1969

State assignments for non-normal asynchronous sequential circuits

Gary Keith Maki

Department:

STATE ASSIGNMENTS FOR NON-NORMAL ASYNCHRONOUS SEQUENTIAL CIRCUITS

by

GARY KEITH MAKI 1943

A DISSERTATION

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI - ROLLA

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

1969

James H. Tracey
Advisor

John S. Gaylen

S. A. Szygenda

Ralph L. Carson

Lyle E. Pursell

Frank J. Kern

ABSTRACT

There is a lack of procedures that can be used to find good internal state assignments for asynchronous sequential circuits operating in the non-normal mode. Presented here, are two generalized state assignments, which are functions only of the number of rows in a flow table. The suggested bounds for the generalized state assignments are $m + [\log_2 m]$ and $m + [m/2]$ internal state variables for a 2^m-row flow table, where [ ] means "next lowest integer". Both generalized state assignments produce group (linear) codes. The algorithms for generating these internal state assignments are easy and straight-forward to implement. It is shown that each of these state assignments satisfactorily encode certain classes of flow tables. Even though a general proof has not been found to show that these assignments are standard, worst-case situations have been constructed, and it has never been necessary to increase the suggested bounds.

An internal state assignment procedure for obtaining non-standard or non-generalized state assignments is also presented. The internal state assignments, using the proposed method, are obtained in a systematic manner; and generally require fewer internal state variables than other procedures presently available.

ACKNOWLEDGEMENT

The author is deeply indebted to Dr. James H. Tracey for his assistance and guidance during the entire graduate program and in the preparation of this thesis.

Thanks also to Dr. Stephen Szygenda and Ross Heitzmann for their careful and prompt reading of this paper.

The author wishes to express his gratitude to his wife, Alice, for her support and assistance during these four years of graduate work. After being the bread winner for four years, she is well-deserving of a rest.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A sequential circuit denotes a class of devices whose outputs depend not only on the present inputs but also on previous inputs. In order to keep track of previous inputs, a sequential circuit will assume one of a set of internal states that will "remember" necessary information regarding the previous inputs. Changes in internal state permit the circuit to produce different outputs for the same inputs and in this way the circuit exhibits a memory quality.

Sequential circuits can be divided into two basic categories: synchronous and asynchronous. In the synchronous design, clock pulses synchronize the operation of the circuit. In the asynchronous design a clock is assumed not to be available or perhaps not desired. A desirable feature of the asynchronous design is that the resulting circuit may take full advantage of the basic device speed since the circuit does not have to wait for the arrival of clock pulses before effecting a transition.

The operation of an asynchronous sequential circuit can be described by means of a flow table such as shown in Figure 1-1. A flow table is a two-dimensional array of next-state entries whose coordinates are present internal state and input state. Each column represents an input state; each row represents an internal state; and the table entries specify the next internal state. If the next internal state is equal to the present internal state, the state is said to be stable and is denoted by a circled entry. Uncircled entries are

called unstable. The combination of input state and present internal state is called the total circuit state. Associated with each stable state is an output state.

	I_1	I_2	I_3
1	(1)/1	3	5
2	1	5	(2)/3
3	1	(3)/2	2
4	(4)/4	5	2
5	4	(5)/1	(5)/2

FIGURE 1-1. Flow table example

The operation of the circuit shown in Figure 1-1 can be described in part as follows: if the present internal state is 1 and the input state is I_1, the next internal state is 1 and the circuit is in a stable condition with an output of 1. As long as the input does not change, the circuit will remain in internal state 1. If the input is changed to I_2, the next state is internal state 3 and the circuit will experience a transition to state 3 and will remain in this state until the input is changed again; the output state would be 2. Note that if the input is changed back to I_1, the circuit will return to internal state 1.

The output states are a function of the input and internal states of the flow table. Since this paper is concerned only with the internal operation of a sequential circuit, the output states will be omitted in all future flow tables.

In some flow tables, particular total circuit states are

never entered and the corresponding next state entries are unspecified. The unspecified next state entry is called a "don't care" state. Flow tables with "don't care" states are called incompletely specified flow tables. Since a "don't care" state is never entered in the synthesis of the actual circuit, it is permissable to assign any value to a "don't care" state to simplify the final design. The material presented in this paper applies to completely specified and to incompletely specified flow tables.

Definition: 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.

Fundamental mode operation is usually required in practical design of these circuits and hence this paper will be concerned only with the design of fundamental mode circuits.

Definition: A flow table with the characteristic that each unstable state leads directly to a stable state is called a normal flow table. A non-normal flow table allows the circuit to assume a sequence of internal states in going from an unstable state to a stable state.

To illustrate the above definition, compare the flow table in Figure 1-1, where all unstable states lead directly to stable states, with that shown in Figure 1-2, where some unstable states lead to other unstable states.

	I_1	I_2	I_3
1	(1)	3	5
2	1	4	(2)
3	2	(3)	4
4	(4)	5	2
5	4	(5)	(5)

FIGURE 1-2. Non-normal flow table

Assume in both circuits that the present internal state is 3 with input I_2; in both cases the circuit is in a stable condition. If the input is changed from I_2 to I_1, both circuits described by the flow tables would finally experience a transition to internal state 1. The difference in operation is that the circuit in Figure 1-1 will experience a direct transition from internal state 3 to internal state 1; where in Figure 1-2, the circuit of that flow table will first experience a transition from internal state 3 to internal state 2 and then to internal state 1.

One of the basic steps in the synthesis procedure of designing an asynchronous sequential circuit is obtaining an internal state assignment. The internal state assignment problem consists basically of encoding each of the internal states of a sequential circuit with a q-tuple (binary code) or set of q-tuples. The q-tuples are encoded by q internal state variables, $y_1, y_2, \ldots, y_q$. With q internal state variables, at most 2^q internal states can be encoded.

The values of the internal state variables for each state can be assigned in many different ways. The way in which these values are assigned can make vast differences in the logical structure of the circuit realization, in the operating speed of the actual circuit, and in the amount of circuitry required to implement the realization. In general, the state assignment method is chosen to achieve one goal or another, such as these: maximum operating speed; minimum hardware requirements for realization of the resulting circuit; or for even the greatest ease of design, which is necessarily a compromise between operating speed and economy for the sake of design simplicity.

Definition: A race condition exists in an asynchronous sequential circuit whenever a transition between a pair of states requires the simultaneous change of two or more internal state variables. If a race condition can lead to false operation of the circuit, it is designated as a critical race.

Every internal state assignment must permit circuit operations free of critical races. One basic approach for doing this is to allow no races; therefore, there can be no critical races. The second approach is to obtain an assignment that permits races, where all races are non-critical.

In an assignment which permits a circuit to operate in the normal mode, all internal state variables that are to change state during a transition from an unstable state to a stable state are excited simultaneously at the beginning of the transition. Such assignments are called single-transition-time assignments [1]. In an assignment which permits a circuit

to operate only in the non-normal mode, all internal state variables that are to change state during a transition between states do not have to be simultaneously excited at the beginning of the transition. In many cases, only one internal state variable at a time is excited to effect a transition between a pair of states. If the average amount of time for an internal state variable to change state is Δt, then for normal operation the average time for each transition between an unstable and stable state would be Δt. For non-normal operation, the average time for each transition between an unstable and stable state would be $\eta \cdot \Delta t$, where η is equal to the number of unstable states the circuit assumed before arriving at the stable state. It is clear, then, that an assignment which permits normal operation results in a circuit with faster operating speed than an assignment which allows only non-normal operation.

Internal state assignment procedures for constructing critical-race-free assignments in normal mode flow tables are well established [1,2,3,4]. Some of these procedures produce minimum variable assignments that are dependent on the characteristics of the flow table. Computer programs are available that generate the state assignment and corresponding design equations [6,7,8]. On the other hand, relatively little has been published concerning the non-normal mode of operation.

Definition: A <u>standard state assignment</u> is one which is a function only of the number of internal states that originally appear in the flow table and is independent of particular

internal state transitions, which may or may not be required.

From the ease of design standpoint, a standard state assignment might be the simplest to use; thus one can be assured that the resulting state assignment is critical-race free, and it takes minimal effort to obtain this assignment. On the other hand, standard state assignments generally require more internal state variables than non-standard assignments. For a 2^m-row flow table, Huffman [2,3] has established upper bounds of $2^m - 1$ internal state variables for a circuit operating in the normal mode, and 2m - 1 internal state variables for a circuit operating in the non-normal mode. The bounds associated with these different assignments imply that the price usually paid for increased circuit speed is an increase in the number of internal state variables with a probable increase in circuit complexity. In general, fewer internal state variables are required to encode a non-normal flow table than a normal flow table. The most serious limitation in attempting to encode a non-normal flow table is the lack of techniques that are easy to apply in obtaining state assignments.

At present, the non-normal standard assignment developed by Huffman [2,3] is the only straight-forward method for obtaining non-normal state assignments. Bounds of 2m - 1 internal state variables for non-normal 2^m-row flow tables are associated with these intermeshed row-set assignments. There is a multiple coding of internal states in this assignment, where each state is assigned several codes. The structure of the

intermeshed row-set assignment is such that at least one of the codes assigned to each internal state is adjacent to a code assigned to every other state in the assignment; therefore, it is clear that the resulting assignment is standard.

Recently a standard assignment for an 8-row flow table has been presented that requires only four internal state variables instead of five produced by Huffman [8]. This assignment was obtained by an exhaustive approach aided by a computer. Program run times imply that the exhaustive approach, used to verify that an assignment is standard, is impractical for larger flow tables.

Hazeltine [9] also treats the internal state assignment problem for non-normal flow tables. Hazeltine's method consists of attempting to construct connecting sequences between all stable states and their corresponding unstable states. The circuit can make the required transitions within each connecting sequence through a change of a single internal state variable at a time. There is trial and error associated with obtaining the assignment and the connecting sequences. The upper bound for Hazeltine's method is also 2m - 1 internal state variables for a 2^m-row flow table.

Presented in this paper are two generalized state assignments which are generated in an easy and straight-forward manner. The suggested bounds for these assignments are $m + [\log_2 m]$ and $m + [m/2]$ for a 2^m-row flow table, where [ ] means "next lowest integer." Since these assignments are functions only of the number of rows in a flow table, they

have characteristics similar to that of standard assignments. Since a general proof has not been found to show that these assignments satisfactorily encode every flow table, they cannot be called standard assignments.

Each of the generalized state assignments can be shown to satisfactorily encode certain classes of flow tables, and these proofs are given in the following chapters. Worst-case flow tables have been constructed, and it has never been necessary to exceed the suggested bounds.

Both generalized state assignments seem to require fewer state variables to encode a 2^m-row flow table than a Huffman assignment. Each generalized state assignment is much easier to produce and frequently requires fewer internal state variables than assignments obtained with Hazeltine's method. For example, a five variable assignment is given for the flow table presented in Hazeltine's paper and both generalized assignments produce a four variable assignment that satisfactorily encode this flow table.

Lastly, a non-normal state assignment procedure is presented that is dependent on flow table structure. An algorithm for finding non-standard or non-generalized state assignments is given and applies to all types of flow tables. The algorithm is designed to be relatively straight-forward, but may not necessarily produce minimum variable assignments.

II. GENERALIZED STATE ASSIGNMENT 1

A. Preliminaries

Before specific details are given concerning either generalized state assignment, a few preliminary ideas and definitions will be presented which are common to both assignments for non-normal mode sequential circuits.

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

Definition: A column partition is a collection of the k-sets that appear in a column of a flow table.

The flow table of Figure 2-1 illustrates the above definitions.

	I_1	I_2
1	(1)	2
2	1	(2)
3	1	(3)
4	(4)	3
5	4	(5)
6	(6)	5

FIGURE 2-1. Flow table to illustrate column partitions

The column partition for the column under input state I_1 is {1,2,3; 4,5; 6} and the column partition for the column under input state I_2 is {1,2; 3,4; 5,6}. Note that the last column partition is composed of only 2-sets.

Definition: A transition path is the set of states that the circuit assumes in undergoing a transition between an unstable and stable state.

An internal state assignment is a satisfactory or valid assignment, if there are no critical races, and if the intersection of the transition paths associated with states of different k-sets is the null set. Each transition path associated with the state assignments presented in this paper will be a sequence of unit-distance states where only one state variable is excited in effecting a transition between the states of the transition path. Since only one state variable is excited in effecting a transition from one state to another, no races, critical or otherwise, can occur. Therefore, insuring that the transition paths associated with different k-sets have no states in common is the only condition that has to be met in showing the existence of a valid assignment.

Saucier [8], in showing that a state assignment for a 12-row flow table is satisfactory for all column partitions which consist of only 2-sets, used a computer to verify that the transition paths of the different 2-sets were disjoint; the run time associated with this program was in the order of five hours. This implies that it would be difficult to verify by an exhaustive approach that all column partitions of even this type for larger flow tables are satisfactorily encoded by a particular assignment.

It would be highly desirable to show that the generalized assignments are standard assignments. From the above dis-

cussion, certainly an exhaustive approach is out of the question due to excessive program run times. In an effort to show that the generalized state assignments satisfactorily code as many cases as possible, worst-case situations will be postulated and these cases will be given most of the consideration.

Definition: If a transition is to be effected between a pair of states S_a and S_b, then the pair of states S_a, S_b is called a transition pair.

Both Hazeltine [9] and Saucier [8] imply that column partitions that consist of only 2-sets are the most difficult to encode. There seems to be more flexibility associated with establishing transition paths between the states of k-sets larger than 2-sets. The transition pairs associated with 2-sets consist of a stable and an unstable state and there is one transition pair for each 2-set. The transition path for a 2-set must not contain any states in the state assignment other than those states of the 2-set. On the other hand, both states of a transition pair can be unstable states in k-sets larger than 2-sets. For example, consider a 3-set composed of states 1,2, and 3, with state 1 being the stable state. One set of transition pairs would be 1,2; 1,3 where the stable state is a member of each transition pair. Two other sets of transition pairs are 1,3; 2,3 and 1,2; 2,3. The operation of the circuit dictated by the set of transition pairs 1,3; 2,3 for example, is described as follows: if the circuit assumed unstable state 2, it would experience a transition to state 3

and then to state 1. In general, for k-sets larger than 2-sets there are many sets of transition pairs that can describe the operation of the circuit. Furthermore, the transition paths associated with transition pairs of the same k-set can have states in common. Consequently, there are more possibilities available in finding a satisfactory transition path for k-sets larger than 2-sets.

There also seems to be considerable flexibility associated with establishing transition paths in flow tables which have "don't care" states. During the design of the circuit, "don't care" states can be specified to any desired value to aid in the synthesis of the circuit, and therefore, can become members of any k-set to facilitate the construction of the transition paths.

In this paper, most attention will be given to column partitions which consist of only 2-sets since they seem to constitute a situation where it is more difficult to construct a satisfactory set of transition paths.

Definition: Mod 2 addition (exclusive OR) is defined as follows:

$0 \oplus 0 = 0$

$0 \oplus 1 = 1$

$1 \oplus 0 = 1$

$1 \oplus 1 = 1$

The symbol denoting this operation is $\oplus$.

As noted in the definition, mod 2 addition is defined over the set of binary digits {0,1}. It can be seen that this

operation is commutative, associative, and closed.

Definition: Consider any sequence of binary digits $a_1, a_2, \ldots, a_\gamma$. The sequence possesses <u>even parity</u> if the number of 1's in the sequence is even, and possesses <u>odd parity</u> if the number of 1's in the sequence is odd.

Consider a set of binary digits $\{a_i\}$, $i = 1,2, \ldots, \xi$ where ξ is an even number. Let the mod 2 sum $a_1 \oplus a_2 \oplus \ldots \oplus a_\xi = b$. Since the operation is closed, $b \in \{0,1\}$. It has been established [12] that if $a_1, a_2, \ldots, a_\xi$ possesses even parity, then $b = 0$ and if $a_1, a_2, \ldots, a_\xi$ possesses odd parity, then $b = 1$. Therefore, the binary sequence $a_1, a_2, \ldots, a_\xi, b$ will always possess even parity.

B. <u>Characteristics of Assignment 1</u>

Generalized State Assignment 1 (hereafter called Assignment 1) requires $m + [\log_2 m]$ state variables to encode a 2^m-row flow table, where [ ] means the "next lowest integer." For example, for a 2^7-row flow table, Assignment 1 requires nine state variables. One distinct advantage associated with this state assignment is the ease at which it can be generated.

Definition: A set of state variables $\{y_1, y_2, \ldots, y_k\}$ is an <u>independent set of state variables</u> if no y_i in the set is equal to the mod 2 sum of a subset of the other state variables of the set.

To illustrate this definition, consider the following set of state variables.

y_1	y_2	y_3	y_4
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Since $y_2 \oplus y_3 = y_4$, this set is not an independent set of state variables. An independent set would be y_1 and any two of the subset $\{y_2, y_3, y_4\}$.

Definition: Consider a set of independent state variables, $\{y_1, y_2, \ldots, y_\xi\}$ where ξ is a non-zero even number ($\xi \geq 2$). If $y_1 \oplus y_2 \oplus \ldots \oplus y_\xi = y_\eta$, then the complete set $\{y_1, y_2, \ldots, y_\xi, y_\eta\}$ is called a <u>parity set</u>. There are ξ independent and one dependent state variable(s) in each parity set. The dependent state variable y_η of the parity set is considered as being generated from the independent state variables.

