1967

Minimization and generation of next-state expressions for asynchronous sequential circuits

Gary Keith Maki

MINIMIZATION AND GENERATION OF NEXT-STATE EXPRESSIONS

FOR

ASYNCHRONOUS SEQUENTIAL CIRCUITS

BY

GARY K. MAKI -1943-

A

THESIS

submitted to the faculty of

THE UNIVERSITY OF MISSOURI AT ROLLA

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

Rolla, Missouri

1967

Approved by

James H. Tracey (advisor) Frank J. Kern

Robert W. Chenoweth S. J. Pagano

ABSTRACT

One step in the synthesis procedure for realizing an asynchronous sequential circuit that is operating in fundamental mode is to obtain an internal-state assignment that will realize the operations of the circuit. Often the procedures that are used in accomplishing the above task generate several satisfactory assignments. The first part of this paper presents a method that will enable one to predict which of the internal-state assignments will yield a simpler set of next-state expressions.

A second topic treated in this paper is one of presenting a method to generate the next-state expressions for an asynchronous sequential circuit directly from the internal-state assignment. An algorithm is presented for generating the next-state expressions without construction of the transition table.

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to Dr. James H. Tracey for his guidance throughout the entire project. The prompt and careful reading of the manuscript is greatly appreciated.

The author also wishes to acknowledge the typing efforts of his wife, Alice, and the help extended by Mrs. Mary Colton in preparing this thesis.

TABLE OF CONTENTS

I. INTRODUCTION

Sequential switching circuits are normally categorized as being either synchronous or asynchronous. In synchronous circuits, clock pulses synchronize the operations of the circuit, while in asynchronous circuits, it is usually assumed that no such clocking is available. A desirable feature of asynchronous design is that the resulting circuit may take full advantage of basic device speed since the circuit does not have to wait for the arrival of clock pulses before effecting a transition. This paper deals with only the asynchronous sequential circuits.

The operation of an asynchronous sequential switching circuit can be described by means of a flow table[1]. An example appears in Figure 1.

	I_1	I_2	I_3
1	1/1	2	4
2	3	2/1	2/1
3	3/2	4	2
4	1	4/2	4/1

Figure 1. Flow table for an asynchronous sequential circuit.

Each column of the flow table represents an input state, each row represents an internal state, and the table entries specify the next internal state and output state. If the next internal state entry is equal to the present internal state, the state is said to be stable and is denoted as such by a circled entry. Uncircled entries are called unstable. For example, in Figure 1, if the circuit is presently stable in internal state 2 with input I_2, the output is in output state 1. A change in input from I_2 to I_1 will cause the circuit to enter unstable state 3. This will be followed by a change in internal state from state 2 to state 3 with a new output state 2. The circuit is now stable and the internal state will undergo no further changes until there is another change in input.

Definition 1: An asynchronous sequential circuit is said to be operating in the fundamental mode if the inputs are never changed unless the circuit is in a stable condition.

Definition 2: A flow table with the characteristic that each unstable state leads directly to a stable state is called a normal flow table.

The sequential circuits in this paper will be considered to be normal fundamental-mode asynchronous sequential circuits.

The problem of selecting an internal-state assignment for the realization of a given normal flow table for an asynchronous sequential circuit that will result in critical-race-free operation is an important step in the design of such circuits. A critical race is a condition that exists when there is a possibility that unequal transmission delays may cause the sequential circuit to reach a stable state other than the one intended.

Huffman recognized the existence of the internal-state assignment problem in his original paper[1] which has been the basis for much of the work in asynchronous sequential circuit design and analysis. A year later, Huffman presented a generalized assignment procedure for coding a 2^n-row normal flow table with a maximum of 2n-1 internal-state variables[2]. This assignment procedure would produce codes with no races and therefore free of critical races.

Liu[3] has developed systematic procedures for constructing noncritical assignments for normal fundamental-mode sequential circuits. These procedures, which are dependent on flow table structure, have been extended by Tracey[4]. One of the three procedures developed by Tracey yields a minimum-variable assignment for normal fundamental-mode flow tables. Often, the algorithms developed by Tracey produce several internal-state assignments with the same number of internal-state variables, all of which produce critical-race-free realizations of a given flow table. It has been observed that the next-state expressions which result from some of these internal-state assignments are simpler than others. One purpose of this paper is to present a method to predict which internal-state assignment for an asynchronous sequential machine will yield simpler next-state expressions than other assignments for the same machine.

The second topic treated in this paper is to find a direct method for obtaining the next-state expressions for an asynchronous sequential machine from a given critical-race-free internal-state assignment. An algorithm is presented in this paper for generating the next-state expressions without the construction of a transition table.

II. SELECTION OF AN INTERNAL-STATE ASSIGNMENT WHICH WILL YIELD SIMPLER NEXT-STATE EXPRESSIONS

The discussion presented throughout this paper will not be concerned with the coding of the input states and will be restricted to finding the next-state expressions on a per-column basis. If there are n internal-state variables and a flow table of m columns and m input states, the general form for the next-state expressions will be

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

where $y_1,y_2,\ .\ .\ .\ ,y_n$ are the present state variables; $Y_1,Y_2,\ .\ .\ .\ ,Y_n$ are the next-state variables; $I_1,I_2,\ .\ .\ .\ ,I_m$ are the input states; $\delta_{11},\delta_{12},\ .\ .\ .\ ,\delta_{nm}$ are functions of the internal-state variables alone.

