

Another disadvantage of Burton and Noaks' method is the informal procedure of simplifying the initial assignment using a dependency diagram. A more formal procedure capable of being programmed on a computer would certainly be welcomed.

From his basic theory, Tan developed a specific iterative procedure which results in a Liu type assignment. Although his assignment may not be a minimum variable assignment, it often results in minimal next-state equations requiring less gate inputs to implement. This is due to the discreet manner in which the state variables are selected for the assignment.

It has also been shown that Tan's basic theory can be used strictly as a next-state equation generation method for any given USTT assignment. However, as in the case of the Maki, Tracey and Smith method, it too requires a considerable amount of work and time for larger flow tables. This is because, in using Tan's basic theory, all dichotomies of transition pairs must be listed and then an exhaustive search for those state variables that minimally cover the relevant dichotomies must be carried out. Even after the covers for the relevant dichotomies have been determined, the derivation of the next-state equations requires another rather complicated step. In this step, the relevant dichotomies have to be further examined to determine whether the uncomplement or complement form of the y-variable covers should be used and whether the product terms will be simple or compound in the final next-state expression. As flow tables become larger, a greater number of dichotomies have to be examined and the amount of work and time required becomes excessive.

The new method developed in this paper seems to provide a better means of coping with larger flow tables. In this method all destination sets (and therefore transition pairs) are grouped into one two-block partition per input column, i.e., the $\alpha$-partition. Therefore, no matter how large the flow table is, in the derivation of a next-state equation it is only necessary to cover one $\alpha$-partition per input column; whereas, in Tan's basic theory, many dichotomies per input column have to be examined and covered. It is this reason that enables the new method to handle larger flow tables more efficiently than the other methods. Since many practical circuits are very large, it is conceivable that this method could become a valuable tool in the future in the synthesis of asynchronous sequential circuits. Also, another attribute of this method is its applicability to be programmed for computer use.

Since the new method is a systematic repetitive procedure, it, too, has the disadvantage of becoming somewhat lengthy for larger flow tables when a large number of $\tau$-partitions and products of $\tau$-partitions have to be considered.

This completes a comparison of the foregoing methods. In deciding which method should be used, the strengths and weaknesses of each method mentioned above would have to be weighed for the particular problem at hand.

## VI. BIBLIOGRAPHY

1. E. J. McCluskey, Introduction to the Theory of Switching Circuits. New York: McGraw-Hill, Inc., 1965.

2. S. H. Unger, Asynchronous Sequential Switching Circuits. New York: John Wiley and Sons., Inc., 1969.

3. G. K. Maki, J. H. Tracey, and R. J. Smith, "Generation of design equations in asynchronous sequential circuits," IEEE Trans. on Computers, vol. C-18, pp. 467-472, May 1969.

4. R. J. Smith, J. H. Tracey, W. L. Schoeffel, and G. K. Maki, "Automation in the design of asynchronous sequential circuits," 1968 Spring Joint Computer Conf., AFIPS Proc., Vol. 32, pp. 55-60, 1968.

5. D. P. Burton and D. R. Noaks, "Complement free STT state assignments for asynchronous sequential machines," Proceedings of the 2nd National Symposium on Logic Design, University of Reading, Sponsored by British Computer Society, March 1969.

6. J. H. Tracey, "Internal state assignments for asynchronous sequential machines," IEEE Trans. on Electronic Computers, vol. EC-15, pp. 551-560, August 1966.

7. C. N. Liu, "A state variable assignment method for asynchronous sequential switching circuits," J. ACM, vol. 10, pp. 209-216, April 1963.

8. C. J. Tan, "Synthesis of asynchronous sequential switching circuits," Dr. Eng. Sc. Dissertation, Columbia University, June 1969.

9. C. J. Tan, P. R. Menon, and A. D. Friedman, "Structural simplification and decomposition of asynchronous sequential circuits," IEEE Trans. on Computers, vol. C-18, pp. 830-838, September 1969.

## VII. VITA

Gregory Martin Bednar was born on June 26, 1944, in St. Louis, Missouri. He was graduated valedictorian of his class from Cuba High School, Cuba, Missouri. He received a Bachelor of Science degree in Electrical Engineering from the University of Missouri - Rolla in August, 1966. He was actively employed by IBM at Rochester, Minnesota, from August 1966 to September 1967, after which he began two years active duty as an officer in the United States Army. One year was spent in Vietnam where he was awarded the bronze star and the bronze star with oak leaf cluster. Presently, he is enrolled in the Graduate School of the University of Missouri - Rolla in Electrical Engineering.

The author is a member of IEEE, Eta Kappa Nu, Tau Beta Pi, and Phi Kappa Phi.