From earlier discussions, each parity set will possess even parity. In the preceding example, state variables y_2, y_3, and y_4 form a parity set and y_4 is said to be generated from y_2 and y_3 ($y_4 = y_2 \oplus y_3$).

Following is an algorithm for generating Assignment 1 for a 2^m-row flow table:

Step 1. Form all possible 2^m code words with m binary

independent state variables. Denote these as state variables $y_1, y_2, \ldots, y_m$.

Step 2. The final code will have $m + [\log_2 m]$ state variables. Two slightly different procedures are given to determine $y_{m+1}, y_{m+2}, \ldots, y_n$, $n = m + [\log_2 m]$.

a) If m is even, divide the set of state variables $\{y_1, y_2, \ldots, y_m\}$ into $[\log_2 m]$ nearly equal disjoint subsets, each subset containing an even number of state variables. The state variables in each of these subsets are considered as the independent variables of a parity set and each of the variables $y_{m+1}, y_{m+2}, \ldots, y_n$ is considered as the dependent variable of one of the $[\log_2 m]$ subsets. Therefore, each of the $[\log_2 m]$ subsets generates one of the variables $y_{m+1}, y_{m+2}, \ldots, y_n$. Every state variable is an element of some parity set.

b) If m is odd, divide the set of state variables $\{y_1, y_2, \ldots, y_{m-1}\}$ into $[\log_2 m]$ nearly equal disjoint subsets, each subset containing an even number of state variables. With each of the $[\log_2 m]$ subsets, generate one of the state variables $y_{m+1}, y_{m+2}, \ldots, y_n$ as in step 2a above. State variable y_m is the only state variable that is not a member of some parity set.

To illustrate this algorithm, a number of assignments will be generated. First consider the assignment for a 2^3-row flow table. Step 1 states that with y_1, y_2, and y_3, form 2^3 unique code words. This is shown in Figure 2-2. Since $[\log_2 3] = 1$, one parity set needs to be formed. This parity set can be $\{y_1, y_2, y_4\}$ and the additional state variable y_4 would be equal to $y_1 \oplus y_2$. The complete assignment is shown in Figure 2-3.

State	y_1	y_2	y_3
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

FIGURE 2-2. First step in generating an assignment

State	y_1	y_2	y_3	y_4
1	0	0	0	0
2	0	0	1	0
3	0	1	0	1
4	0	1	1	1
5	1	0	0	1
6	1	0	1	1
7	1	1	0	0
8	1	1	1	0

FIGURE 2-3. Complete code for an 8-row flow table

An equally valid assignment would have parity sets of $\{y_1, y_3, y_4\}$ or $\{y_2, y_3, y_4\}$ in which cases, $y_4 = y_1 \oplus y_3$ and $y_4 = y_2 \oplus y_3$ respectively. Each of these assignments is similar to the standard assignment known to exist for an 8-row flow table and requires one less state variable than that produced by Huffman [8]. Saucier obtained and proved that this assignment is a

standard assignment by an exhaustive approach aided by a computer.

To illustrate the assignment algorithm even further, the code for a 2^5-row flow table is shown in Figure 2-4.

State	y_1	y_2	y_3	y_4	y_5	y_6	y_7
1	0	0	0	0	0	0	0
2	0	0	0	0	1	0	0
3	0	0	0	1	0	0	1
4	0	0	0	1	1	0	1
5	0	0	1	0	0	0	1
6	0	0	1	0	1	0	1
7	0	0	1	1	0	0	0
8	0	0	1	1	1	0	0
9	0	1	0	0	0	1	0
10	0	1	0	0	1	1	0
11	0	1	0	1	0	1	1
12	0	1	0	1	1	1	1
13	0	1	1	0	0	1	1
14	0	1	1	0	1	1	1
15	0	1	1	1	0	1	0
16	0	1	1	1	1	1	0
17	1	0	0	0	0	1	0
18	1	0	0	0	1	1	0
19	1	0	0	1	0	1	1
20	1	0	0	1	1	1	1
21	1	0	1	0	0	1	1
22	1	0	1	0	1	1	1
23	1	0	1	1	0	1	0
24	1	0	1	1	1	1	0
25	1	1	0	0	0	0	0
26	1	1	0	0	1	0	0
27	1	1	0	1	0	0	1
28	1	1	0	1	1	0	1
29	1	1	1	0	0	0	1
30	1	1	1	0	1	0	1
31	1	1	1	1	0	0	0
32	1	1	1	1	1	0	0

FIGURE 2-4. State assignment for a 2^5-row flow table

In this assignment, the parity sets are $\{y_1, y_2, y_6\}$ and $\{y_3, y_4, y_7\}$ where $y_1 \oplus y_2 = y_6$ and $y_3 \oplus y_4 = y_7$.

In generating an assignment for a 2^6-row flow table, the

independent state variables are $y_1, y_2, y_3, y_4, y_5, y_6$ and the parity sets are $\{y_1, y_2, y_7\}$ and $\{y_3, y_4, y_5, y_6, y_8\}$ with $y_1 \oplus y_2 = y_7$ and $y_3 \oplus y_4 \oplus y_5 \oplus y_6 = y_8$.

It will be shown that the code generated with the above algorithm constitutes a group. A group is a set of elements and an operation where –

1) the operation over the set of elements is closed,

2) the operation is associative,

3) there exists an identity element,

4) each element has an inverse.

In Assignment 1 for a 2^m-row flow table, the elements of the group are the binary n-tuples (the codes for the states), $n = m + [\log_2 m]$, formed by the assignment algorithm and the operation is mod 2 addition (exclusive OR).

Theorem 2-1: Mod 2 addition over any set of binary n-tuples is commutative and associative. If the all-zero n-tuple S_o is in the set, then S_o is the identity element and each n-tuple is its own inverse.

Proof: Since $a_i \oplus b_i = b_i \oplus a_i$, where $a_i, b_i \in \{0,1\}$,

$$S_a \oplus S_b = (a_1, a_2, \ldots, a_n) \oplus (b_1, b_2, \ldots, b_n) =$$

$$(a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n) =$$

$$(b_1 \oplus a_1, b_2 \oplus a_2, \ldots, b_n \oplus a_n) = S_b \oplus S_a,$$

and the commutative law holds. By definition, an identity element e of a set satisfies the relationship

$$S_i \oplus e = e \oplus S_i = S_i,$$

for all S_i in the set. Since for $a_i \in \{0,1\}$,

$$a_i \oplus 0 = 0 \oplus a_i = a_i,$$

n-tuple $S_o = (0, 0, \ldots, 0)$ is the identity element since

$$S_o \oplus S_a = (0, 0, \ldots, 0) \oplus (a_1, a_2, \ldots, a_n) =$$

$$(0 \oplus a_1, 0 \oplus a_2, \ldots, 0 \oplus a_n) = (a_1, a_2, \ldots, a_n) = S_a.$$

By definition, the inverse of element S_i is S_i^{-1}, where

$$S_i \oplus S_i^{-1} = S_i^{-1} \oplus S_i = S_o,$$

S_o being the identity element. Since $a \oplus b = 0$, $a,b \in \{0,1\}$ only when $a = b$,

$$S_a \oplus S_a = (a_1, a_2, \ldots, a_n) \oplus (a_1, a_2, \ldots, a_n) =$$

$$(a_1 \oplus a_1, a_2 \oplus a_2, \ldots, a_n \oplus a_n) = (0, 0, \ldots, 0) = S_o,$$

for all S_a. Therefore each element is its own inverse. Since the associative law holds for mod 2 addition over the binary elements, all n-tuples under mod 2 addition satisfy the associative law. ##

From theorem 2-1, only closure needs to be established to show that a set of n-tuples that satisfy the hypothesis of this theorem is a group.

Lemma 2-1: 2^m code words coded with m independent state variables and mod 2 addition constitute a group.

Proof: From theorem 2-1, only closure need be shown. Since the maximum number of code words that can be generated with m state variables is 2^m (all possible m-tuples), closure must hold. ##

Theorem 2-2: The 2^m elements of the state assignment produced by Assignment 1 together with the operation of mod 2 addition form a group.

Proof: From theorem 2-1, only closure need be shown.

Consider the m-tuples S_1', S_2', and S_i', such that

$$S_1' = (a_1, a_2, \ldots, a_m), \quad S_2' = (b_1, b_2, \ldots, b_m)$$

and

$$S_i' = (c_1, c_2, \ldots, c_m).$$

From lemma 2-1, since the set of m-tuples is closed, there exists an S_i' such that

$$S_1' \oplus S_2' = S_i'$$

or

$$(a_1, a_2, \ldots, a_m) \oplus (b_1, b_2, \ldots, b_m) =$$
$$(c_1, c_2, \ldots, c_m).$$

Then $a_i \oplus b_i = c_i$ for all i, i = 1, 2, . . ., m. Let the parity sets be $\{y_1, y_2, \ldots, y_k, y_{m+1}\}$, $\{y_{k+1}, y_{k+2}, \ldots, y_\ell, y_{m+2}\}$ etc. Then the states S_1, S_2, and S_i in Assignment 1 are:

$$S_1 = (a_1, a_2, \ldots, a_m, a_1 \oplus a_2 \oplus \ldots \oplus a_k, a_{k+1} \oplus a_{k+2} \oplus \ldots \oplus a_\ell, \ldots)$$

$$S_2 = (b_1, b_2, \ldots, b_m, b_1 \oplus b_2 \oplus \ldots \oplus b_k, b_{k+1} \oplus b_{k+2} \oplus \ldots \oplus b_\ell, \ldots)$$

$$S_i = (c_1, c_2, \ldots, c_m, c_1 \oplus c_2 \oplus \ldots \oplus c_k, c_{k+1} \oplus c_{k+2} \oplus \ldots \oplus c_\ell, \ldots).$$

Since

$$(a_1 \oplus a_2 \oplus \ldots \oplus a_k) \oplus (b_1 \oplus b_2 \oplus \ldots \oplus b_k) =$$
$$(c_1 \oplus c_2 \oplus \ldots \oplus c_k)$$

and

$$(a_{k+1} \oplus a_{k+2} \oplus \ldots \oplus a_\ell) \oplus (b_{k+1} \oplus b_{k+2} \oplus \ldots \oplus b_\ell) =$$
$$(c_{k+1} \oplus c_{k+2} \oplus \ldots \oplus c_\ell),$$

then $S_1 \oplus S_2 = S_i$. Since this is valid for all S_1 and S_2 in

Assignment 1, closure is assured. ##

Assignment 1 is denoted as group G.

The state assignment produced by Assignment 1 forms a group with a maximum distance property. There are no states a distance greater than some maximum and it will be shown that this maximum value is m for the state assignment for a 2^m-row flow table.

Definition: The distance between two states S_i and S_j is the number of state variables in which the binary code representation of those states differ. The distance is also equal to the number of 1's that appear in the mod 2 sum of S_i and S_j. Let the distance between S_i and S_j be denoted as $||S_i \oplus S_j||$.

If the distance between two states is d, then d state variables must be excited to effect a transition between the states. As the distance between states of a transition pair increases, the number of states in the resulting transition path also increases which means that more "spare" states are needed to produce a satisfactory transition. It will be shown later that it is desirable for the distance to be a minimum between states in a state assignment.

The minimum maximum distance between states in a set of 2^m states is m. The "best" assignment that could be achieved is one which encodes the 2^m states with m state variables. In this case, each m-tuple is a distance m from another m-tuple. It will be shown that encoding the states with Assignment 1 does not increase the maximum distance. This means that the

same number of state variables must be excited to effect a maximum distance transition for states encoded with Assignment 1 as if the states were encoded with m state variables. Encoding the states with Assignment 1 increases the number of "spare" states that are available, but does not increase the number of states needed in a transition path between maximum distance states. Following are some theorems which prove that the maximum distance between states encoded with Assignment 1 is m.

Definition: The weight of a state is equal to $||S_o \oplus S_i||$, where S_o is the identity element in the group. The weight is also equal to the number of 1's in the code for state S_i. The weight is also denoted as $||S_i||$.

Theorem 2-3: If $||S_o \oplus S_i|| \leq d$, some distance, for all S_i in a group code, then $||S_k \oplus S_j|| \leq d$, for all S_k and S_j in the group.

Proof: Since S_o is the identity element, $S_o \oplus S_i = S_i$ for all S_i in the group. From the closure property of groups, there exists an S_i such that $S_k \oplus S_j = S_i$. Then from the hypothesis, $||S_o \oplus S_i|| \leq d$ and

$$||S_o \oplus S_i|| = ||S_o \oplus S_j \oplus S_k|| = ||S_j \oplus S_k||.$$

Therefore, $||S_j \oplus S_k|| \leq d$. ##

The significance of theorem 2-3 is that the maximum weight of a code word in a group is the maximum distance between the code words. To show that the state assignment for a 2^m-row flow table has a maximum distance of m, one need only show that the maximum weight of any code word is m.

Lemma 2-2: A parity set containing k independent state variables has a maximum weight equal to the number of independent state variables k.

Proof: Let a parity set be composed of $\{y_1, y_2, \ldots, y_k, y_p\}$, where $y_1, y_2, \ldots, y_k$ are the independent state variables and $y_p = y_1 \oplus y_2 \oplus \ldots \oplus y_k$. Since there is even parity over each parity set and there is an even number of independent state variables, the maximum weight of the parity set is k. (Since k+1 is odd, k is the maximum number of 1's that can appear in the parity set.) ##

Theorem 2-4: The maximum distance between any pair of states in the state assignment obtained from Assignment 1 is m for a 2^m-row flow table.

Proof: Since every state variable is in one of the parity sets for a code for a 2^m-row flow table when m is even, the maximum weight of a state is equal to the sum of the maximum weights of the parity sets. From lemma 2-2, the maximum weight of a parity set is equal to the number of independent variables in that parity set. It follows that since each parity set is disjoint, the sum of the maximum weights of several parity sets is equal to the sum of the independent state variables in these parity sets. Then for m even, there are $[\log_2 m]$ parity sets, with m independent state variables in these parity sets and the maximum weight is m. For m odd, y_m is not in any parity set; therefore the maximum weight is equal to the sum of the maximum weights of the parity sets plus the maximum weight of y_m which is 1. Since the $[\log_2 m]$

parity sets are coded with m-1 independent state variables, the sum of the maximum weights of the parity sets is m-1. Therefore, the maximum weight of any state is m, for m odd or even. From theorem 2-3, if the maximum weight is m, then the maximum distance between states is m. ##

As stated previously, a transition path is the set of states the circuit can assume during a transition from one state to another. For the assignment presented here, the transition paths are composed of a sequence of unit-distance states, where only one state variable is excited in going from one state in this sequence to another. For example, if a transition occurred between states S_a and S_b and the transition path was S_a,S_1,S_2,S_3,S_b, with the circuit going from S_a to S_1, then to S_2, then to S_3, and finally to S_b, then each state in the sequence S_a,S_1,S_2,S_3,S_b would differ in only one state variable in their binary code representations from the state immediately before and after it in the sequence. S_a is called the initial state and S_b, the final state. S_1 is called the first state in the transition path; S_2, the second; and S_3, the third. With this type of transition path no races, critical or otherwise, can occur. Therefore, the only problem left to be concerned about in obtaining a valid assignment is to guarantee that the transition paths of different 2-sets in a column partition have no states in common.

Definition: Let the distance between two states be d. If d state variables are excited, each one only once, in effecting a transition between the states, then the transition is called

a minimum length (ML) transition and the transition path is called a minimum length (ML) transition path. Including the states of the transition pair, the number of states in a ML transition path is d + 1.

Non-ML transitions would require more states in the transition path than ML transitions and therefore, the latter are more desirable than the former.

C. Principal-Column Partitions

It can be shown that Assignment 1 satisfactorily encodes a type of column partition that is considered next.

Definition: A principal-column partition of a flow table is composed of 2-sets for which the same set of state variables must be excited in a transition from the initial to final state of any 2-set.

To illustrate a principal-column partition, consider the flow table and state assignment of Figure 2-5.

Y_1	Y_2	Y_3	Y_4		I_1	I_2
0	0	0	0	1	(1)	8
0	0	1	0	2	(2)	3
0	1	0	1	3	6	(3)
0	1	1	1	4	(4)	5
1	0	0	1	5	4	(5)
1	0	1	1	6	(6)	7
1	1	0	0	7	2	(7)
1	1	1	0	8	1	(8)

FIGURE 2-5. Flow table with a principal-column partition

The column partitions under input I_1 and I_2 are α_1 = {1,8; 2,7; 3,6; 4,5} and α_2 = {1,8; 2,3; 4,5; 6,7} respectively. In column partition α_1, state variables y_1, y_2, and y_3 must be excited to effect a transition from one state of any 2-set to the other state of the same 2-set. However, in column partition α_2, a transition from state 1 to state 8 requires a change in variables y_1, y_2, and y_3 and a transition from state 2 to state 3 requires a change in variables y_2, y_3, and y_4. Therefore, α_1 is a principal-column partition and α_2 is not.

Definition: A maximum-distance column partition is composed of 2-sets and the distance between the states of each 2-set is the maximum distance of the code.