The intent of this section is to obtain a figure of merit that will predict which internal-state assignment for a normal fundamental-mode asynchronous sequential machine will yield simpler next-state expressions than other assignments for the same machine. In this paper the assignment-selection process is considered to be the selection of that assignment which will minimize the functions $\delta_{11}, \delta_{12}, \ldots \delta_{ij}, \ldots, \delta_{nm}$ into a simplest sum-of-products expression. The assignment which tends to minimize the complete set of functions will be the one that will be said to yield the simplest next-state expressions. It is realized that the coding of the input states will affect the complexity of the next-state expressions, but there is some positive value in choosing the assignment with the simplest δ_{ij} coefficients of Eq.(1), even though it cannot be guaranteed to result in a minimal set of equations. Optimum coding of the input states will not be part of the study in this paper.

The determination of a figure of merit that will be used to judge each internal-state assignment will be based on the following characteristics:

1) The number of internal-state variables which have the characteristic that the next state is equal to the present state for all transitions in a particular column of a flow table.
2) The number of internal-state variables that remain constant for all transitions in a particular column of a flow table.

An internal-state variable y_j which has the characteristic that the next state is equal to the present state in a column of a flow table under input state I_i will have as part of the next-state expression for the next-state variable Y_j

$$Y_j = y_j I_i, \tag{2}$$

or $\delta_{ij} = y_j$ as one of the coefficients in Eq.(1). Another way of stating this is that the internal-state variable y_j will not change state in any transition in the column of the flow table with input state I_i.

An internal-state variable y_j which remains constant for all transitions in a particular column with input I_i will have as part of the next-state expression for the next-state variable Y_j

$$Y_j = 1(0) I_i, \tag{3}$$

or $\delta_{ij} = 1(0)$ as one of the coefficients in Eq.(1). Another way of stating this is that if a transition table was formed, the next-state variable Y_j would have a next-state entry of 1 or 0 for all the specified internal states in the column with input I_i.

The discussion following will describe exactly how the characteristic of the next state being equal to the present state can be determined. It is advantageous to use partition theory as it has been employed to describe certain aspects in switching theory.

Definition 3: A partition Π on a set S is a collection subsets of S such that their pairwise intersection is the null set. The disjoint subsets are called the blocks of Π.

If the set union of these subsets is S, the partition is completely specified; otherwise, the partition is incompletely specified. Elements of S that do not appear in Π are called unspecified or optional elements with respect to that partition.

Definition 4: The two-block partition $\tau_1, \tau_2, \ldots, \tau_n$ induced by the internal-state variables $y_1, y_2, \ldots, y_n$, respectively, are called the set of τ-partitions of that assignment.

The following example in Figure 2 will help illustrate the above definition. Here the first block in each τ-partition is a set of the internal states that have been coded with a 0 by each internal-state variable and the second block is a similar set of the states that have been coded with a 1 by each internal-state variable. It should be pointed out that the ordering of the blocks is unimportant.

Internal states	Internal-state variables y_1	y_2	y_3
a	0	0	0
b	0	1	0
c	1	1	0
d	1	0	0
e	0	1	1
f	1	0	1

$$\tau_1 = \{\overline{a,b,e};\ \overline{c,d,f}\}$$

$$\tau_2 = \{\overline{a,d,f};\ \overline{b,c,e}\}$$

$$\tau_3 = \{\overline{a,b,c,d};\ \overline{e,f}\}$$

Figure 2. Internal-state assignment and corresponding τ-partitions.

Definition 5: A *k-set* of a single column of a flow table consists of all k-1 unstable entries leading to the same stable state, together with that stable state.

Definition 6: A column partition α_i is a collection of the k-sets of the column of a flow table with input state I_i, where each k-set is contained in a single block.

	I_i
a	1
b	2
c	1
d	2
e	2
f	3
g	3
h	4
j	5

Figure 3. Partial flow table and corresponding column partition $\alpha_i = \{\overline{a,c};\ \overline{b,d,e};\ \overline{f,g};\ \overline{h};\ \overline{j}\}$.

Definition 7: Partition θ_2 is less than or equal to θ_1 ($\theta_2 \leq \theta_1$) where θ_1 and θ_2 may be incompletely specified, if and only if all elements specified in θ_2 are also specified in θ_1 and each block of θ_2 appears in a unique block of θ_1.

Theorem 1: An internal-state variable y_j will not change state in any transition in a column of a flow table with input state I_i if $\alpha_i \leq \tau_j$.

Proof: From the definition of a k-set, all transitions of the column of a flow table with input state I_i will take place within some k-set of the column partition α_i. If the column partition α_i is less than or equal to a τ-partition τ_j, each block of the column partition α_i, which is a k-set of the column partition, is included in one of the blocks of the partition τ_j. Therefore, the internal-state variable y_j cannot undergo a change of state in any transition in the column with input I_i, because each pair of states that have a transition between them are listed in the same block of the partition τ_j and are coded idenically with internal-state variable y_j.∇

To demonstrate the use of theorem 1, the τ-partitions

$$\tau_1 = \{\overline{a,c,f,g,h};\ \overline{b,d,e,j}\}$$

$$\tau_2 = \{\overline{a,b,d,e};\ \overline{c,f,g,h,j}\}$$

will be compared to the column partition of Figure 2

$$\alpha_i = \{\overline{a,c};\ \overline{b,d,e};\ \overline{f,g};\ \overline{h};\ \overline{j}\}.$$

Each block of α_i is contained in a block of τ_1 or $(\alpha_i \leq \tau_1)$. This means that in all the transitions of this column, internal-state variable y_1 will not change state, or the next state will always be equal to the present state for any transition in the column with input I_i. However, $\tau_2$does not satisfy theorem 1 in that the block $\overline{a,c}$ of α_i does not appear in a block of τ_2. The internal-state variable y_2 will therefore, undergo a change of state during the transition state a to state c.