In Figure 2-5, both α_1 and α_2 are maximum-distance column partitions. Column partition α_1 can be called a maximum-distance principal-column partition.

From theorem 2-4, the distance between the states of any 2-set is m for a maximum-distance column partition. For a maximum-distance principal-column partition, the same m state variables would be excited in effecting a transition between the states of any 2-set.

In an effort to describe a transition path in mathematical terms, the following notation and idea is presented. If S_a is an initial state, then p_1 is defined as a quantity such that the first state A_1 in the transition path can be expressed as $p_1 \oplus S_a = A_1$, where p_1 is an n-tuple. Since A_1 is a distance of 1 from S_a, it follows that $||p_1|| = 1$. In a like manner,

the second state A_2 in the transition path can be represented as $p_2 \oplus S_a = A_2$. p_2 is also an n-tuple, but has a weight of 2 since A_2 is a distance 2 from S_a. In general, the i-th state A_i in the transition path can be represented as $p_i \oplus S_a = A_i$. If the distance between the initial and final states is m, then the last unit transition can be expressed as $p_m \oplus S_a$. The states of the transition path would be

$$S_a, p_1 \oplus S_a, p_2 \oplus S_a, \ldots, p_{m-1} \oplus S_a, p_m \oplus S_a.$$

To illustrate this idea, let S_a = 1001 and S_b = 0111, where S_a is the initial and S_b is the final state. Note that the first three state variables that code S_a must be excited to effect a transition to S_b. The transition path consists of four states, and for this example let them be (1001, 0001, 0011, 0111) = (S_a, A_1, A_2, S_b). The first state in the transition path is 0001 and can be expressed as $p_1 \oplus S_a$, where p_1 is 1000. In a like manner, $A_2 = p_2 \oplus S_a$, where p_2 = 1010 and $S_b = p_3 \oplus S_a$, where p_3 = 1110.

Several characteristics may be noted concerning the p_i's. It was noted above that $||p_1|| = 1$ and p_1 is an n-tuple that contains a single 1. Likewise, p_i is an n-tuple with i 1's, but can be represented as a sum of n-tuples

$$p_i = c_1 \oplus c_2 \oplus \ldots \oplus c_i,$$

where each c_j is an n-tuple with a weight of 1. To show that p_{i+1} represents a unit transition from p_i, consider $p_i \oplus p_{i+1}$

$$p_i \oplus p_{i+1} = (c_1 \oplus c_2 \oplus \ldots \oplus c_i) \oplus (c_1 \oplus c_2 \ldots \oplus c_{i+1}),$$

then by the associative and commutative laws, this becomes

$$(c_1 \oplus c_1) \oplus (c_2 \oplus c_2) \ldots (c_i \oplus c_i) \oplus c_{i+1} = c_{i+1},$$

which has a weight of 1. Therefore, $||p_i \oplus p_{i+1}|| = 1$, for all p_i.

Consider a set H to contain all the initial states of a column partition that is associated with a 2^m-row flow table. If the column partition contains only 2-sets, then there are 2^{m-1} states in H. For principal-column partitions, it is possible to represent the i-th state in the transition path of each state in H as $p_i \oplus h$, for each h in H. For those column partitions where the same set of state variables are not excited in all the transitions, it is not possible to represent the states in the transition path in this manner.

In the subsequent discussion, the bounds associated with Assignment 1 will be obtained, and a proof showing that all maximum-distance principal-column partitions are successfully encoded by Assignment 1.

For maximum-distance principal-column partitions, the i-th state in the transition path for each state in the initial set H can be represented as $p_i \oplus h$ for all $h \in H$. The entire set of transition paths can be represented as

$$H, p_1 \oplus H, p_2 \oplus H, \ldots, p_i \oplus H, \ldots, p_m \oplus H,$$

where $p_j \oplus H$ means $p_j \oplus h$ for all $h \in H$. This allows all the transition paths to be found in a parallel fashion.

The set of initial states is H and the set of final states must be $p_m \oplus H$; each set contains 2^{m-1} states. Each $p_j \oplus H$, $j < m$, must not contain any states of G. Since the transition paths cannot contain states of G (other than initial and final states), each $p_j \oplus H \notin G$, $j < m$. Therefore, each $p_j \oplus H$ must

be an n-tuple of the n-cube, that is not a state of G, $n = m + [\log_2 m]$. The elements of the n-cube possess the group properties.

Lemma 2-3: The set of all n-tuples of an n-cube N and the operation of mod 2 addition constitute a group of order 2^n.

Proof: Since all n-tuples of an n-cube, 2^n of them, are in N, including the identity element, the closure property of groups is assured. By theorem 2-1, N and the operation of mod 2 addition form a group. ##

The state assignment is a group G and since the elements of G are a subset of the elements of N, G is a subgroup of N.

If the set of initial states H has the properties of a group (i.e. a subgroup of G), then the i-th set of transition paths, $p_i \oplus H$, can be called a coset of the group N with respect to subgroup H [10]. Each state of a transition path is an element of some coset of N with respect to subgroup H, and the set of transition paths can be considered as a set of cosets. The following theorems are proven in many abstract algebra or modern algebra textbooks and, therefore, are just stated [10,11].

Theorem 2-5a: The number of elements in any two cosets of group N with respect to subgroup H is the same.

Theorem 2-5b: Any two cosets of group N with respect to subgroup H are either identical or disjoint.

Theorem 2-5c: The number of distinct cosets of group N with respect to subgroup H divides the order of N such that the number of distinct cosets = Ord(N)/Ord(H).

Definition: The set of all states in their respective transition path a distance i from the initial states are said to belong to the i-th level of the transition paths. This corresponds to the set of states that result after each state in the initial set of states has experienced i unit transitions.

From theorem 2-5b, each level of the transition paths must be a member of a different coset to insure that the intersection of any two transition paths is the null set. The number of distinct cosets needed then is equal to the number of levels needed for the transition paths. Therefore, the maximum number of cosets needed must be equal to or greater than the maximum number of states in a transition path. The maximum number of states in a transition path correspond to those in a maximum-distance column partition. Theorem 2-6 shows how the bound associated with Assignment 1 was arrived at.

Theorem 2-6: If a set of initial states H form a subgroup of order 2^{m-1} of a group N of order 2^n, then the minimum value of n to provide enough distinct cosets for a maximum-distance column partition is $n = m + [\log_2 m]$.

Proof: The order of the transition path of a maximum-distance column partition is m + 1. Let δ denote the number of distinct cosets of the group N with respect to subgroup H. From theorem 2-5c

$$\delta \cdot 2^{m-1} = 2^n,$$

or $$\delta = 2^{n-m+1}.$$

However, δ must be greater than or equal to m + 1, the number of states in a maximum-distance transition path, therefore,

$$m + 1 \leq 2^{n-m+1}.$$

This can be rewritten as

$$\log_2(m + 1) \leq n - m + 1,$$

or $$n \geq m - 1 + \log_2(m + 1).$$

Since n must be an integer,

$$n \geq m - 1 + [[\log_2(m + 1)]],$$

where [[ ]] means "next largest integer." The minimum value of n is

$$n = m - 1 + [[\log_2(m + 1)]].$$

It can be easily verified that for all m,

$$[[\log_2(m + 1)]] - 1 = [\log_2 m],$$

where [ ] means "next lowest integer." Therefore, the minimum value of n is

$$n = m + [\log_2 m]. \quad \#\#$$

The set of transition paths can be obtained from the cosets of the group N with respect to H. The set of transition paths can be represented as

$$H, p_1 \oplus H, p_2 \oplus H, \ldots, p_i \oplus H, \ldots, p_m \oplus H$$

for maximum-distance principal-column partitions. There are two conditions that must be placed on each p_i to insure a valid set of transition paths. A valid transition path cannot share any states with another transition path (sometimes called "cross-over") and cannot contain any states of G in the path, other than the initial and final states. Each transition path can be described as follows:

$$h, p_1 \oplus h, p_2 \oplus h, \ldots, p_i \oplus h, \ldots, p_m \oplus h,$$

where h is an initial state in H and $p_m \oplus h$ is the final state

in the 2-set. To insure that a transition path does not contain a state of G, other than h and $p_m \oplus h$, the following condition must be met:

$$p_j \oplus h \notin G, \quad j < m.$$

Since $h \in G$,

$$p_j \notin G, \quad j < m.$$

Transition paths will have common states if two cosets are equal, or if $p_i \oplus H = p_j \oplus H$. This condition never occurs if

$$p_i \oplus p_j \notin H.$$

In summary, the two conditions placed on the p_i's in order to produce a valid set of transition paths are:

1) $p_i \notin G, \quad i < m$

2) $p_i \oplus p_j \notin H$, for all i and j, $i \neq j$.

It should be noted that the p_i's are a function of the subgroup H, which is the set of initial states.

At this point it will be shown that for each maximum-distance principal-column partition a set of p_i's exist to obtain a set of transition paths. In a principal-column partition, the same set of state variables are excited to effect a transition from the initial to final state of each 2-set.

Definition: If S_a and S_b are states of a 2-set, then $\underline{S_m} = S_a \oplus S_b$.

From the closure property of groups, it can be seen that S_m must be an element of G since $S_m = S_a \oplus S_b$. Also since, $p_m \oplus S_a = S_b$ and $S_m \oplus S_a = S_b$, it follows that $p_m = S_m$.

Let $S_a = (a_1, a_2, \ldots, a_n)$ and $S_b = (b_1, b_2, \ldots, b_n)$, both being in the same 2-set and each coded with state variables

$y_1, y_2, \ldots, y_n$. Then

$$S_m = S_a \oplus S_b = (a_1, a_2, \ldots, a_n) \oplus (b_1, b_2, \ldots, b_n) = (a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n).$$

If a_i differs from b_i, then $a_i \oplus b_i = 1$, and if they are the same, $a_i \oplus b_i = 0$. In general, S_m has 1's in those bit positions where a_i and b_i differ and 0's where a_i and b_i are the same. If a_i and b_i differ, then in effecting a transition from S_a to S_b, state variable y_i must be excited. Therefore, S_m indicates those state variables which are to be excited in a transition between the states of a 2-set by the presence of 1's in those bit positions.

Let S_i and S_j be elements of a 2-set in a maximum-distance principal-column partition. It follows from the definition of a principal-column partition that S_m is the same for all S_i and S_j in the column partition, $S_i \oplus S_j = S_m$. Furthermore, since the distance between the elements of each 2-set is m for a maximum-distance column partition of a 2^m-row flow table, the weight of S_m is m. For example, a maximum-distance principal-column partition and the corresponding state assignment for a 2^4-row flow table is shown in Figure 2-6. Note that S_m = 111100 and $S_m = S_i \oplus S_j$, for all S_i and S_j which are members of the same 2-set.

	y_1	y_2	y_3	y_4	y_5	y_6
1	0	0	0	0	0	0
2	0	0	0	1	0	1
3	0	0	1	0	0	1
4	0	0	1	1	0	0
5	0	1	0	0	1	0
6	0	1	0	1	1	1
7	0	1	1	0	1	1
8	0	1	1	1	1	0
9	1	0	0	0	1	0
10	1	0	0	1	1	1
11	1	0	1	0	1	1
12	1	0	1	1	1	0
13	1	1	0	0	0	0
14	1	1	0	1	0	1
15	1	1	1	0	0	1
16	1	1	1	1	0	0

α = {1,16; 2,15; 3,14; 4,13; 5,12; 6,11; 7,10; 8,9}

FIGURE 2-6. Principal-column partition for a 2^4-row flow table

At this point, a particular maximum-distance principal-column partition will be considered. This column partition has an S_m, denoted as S_{mp}, where the first m bit positions are 1 and the m+1 through n bit positions are 0. This means that state variables $y_1, y_2, \ldots, y_m$ are to be excited during the transition between the states of all 2-sets. For a 2^4-row flow table, this S_m is 111100 and the principal-column partition is shown in Figure 2-6. After the p_i's have been determined for this type of maximum-distance principal-column partition, a mapping of this case onto other maximum-distance principal-column partitions will be established.

Since the p_i's depend on the subgroup H, it is important to determine the characteristics of H and the method for generating this subgroup. The set of elements that compose H are to represent all the initial states of a column partition. Certainly then, H must not contain both elements of a 2-set. If the S_m of a maximum-distance principal-column partition is included in H, then from the closure property of groups, there exist a pair of states S_a and S_b in H such that $S_a \oplus S_b = S_m$. Clearly, this implies that S_a and S_b, which are in the same 2-set, are both in H; therefore, S_m cannot be a member of H.

Another point, not quite so obvious, is that states with a weight of 1 cannot be in H. For m-even there are no states with a weight of 1 in G, and therefore, none can ever appear in H either. For m odd, there is one state in G that has a weight of 1. Since state variable y_m is not in a parity set, there is a state where $y_m = 1$ and all the other state variables are 0.

To see where the problem arises, one must consider the structure of the p_i's. From previous discussions, it was shown that $||p_i \oplus p_{i+1}|| = 1$, for all i. S_m is a maximum weighted code word of G, and from theorem 2-4, $y_m = 1$ in each maximum-weighted code word. This implies that in each maximum-distance column partition, state variable y_m is excited in a transition between the states of each 2-set. Then someplace in every transition path, y_m will change state; let $p_{i+1} \oplus H$ represent the set of states of the transition path just after y_m has changed state. Then $p_{i+1} \oplus p_i$ is an n-tuple with a weight of 1 where $y_m = 1$ and all other state variables are 0.

This n-tuple is a state of G. Therefore, a state of weight 1 cannot appear in the subgroup H if the conditions on the p_i's are to be met for maximum-distance principal-column partitions.

For less than maximum-distance principal-column partitions where y_m is not excited in the transitions, $p_{i+1} \oplus p_i$ can never equal the n-tuple with a weight of 1 where $y_m = 1$. Therefore, for those cases where y_m is not excited, the restriction of not allowing a state of weight 1 in H can be removed.

In summary, the two conditions placed on H, the subgroup containing all the initial states of a principal-column partition are:

1) S_m for the column partition cannot be in H.

2) No state with a weight of 1 can be in H when m is odd and y_m is excited to effect the transitions.

The algorithm for generating H for the maximum-distance principal-column partition with an S_{mp}, which has 1's in the first m bit positions and 0's in the rest is stated as follows:

Step 1. With m - 1 state variables $y_1, y_2, \cdots, y_{m-1}$ generate 2^{m-1} unique code words. This is all possible combinations of the m - 1 state variables. Let state variable $y_m = 0$ in all the code words.

Step 2. Generate the parity sets for each state in an identical manner used in generating the total assignment.

The order of H must be 2^{m-1} for a 2^m-row flow table, and obviously the above algorithm produces the proper order for H. By the same means to show that G is a group (Theorem 2-2), H is also a group and since each element of H is contained in G,

H is a subgroup of G.

Since the value of state variable y_m in the code for S_{mp} is 1 and the value of y_m in all the states of H is 0, S_{mp} cannot appear in H. The only assignments which have a state in G with a weight of 1 are those when m is odd. The state with a weight of 1 has state variable $y_m = 1$ and all other state variables are 0. Since $y_m = 0$ in all the states of H, no state of weight 1 can appear in H. Therefore, the above procedure for generating H meets the two specified conditions.

In Figure 2-7 is the subgroup H for the assignment associated with the 2^4-row flow table and column partition shown in Figure 2-6.

	y_1	y_2	y_3	y_4	y_5	y_6
1	0	0	0	0	0	0
3	0	0	1	0	0	1
5	0	1	0	0	1	0
7	0	1	1	0	1	1
9	1	0	0	0	1	0
11	1	0	1	0	1	1
13	1	1	0	0	0	0
15	1	1	1	0	0	1

FIGURE 2-7. Subgroup H for 2^4-row flow table and column partition of Figure 2-6

In Figure 2-8 is the subgroup for the assignment associated with a maximum-distance principal-column partition of a 2^5-row flow table. The above algorithm is used to obtain this subgroup.

Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	0
0	1	0	0	0	1	0
0	1	0	1	0	1	1
0	1	1	0	0	1	1
0	1	1	1	0	1	0
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	0	1	1
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	0	0

FIGURE 2-8. Subgroup H for a principal-column partition of a 2^5-row flow table.

Note that $y_m = 0$ in all the states of each H above, thereby making it impossible to have S_{mp} or a state of weight 1 appear in H.

From the constructed subgroup H, a set of p_i's can be obtained. A satisfactory set of p_i's needed to generate distinct cosets of the group G with respect to the subgoup H must meet two conditions:

1) $p_i \notin G$, $i < m$

2) $p_i \oplus p_j \notin H$, for all i and j, $i \neq j$.

Theorem 2-7: If each p_i has odd parity over at least one of the parity sets and does not have odd parity over the same parity sets as some other p_j, $i \neq j$, then both of the above

conditions are met.

Proof: Since each state of G has even parity over the parity sets, if each p_i has odd parity in at least one parity set, the $p_i \notin G$. If p_i has odd parity in the same parity sets as some p_j, then $p_i \oplus p_j$ has even parity in all parity sets. However, if p_i and p_j do not have odd parity in the same parity sets, the $p_i \oplus p_j$ would have odd parity in at least one parity set and therefore cannot be in H. ##

From theorem 2-7, if the parity combination over the parity sets of two n-tuples, n_1 and n_2, are different, then the codes for n_1 and n_2 cannot be the same. For example, assume there are three parity sets in n_1 and n_2, and let 1 represent odd parity and 0 represent even parity in a parity set. If the parity combination for each n-tuple is described as follows:

	Parity Set		
	1	2	3
n_1:	0	1	1
n_2:	1	0	1

The codes for n_1 and n_2 must be different. It should be pointed out that n-tuples with the same parity combination do not necessarily have the same code.

Since the number of distinct cosets needed for a maximum-distance column partition is m, m p_i's are needed. Since $p_m = S_m$, a state of G, m - 1 p_i's must have odd parity. Then to meet the conditions of theorem 2-7, at least m - 1 odd parity combinations of the parity sets are needed. Since there are $[\log_2 m]$

parity sets in each state of G, there are $2^{[\log_2 m]}-1$ different odd parity combinations available. For some values of m, there are not enough odd parity combinations available. For example, when m = 14, there are $2^{[\log_2 14]} - 1 = 7$ different odd parity combinations and 13 odd parity p_i's are needed. Let $\lambda = 2^{[\log_2 m]}$. To solve this problem for those values of m where $m > \lambda$, one first generates $\lambda - 1$ p_i's with different odd parity combinations with $y_m = 0$ in all p_i's. To obtain p_λ, one lets $y_m = 1$ in $p_{\lambda-1}$. Since there are no states in H where $y_m = 1$, $p_\lambda \oplus p_i$, $i = 1,2, \ldots,\lambda-1$, cannot be in H. The rest of the p_i's, $i = \lambda+1, \ldots,m$, are generated in such a manner that each p_i and p_j, $i,j > \lambda$ has a different odd parity combination.

There are two questions that still must be answered in using this procedure:

1) For $i < \lambda$ and $j > \lambda$, does there exist an i and a j such that $p_i \oplus p_j \in H$.

2) Are there enough p_i's available for maximum distance transitions.

In those p_i's where $i < \lambda$ $y_m = 0$, and in those p_j's where $j > \lambda$ $y_m = 1$, therefore $p_i \oplus p_j$ is an n-tuple with $y_m = 1$ and is not in H. In the algorithm above, a maximum of $2\cdot\lambda$ or $2^{[\log_2 m] + 1}$ p_i's are available, and since $2^{[\log_2 m] + 1}$ is always greater than m, enough p_i's can be generated for maximum-distance column partitions.

Following is a general statement of a procedure to obtain a satisfactory set of p_i's: If $\lambda \geq m$, determine m different parity combinations for the p_i's. If $\lambda < m$, first generate

$\lambda - 1$ p_i's with different odd parity combinations keeping $y_m = 0$ in each p_i; p_λ will be $p_{\lambda-1}$ with $y_m = 1$.. Generate each p_j, $j = \lambda + 1, \ldots, m$, such that they exhibit a different parity combination.

Theorem 2-8: A satisfactory set of p_i's exist for a maximum-distance principal-column partition for 2^m-row flow tables.

Proof: The proof is included in the above discussion. ##

To illustrate the above procedure, consider a maximum-distance principal-column partition of a 2^4-row flow table with S_{mp} = 111100 of Figure 2-6. The parity sets are$\{y_1, y_2, y_5\}$ and $\{y_3, y_4, y_6\}$. A satisfactory set of p_i's are:

	y_1	y_2	y_3	y_4	y_5	y_6
p_1 =	1	0	0	0	0	0
p_2 =	1	0	1	0	0	0
p_3 =	1	1	1	0	0	0
p_4 =	1	1	1	1	0	0

The corresponding transition paths are shown in Figure 2-9.

H	$p_1 \oplus$ H	$p_2 \oplus$ H	$p_3 \oplus$ H	$p_4 \oplus$ H
0 0 0 0 0 0	1 0 0 0 0 0	1 0 1 0 0 0	1 1 1 0 0 0	1 1 1 1 0 0
0 0 1 0 0 1	1 0 1 0 0 1	1 0 0 0 0 1	1 1 0 0 0 1	1 1 0 1 0 1
0 1 0 0 1 0	1 1 0 0 1 0	1 1 1 0 1 0	1 0 1 0 1 0	1 0 1 1 1 0
0 1 1 0 1 1	1 1 1 0 1 1	1 1 0 0 1 1	1 0 0 0 1 1	1 0 0 1 1 1
1 0 0 0 1 0	0 0 0 0 1 0	0 0 1 0 1 0	0 1 1 0 1 0	0 1 1 1 1 0
1 0 1 0 1 1	0 0 1 0 1 1	0 0 0 0 1 1	0 1 0 0 1 1	0 1 0 1 1 1
1 1 0 0 0 0	0 1 0 0 0 0	0 1 1 0 0 0	0 0 1 0 0 0	0 0 1 1 0 0
1 1 1 0 0 1	0 1 1 0 0 1	0 1 0 0 0 1	0 0 0 0 0 1	0 0 0 1 0 1

FIGURE 2-9. Transition paths for a maximum-distance principal-column partition from Figure 2-6

The p_i's for a maximum-distance principal-column partition of a 2^6-row flow table with S_{mp} = 11111100 are shown next. The parity sets are $\{y_1, y_2, y_7\}$ and $\{y_3, y_4, y_5, y_6, y_8\}$. A satisfactory set of p_i's are:

	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
p_1 =	0	0	1	0	0	0	0	0
p_2 =	1	0	1	0	0	0	0	0
p_3 =	1	0	1	1	0	0	0	0
p_4 =	1	0	1	1	0	1	0	0
p_5 =	1	1	1	1	0	1	0	0
p_6 =	1	1	1	1	1	1	0	0

The p_i's for a maximum-distance principal-column partition with S_{mp} of a 2^{13}-row flow table are shown next. The purpose is to illustrate the construction of a set of p_i's for a relatively large flow table. The parity sets are $\{y_1, y_2, y_3, y_4, y_{14}\}$, $\{y_5, y_6, y_7, y_8, y_{15}\}$, and $\{y_9, y_{10}, y_{11}, y_{12}, y_{16}\}$.

	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}
p_1 =	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p_2 =	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
p_3 =	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
p_4 =	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0
p_5 =	1	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0
p_6 =	1	1	1	0	1	1	0	0	1	0	0	0	0	0	0	0
p_7 =	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0
p_8 =	1	1	1	0	1	1	1	0	1	0	0	0	1	0	0	0
p_9 =	1	1	1	0	1	1	1	0	1	1	0	0	1	0	0	0
p_{10} =	1	1	1	1	1	1	1	0	1	1	0	0	1	0	0	0
p_{11} =	1	1	1	1	1	1	1	0	1	1	1	0	1	0	0	0
p_{12} =	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0
p_{13} =	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0

It has been shown that a satisfactory set of transition paths exist for maximum-distance principal-column partitions with S_{mp}. The proof showing the existence of a similar set of transition paths for any other S_m of a maximum-distance principal-column partition is identical to that for the one with S_{mp}. It is concluded then that Assignment 1 satisfactorily encodes all maximum-distance principal-column partitions.

It is possible to find a subgroup H' and associated p_i's for any S_m of a maximum-distance principal-column partition from the subgroup H and associated p_i's of a maximum-distance principal-column partition with S_{mp}. A mapping does exist between these two cases. For the example of Figure 2-6 with S_m = 111100, the question is how to map this case onto the case where S_m = 101011 for instance. In the first case, state variables y_1, y_2, y_3, and y_4 are excited to effect a transition between the states of each 2-set and in the latter case, y_1, y_3, y_5, and y_6 are excited. Note that still two of the three state variables of each parity set are excited in a maximum-distance transition. Almost obvious, the proposed mapping is a simple permutation of the state variables coding the parity sets.

If there are k state variables in a parity set, then in a maximum-distance transition k - 1 of these state variables are excited. This means that only one state variable from each parity set is not excited in a maximum-distance transition. In the case already presented, the state variable that was not excited in each parity set is the dependent state variable.

In the case where the dependent state variable of a parity set is to be excited in a maximum-distance transition, the mapping is a permutation of the dependent state variable with the independent state variable that is not excited. The following theorem shows that after a permutation of this type, the resulting n-tuple is a state of G.

Theorem 2-9: The permutation of any two bits within a parity set of a state of G will produce a state of G.

Proof: Let a parity set consist of bits $\{a_1, a_2 \ldots, a_k, a_p\}$ which are values of the state variables $\{y_1, y_2, \ldots, y_k, y_p\}$. Since $y_1, y_2, \ldots, y_k$ are the independent state variables, $a_1, a_2, \ldots, a_k$ can assume 2^k possible values in G. The relationship between the bits of the parity set is

$$a_1 \oplus a_2 \oplus \ldots \oplus a_j \oplus \ldots \oplus a_k = a_p.$$

Suppose that a_j was permuted with a_p. The parity set would be $\{a_1, a_2, \ldots, a_p, \ldots, a_k, a_j\}$. This parity set is one of the 2^k parity sets if

$$a_1 \oplus a_2 \oplus \ldots \oplus a_p \oplus \ldots \oplus a_k = a_j.$$

This must hold since $\{0,1\}$ and $\oplus$ form an abelian group where each element $a_i \in \{0,1\}$ is its own inverse. The parity set obtained after the permutation is one of the 2^k parity sets and therefore, the resulting n-tuple is a state of G. ##

To obtain the subgroup H' of any maximum-distance principal-column partition from the subgroup H that already has been generated for S_{mp}, those dependent state variables that are excited are permuted with the independent state variable that is not excited in each parity set. The total number of

permutations is equal to or less than $[\log_2 m]$, the number of parity sets. For example, consider the subgroup of Figure 2-7 with the corresponding S_m = 111100. To obtain the subgroup H' for S_m = 101011, simply permute state variables y_2 with y_5 and y_4 with y_6. This is shown in Figure 2-10.

Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	0	0	0
1	1	1	0	0	1

Subgroup H

Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
0	0	0	0	0	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	1	1	1	0
1	1	0	0	0	0
1	1	1	1	0	0
1	0	0	0	1	0
1	0	1	1	1	0

Subgroup H'

FIGURE 2-10. Comparison of H with S_m= 111100 and H' with S_m = 101011

Simple permutation of the state variables within a parity set does not affect the group properties since the set is still closed under mod 2 addition. Also under this type of permutation, there are no states with a weight of 1 in H', and furthermore S_m is not a member of H'; therefore, H' satisfies the conditions for a subgroup that contains a set of initial states.

In a similar manner, the set of p_i's for the subgroup H' are obtained from the p_i's for H by an identical permutation . This is illustrated in the following example where the p_i's for

H' of Figure 2-10 are obtained from the p_i's of H by permuting y_2 with y_5 and y_4 with y_6.

	y_1	y_2	y_3	y_4	y_5	y_6
p_1 =	1	0	0	0	0	0
p_2 =	1	0	1	0	0	0
p_3 =	1	1	1	0	0	0
p_4 =	1	1	1	1	0	0

p_i's for H

	y_1	y_2	y_3	y_4	y_5	y_6
p_1 =	1	0	0	0	0	0
p_2 =	1	0	1	0	0	0
p_3 =	1	0	1	0	1	0
p_4 =	1	0	1	0	1	1

p_i's for H'

FIGURE 2-11. Comparison of p_i's for H with p_i's for H'

The transition paths for the maximum-distance principal-column partition with S_m = 101011 are obtained directly from the cosets.

From the above discussions, a satisfactory set of transition paths can be obtained for any maximum-distance principal-column partition. Each of these transition paths is minimum length. There are some less than maximum distance principal-column partitions where the transition paths cannot be minimum length. To demonstrate this, consider the assignment for a 2^6-row flow table where the parity sets are $\{y_1, y_2, y_7\}$ and $\{y_3, y_4, y_5, y_6, y_8\}$ (called parity set 1 and 2 respectively). A ML transition is not possible between states 00000000 and 00111100. Note that in making this transition, only state variables that code parity set 2 must be excited. Let 00000000 be the initial state and assume that state variable y_3 is excited to obtain the first state A_1 in the transition path. Thus, A_1 = 00100000. Since there is odd parity in parity set 2,

A_1 is not a state of G. If a ML transition path is desired, state variable y_4, y_5, or y_6 must be excited to obtain the second state A_2 in the transition path. However, if any one of these state variables is excited to obtain A_2, even parity would result in parity set 2 and A_2 would be a state in G which does not represent a satisfactory transition path. To insure that there is odd parity in at least one parity set of each state in the transition path, it will be necessary to obtain A_2 by exciting a state variable from parity set 1. This results in a transition path that is not minimum length.

The above situation occurs whenever four or more state variables from one parity set and no state variables from another parity set must excited in making a transition. All state assignments for flow tables larger than 2^5-rows possess this characteristic. The transition path associated with this type transition cannot be minimum length.

To illustrate a procedure for finding the transition paths associated with a principal-column partition where the transition paths cannot be minimum length, consider the assignment for a 2^6-row flow table where S_m= 00111100; state variables y_3, y_4, y_5, and y_6 must be excited in effecting each transition between the states of the 2-sets. If S_i and S_j are states of the same 2-set, then $S_i \oplus S_j = S_m$. As stated before, the subgroup H must not contain S_m if it is to represent a set of initial states. For this example, H can be obtained by generating 2^5 code words with state variables y_1, y_2, y_3, y_4, and y_5. Then letting y_6 = 0 in all the code words and generating the parity sets will

complete the codes for the states of H. Since $y_6 = 0$ in all the states of H, S_m cannot be in H, and therefore, H is a satisfactory set of initial states. A satisfactory set of p_i's are:

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8
$p_1 =$	0	0	1	0	0	0	0	0
$p_2 =$	0	1	1	0	0	0	0	0
$p_3 =$	0	1	1	1	0	0	0	0
$p_4 =$	0	1	1	1	0	1	0	0
$p_5 =$	0	0	1	1	0	1	0	0
$p_6 =$	0	0	1	1	1	1	0	0

Since each p_i has odd parity in at least one parity set, $p_i \notin G$. Furthermore, it can be seen that each $p_i \oplus p_j \notin H$. This meets the conditions placed on the p_i's. Note that there are seven states in each transition path, including the initial and final states. This results from exciting state variable y_2 twice during the total transition. The complete set of transition paths are represented by

$$H,\ p_1 \oplus H,\ p_2 \oplus H,\ p_3 \oplus H,\ p_4 \oplus H,\ p_5 \oplus H,\ p_6 \oplus H.$$

Since $p_i \oplus p_j \notin H$, $i,j \in \{1,2,3,4,5,6\}$, seven distinct cosets (including H) are obtained and the set of transition paths is satisfactory.

The number of p_i's that can be generated is $2^{[\log_2 m]} + 1$. For maximum-distance principal-column partitions, the number of p_i's available is always equal to or greater than m. All maximum-distance transitions can result in ML transition paths. For those transitions less than maximum distance which

cannot produce ML transition paths, it appears that the number of p_i's needed is less than or equal to m. Even though a general proof is lacking, an extensive search has been made of principal-column partitions for flow tables up to and including 2^{25}-rows, and the number of p_i's needed was always less than or equal to m. Due to the size of the problems, a less extensive search has been made for flow tables larger than 2^{25}-rows, but the many flow tables checked confirm that the number of p_i's needed is less than or equal to m. Therefore, it seems safe to conclude that there always are enough p_i's available to generate distinct cosets to yield a satisfactory set of transition paths for those cases where ML transition paths are not possible.

For less than maximum-distance principal-column partitions where the transitions can result in ML transition paths, the number of p_i's needed is less than m. Since $2^{[\log_2 m] + 1}$ is greater than or equal to m, there are enough p_i's available to generate distinct cosets to yield a satisfactory set of transition paths, and therefore, Assignment 1 satisfactorily encodes this type of principal-column partition also. With this, it seems safe to conclude that all principal-column partitions are satisfactorily encoded with Assignment 1.

The procedure for obtaining a set of p_i's for a principal-column partition (less than maximum distance) is identical to the procedure used in the maximum distance case. The only exception would occur for those cases where ML transition paths are not possible, in which case, odd parity is produced

in parity sets where state variables originally were not to be excited.

The generation of a suitable subgroup of initial states for any principal-column partition will be discussed in detail in the subsequent chapter.

III. GENERALIZED STATE ASSIGNMENT 2

A. Characteristics of Assignment 2

Generalized State Assignment 2 (hereafter called Assignment 2) requires m + [m/2] state variables to encode a 2^m-row flow table, where [ ] means the "next lowest integer." Like Assignment 1, it has the distinct advantage in being easy to generate. An advantage Assignment 2 has in comparison to Assignment 1 is that all transitions are minimum length. Assignment 2 requires more state variables than Assignment 1 for flow tables greater than 2^5 rows. In Figure 3-1 there is a comparison of the bounds associated with Huffman's intermeshed row-set assignment, Assignment 1, and Assignment 2 for various 2^m-row flow tables.

m	Huffman's	Assignment 1	Assignment 2
2	3	3	3
3	5	4	4
4	7	6	6
5	9	7	7
6	11	8	9
7	13	9	10
8	15	11	12
.	.	.	.
.	.	.	.
.	.	.	.
15	29	17	22
.	.	.	.
.	.	.	.
.	.	.	.
m	2m-1	$m+[\log_2 m]$	m+[m/2]

FIGURE 3-1. Comparison of the bounds associated with Huffman's assignment, Assignment 1 and Assignment 2

The assignment algorithm for Assignment 2 is very similar to that of Assignment 1. The only difference is that each parity set is restricted to only three state variables - two independent and one dependent. Following is an algorithm for generating Assignment 2 for a 2^m-row flow table:

Step 1. Form all possible 2^m code words with m binary independent state variables. Denote these as state variables $y_1, y_2, \ldots, y_m$.