It might be noted that in making the test for $\alpha_i \leq \tau_j$ for all i and j, one only has to be able to show that each block of the column partition α_i is contained in a block of the τ-partition τ_j. Extending this even further, only the blocks of α_i which contain two or more elements need be considered.

Let $(\#y^i)$ be the number of internal-state variables that meet the conditions of theorem 1 under each column with input I_i. The total number of terms like that of Eq.(2), which will appear in the next-state expressions of Eq.(1), will be

$$D_k = \sum_{i=1}^{m} (\#y^i), \tag{4}$$

where m is the number of columns in the flow table and D_k is the total number of internal-state variables that do not change state for internal-state assignment k. Following is an algorithm that can be used to obtain D_k for internal-state assignment k:

1) Form the partitions α_i and τ_j for all values of i and j.
2) Determine $(\#y^1)$, which is the number of internal-state variables under input I_1 that meet the conditions of theorem 1. Repeat for $i = 2,3,\ldots,m$.
3) D_k for internal-state assignment k is given by Eq.(4).

At this point, attention will be given to the internal-state variables which remain constant for all transitions in a particular column of a flow table, or can be considered as constant next-state terms. In determining the constant next-state terms in a particular column of a flow table, consideration will be given to the actual next-state entry for each internal state in a column of the flow table as it would appear in a transition table.

It has been extablished that all the transitions in a column of a flow table take place within some k-set k_r of that column. Each of the unstable states of the k-set k_r can experience a transition to the stable state of k_r. The next-state entry for each of the unstable states of k_r will have to be the same as the code assigned to the stable state within k_r in order to insure a transition from the unstable states to the stable state will be independent of transmission delays. In other words, the next-state entry for each of the unstable states of a k-set k_r will be determined completely by the code assigned to the stable state of k_r. If the internal-state variable y_j in the code assigned to the stable state of k_r in a certain column of a flow table is 1(0), then all the internal states of k_r will have a next-state entry of 1(0) for the next-state variable Y_j.

Extending the above argument even further and assuming there are $(\#k^i)$ k-sets in a particular column of a flow table with input I_i, the next-state variable Y_j will have a next-state entry of 1(0) in all the specified states of the column with input I_i if the internal-state variable y_j is 1(0) in the code assigned to all $(\#k^i)$ stable states of the same column. The resulting next-state expression would be

$$Y_j = 1(0)I_i.$$

The following discussion will explain a method that can be used to obtain the constant terms in each column of a flow table. First list the stable states and their respective codes for each column of the flow table and identify each set of lists with the respective input state of the column. From these lists, one can determine which, if any, of the internal-state variables are 1(0) in the codes for the stable states in each of the respective columns. To illustrate this, assume the internal states with their respective codes shown in Figure 4 are the stable states in a column of a flow table.

	y_1	y_2	y_3	y_4	y_5
a	1	0	1	0	1
c	1	1	0	0	1
d	1	0	1	0	0
f	1	1	1	0	0

Figure 4. Stable states and their respective codes.

By inspection, one can determine that there are two internal-state variables that are 1(0) in the states listed; they are $y_1 = 1$ and $y_4 = 0$.

Let ψ_i be the number of internal-state variables that are 1(0) in the codes for the stable states in the column with input I_i. In Figure 4, $\psi_i = 2$. Let C_k be the total number of internal-state variables found in the columns to be 1(0) for internal-state assignment k;

$$C_k = \sum_{i=1}^{m} \psi_i, \tag{5}$$

where m is the number of columns in the flow table.

Both of the characteristics discussed for determing a weight to attach to each internal-state assignment for a asynchronous sequential machine are valuable. The relative weight to attach to D_k and C_k for internal-state assignment k may vary with each type of implementation. The weight for an internal-state assignment k will be defined as

$$W_k = D_k + \xi C_k, \tag{6}$$

where W_k will be the weight attached to internal-state assignment k and ξ is a variable that would allow the adjust- of the relative values of C_k in respect to D_k. It seems safe to conclude that a constant coefficient, as represented in Eq.(3), would require a lesser amount of combinational logic for synthesis than a literal coefficient, as represented in Eq.(2). The designer of a sequential circuit will have to decide on a value for ξ in determining how much easier it is to implement a constant coefficient as opposed to a literal coefficient. This could be done by obtaining a cost figure to compare the coefficients. This cost figure would depend on the number of literals associated with each input state.

In general, ξ will vary with each type of implementation. For purposes of illustration in this paper, ξ will take on a value of 2, which is arbitrary, to demonstrate the assignment-selection procedure. The weight for an internal-state assignment k will be defined in the examples shown in this paper as

$$W_k = D_k + 2C_k \tag{7}$$

The internal-state assignment with the largest weight associated with it will be predicted to yield the simplest next-state expressions, because it would have the largest number of terms like those shown in Eq.(2) and Eq.(3).

Sequential machine A in Figure 5 can be coded with either of the two internal-state assignments shown. The criteria developed above will be used to predict which assignment will produce the simplest next-state expressions.

	I_1	I_2	I_3
a	A	F	D
b	C	D	B
c	C	C	B
d	A	D	D
e	F	E	E
f	F	F	E

Assignment 1				Assignment 2		
y_1	y_2	y_3		y_1	y_2	y_3
0	0	0	a	0	0	0
1	0	1	b	1	0	1
1	0	0	c	1	1	1
0	0	1	d	0	0	1
0	1	1	e	1	1	0
0	1	0	f	0	1	0

Figure 5. Machine A with two assignments.