Step 2. The final code will have m + [m/2] state variables. To determine $y_{m+1}, y_{m+2}, \ldots, y_n$, n = m +[m/2], divide the set of independent state variables $\{y_1, y_2, \ldots, y_m\}$ into [m/2] subsets, each subset containing two state variables y_{2i-1}, y_{2i}, where i = 1,2, . . .,[m/2]. The state variables in each of these subsets is considered to contain the independent state variables of a parity set; and each of the state variables $y_{m+1}, y_{m+2}, \ldots, y_n$ is considered as the dependent variable to one of the [m/2] subsets. Therefore, the independent state variables in each of the [m/2] subsets generates one of the state variables $y_{m+1}, y_{m+2}, \ldots, y_n$. If m is even, every state variable is in a parity set; if m is odd, y_m is the only state variable that is not an element of some parity set.

Since [m/2] = [$\log_2 m$] for m = 3, 4, and 5, Assignment 1 and Assignment 2 are identical for these flow tables. Figure 2-3 and Figure 2-4 show the codes for 2^4 and 2^5-row flow tables respectively. To illustrate the assignment

procedure in a situation where it differs from Assignment 1, consider the code to be generated for a 2^6-row flow table. The independent state variables are $\{y_1,y_2,y_3,y_4,y_5,y_6\}$ and are used to generate 2^6 unique code words. Since [m/2] = 3 for a 2^6-row flow table, there are three parity sets in this state assignment, $\{y_1,y_2,y_7\}$, $\{y_3,y_4,y_8\}$, and $\{y_5,y_6y_9\}$, where $y_1 \oplus y_2 = y_7$, $y_3 \oplus y_4 = y_8$, and $y_5 \oplus y_6 = y_9$. With this information the state assignment can easily be generated.

It can be shown that the 2^m states encoded by Assignment 2 and the operation of mod 2 addition form a group. The proof is identical to that of theorem 2-2, except each parity set in Assignment 2 contains only three state variables. Let the group formed by Assignment 2 be denoted hereafter as G*.

Even though Assignment 2 has more state variables associated with it for large values of m, it still has the same maximum distance characteristic as Assignment 1.

Theorem 3-1: The maximum distance between any pair of states in the state assignment obtained from Assignment 2 is m for a 2^m-row flow table.

Proof: The set of independent state variables is $\{y_1,y_2, \ldots,y_m\}$. For m even, each of the independent state variables is in one and only one parity set. Then, from lemma 2-2 and the proof of theorem 2-4, the maximum weight is m. For m odd, each of the independent state variables, except y_m, is in one and only one parity set and y_m is not in any parity

set. Then the maximum weight of a state is m - 1 plus the maximum weight of y_m which is 1; the maximum weight then is m. Therefore, the maximum weight of a state is m and from theorem 2-3, the maximum distance is also m. ##

Since the distance between two states is equal to the number of state variables that must be excited in effecting a transition between the states, the number of state variables needed to be excited in effecting a transition between states that are a maximum distance apart is the same for Assignment 1 and Assignment 2. For both assignments, the number of states in a ML transition path for maximum-distance states is the same.

From theorem 3-1 and theorem 2-4, if a state variable is in at most one parity set, the maximum distance between code words is m. It can be shown that if a state variable is in more than one parity set, then the maximum distance is greater than m. Since the smallest parity set contains three state variables, while allowing a state variable to be in at most one parity set, the maximum number of these parity sets that can be obtained from m independent state variables is [m/2]. Since each parity set contains a generated or dependent state variable and Assignment 2 contains a maximum number of parity sets, then Assignment 2 generates the largest n-tuples (largest values of n) of any group code with the maximum distance being m for 2^m code words. A group code, code words and an operation that constitute a group, is sometimes referred to as a linear code [12]. Assignment 1 and Assignment 2

produce linear codes that can be called maximum-distance codes. This appears to be the dual of the problem associated with producing error-detecting/error-correcting linear codes with a minimum-distance characteristic.

B. Principal-Column Partitions

Most of the ideas developed for Assignment 1 concerning principal-column partitions can be applied directly to Assignment 2. In the treatment of Assignment 2, specific algorithms are given for the generation of the subgroup H of initial states and the associated p_i's for any principal-column partition, Assignment 2 will be shown to satisfactorily encode all principal-column partitions.

If A_i is the i-th state of a transition path and S_a is the initial state ($A_o = S_a$), then A_i will be represented as $p_i \oplus S_a$. The p_i's are used in an identical manner in finding the transition paths of principal-column partitions for states of Assignment 2 as they were in Assignment 1. The set of transition paths are represented as

$$H,\ p_1 \oplus H,\ \ldots,p_m \oplus H,$$

where H is the set of initial states.

In an effort to present a specific algorithm for obtaining the subgroup H for any maximum-distance principal-column partition, the following theorem will be utilized.

Theorem 3-2: An even number of (zero or two) state variables in each parity set must be excited to effect a transition between any two states of G*.

Proof: Since every state of G* possesses even parity over each parity set, an even number of state variables must be excited in each parity set to effect a transition between any two states of G*. Since the number of state variables in each parity is three, there are either no state variables or two state variables from each parity set that must be excited to effect a transition between any pair of states. ##

Let H represent a set of initial states. For principal-column partitions, H can constitute a subgroup of the group G *. The conditions placed on the subgroup H are:

1) S_m for the column partition cannot be in H

2) If m is odd with state variable y_m excited, then no state with a weight of 1 can be in H.

Two algorithms are given for obtaining the subgroup H, depending on whether m is even or odd. For m odd, the subgroup H for maximum-distance principal-column can be generated as follows:

Step 1. With m - 1 independent state variables $y_1, y_2, \ldots, y_{m-1}$ generate 2^{m-1} distinct code words. Let state variable $y_m = 0$ in all code words.

Step 2. Generate the parity sets to obtain the rest of the code for each state in H.

In maximum-distance transitions, state variable y_m must be excited and induce a 1 in the code for S_m. Since $y_m = 0$ in all the states of H, S_m cannot appear in H, and for the same reason there cannot be any states of weight 1 in H.

Therefore, both conditions placed on H are met.

For m even, only S_m for the maximum-distance principal-column partition cannot appear in H. In a maximum-distance transition, m state variables must be excited. Since there are three state variables in each parity set and from theorem 3-2, it follows that at least one independent state variable from each parity set must be excited in a maximum-distance transition. Let y_e be any one of the independent state variables that must be excited in every transition. Following is an algorithm for generating the subgroup H when m is even:

Step 1. Generate 2^{m-1} distinct code words with the set of m - 1 independent state variables $\{y_1, y_2, \ldots, y_m\}$, where y_e is not in the set. Let $y_e = 0$ in all code words.

Step 2. Generate the parity sets to obtain the rest of the code for each state in H.

State variable y_e induces a 1 in the code for S_m and since $y_e = 0$ in all states of H, S_m cannot be in H; and therefore, a suitable H is obtained with this algorithm. It should be pointed out that the above algorithm is not the only one that could be used to generate a suitable subgroup H. Any method is acceptable, as long as the set of initial states satisfy the two conditions stated earlier.

Examples of subgroups H for 2^4 and 2^5-row flow tables are shown in Figure 2-7 and Figure 2-8 respectively, where the independent state variables are the excited state variables. To illustrate the algorithm more specifically, consider a maximum-distance principal-column partition of a 2^6-row flow

table with S_m = 101110101. The parity sets are $\{y_1, y_2, y_7\}$,

Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9
0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1	0
0	0	0	1	0	1	0	1	1
0	0	1	0	0	0	0	1	0
0	0	1	0	0	1	0	1	1
0	0	1	1	0	0	0	0	0
0	0	1	1	0	1	0	0	1
0	1	0	0	0	0	1	0	0
0	1	0	0	0	1	1	0	1
0	1	0	1	0	0	1	1	0
0	1	0	1	0	1	1	1	1
0	1	1	0	0	0	1	1	0
0	1	1	0	0	1	1	1	1
0	1	1	1	0	0	1	0	0
0	1	1	1	0	1	1	0	1
1	0	0	0	0	0	1	0	0
1	0	0	0	0	1	1	0	1
1	0	0	1	0	0	1	1	0
1	0	0	1	0	1	1	1	1
1	0	1	0	0	0	1	1	0
1	0	1	0	0	1	1	1	1
1	0	1	1	0	0	1	0	0
1	0	1	1	0	1	1	0	1
1	1	0	0	0	0	0	0	0
1	1	0	0	0	1	0	0	1
1	1	0	1	0	0	0	1	0
1	1	0	1	0	1	0	1	1
1	1	1	0	0	0	0	1	0
1	1	1	0	0	1	0	1	1
1	1	1	1	0	0	0	0	0
1	1	1	1	0	1	0	0	1

FIGURE 3-2. Subgroup H for S_m = 101110101

$\{y_3, y_4, y_8\}$, and $\{y_5, y_6, y_9\}$; let them be denoted as parity set 1, 2, and 3 respectively. The excited state variables are y_1 and y_7 in parity set 1, y_3 and y_4 in parity set 2, and y_5 and y_9 in parity set 3. The 2^{m-1} states can be generated with the set of independent state variables $\{y_1, y_2, y_3, y_4, y_6\}$, letting $y_5 = 0$, and forming the parity sets. This subgroup is

shown in Figure 3-2. Note that S_m does not appear in the subgroup H since $y_e = y_5 = 0$. An equally valid subgroup could be formed by letting $y_e \in \{y_1, y_3, y_4\}$.

From the constructed subgroup H, a set of p_i's can be obtained. A satisfactory set of p_i's needed to generate distinct cosets for principal-column partitions must meet two conditions:

1) $p_i \notin G_1$, $i < m$

2) $p_i \oplus p_j \notin H$, for all i and j, $i \neq j$.

These are the same conditions placed on the p_i's for Assignment 1.

Obtaining a satisfactory set of p_i's for Assignment 2 is an easier task than that of Assignment 1. In Assignment 1, exciting y_m was a critical factor in larger flow tables. For Assignment 2, one can rely almost completely on theorem 2-7 for obtaining a satisfactory set of p_i's. Theorem 2-7 is restated below.

Theorem 2-7: If each p_i has odd parity in at least one of the parity sets and does not have odd parity in the same parity sets as some other p_j, $i \neq j$, then both conditions placed on the p_i's are met.

From chapter 2, the number of p_i's that can be generated for Assignment 1 is $2^{[\log_2 m] + 1}$, where the number of parity sets in the codes of Assignment 1 is $[\log_2 m]$. For Assignment 2, the number of parity sets if $[m/2]$. By the same arguments used to determine the number of p_i's for Assignment 1, the number of p_i's that can be generated for Assignment 2 is

$2^{[m/2]+1}$. The number of p_i's needed in a maximum-distance principal-column partition is m; since $[m/2] \geq [\log_2 m]$ and $2^{[\log_2 m]+1} \geq m$, then $2^{[m/2]+1} \geq m$ and there exist a set of p_i's to generate distinct cosets for each maximum-distance principal-column partition.

In an effort to present a specific algorithm for obtaining a set of p_i's that would apply for any maximum-distance principal-column partition, the following definition will be utilized.

Definition: A parity set is said to be excited (sometimes called an excited parity set) if at least two of the state variables that code the parity set must change state in a transition. Those state variables that code the excited parity set and must change state to effect the transition are called the excited state variables.

In maximum-distance column partitions there are [m/2] excited parity sets. The parity sets are to be numbered from 1 to k, k = [m/2]. From theorem 3-2, there are two excited state variables in each excited parity set. In excited parity set i, the excited state variables are to be labeled y_{i1} and y_{i2}, i = 1, 2, . . ., k. In the preceding example, the excited state variables are labeled y_{11}, y_{12}, y_{21}, y_{22}, y_{31}, and y_{32} with the following relationship:

$y_{11} = y_1$ $\qquad$ $y_{12} = y_6$

$y_{21} = y_3$ $\qquad$ $y_{22} = y_4$

$y_{31} = y_5$ $\qquad$ $y_{32} = y_9$.

Labeling the state variables in this manner allows a set of p_i's to be directly for any S_m. In Figure 3-3 is shown a set of p_i's for maximum-distance principal-column partitions where m is even. All unexcited state variables are 0 in the p_i's and are not shown in Figure 3-3.

	Excited Parity Sets								
	1		2		3		. . .	k	
	y_{11}	y_{12}	y_{21}	y_{22}	y_{31}	y_{32}	. . .	y_{k1}	y_{k2}
p_1 =	1	0	0	0	0	0	. . .	0	0
p_2 =	1	0	1	0	0	0	. . .	0	0
p_3 =	1	0	1	0	1	0	. . .	0	0
.	.	.	.	.	.	.		.	.
.	.	.	.	.	.	.		.	.
.	.	.	.	.	.	.		.	.
$p_{k/2}$ =	1	0	1	0	1	0	. . .	1	0
$p_{k/2 + 1}$ =	1	1	1	0	1	0	. . .	1	0
$p_{k/2 + 2}$ =	1	1	1	1	1	0	. . .	1	0
.	.	.	.	.	.	.		.	.
.	.	.	.	.	.	.		.	.
.	.	.	.	.	.	.		.	.
$p_{2k - 1}$ =	1	1	1	1	1	1	. . .	1	0
p_{2k} =	1	1	1	1	1	1	. . .	1	1

FIGURE 3-3. Set of p_i's for principal-column partitions when m is even, and when m is odd if y_m is not excited

For m odd, state variable y_m is not in any parity set. A set of p_i's is shown in Figure 3-4 for m odd with y_m excited

in each transition. All unexcited state variables are 0 in the p_i's and are not shown in Figure 3-4.

		Excited Parity Sets									
		1		2		3		. . .	k		
		y_{11}	y_{12}	y_{21}	y_{22}	y_{31}	y_{32}	. . .	y_{k1}	y_{k2}	y_m
p_1	=	1	0	0	0	0	0	. . .	0	0	0
p_2	=	1	0	1	0	0	0	. . .	0	0	0
p_3	=	1	0	1	0	1	0	. . .	0	0	0
.		.	.	.	.	.	.		.	.	.
.		.	.	.	.	.	.		.	.	.
.		.	.	.	.	.	.		.	.	.
$p_{k/2}$	=	1	0	1	0	1	0	. . .	1	0	0
$p_{k/2 + 1}$	=	1	0	1	0	1	0	. . .	1	0	1
$p_{k/2 + 2}$	=	1	1	1	0	1	0	. . .	1	0	1
$p_{k/2 + 3}$	=	1	1	1	1	1	0	. . .	1	0	1
.		.	.	.	.	.	.		.	.	.
.		.	.	.	.	.	.		.	.	.
.		.	.	.	.	.	.		.	.	.
p_{2k}	=	1	1	1	1	1	1	. . .	1	0	1
$p_{2k + 1}$	=	1	1	1	1	1	1	. . .	1	1	1

FIGURE 3-4. Set of p_i's for principal-column partitions for m odd with y_m excited

For maximum-distance principal-column partitions, k = [m/2]. It will be shown shortly that the above p_i's are satisfactory for any principal column partition. The distance between the states of each 2-set is 2k + 1 when m is odd with

y_m excited, and 2k otherwise. If y_m must be excited and m is odd, then the p_i's of Figure 3-4 are to be used; otherwise, the p_i's are obtained from Figure 3-3.

In Figure 3-3 and Figure 3-4, one can note that each p_i, $i < m$, has odd parity in at least one parity set. In Figure 3-3, each p_i has a different parity combination from every other p_j, and therefore the conditions of theorem 2-7 are met and the distinct cosets can be obtained. In Figure 3-4, each p_i has a different parity combination from every other p_j, except when $i = k/2$ and $j = k/2 + 1$. However, $||p_{k/2} \oplus p_{k/2+1}|| = 1$, and since there are no states of weight 1 in H, $p_{k/2} \oplus p_{k/2+1}$ cannot be in H. Therefore, the conditions placed on the p_i's are met. Therefore, Assignment 2 satisfactorily encodes all maximum-distance principal-column partitions.

The p_i's associated with the previous example of a maximum-distance principal-column partition for a 2^6-row flow table with S_m = 101110101 are shown in Figure 3-5.