First D_k will be obtained by following the procedure developed in this paper.

Step 1. The column partitions are

$$\alpha_1 = \{\overline{a,d};\ \overline{b,c};\ \overline{e,f}\}$$

$$\alpha_2 = \{\overline{a,f};\ \overline{b,d};\ \overline{c};\ \overline{e}\}$$

$$\alpha_3 = \{\overline{a,d};\ \overline{b,c};\ \overline{e,f}\}.$$

The τ-partitions for assignment 1 are

$$\tau_1 = \{\overline{a,d,e,f};\ \overline{b,c}\}$$

$$\tau_2 = \{\overline{a,b,c,d};\ \overline{e,f}\}$$

$$\tau_3 = \{\overline{a,c,f};\ \overline{b,d,e}\}$$

and for assignment 2 are

$$\tau_1 = \{\overline{a,d,f};\ \overline{b,c,e}\}$$

$$\tau_2 = \{\overline{a,b,d};\ \overline{c,e,f}\}$$

$$\tau_3 = \{\overline{a,e,f};\ \overline{b,c,d}\}.$$

Step 2. The τ-partitions for assignment 1 that meet the conditions of theorem 1 are

$$\alpha_1 \leq \tau_1$$

$$\alpha_1 \leq \tau_2$$

$$\alpha_2 \leq \tau_3$$

$$\alpha_3 \leq \tau_1$$

$$\alpha_3 \leq \tau_2,$$

and for assignment 2

$$\alpha_2 \leq \tau_3.$$

Step 3. D_1 and D_2 are

$$D_1 = 2 + 1 + 2 = 5$$

$$D_2 = 0 + 1 + 0 = 1.$$

C_k is obtained by following the procedure given earlier. The stable states and their respective codes in assignment 1 under each input state are

I_1		I_2		I_3	
a	0 0 0	c	1 0 0	b	1 0 1
c	1 0 0	d	0 0 1	d	0 0 1
f	0 1 0	e	0 1 1	e	0 1 1
		f	0 1 0		

and in assignment 2 are

I_1		I_2		I_3	
a	0 0 0	c	1 1 1	b	1 0 1
c	1 1 1	d	0 0 1	d	0 0 1
f	0 1 0	e	1 1 0	e	1 1 0
		f	0 1 0		

Internal-state variable y_3 is constant in the above lists in the columns with input I_1 and I_3 for assignment 1. There are no internal-state variables constant in the above lists for assignment 2. C_1 and C_2 are

$$C_1 = 2$$

$$C_2 = 0.$$

The weight for each assignment is

$$W_1 = 5 + 4 = 9,$$

and $$W_2 = 1 + 0 = 1.$$

From the above information, assignment 1 would be predicted to have the simpler next-state expressions.

The next-state expressions for assignment 1 are

$$
\begin{aligned}
Y_1 &= y_1I_1 + y_1y_3'I_2 + y_1I_3\\
Y_2 &= y_2I_1 + (y_2 + y_1'y_3')I_2 + y_2I_3\\
Y_3 &= y_3I_2 + I_3,
\end{aligned}
$$

and for assignment 2 are

$$
\begin{aligned}
Y_1 &= y_1y_3I_1 + y_1y_2I_2 + (y_1 + y_2)I_3\\
Y_2 &= (y_1 + y_2)I_1 + (y_2 + y_3')I_2 + y_2y_3'I_3\\
Y_3 &= y_1y_3I_1 + y_3I_2 + (y_3 + y_2')I_3.
\end{aligned}
$$

Clearly, assignment 1 yields the simpler next-state expressions.

Presented in this section was a method to predict which internal-state assignment, from several such assignments for the same asynchronous sequential machine, will yield the simplest next-state expressions.

III. GENERATION OF NEXT-STATE EXPRESSIONS

The problem treated in this section is one of obtaining the next-state expressions for a normal fundamental-mode asynchronous sequential machine directly from a given critical-race-free internal-state assignment. An algorithm is presented for generating the next-state expressions without the construction of a transition table. The next-state expressions are generated in the form shown in Eq.(1). The input states are shown uncoded, but would be coded and simplified before realization of the flow table is attempted. Each of the functions δ_{ij} in Eq.(1) will consist of a sum-of-products expression representing the 1-cells and a similar expression representing the don't-care states for a next-state variable Y_j in the column of a flow table with input state I_j. In general, these expressions are not minimal, but the minimal expressions can certainly be obtained from these equations with conventional simplification algorithms.

Consider the example shown in Figure 6 to be part of a flow table with the corresponding internal-state assignment.

y_1	y_2	y_3		I_i
0	0	0	a	1
0	0	1	b	1
0	1	1	c	2
1	0	1	d	3
1	1	0	e	2

Figure 6. State assignment and partial flow table.

In this example, there are transitions between states *a* and *b* and states *c* and *e*. One will note that a race condition exists at state *c*. Because of unequal transmission delays, any of the internal states -1-, where the dashes represent all combination of 1's and 0's, could momentarily be assumed during the transition between states *c* and *e*. Internal states *c*, 011, and *e*, 110, must have a next-state entry of 110; but to insure that the circuit reaches the proper terminal state, internal states 010 and 111 must also have the next-state entry of 110. If these states had any other next-state entry, improper operation could result. A complete transition table, if formed, would have to show states 010 and 111 with the proper next-state entry of 110.

This set of states that have the same next-state entry can be represented as a *p*-subcube of an *n*-cube. The *n*-cube would represent all possible internal states in a particular column of a flow table, and the *p*-subcube would be a subset of the states represented by the *n*-cube. Each subcube can be represented, in turn, by a product function of the internal-state variables. In the above example, the p-subcube would be -1-, or as a product of internal-state variables, it is y_2. In this example then, all internal states where $y_2 = 1$ will have the same next-state entry as stable state e; including internal state e, there are 4 such states. For the transition b to a, the subcube that represents all the states which will have the same next-state entry as stable state a is 00- or $y_1'y_2'$. All internal states where both y_1 and y_2 are zero will have a next-state entry of 000, which is the code induced by the internal-state variables for stable state a.

It has been established that all the transitions in a column of a flow table take place within some k-set k_h of that column. Each of the unstable states of k_h can experience a transition to the stable state of k_h. The next-state entry for each of the unstable states of k_h will have to be the same as the code assigned to the stable state within k_h in order to insure a transition from the unstable states to the stable state will be independent of transmission delays. In some cases a single p-subcube can represent all the internal states of a k-set that must have the same next-state entry, like in the example above with only two internal states per k-set in the original flow table. In general, it may take several p-subcubes to represent the internal states of a k-set with more than two internal states. The following example shown in Figure 7 will illustrate this point.

y_1	y_2	y_3		I_i
0	0	0	a	A
1	1	0	b	D
0	1	1	c	A
1	1	1	d	D
1	0	1	e	A

Figure 7. State assignment and partial flow table.

Internal states *a*, *c*, and *e* are in the same *k*-set and will have the same next-state entry of 000; internal states *b* and *d* are in the same *k*-set and will have the next-state entry of 111. During the transitions between states *c* and *a*, any of the internal states 0-- could momentarily be assumed and must have the next-state entry of 000 to insure proper operation of the circuit. These states can be represented by the *p*-subcube 0--, or as a product of internal-state variables y_1'. During the transition from *e* to *a*, any of the states -0- could momentarily appear and must have the next-state entry of 000. This *p*-subcube can be represented as -0- or as y_2'. In this example, a sum-of-products expression that can represent all the internal states that must have the next-state entry of 000 is

$$y_1' + y_2'.$$

The *p*-subcube that would represent the internal states that must have a next-state entry of 111 is 11- or $y_1 y_2$.

A p-subcube represents all the internal states that could momentarily appear due to unequal transmission delays during the transition between an unstable and stable state within a k-set. The internal states represented by a p-subcube must have the same next-state entry. A p-subcube may represent even internal states which may not be assigned to the rows of the original flow table. These spare states may be entered during a transition between internal states when unequal transmission delays cause internal states other than those assigned to the rows of the original flow table to be assumed. However, it will not be necessary to identify the spare states individually because they will be represented in a p-subcube.

Each transition between an unstable and stable state of a k-set k_h has a unique p-subcube p_h which represents all the internal states that could be assumed. None of the internal states represented in p_h can be represented in another p-subcube. If the situation did occur where two subcubes had an internal state in common, this internal state would be required to have two different next-state entries. Such an assignment does not constitute a satisfactory code for flow tables operating in normal fundamental-mode.

In the previous example it can be seen that for k-sets of three or more internal states, it will require more than one p-subcube to represent all the internal states that must have the same next-state entry in a column of a flow table.

Definition 8: K_r is defined to be a sum-of-products expression representing all the internal states of k-set k_r which have the same next-state entry, namely that of the stable state of k_r.

A method to represent the p-subcube as a product of the internal-state variables which represents all possible internal states that could appear during a transition between an unstable and a stable state in a k-set is as follows:

1) List the codes assigned to the stable and unstable states involved in the transition within k-set k_r.
2) The product expression that will represent the p-subcube will be a subset of the internal-state variables (y_h . . . y_i . . . y_k). If the internal-state variable y_j is a 1 in both of the states in question of k_r, it will appear uncomplemented in the product expression. If the internal-state variable y_j appears as a 0 in both of the states in question of k_r, its complement will appear in the product expression. If the internal-state variable y_j appears as both 1's and 0's in the states in question of k_r, it is considered a don't-care variable and does not appear in the product expression.

Consider the following single column of a flow table with the codes listed for each internal state:

y_1	y_2	y_3	y_4	y_5		I_i
1	0	0	0	1	a	1
1	0	1	1	1	b	1
0	0	1	0	1	c	2
1	0	0	1	1	d	1
1	0	1	0	1	e	1
0	1	0	1	0	f	2

Figure 8. Partial flow table and corresponding state assignment.

Internal states a, b, d and e are in the same k-set k_0. There are transitions a to b, d to b and e to b in k_0. The p-subcube representing the states of the transition a to b is obtained as follows: Internal-state variable y_1 and y_5 are 1 in both states a and b; therefore, y_1 and y_5 will appear in the product expression. Internal-state variable y_2 is 0 in both states; therefore, y_2' will appear in the product expression. Internal-state variables y_3 and y_4 appear as 1's and 0's in both states, so neither will appear in the product expression. The product expression for this p-subcube, which represent the internal states that may appear during the transition from state a to state b, is $y_1y_2'y_5$. The p-subcube which will represent the states that may appear during the transition from state d to state b is obtained in the same manner just described and is $y_1y_2'y_4y_5$. Obtained in the same manner, the p-subcube that represents the states for the transition from e to b is $y_1y_2'y_3y_5$. K_0 which represents all the internal states of k_0 that have the same next-state entry 10111 is

$$y_1y_2'y_5 + y_1y_2'y_4y_5 + y_1y_2'y_3y_5.$$

A tabular method to obtain the same subcubes is shown for the internal states of k_0 from Figure 8 as follows:

b	1 0 1 1 1	b	1 0 1 1 1	b	1 0 1 1 1
a	1 0 0 0 1	d	1 0 0 1 1	e	1 0 1 0 1
	1 0 - - 1		1 0 - 1 1		1 0 1 - 1

Record the value of the internal-state variable in those columns where it is the same; where there is a difference in the internal-state variable, place a don't-care (-). The sum-of-products expression can be obtained directly from above and K_0 for k_0 is

$$y_1y_2'y_5 + y_1y_2'y_4y_5 + y_1y_2'y_3y_5.$$

As stated before, all the unstable internal states represented by K_h must have the same next-state entry, namely that of the code for the stable state of the corresponding k-set, k_h. It follows that if the internal-state variable y_j in the code for the stable state of k-set k_h is 1(0), then all the internal states represented by K_h will have a next-state entry of 1(0) for the next-state variable Y_j.