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9
p_1 =	1	0	0	0	0	0	0	0	0
p_2 =	1	0	1	0	0	0	0	0	0
p_3 =	1	0	1	0	1	0	0	0	0
p_4 =	1	0	1	0	1	0	1	0	0
p_5 =	1	0	1	1	1	0	1	0	0
p_6 =	1	0	1	1	1	0	1	0	1

FIGURE 3-5. p_i's for principal-column partition with S_m = 101110101

It has been established that a satisfactory set of transition paths exist for all maximum-distance principal-column partitions. The next step is to show that a satisfactory set of transition paths exist for all principal-column partitions less than maximum distance. In Assignment 1, ML transition paths were not possible for certain transitions when there was five or more state variables in a parity set. However, with three state variables in each parity set in the codes for Assignment 2, all transitions will be ML transitions. Since all transitions are ML transitions, the number of p_i's needed for principal-column partitions is less than or equal to m. Then since the number of p_i's that can be generated is $2^{[m/2]+1}$, which is greater than or equal to m, enough cosets can always be formed to obtain a satisfactory set of transition paths. Following are specific algorithm that can be used to generate the subgroup H of initial states and associated p_i's for less than maximum-distance principal-column partitions.

The conditions placed on the subgroup H for any principal-column partition are:

1) S_m for the column partition cannot be in H

2) If m is odd and y_m is excited, then no state with a weight of 1 can be in H.

The algorithms that are given for generating the subgroup H depend on whether m is even or odd. For m odd with y_m excited, H can be generated as follows:

Step 1. With m - 1 independent state variables $y_1, y_2, \ldots, y_{m-1}$ generate 2^{m-1} distinct code words. Let state variable

$y_m = 0$ in all code words.

Step 2. Generate the parity sets to obtain the rest of the code for each state in H.

Since state variable y_m is excited in each transition, y_m induces a 1 in the code for S_m. Since $y_m = 0$ in all the states of H, S_m cannot appear in H and for the same reason, there cannot be any states of weight 1 in H. Therefore, both conditions placed on H are met.

For m even, and for m odd where y_m is not excited, only S_m of the principal-column partition cannot appear in H. Since there are three state variables in each parity set and from theorem 3-2, it follows that at least one independent state variable from each excited parity set must be excited. Let y_e be any one of the independent state variables that must be excited. Following then is an algorithm for generating H:

Step 1. Generate 2^{m-1} distinct code words with the set of m - 1 independent state variables $\{y_1, y_2, \ldots, y_m\}$, where y_e is not in the set. Let $y_e = 0$ in all code words.

Step 2. Generate the parity sets to obtain the rest of the code for each state in H.

The above procedures can be used for generating subgroups of initial states for principal-column partitions for Assignment 1 without any modifications.

To illustrate the procedure for generating a subgroup H, consider the state assignment for a 2^5-row flow table shown in Figure 3-6, and the principal-column partition:

α = {0,36; 1,37; 2,34; 3,35; 4,32; 5,33; 6,30; 7,31; 10,26; 11,27; 12,24; 13,25; 14,22; 15,23; 16,20; 17,21},

where S_m= 1111000. The numbers of the states are octal representations of the code associated with the independent state variables y_1, y_2, y_3, y_4, and y_5.

State	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0
2	0	0	0	1	0	0	1
3	0	0	0	1	1	0	1
4	0	0	1	0	0	0	1
5	0	0	1	0	1	0	1
6	0	0	1	1	0	0	0
7	0	0	1	1	1	0	0
10	0	1	0	0	0	1	0
11	0	1	0	0	1	1	0
12	0	1	0	1	0	1	1
13	0	1	0	1	1	1	1
14	0	1	1	0	0	1	1
15	0	1	1	0	1	1	1
16	0	1	1	1	0	1	0
17	0	1	1	1	1	1	0
20	1	0	0	0	0	1	0
21	1	0	0	0	1	1	0
22	1	0	0	1	0	1	1
23	1	0	0	1	1	1	1
24	1	0	1	0	0	1	1
25	1	0	1	0	1	1	1
26	1	0	1	1	0	1	0
27	1	0	1	1	1	1	0
30	1	1	0	0	0	0	0
31	1	1	0	0	1	0	0
32	1	1	0	1	0	0	1
33	1	1	0	1	1	0	1
34	1	1	1	0	0	0	1
35	1	1	1	0	1	0	1
36	1	1	1	1	0	0	0
37	1	1	1	1	1	0	0

FIGURE 3-6. State assignment for 2^5-row flow table

The excited state variables are y_1, y_2, y_3, and y_4 when S_m = 1111000. In Figure 3-7 is the subgroup H for the above

column partition with S_m = 1111000.

State	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0
4	0	0	1	0	0	0	1
5	0	0	1	0	1	0	1
10	0	1	0	0	0	1	0
11	0	1	0	0	1	1	0
14	0	1	1	0	0	1	1
15	0	1	1	0	1	1	1
20	1	0	0	0	0	1	0
21	1	0	0	0	1	1	0
24	1	0	1	0	0	1	1
25	1	0	1	0	1	1	1
30	1	1	0	0	0	0	0
31	1	1	0	0	1	0	0
34	1	1	1	0	0	0	1
35	1	1	1	0	1	0	1

FIGURE 3-7. Subgroup H for principal-column partition with S_m = 1111000

Note that y_4 = 0 in all states of H, thereby insuring that S_m is not in H.

Obtaining the p_i's is as straight-forward for the less than maximum-distance case as it is for the maximum-distance case. Figure 3-3 and Figure 3-4 can be used directly to find the p_i's for any principal-column partition. The excited parity sets are numbered from 1 to k; these are the parity sets shown in Figure 3-3 and Figure 3-4. The excited state variables are labeled in the same manner as they were in the maximum-distance case. The p_i's of Figure 3-3 and Figure 3-4 can be

used directly regardless of which parity sets are to be excited or which state variables (dependent or independent) within each excited parity set are to be excited. If m is odd and state variable y_m is an excited state variable, then the p_i's of Figure 3-4 apply; otherwise, the p_i's of Figure 3-3 are to be used. The p_i's for the subgroup H of Figure 3-7 are:

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
p_1 =	1	0	0	0	0	0	0
p_2 =	1	0	1	0	0	0	0
p_3 =	1	1	1	0	0	0	0
p_4 =	1	1	1	1	0	0	0.

For this example, k = 2 and

$$Y_{11} = Y_1 \qquad Y_{12} = Y_2$$

$$Y_{21} = Y_3 \qquad Y_{22} = Y_4.$$

The corresponding set of transition paths is shown in Figure 3-8.

With the above discussions and algorithms, it is concluded that each principal-column partition is satisfactorily encoded by Assignment 2. Furthermore, the transition paths can be obtained in an easy, straight-forward manner.

0	0000000 →	1000000 →	1010000 →	1110000 →	1111000	36
1	0000100 →	1000100 →	1010100 →	1110100 →	1111100	37
4	0010001 →	1010001 →	1000001 →	1100001 →	1101001	32
5	0010101 →	1010101 →	1000101 →	1100101 →	1101101	33
10	0100010 →	1100010 →	1110010 →	1010010 →	1011010	26
11	0100110 →	1100110 →	1110110 →	1010110 →	1011110	27
14	0110011 →	1110011 →	1100011 →	1000011 →	1001011	22
15	0110111 →	1110111 →	1100111 →	1000111 →	1001111	23
20	1000010 →	0000010 →	0010010 →	0110010 →	0111010	16
21	1000110 →	0000110 →	0010110 →	0110110 →	0111110	17
24	1010011 →	0010011 →	0000011 →	0100011 →	0101011	12
25	1010111 →	0010111 →	0000111 →	0100111 →	0101111	13
30	1100000 →	0100000 →	0110000 →	0010000 →	0011000	6
31	1100100 →	0100100 →	0110100 →	0010100 →	0011100	7
34	1110001 →	0110001 →	0100001 →	0000001 →	0001001	2
35	1110101 →	0110101 →	0100101 →	0000101 →	0001101	3
	H	$p_1 \oplus H$	$p_2 \oplus H$	$p_3 \oplus H$	$p_4 \oplus H$	

FIGURE 3-8. Transition paths for less than maximum-distance principal-column partition

IV. TRANSITION PATHS - GENERAL CASE

The comments of this section pertain to both Assignment 1 and Assignment 2. The most general case includes column partitions which are composed of k-sets, some or all of which may be larger than 2-sets. Also the set of state variables that must be excited in effecting transitions may vary from k-set to k-set. Since the same state variables are not excited to effect the transitions between the states of all the k-sets, forming a subgroup H of initial states and obtaining a set of p_i's provides little help in finding the transition paths for the general case.

Trial and error is involved in finding the transition paths for the more general cases. Of course, a certain amount of checking is associated with a trial and error approach, and any means that can reduce the number of checks that have to be made is usually welcomed. The checking that must be done in finding a set of transition paths is to insure that crossover does not occur.

Definition: When transition paths of different k-sets have a state in common, it is called crossover.

A trial and error approach for finding the transition paths of a column partition in its crudest form would involve checking each state of the transition paths associated with each k-set with every state in the transition paths of all other k-sets. With the codes produced by Assignment 1 and Assignment 2, it is not necessary to do quite as much check-

ing.

Theorem 4-1: If the parity combinations of two n-tuples are different, then the n-tuples cannot have the same code.

Proof: If there is just one parity set which has odd parity in one n-tuple and even parity in another, then the codes for these parity sets are different and the corresponding n-tuples also have different codes. ##

From theorem 4-1, only those states with the same parity combination need be checked for possible crossover. The converse of the above theorem does not hold, for two n-tuples can have the same parity combination and still have different codes. For example, let n-tuple n_1 = 010000 and n_2 = 101100 which are encoded by y_1, y_2, y_3, y_4, y_5, and y_6, with parity set 1 and 2 being $\{y_1,y_2,y_5\}$ and $\{y_3,y_4,y_6\}$ respectively. Both n-tuples have the same parity combination (odd parity in parity set 1 and even parity in parity set 2), but certainly they have different codes.

Two n-tuples with the same parity combination could be thought of as being in an equivalence class. Let the following define the relation between n-tuples.

Definition: Let the symbol $\sim$ denote a specific relation defined on the set N (all n-tuples on the n-cube). Let n_1 and n_2 be any two n-tuples. $n_1 \sim n_2$ if n_1 has the same parity combination as n_2.

A relation is an equivalence relation if the relation is reflexive, symmetric, and transitive [10]. It can be seen

that the above defined relation is an equivalence relation. An equivalence relation partitions the set it is defined on into equivalence classes [10]. The n-tuples in each equivalence class here have the same parity combinations.

Theorem 4-2: The number of n-tuples in each equivalence class is 2^m for both Assignment 1 and Assignment 2.

Proof: The number of n-tuples in the n-cube associated with Assignment 1 is $2^{m+[\log_2 m]}$. In Assignment 1 there are $[\log_2 m]$ parity sets. The number of parity combinations that can be assumed by each n-tuple is therefore $2^{[\log_2 m]}$. Then, the number of n-tuples in each equivalence class in the code for Assignment 1 is

$$\frac{2^{m+[\log_2 m]}}{2^{[\log_2 m]}} = 2^m.$$

For Assignment 2, the number of n-tuples in the associated n-cube is $2^{m+[m/2]}$. In Assignment 2 there are $[m/2]$ parity sets. The number of parity combinations that can be assumed by each n-tuple is $2^{[m/2]}$. Therefore, the number of n-tuples in each equivalence class in the code for Assignment 2 is

$$\frac{2^{m+[m/2]}}{2^{[m/2]}} = 2^m. \quad \#\#$$

In each assignment, the number of n-tuples in each equivalence class is the same. Since $[m/2] \geq [\log_2 m]$, the number of equivalence classes in Assignment 2 is greater than or equal to the number of equivalence classes in Assignment 1. The number of states encoded by both assignments is 2^m, and each of these states has even parity in each parity set.

Therefore, the states encoded by each assignment make up one of the equivalence classes.

Since each equivalence class represents a disjoint set of n-tuples, crossover only has to be checked between n-tuples of the same equivalence class. The number of n-tuples in each equivalence class that must be checked then is 2^m. This is significantly less than having to check 2^n n-tuples.

For the general case, the process of obtaining the transition paths consists mainly of trial and error. Each n-tuple in the transition path of a k-set must be checked with the n-tuples in the transition paths of all other k-sets. The checking for crossover only needs to be done between n-tuples of the same equivalence class.

For column partitions that consist of only 2-sets with the states of each 2-set a maximum distance apart, a slight modification of this approach can be taken. If all, or at least most, of the states of each level of the transition paths are n-tuples from the same equivalence class, then the checking is primarily restricted to the levels of the transition paths. If the states at different levels of the transition paths belong to different equivalence classes, there can be no crossover between levels. The advantage of this method is that it allows the transition paths to be generated in a parallel fashion. To illustrate this procedure, two examples are presented.

The first column partition for consideration is

α_1 = {1,17; 2,10; 3,5; 12,14; 4,11; 6,15; 0,13; 7,16},

where the state assignment is shown in Figure 4-1 with the states given the octal number associated with the independent state variables y_1, y_2, y_3, and y_4.

State	y_1	y_2	y_3	y_4	y_5	y_6
0	0	0	0	0	0	0
1	0	0	0	1	0	1
2	0	0	1	0	0	1
3	0	0	1	1	0	0
4	0	1	0	0	1	0
5	0	1	0	1	1	1
6	0	1	1	0	1	1
7	0	1	1	1	1	0
10	1	0	0	0	1	0
11	1	0	0	1	1	1
12	1	0	1	0	1	1
13	1	0	1	1	1	0
14	1	1	0	0	0	0
15	1	1	0	1	0	1
16	1	1	1	0	0	1
17	1	1	1	1	0	0

FIGURE 4-1. 2^4-row state assignment

The two parity sets are $\{y_1, y_2, y_5\}$ and $\{y_3, y_4, y_6\}$; let them be called parity set 1 and 2 respectively. Listed are the 2-sets and the state variables that are to be excited in the corresponding maximum-distance transition along with the corresponding S_m.

2-sets	Excited Variables	S_m
1,17	Y_1, Y_2, Y_3, Y_6	111001
2;10	Y_1, Y_3, Y_5, Y_6	101011
3,5 & 12,14	Y_2, Y_3, Y_5, Y_6	011011
4,11	Y_1, Y_2, Y_4, Y_6	110101
6,15 & 0,13	Y_1, Y_3, Y_4, Y_5	101110
7,16	Y_1, Y_3, Y_4, Y_5	100111

In maximum-distance principal-column partitions, the same set of state variables is excited in each 2-set. This example represents the opposite extreme with different sets of state variables being excited in almost every 2-set. Figure 4-2 shows the set of transition paths for this column partition. The bar $^{(-)}$ denotes those bits in the code for a state which are to change state at some point in the total transition.

The first level of the transition paths is obtained from the initial set by changing a bit in parity set 2 such that there would be no crossover within the first level of the transition paths. The first level then has even parity in parity set 1 and odd parity in parity set 2. An appropriate bit in parity set 1 is changed in each of the states of level 1 to obtain the states of level 2. In level 2, there is odd parity in both parity sets. Level 3 is obtained from level 2 by changing the last bit to be changed in parity set 2, except for the state associated with 15 which had a bit changed in parity set 1. This is necessary to produce a set of states at level 3 where there is no crossover. The state at level 3 associated with 15, marked with an asterisk, has the same

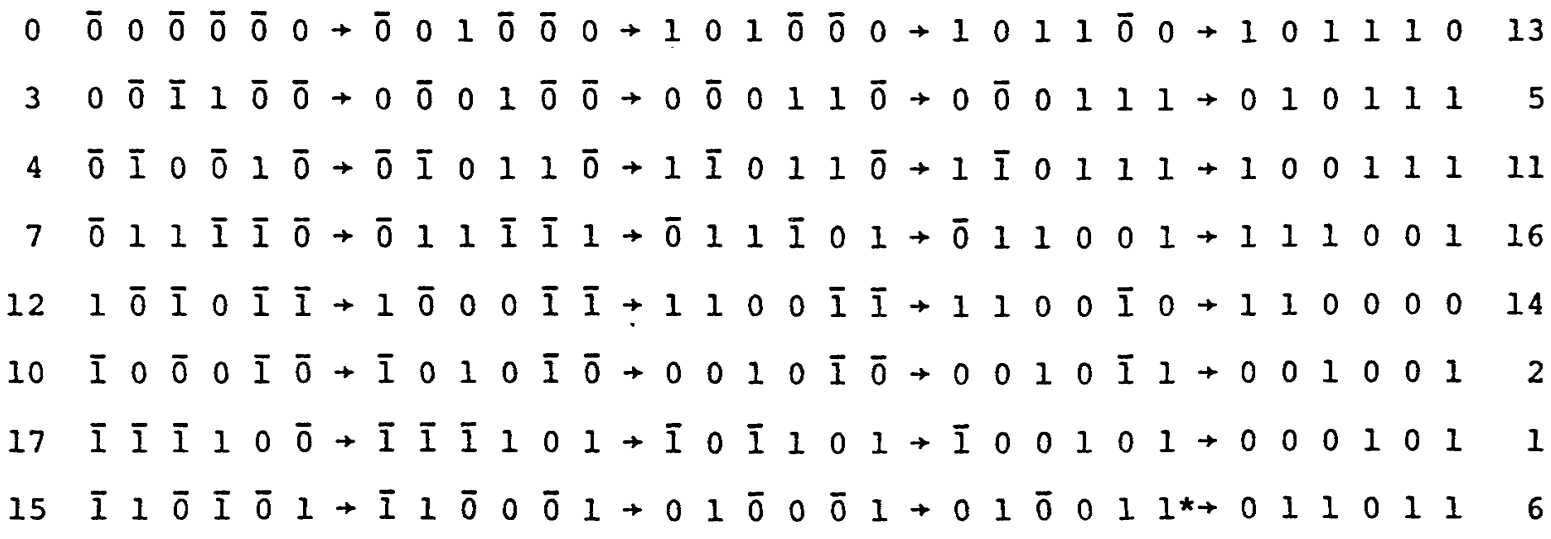

FIGURE 4-2. Maximum distance example for 2^4-row flow table

parity combination as those states at level 1. However, there is no crossover. Note that the states at the different levels belong to different equivalence classes with only one exception at level 3; and therefore, there is no crossover between levels.