Definition 9: An internal state which has a specified next-state entry in a particular column of a flow table will be called a <u>specified internal state</u> of that column.

The remaining internal states are said to be unspecified for that column.

Each K_r represents a unique set of internal states and each of these internal states is specified. The total number of specified internal states S_i in a column of a flow table is equal to the sum of the number of the internal states represented by each K_r of that column. If there are k internal-state variables in the internal-state assignment, the number of internal states in a column of a flow table that are not specified is $2^k - S_i$, where 2^k is the total number of internal states possible with k internal-state variables.

Each K_r which represents all the internal states that have the same next-state entry in the k-set k_r, represents a unique set of internal states that are specified in a particular column of a flow table. K_r can be expressed as a sum-of-products expression in terms of the internal-state variables. It follows that a sum of the product expressions representing all the K_r's of a particular column of a flow table would be an expression to logically represent all the specified internal states of that column. The unspecified entries would be simply the logical complement of the above expression obtained from the K_r's. Consider the flow table with m input states and the corresponding internal-state assignment in Figure 9:

y_1	y_2	y_3	y_4		I_1	I_2	. . .	I_m
0	0	0	0	a	B	C	. . .	A
1	0	1	0	b	B	B	. . .	-
0	0	1	0	c	B	C	. . .	J
0	0	0	1	d	D	C	. . .	D
1	0	1	1	e	D	B	. . .	-
1	1	0	0	f	G	F	. . .	A
1	1	1	0	g	G	F	. . .	J
0	1	0	0	h	H	F	. . .	-
0	1	1	0	j	H	-	. . .	J

Figure 9. Sequential machine B.

The k-sets under input state I_1 are

k_1 = abc, k_2 = de, k_3 = fg, and k_4 = hj.

The k-sets under input state I_2 are

k_5 = acd, k_6 = be, and k_7 = fgh.

The k-sets under input state I_m are

k_8 = af, k_9 = cgj, and k_{10} = d.

The p-subcube for each transition pair under input I_1

in k_1 a to b is -0-0 or $y_2'y_4'$

c to b is -010 or $y_2'y_3y_4'$

in k_2 e to d is -0-1 or $y_2'y_4$

in k_3 f to g is 11-0 or $y_1y_2y_4'$

in k_4 j to h is 01-0 or $y_1'y_2y_4'$

The p-subcube for each transition pair under input I_2

in k_5 a to c is 00-0 or $y_1'y_2'y_4'$

d to c is 00-- or $y_1'y_2'$

in k_6 e to b is 101- or $y_1 y_2' y_3$

in k_7 g to f is 11-0 or $y_1 y_2 y_4'$

h to f is -100 or $y_2 y_3' y_4'$

The p-subcube for each transition pair under input I_m

in k_8 f to a is --00 or $y_3' y_4'$

in k_9 g to j is -110 or $y_2 y_3 y_4'$

c to j is 0-10 or $y_1' y_3 y_4'$

in k_{10} d to d is 0001 or $y_1' y_2' y_3' y_4$

Each K_r representing the k-set k_r under the respective input state is given as follows:

Under input I_1

$$K_1 = y_2' y_4' + y_2' y_3 y_4',$$
$$K_2 = y_2' y_4,$$
$$K_3 = y_1 y_2 y_4',$$
$$K_4 = y_1' y_2 y_4'.$$

Under input I_2

$$K_5 = y_1' y_2' y_4' + y_1' y_2',$$
$$K_6 = y_1 y_2' y_3,$$
$$K_7 = y_1 y_2 y_4' + y_2 y_3' y_4'.$$

Under input I_m

$$K_8 = y_3' y_4',$$
$$K_9 = y_2 y_3 y_4' + y_1' y_3 y_4',$$
$$K_{10} = y_1' y_2' y_3' y_4.$$

The sum of the product expressions of the K_r's in the column with input I_i will logically represent all the specified internal states of that column. In this case, the expressions that represent the specified states are for

Input I_1: $y_2'y_4' + y_2'y_3y_4' + y_2'y_4 + y_1y_2y_4' + y_1'y_2y_4'$

Input I_2: $y_1'y_2'y_4' + y_1'y_2' + y_1y_2'y_3 + y_1y_2y_4' + y_2y_3'y_4'$

Input I_m: $y_3'y_4' + y_1'y_2'y_3'y_4 + y_2y_3y_4' + y_1'y_3y_4'$.

The logical complement of the above expressions would represent the unspecified states in each column. The simplified expressions that represent the unspecified states are for

Input I_1: y_2y_4

Input I_2: $y_2y_4 + y_1y_2'y_3' + y_1'y_2y_3$

Input I_m: $y_1y_4 + y_2y_4 + y_3y_4 +$

The above terms represent the unspecified or don't-care states of each column as a simplified sum-of-products expression.

After obtaining the unspecified or don't-care states, it is necessary to obtain the 1-cells for each next-state variable in each column. As noted before, all the internal states represented in K_h will have the same next-state entry, namely that of the stable state of k_h. Consider the binary code for the stable state of the k-set k_h in a certain column to be $c_1\ c_2\ .\ .\ .\ c_k$. The next-state variable Y_1 will have a 1 as the next-state entry for all internal states represented in the corresponding K_h if c_1 of the stable state is 1. If c_1 is 0, then the next-state variable Y_1 will be 0 for all the internal states represented by K_h. This same reasoning can be applied to any next-state variable Y_j. In general, the next-state variable Y_j will be 1(0) for all internal states represented in K_h if c_j of the stable state of k_h is 1(0).

All the 1-cells of the next-state variable Y_j in a particular column of a flow table can be represented by the sum of the product expressions of those K_h's where $Y_j = 1$. Following is an algorithm that can be used to generate the next-state expressions in the form of Eq.(1):

1) List the k-sets of each column and the stable states of each k-set.

2) Determine the p-subcube corresponding to each transition pair in each k-set indentified with input state I_1. Retain the identity of this set of p-subcubes with input I_1 and their respective k-set. Repeat this procedure for input states $I_2, I_3, \ldots, I_m$.

3) Determine the K_h corresponding to k-set k_h indentified with input state I_1. Each K_h is obtained by the sum of the product expressions of the p-subcubes associated with k-set k_h. Retain the identity of this set of K_h's with input state I_1. Repeat this procedure for input states $I_2, I_3, \ldots, I_m$.

4) Determine the don't-care states associated with each column as follows:

   a. Form a sum-of-products expression of all the K_h's corresponding to the k-sets under input I_1. Retain the identity of this set of expressions with input I_1. Repeat this for input states $I_2, I_3, \ldots, I_m$.

b. Find the logical complement of the expression identified with input I_1. This will represent the don't-care states in this column of the flow table. Repeat for input states $I_2, I_3, \ldots, I_m$.

5) The 1-cells for each next-state variable Y_j for $j = 1, 2, \ldots, n$ can be found as follows:

   a. Determine the k-sets where $c_1 = 1$ in the stable states in the column with input I_1 to be identified with Y_1. Repeat for $c_j = 1$ for $j = 2, 3, \ldots, n$ under input I_1. Retain the identity of this set of k-sets with input I_1 and the respective Y_j. Repeat for input states $I_2, I_3, \ldots, I_m$.

   b. Form a sum-of-products expression of the K_h's that represent each of the k-sets of step 4a under input I_1 for the next-state variable Y_1. Repeat for $Y_2, Y_3, \ldots, Y_n$. Retain the identity of each expression with input I_1. Repeat for inputs $I_2, I_3, \ldots, I_m$.

6) Determine the next-state expressions for the next-state variable Y_j for $j = 1, 2, \ldots, n$ under each column as follows:

   a. Form a sum-of-products expression of the 1-cells for each Y_j that are associated with input I_1 of step 4a and the don't-care states associated with the same input of step 3. Repeat for input states $I_2, I_3, \ldots, I_m$.

   b. Perform the logical AND operation with the input I_1 and the sum-of-products expression that is associated with I_1 of step 5a. Repeat for input states $I_2, I_3, \ldots, I_m$.

7) Determine the next-state expression for the next-state variable Y_j for $j = 1, 2, \ldots, n$ by performing the logical OR operation with the respective Y_j terms from step 5b. The results will be in the form of Eq.(1).

At this point, the input states can be coded and the minimal next-state expressions can be obtained by use of a computer simplification program or some other simplification technique.

The next-state expressions for sequential machine B, shown in Figure 7, are obtained here using the algorithm as follows:

Steps 1,2,3, and 4 have already been completed in the algorithm.

<u>Step 5a</u>. The next-state variables have 1-cells in the following k-sets:

In the column with input I_1,

Y_1 has 1-cells in k_1 and k_3,

Y_2 has 1-cells in k_3 and k_4,

Y_3 has 1-cells in k_1 and k_3,

Y_4 has 1-cells in k_2.

In the column with input I_2,

Y_1 has 1-cells in k_6 and k_7,

Y_2 has 1-cells in k_7,

Y_3 has 1-cells in k_5 and k_6,

Y_4 has no 1-cells.

In the column with input I_m,

Y_1 has no 1-cells,

Y_2 has 1-cells in k_9,

Y_3 has 1-cells in k_9,

Y_4 has 1-cells in k_{10}.

<u>Step 5b</u>. The sum-of-products expression that represents the 1-cells for each next-state variable are as follows:

The expression for the 1-cells in the columns with input I_1

for Y_1 is $\quad y_2'y_4' + y_1y_2y_4' + y_2'y_3y_4'$,

for Y_2 is $\quad y_1y_2y_4' + y_1'y_2y_4'$,

for Y_3 is $\quad y_2'y_4' + y_1y_2y_4' + y_2'y_3y_4'$,

and for Y_4 is $\quad y_2'\, y_4$.

The expression for the 1-cells in the column with input I_2

for Y_1 is $y_1y_2'y_3 + y_1y_2y_4' + y_2y_3'y_4'$,

for Y_2 is $y_1y_2y_4' + y_2y_3'y_4'$,

for Y_3 is $y_1'y_2'y_4' + y_1'y_2' + y_1y_2'y_3$,

and for Y_4 there is none.

The expression for the 1-cells in the column with input I_m

for Y_1 is none,

for Y_2 is $y_2y_3y_4' + y_1'y_3y_4'$,

for Y_3 is $y_2y_3y_4' + y_1'y_3y_4'$,

and for Y_4 is $y_1'y_2'y_3'y_4$.