For m odd, state variable y_m is not in a parity set, and when y_m needs to be excited to effect a transition greater than a distance 1, a special problem occurs. For 2^m-row flow tables state variable y_m cannot be the first or last state variable excited in effecting a transition, unless it is the only one that has to be excited. Since state variable y_m is not in a parity set, exciting y_m first produces an n-tuple that has even parity in all parity sets and is in G (or G*); this represents a crossover if the transition is greater than a distance of 1. If y_m is the last state variable that is excited, then the resulting n-tuple has even parity in each parity set, which again is in G (or G*); this represents a crossover also. Therefore, for 2^m-row flow tables, if y_m is to be excited in a transition greater than a distance 1, then it cannot be the first or the last state variable excited. For flow tables that do not contain exactly 2^m-rows, and for incompletely specified flow tables, y_m can be excited first or last in certain transitions.

The second maximum-distance example illustrating the idea of attempting to keep the n-tuples of the same level of the transition paths in the same equivalence class is a column partition from a 2^5-row flow table.

α_2 = {0,13; 26,35; 2,11; 1,16; 32,25; 34,23; 10,5; 22,37; 15,36; 4,21; 20,7; 14,33; 24,17; 3,30; 6,31; 12,27}.

The state assignment for the 2^5-row flow table that corresponds to these numbered states is shown in Figure 3-6. First a word about this particular example. Several algorithms were developed to obtain the transition paths in a straight-foward manner involving as little trial and error as possible. Some of these algorithms worked reasonably well for a limited number of cases. Every method developed failed to produce the transition paths for this example, except one which becomes so involved that it is questionable as to its value. Therefore, this column partition seems to constitute a worst or near-worst case. The set of transition paths are shown in Figure 4-3.

If parity set 1 is $\{y_1, y_2, y_6\}$ and parity set 2 is $\{y_3, y_4, y_7\}$, the parity combinations for the different levels of the transition paths are shown next, where 0 represents even parity and 1 represents odd parity.

Level	Parity set 1	Parity set 2	
0	0	0	Initial set
1	0	1	
2	1	1	
3	1	1	y_5 excited to produce level 3 from level 2
4	1	0	states 0 and 37 have parity 01 at this level
5	0	0	final set

From the above parity combinations, it can be seen in which parity set the state variables were changed to produce a succeeding level of states in the transition path. Except for level 2 and 3, the parity combinations are different, insuring

0	$0\bar{0}0\bar{0}\bar{0}\bar{0}\bar{0}$ →	$0\bar{0}01\bar{0}\bar{0}\bar{0}$ →	$0101\bar{0}\bar{0}\bar{0}$ →	$01011\bar{0}\bar{0}$ →	0101110^{*}→	0101111	13
2	$0\bar{0}0\bar{1}\bar{0}\bar{0}\bar{1}$ →	$0\bar{0}00\bar{0}\bar{0}\bar{1}$ →	$0100\bar{0}\bar{0}\bar{1}$ →	$0100\bar{1}\bar{0}\bar{1}$ →	$0100\bar{1}\bar{0}0$ →	0100110	11
14	$\bar{0}1\bar{1}\bar{0}\bar{0}\bar{1}1$ →	$\bar{0}1\bar{1}1\bar{0}\bar{1}1$ →	$\bar{0}1\bar{1}1\bar{0}01$ →	$\bar{0}1\bar{1}1101$ →	$\bar{0}101101$ →	1101101	33
16	$0\bar{1}\bar{1}\bar{0}\bar{1}0$ →	$0\bar{1}\bar{1}00\bar{1}0$ →	$00\bar{1}0\bar{0}\bar{1}0$ →	$00\bar{1}01\bar{1}0$ →	$00001\bar{1}0$ →	0000100	1
20	$\bar{1}0\bar{0}\bar{0}0\bar{1}0$ →	$\bar{1}00\bar{1}0\bar{1}0$ →	$00\bar{0}1\bar{0}\bar{1}0$ →	$00\bar{0}11\bar{1}0$ →	$00111\bar{1}0$ →	0011100	7
26	$1\bar{0}1\bar{1}\bar{0}\bar{1}\bar{0}$ →	$1\bar{0}100\bar{1}\bar{0}$ →	$1\bar{0}10\bar{0}0\bar{0}$ →	$1\bar{0}1010\bar{0}$ →	$1\bar{0}10101$ →	1110101	35
32	$1\bar{1}\bar{0}1\bar{0}0\bar{1}$ →	$1\bar{1}\bar{0}0\bar{0}\bar{0}1$ →	$1\bar{1}\bar{0}0\bar{0}11$ →	$1\bar{1}\bar{0}0111$ →	$1\bar{1}10111$ →	1010111	25
34	$1\bar{1}\bar{1}\bar{0}\bar{0}\bar{0}1$ →	$1\bar{1}\bar{1}1\bar{0}\bar{0}1$ →	$10\bar{1}1\bar{0}\bar{0}1$ →	$10\bar{1}11\bar{0}1$ →	$1001\bar{1}\bar{0}1$ →	1001111	23
3	$\bar{0}\bar{0}0\bar{1}\bar{1}0\bar{1}$ →	$\bar{0}\bar{0}\bar{0}0\bar{1}0\bar{1}$ →	$1\bar{0}0\bar{0}1\bar{0}\bar{1}$ →	$1\bar{0}0000\bar{1}$ →	$1\bar{0}00000$ →	1100000	30
5	$0\bar{0}\bar{1}0\bar{1}\bar{0}\bar{1}$ →	$0\bar{0}\bar{1}0\bar{1}\bar{0}0$ →	$01\bar{1}0\bar{1}\bar{0}0$ →	$01\bar{1}00\bar{0}0$ →	$0100\bar{0}0$ →	0100010	10
15	$\bar{0}110\bar{1}\bar{1}\bar{1}$ →	$\bar{0}11\bar{0}\bar{1}\bar{1}0$ →	$111\bar{0}\bar{1}\bar{1}0$ →	$1110\bar{0}\bar{1}0$ →	$1111\bar{0}\bar{1}0$ →	1111000	36
17	$\bar{0}\bar{1}1\bar{1}\bar{1}1\bar{0}$ →	$\bar{0}\bar{1}1\bar{1}\bar{1}11$ →	$\bar{0}01\bar{1}\bar{1}11$ →	$\bar{0}01\bar{1}011$ →	$\bar{0}0100\bar{1}1$ →	1010011	24
21	$\bar{1}0\bar{0}0\bar{1}\bar{1}\bar{0}$ →	$\bar{1}0\bar{0}0\bar{1}\bar{1}1$ →	$00\bar{0}0\bar{1}\bar{1}1$ →	$00\bar{0}00\bar{1}1$ →	$0010\bar{0}\bar{1}1$ →	0010001	4
27	$\bar{1}\bar{0}\bar{1}1\bar{1}1\bar{0}$ →	$\bar{1}\bar{0}01\bar{1}1\bar{0}$ →	$\bar{1}101\bar{1}1\bar{0}$ →	$\bar{1}1010\bar{1}\bar{0}$ →	$\bar{1}101011$ →	0101011	12
31	$\bar{1}\bar{1}\bar{0}\bar{0}\bar{1}00$ →	$\bar{1}\bar{1}\bar{0}1\bar{1}00$ →	$\bar{1}0\bar{0}1\bar{1}00$ →	$\bar{1}0\bar{0}1000$ →	$\bar{1}011000$ →	0011000	6
37	$1\bar{1}\bar{1}1\bar{1}\bar{0}\bar{0}$ →	$1\bar{1}\bar{1}1\bar{1}\bar{0}1$ →	$1\bar{1}\bar{1}1\bar{1}\bar{1}1$ →	$1\bar{1}\bar{1}1011$ →	$10\bar{1}1011^{*}$→	1001011	22
	Initial set	Level 1	Level 2	Level 3	Level 4	Final set	

FIGURE 4-3. Maximum distance example for 2^5-row flow table

that there can be no crossover between those levels. Level 3 was obtained from level 2 by changing only state variable y_5. Level 2 had to be obtained such that this could be done. The states of level 2 and level 3 are all 2^5 n-tuples that are contained in the equivalence class that has odd parity in both parity sets. State 0 and 37 have a different parity combination at level 4 than the other states of that level; they are marked with an asterisk. It was necessary to check these two n-tuples with the n-tuples of level 1 to insure that there is no crossover.

The advantage of the method illustrated in the two examples above is that it allows one to produce the transition paths in a parallel fashion for maximum distance column partitions.

In principal-column partitions, the same set of state variables are excited in a transition between the states of each 2-sets. For 2^4 and 2^5-row flow tables, many maximum-distance column partitions were considered where the state variables that must be excited in the transitions varied as much as possible. These problems were constructed to represent the opposite type of situation presented by principal-column partitions. Two examples have been shown. In every case, there was a non-unique solution with considerable flexibility associated with generating a set of transition paths. Also, finding a set of transition paths was relatively easy.

A procedure which involves trial and error must also be used for column partitions where k-sets are not restricted to 2-sets. The first step in obtaining the transition paths is

to form a set of transition pairs for each k-set, and generally the transition pairs are selected such that the total number of states in the resulting transition path associated with the states of each k-set is minimal. For example, consider a k-set that consits of the states S_a, S_b, and S_c, with S_a being the stable state. If the distance between S_a and S_b is 2, between S_a and S_c is 4, and between S_b and S_c is 2, then one might form transition pairs of S_a,S_b and S_b,S_c where there would be a transition path between each pair of states of the transition pairs. In this example, the total number of states in the transition path for this k-set is 5. If the transition pairs were S_a,S_b and S_a,S_c, the total number of states in the transition path could be as large as 7. It should be pointed out that the pairing of the states in forming a set of transition pairs for a k-set can be changed if it is discovered that another set would produce a satisfactory transition path.

The transition paths associated with transition pairs of the same k-set can have states in common. In fact, it is desirable for transition pairs of the same k-set to share states in the transition path since this would result in fewer total states in the transition path for the k-set. Trial and error is involved in finding the transition paths for each transition pair. The only checking that needs to be done is between n-tuples in the same equivalence class. Experience indicates that it is easier to find a set of transition paths for this case than for maximum-distance column partitions.

V. NON-GENERALIZED STATE ASSIGNMENT

The only method for finding non-standard state assignments presently available that is partly systematic is that presented by Hazeltine [9]. Even though the general approach is systematic, a significant amount of trial and error frequently results in finding the transition paths and state assignment. The upper bound associated with this method is 2m - 1 state variables for a 2^m-row flow table.

In this chapter, a method is presented for finding non-standard or non-generalized state assignments for flow tables. The state assignment is a function of the characteristics of a flow table. The method presents a systematic means of obtaining the state assignment, but may not produce minimum variable assignments. Trial and error is required in finding the transition paths but not in generating the state assignment. The assignment procedure has a suggested upper bound of m + [m/2] state variables for a 2^m-row flow table, which is Assignment 2.

The basis of the assignment procedure is to find a state assignment where all, or at least most, of the transitions can be accomplished such that the resulting transition paths are minimum length. ML transitions are desirable in that the state variables which must change state are the only ones excited in effecting a transition. Also the number of states in a ML transition path is less than the number of states in a non-ML transition path for the same transition pair. Since a satisfactory set of transition paths is more likely when fewer

states are required in the corresponding transition paths, ML transitions are more desirable.

In general, flow tables may not contain 2^m rows, but rather something between 2^{m-1} and 2^m rows where at least m state variables are needed to encode it. The procedures presented are designed to encode 2^m-row flow tables. Certainly these procedures will apply, perhaps in an easier fashion, to flow tables where the number of rows is between 2^{m-1} and 2^m rows. The procedures are also applicable to incompletely specified flow tables.

The first step in finding an assignment is to encode the states of the flow table with m state variables $y_1, y_2, \ldots, y_m$. This corresponds to assigning codes of a m-cube to the states of the flow table. The m state variables that encode the m-cube are considered as a set of independent state variables. This encoding is called the initial code.

State variables are added to the initial code in such a manner that transition paths between states can be minimum length. Each state variable added to the assignment is a dependent state variable and it, with the independent state variables that are used to generate it, form a parity set.

As stated before, the primary goal is to achieve an assignment such that all or at least most of the transitions between the states are minimum length. Since there exist states where it is impossible to obtain ML transition paths in state assignments, which have five or more state variables in a parity set, the parity sets will be restricted to containing only

three state variables.

Initially a procedure will be developed for column partitions that consist of only 2-sets. The states of a 2-set are a transition pair and each 2-set has only one transition pair. Necessary modifications of this procedure will be made to apply to the more general case where k-sets are not restricted to 2-sets.

The first step in the assignment procedure is to encode the states with m independent state variables. Each state is given a specific m-tuple in the m-cube. The procedure works even if the codes for the states are assigned arbitrarily, but a judicious selection of codes for the states can reduce the number of state variables in the final code. The number of states in the transition paths of a transition pair is directly proportional to the distance between the states of a transition pair. If the distance between the states of a transition pair is d, then the number of states in a ML transition path is d + 1, and in a non-ML transition path, it is greater than d + 1. More states in a transition path generally imply that it is more difficult to construct transition paths that are void of crossovers. In general then, the closer one can encode the states of transition pairs, the easier it is to construct a satisfactory set of transition paths.

Transitions are not usually required between all pairs of states in a flow table. A judicious initial code would assign those states, between which transitions are not re-

quired, m-tuples that are relatively far apart. Certainly, one should attempt to avoid assigning states of a transition pair m-tuples which are maximum distance apart. In general, fewer state variables are required in the final code, if the states of the transition pairs are given codes close together.

It is hard, sometimes, to determine how much effort should go into trying to find a good initial code. A minimal effort would involve attempting to keep the distances between the states of the 2-sets less than maximum.

One further point is worth mentioning. If particular transitions appear in the flow table many more times than others, then there is some positive value in assigning adjacent codes to these states. This will insure that these transitions can always be accomplished and, since they are relatively large in number, this initial coding may reduce the problem complexity.

If the initial coding produces an assignment such that there exists a satisfactory set of transition paths for each column of the flow table, then the assignment process is complete and no additional state variables have to be added. In most cases, the initial coding will not produce a satisfactory set of transition paths and additional state variables must be added. Each state variable that is added is a dependent or generated state variable of a parity set. The two independent state variables and the generated state variable form a parity set, with each parity set possessing even parity. The two independent state variables, which are used to generate

the dependent state variable, are selected such that a maximum number of transitions result in ML transition paths.

The final code will consist of a set of n state variables $\{y_1, y_2, \ldots, y_m, y_{m+1}, \ldots, y_n\}$. State variables $y_{m+1}, \ldots, y_n$ are the dependent state variables generated by the independent state variables of the parity sets. In general, there would be some state variables that do not belong to any parity set. Since the final code is formed in a manner similar to that of Assignment 2, the code for a 2^m-row flow table would constitute a group code. Since there is an effort to keep the distance between states of transition pairs as small as possible, the final code should not have a maximum distance greater than m. This implies that an independent state variable can be in at most one parity set, for if a state variable could be in even two parity sets, the maximum distance would be greater than m for 2^m-row flow tables.

Definition: Those state variables that must be excited to produce a ML transition path between the states of a transition pair are called the excited state variables.

To emphasize the above definition, consider the following assignment and the transition pair consisting of states 0 and 3. The assignment is shown on the following Karnaugh map.

y_3y_4 \ y_1y_2	00	01	11	10
00	0•	• •	4	
01	1	•	5	
11	3•	• •	7	
10	2		6	

A transition path for the transition pair, 0,3 could be denoted by the sequence

$$0000 \rightarrow 0100 \rightarrow 0101 \rightarrow 0111 \rightarrow 0011$$

and is shown by the dotted line on the Karnaugh map. The excited state variables are y_3 and y_4, even though state variable y_2 was excited in the transition path. The transition path is not minimum length.

Transition paths, associated with states that have adjacent codes, consist of just the adjacent states, and no crossover can possibly result. Furthermore, these are ML transition paths. Therefore, the only type of transitions that could result in non-ML transitions are those between states which are a distance 2 or greater.

Theorem 5-1: Consider a transition pair S_a, S_b, where the distance between S_a and S_b is greater than 1. (There are at least two excited state variables.) If, for this transition, there is at least one excited state variable in at least one parity set, then a ML transition path exists for S_a, S_b.