<u>Step</u> <u>6a</u> <u>and</u> <u>6b</u>. The unsimplified next-state expressions under each column of the flow table, including the don't-care states, are for

Input I_1:

$$Y_1 = [y_2'y_4' + y_1y_2y_4' + y_2'y_3y_4' + d(y_2y_4)]I_1$$

$$Y_2 = [y_1y_2y_4' + y_1'y_2y_4' + d(y_2y_4)]I_1$$

$$Y_3 = [y_2'y_4' + y_2'y_3y_4' + y_1y_2y_4' + d(y_2y_4)]I_1$$

$$Y_4 = [y_2'y_4 + d(y_2y_4)]I_1.$$

Input I_2:

$$Y_1 = [y_1y_2'y_3 + y_2y_3'y_4' + y_2y_4' + d(y_2y_4 + y_1'y_2y_3 + y_1y_2'y_3')]I_2$$

$$Y_2 = [y_1y_2y_4' + y_2y_3'y_4' + d(y_2y_4 + y_1y_2'y_3' + y_1'y_2y_3)]I_2$$

$$Y_3 = [y_1'y_2'y_4' + y_1'y_2' + y_1y_2'y_3 + d(y_2y_4 + y_1y_2'y_3' + y_1'y_2y_3)]I_2$$

$$Y_4 = 0I_2.$$

Input I_m:

$$Y_1 = 0I_m$$

$$Y_2 = [y_2y_3y_4' + y_1'y_3y_4' + d(y_1y_4 + y_2y_4 + y_3y_4)]I_m$$

$$Y_3 = [y_2y_3y_4' + y_1'y_3y_4' + d(y_1y_4 + y_2y_4 + y_3y_4)]I_m$$

$$Y_4 = [y_1'y_2'y_3'y_4 + d(y_1y_4 + y_2y_4 + y_3y_4)]I_m$$

Step 7. The next-state expressions are

$$Y_1 = [y_2'y_4' + y_1y_2y_4' + y_2'y_3y_4' + d(y_2y_4)]I_1$$
$$+ [y_1y_2'y_3 + y_2y_3'y_4' + y_2y_4' + d(y_2y_4 + y_1'y_2y_3 + y_1y_2'y_3')]I_2 + \ldots$$
$$+ 0I_m$$

$$Y_2 = [y_1y_2y_4' + y_1'y_2y_4' + d(y_2y_4)]I_1$$
$$+ [y_1y_2y_4' + y_2y_3'y_4' + d(y_2y_4 + y_1y_2'y_3' + y_1'y_2y_3)]I_2 + \ldots$$
$$+ [y_2y_3y_4' + y_1'y_3y_4' + d(y_1y_4 + y_2y_4 + y_3y_4)]I_m$$

$$Y_3 = [y_2'y_4' + y_2'y_3y_4' + y_1y_2y_4' + d(y_2y_4)]I_1$$
$$+ [y_1'y_2'y_4' + y_1'y_2' + y_1y_2'y_3 + d(y_2y_4 + y_1y_2'y_3' + y_1'y_2y_3)]I_2 + \ldots$$
$$+ [y_2y_3y_4' + y_1'y_3y_4' + d(y_1y_4 + y_2y_4 + y_3y_4)]I_m$$

$$Y_4 = [y_2'y_4 + d(y_2y_4)]I_1$$
$$+ 0I_2 + \ldots$$
$$+ [y_1'y_2'y_3'y_4 + d(y_1y_4 + y_2y_4 + y_3y_4)]I_m.$$

At this point one can code the input states and find the minimal next-state expressions by using simplification techniques.

The algorithm just presented generates the next-state expressions, complete with don't-cares, directly from the internal-state assignment and the flow table, without requiring the construction of the transition table.

IV. SUMMARY

This paper has presented a method that will enable one to predict which of many internal-state assignments for a normal fundamental-mode asynchronous sequential machine will yield a simpler set of next-state expressions. This can be done by using the assignment weighting scheme described herein, which is based on two simple algorithms that determine important characteristics of each internal-state assignment.

The second problem treated in this paper is one of going directly from an internal-state assignment to the next-state expressions for a normal fundamental-mode asynchronous sequential machine. An algorithm is presented for generating the next-state expressions without requiring the construction of a transition table. This algorithm would seem to make the problem of generating the next-state expressions an easier one to program on a computer, because the algorithm is given as a sequence of steps and no decisions are required in following the algorithm. Currently, a computer program is being written to implement this algorithm.

BIBLIOGRAPHY

1. HUFFMAN, D.A., "The Synthesis of Sequential Switching Circuits," J. of the Franklin Institute, Volume 257, pp. 161-190 and 257-303, March and April 1954.

2. HUFFMAN, D.A., "A Study of the Memory Requirements of Sequential Switching Circuits," Massachusetts Institute of Technology, Technical Report No. 293, March 1955.

3. LIU, C.N., "A State Variable Assignment for Asynchronous Sequential Circuits," J. of the ACM, Volume 10, pp. 209-216, April 1963.

4. TRACEY, J.H., "The Internal State Assignment for Asynchronous Sequential Machines," IEEE Transactions on Electronic Computers, Volume EC-15, pp. 551-560, August 1966.

VITA

The author was born on July 25, 1943 in Marquette, Michigan. He received his primary and secondary education in Munising, Michigan. He received a Bachelor of Science degree in Electrical Engineering from Michigan Technological University in June, 1965. The author has been enrolled in the Graduate School at the University of Missouri at Rolla and has been on the staff in the Electrical Engineering Department since September, 1965.

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