Proof: A ML transition path would not be possible, if in changing the state of the excited state variables, one could not avoid producing a n-tuple that is a state encoded

by the state assignment. From theorem 3-2, there must be two excited state variables in a parity set. The first n-tuple in the transition path can be obtained by changing state of one of the excited state variables in a parity set; this n-tuple then has odd parity in a parity set. All the states encoded by the assignment have even parity in the parity sets. Therefore, the first n-tuple in the transition path cannot be a state encoded by the assignment; furthermore, if all the n-tuples of the transition path have odd parity in a parity set, then they cannot be states encoded by the assignment. Producing the rest of the transition path where there is odd parity in a parity set of each n-tuple can be accomplished simply by changing state of all the other excited state variables, except the one in the parity set; the last state in the transition path is obtained by changing the state of the last excited state variable in the parity set. Since only those state variables which must change state are excited in the transition, a ML transition path results. ##

Theorem 5-1 does not imply that it is impossible to have a ML transition path if none of the excited state variables are in a parity set, but rather that a ML transition path is always possible if an excited state variable is in a parity set. For incompletely specified flow tables and in flow tables where there are not exactly 2^m states, it is possible, in some cases, to obtain ML transition paths without having an excited state variable in a parity set. For 2^m-row flow tables without "don't care" states, a state variable in a parity set must

be excited to effect a distance greater than 1 transition.

Theorem 5-1 does not imply that if there is at least one excited state variable in a parity set for each transition pair, then a satisfactory set of transition paths exist. Counter examples can be easily constructed to show this.

All transition pairs do not have to produce ML transition paths in a satisfactory state assignment. Since ML transitions require fewer states in transition paths, it is more likely to obtain a satisfactory set of transition paths if most of the transition pairs produce ML transitions. Therefore, each generated state variable is added in such a manner that most of the transitions produce ML transition paths. This is done by first counting the number of times each state variable must be excited in effecting transitions between states a distance greater than 1. This is referred to as the count list; the state variables that have the highest value in the count list are excited most often. The first parity set is formed from the two highest value state variables in the count list. These two state variables are used to generate a dependent state variable, and the three state variables form a parity set. Forming a parity set in this manner allows a maximum number of transitions to result in ML transition paths.

After generating each new state variable, one should attempt to find a satisfactory set of transition paths for each column partition of the flow table. Even though some of the transition pairs may not produce ML transition paths, it still may be possible to find a satisfactory set of transition paths. Producing the transition paths is mostly a trial and error

process. If all the transition paths can be obtained without any crossover, then the process is complete, and the state assignment is valid for that flow table. If there are some transition paths that cannot be obtained, then there are two procedures one can use. The easiest would be to generate another dependent state variable from the two highest value state variables in the count list that are not yet in a parity set. This forms another parity set. The next step would again attempt to obtain a satisfactory set of transition paths. The process continues until one reaches the suggested bound associated with Assignment 2, m + [m/2], or finds a satisfactory set of transition paths.

A second method of determining which independent state variables are to be used in generating an additional state variable would be particularly useful if there is only one or just a few column partitions for which a satisfactory set of transition paths cannot be obtained. If this is the case, then recount the number of times each state variable, that is not in a parity set yet, needs to be excited in effecting a transition between states of those column partitions for which a satisfactory set of transition paths cannot be found. Select the two state variables that are excited most frequently to generate another state variable, and form a new parity set. After this is done, attempt to find a satisfactory set of transition paths for all column partitions. The process continues until the upper bound is achieved, or a satisfactory set of transition paths is found.

To illustrate the above procedure somewhat, consider the

states of a 4-cube, numbered as shown in Figure 5-1.

y_3y_4 \ y_1y_2	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

FIGURE 5-1. States of a 4-cube

Note that the only transitions that can be accomplished are those between states which have adjacent codes. Suppose that it was determined that state variables y_1 and y_2 were excited most often in effecting transitions between states. From the above method, state variable y_5 would be generated such that $y_5 = y_1 \oplus y_2$. The Karnaugh map representation of the 16 states from Figure 5-1 would now appear as shown in Figure 5-2.

$y_5 = 0$

y_3y_4 \ y_1y_2	00	01	11	10
00	0		12	
01	1		13	
11	3		15	
10	2		14	

$y_5 = 1$

y_3y_4 \ y_1y_2	00	01	11	10
00		4		8
01		5		9
11		7		11
10		6		10

FIGURE 5-2. States of Figure 5-1 in a 5-cube with $y_5 = y_1 \oplus y_2$

Note that each of the transitions, where state variables in the parity set $\{y_1,y_2,y_5\}$ are excited, can be accomplished in a fashion such that a ML transition path can be obtained. In those transitions where y_3 and y_4 are the excited state variables, it is impossible to obtain ML transition paths. In particular, note that a ML transition path between states 0 and 3 would contain state 1 or 2.

Following is a complete statement of the procedure for finding a non-standard or non-generalized state assignment for column partitions that contain only 2-sets:

Step 1. Code the states with m state variables $y_1,y_2,$ . . .,y_m in such a manner that one tries to avoid assigning codes to states in 2-sets which are maximum distance apart. In general, best results are obtained if the states of the 2-sets can be assigned codes which are close together.

Step 2. Count the number of times each state variable y_i, i = 1,2, . . .,m, must be excited to effect a distance 2 or greater transition. This forms the count list.

Step 3. Let y_j and y_k be two highest value state variables in the count list that are not yet in a parity set. Generate a new state variable with y_j and y_k $(y_j \oplus y_k)$ which, together with y_j and y_k, form a parity set.

Step 4. Attempt to find the transition paths for each column partition. If a satisfactory set of transition paths exist, then the assignment procedure is complete. If not, then return to step 3 and generate another state variable.

The process continues until a set of transition paths is found, or the suggested upper bound of m + [m/2] is obtained.

For an 8-row flow table, one has a choice of trying to find a three-variable assignment, which in most problems is impossible, or to use a four-variable standard assignment. Unless the number of columns in the flow table is small, there is not much hope in easily finding a three-variable assignment. In most cases, it may be easier to use the standard four-variable assignment. The above procedure, however, will always generate this standard assignment.

Following is an example to illustrate the above procedure. Consider the six column partitions which were obtained from a flow table called Machine A.

α_1 = {0,9; 7,13; 3,1; 10,4; 2,8; 11,15; 6,14; 5,12}

α_2 = {0,6; 2,7; 3,13; 9,1; 10,15; 4,12; 14,8; 11,5}

α_3 = {9,4; 7,10; 0,12; 3,2; 15,6; 5,13; 8,14; 1,11}

α_4 = {0,14; 7,15; 9,12; 3,5; 4,6; 11,13; 10,8; 1,2}

α_5 = {7,10; 2,4; 15,12; 0,6; 3,13; 1,9; 8,14; 11,5}

α_6 {0,13; 7,15; 3,5; 9,14; 4,8; 1,11; 10,12; 6,2}

The codes assigned to the states of the m-cube (4-cube) are shown in Figure 5-3. No states that are in a 2-set are a maximum distance apart.

y_3y_4 \ y_1y_2	00	01	11	10
00	0	9	13	14
01	6	11	1	2
11	15	4	5	7
10	12	10	3	8

FIGURE 5-3. Initial coding for Machine A

The number of times each state variable is excited in a distance 2 or greater transition is tabulated next.

State Variable	Number of Times Excited
y_1	14
y_2	15
y_3	12
y_4	16

On the basis of this count list, a new state variable y_5 should be generated from state variables y_2 and y_4. Therefore, $y_5 = y_2 \oplus y_4$; the Karnaugh map representation of this assignment is shown in Figure 5-4.

$y_5 = 0$

y_3y_4 \ y_1y_2	00	01	11	10
00	0			14
01		11	1	
11		4	5	
10	12			8

$y_5 = 1$

y_3y_4 \ y_1y_2	00	01	11	10
00		9	13	
01	6			2
11	15			7
10		10	3	

FIGURE 5-4. State assignment for Machine A

It can be shown that a set of satisfactory transition paths exist for each column partition, and therefore, the assignment is satisfactory. The amount of work involved in finding this state assignment is significantly less than that associated with using Hazeltine's method, or a completely trial and error approach.

For the more general case where k-sets are not restricted to 2-sets, the above algorithm can be modified by considering transition pairs instead of 2-sets. Each k-set can be described by a set of transition pairs. For example, consider a 3-set consisting of states 1, 2, and 3, with state 1 as the stable state. There are three possible sets of transition pairs for a 3-set: (1,2)(1,3); (1,2)(2,3); (1,3)(2,3), where the parenthesis ( ) denotes the transition pair. On the other hand, there is only one transition pair for each 2-set. It is apparent that more flexibility is associated with k-sets larger than 2-sets. To account for this, in the initial coding, attention is given primarily to keeping the codes of the 2-sets as close together as possible. Little effort is applied to keeping the states of the k-sets larger than 2-sets close together.

From the initial coding, a set of transition pairs is obtained for each k-set. The transition pairs are selected such that the total number of states in the resulting transition path associated with each k-set is minimal. Consider a 3-set of states 1, 2, and 3 to illustrate this point. The states are encoded in the following 4-cube and all other states

are ignored.

Y_3Y_4 \ Y_1Y_2	00	01	11	10
00	1		2	
01				
11			3	
10				

Let state 3 be the stable state. If the transition pairs for this k-set are (1,3)(2,3) and each transition pair has an independent path, the total number of states in the transition path would be 7. However, if the transition pairs were (1,2)(2,3), the total number of states in the transition path would be 5. Therefore, the second set of transition pairs is preferred.

Transition pairs which consist of states that are adjacent are most desirable in that the resulting transition path contains just the two states. It should be pointed out that the pairing of the states in forming the transition pairs for a k-set can be changed, if it is discovered that another set would produce a satisfactory set of transition paths.

The means of determing which independent state variables are to be used in generating the additional state variables is identical to the procedure already discussed. The two state variables that are excited most frequently in effecting a distance 2 or greater transition, and not already in a parity set, are used to generate each additional state variable. After each additional state variable is obtained, an attempt

is made to find a satisfactory set of transition paths for each column partition. The transition paths associated with transition pairs of the same k-set can have states in common. In fact, it is desirable for transition pairs of the same k-set to share states in the transition path, since this would result in fewer total states in the transition path for the k-set. Trial and error is involved in finding the transition path for each transition pair.

Following is the complete procedure for finding the assignment for the general case:

Step 1. Code the states with m state variables in such a manner than one tries to avoid assigning code words a maximum distance apart to states of 2-sets. Little attention is given to k-sets larger than 2-sets other than a small effort to keep the states of the k-sets as close together as possible.

Step 2. A set of transition pairs is obtained for each k-set in a column partition and is selected in such a manner that the total number of states in the resulting transition path for each k-set would be minimal. One can change the pairing of transition pairs for a k-set at any point in the procedure, if it appears that a different set of transition pairs is more desirable.

Step 3. Count the number of times each state variable must be excited to effect a distance 2 or greater transition. This forms the count list.

Step 4. Let y_j and y_k be the two highest value state variables in the count list that are not yet in a parity set.

Generate a new state variable with y_j and y_k ($y_j \oplus y_k$), which together with y_j and y_k, form a parity set.

Step 5. Attempt to find the transition paths for each column partition. If a satisfactory set of transition paths exist, then the assignment procedure is complete. If not, return to step 4 and generate another state variable. The process continues until a set of transition paths is found, or the suggested upper bound m + [m/2] is obtained.

To illustrate this procedure, consider the five column partitions taken from a flow table called Machine B. The stable state of each k-set is underlined (_).

α_1 = {0,1; 3,4,5; 6,7; 8,9,10; 12,13,14; 11,15,2}

α_2 = {3,15; 5,10,12; 0,9; 6,8,14; 1,7,11; 13,2,4}

α_3 = {4,14,8; 2,12,6; 1,11,5; 3,13,15; 7,9; 0,10}

α_4 = {2,4; 6,8; 10,11,12; 14,15; 1,13,7; 0,3,5,9}

α_5 = {10,9,8; 4,5,13; 11,12,6; 0,3,7; 1,13,14,15}

The first step is to encode the states into a 4-cube. Since there are few 2-sets, it is possible to place the states of each 2-set adjacent in the 4-cube. The rest of the states are encoded at a somewhat random manner with some attention being given to states that appear in the same k-set. It is desirable to give the states in the same k-set codes which are a minimum distance apart. In this example, it is done to some extent, but not much effort was made in this direction. The initial code is shown in Figure 5-5.

Y_3Y_4 \ Y_1Y_2	00	01	11	10
00	0	10	8	14
01	1	5	4	15
11	13	11	2	3
10	9	12	6	7

FIGURE 5-5. Initial code for Machine B

A set of transition pairs for the states of each column partition is obtained in accordance with step 2. States that are adjacent make ideal transition pairs. In column partition α_3 it turned out, somewhat unexpectedly, that the states in each transition pair of all the k-sets are adjacent. Following are the column partitions, where the parenthesis ( ) denotes the transition pairs in each k-set.

α_1 = {(0,1); (3,4)(4,5); (6,7); (9,10)(10,8); (12,14)(14,13); (11,2)(2,15)}

α_2 = {(3,15); (5,10)(10,12); (0,9); (11,7)(1,7); (6,8)(8,14); (4,2)(2,13)}

α_3 = {(4,8)(8,14); (2,6)(6,12); (1,5)(5,11); (13,3)(3,15); (7,9); (0,10)}

α_4 = {(2,4); (6,8); (11,12)(12,10); (1,13)(13,7); (14,15); (9,0)(0,5)(9,3)}

α_5 = {(10,9)(10,8); (4,5)(5,13); (11,12)(12,6); (0,7)(3,7); (1,13)(1,15)(15,14)}

The next step is to count the number of times each state

variable y_1, y_2, y_3, and y_4 must be excited to effect a transition between states of the transition pairs that are a distance 2 or greater. State variables, which must be excited to effect a transition between states of transition pairs that are adjacent, are not counted in this procedure. Note that the transitions of column partition α_3 add nothing to this count. Listed is the count obtained in this step.

State Variable	Number of Times Excited
y_1	7
y_2	9
y_3	9
y_4	6

On the basis of this count list, state variable y_5 can be obtained from y_2 and y_3; therefore, $y_5 = y_2 \oplus y_3$. Shown below is the Karnaugh map representation for this assignment. It turns out that all transition paths are obtained easily.

$y_5 = 0$

y / y_3y_4 \ y_1y_2	00	01	11	10
00	0			14
01	1			15
11		11	2	
10		12	6	

$y_5 = 1$

y_3y_4 \ y_1y_2	00	01	11	10
00		10	8	
01		5	4	
11	13			3
10	9			7

FIGURE 5-6. State assignment for Machine B

The final code depends on the flow table characteristics and the initial encoding. On some occasions, when only one or very few transitions in a column partition cannot produce a set of transition paths, there might be some advantage in considering a slight change in the initial code. A slight change in the initial code may produce a satisfactory assignment and avoid the addition of another state variable to the assignment.

VI. SUMMARY

All the research effort was concerned with finding state assignments for non-normal asynchronous sequential circuits. Generally, state assignments for flow tables that may operate in the non-normal mode require fewer internal state variables than assignments for flow tables that operate in the normal mode. There is a lack of techniques for finding good assignments for the non-normal mode case. Presented in this paper are two generalized state assignments, which are functions only of the number of rows in a flow table, and an assignment method for finding non-standard or non-generalized assignments for non-normal flow tables.

Assignment 1 has a suggested bound of $m + [\log_2 m]$ state variables for a 2^m-row flow table, where [ ] means "next lowest integer". The algorithm for generating the codes is easy and straight-forward to implement. It is shown that Assignment 1 satisfactorily encodes all maximum-distance principal-column partitions, and the less than maximum-distance principal-column partitions that produce ML transition paths. An extensive search was made of flow tables up to and including 2^{25} states where ML transitions are not possible, and it was found that Assignment 1 also satisfactorily encodes these cases. From this, it seems safe to conclude that Assignment 1 satisfactorily encodes all principal-column partitions.

Assignment 2 has a suggested bound of $m + [m/2]$ state variables for a 2^m-row flow tables. In addition to being easy

to generate the code, all the transitions between states encoded by Assignment 2 can be ML transitions. Assignment 2 satisfactorily encodes all principal-column partitions, and furthermore, the transition paths are easy to obtain for this case; specific algorithms are given to accomplish this.

Both generalized state assignments are group (linear) codes with the maximum distance being m for states of a 2^m-row flow table. A general proof showing that these assignments are standard assignments has not been found; however, it is the author's opinion that these assignments are standard. Worst-case situations have been constructed and it has never been necessary to exceed the suggested bounds.

Trial and error is mainly used in finding the transition paths for the general case. Both generalized state assignments have a desirable feature of allowing the n-tuples of the n-cube to be partitioned into equivalence classes according to parity combinations in the parity sets. Crossover can occur only between n-tuples (states of transition paths) that are in the same equivalence class. This significantly reduces the amount of work involved in finding the transition paths.

Good techniques for finding state assignments for non-normal flow tables, which are not standard assignments, do not exist in the literature. A state assignment technique is presented that, in the author's opinion, is better than anything previously developed. The assignment procedure generally produces assignments which require fewer state variables, and is more systematic than the procedures presently available.

BIBLIOGRAPHY

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

2. D.A. Huffman, "The synthesis of sequential switching circuits," J. Franklin Inst., vol. 257, pp. 161-190, 275-303, March and April 1954.

3. D.A. Huffman, "A study of the memory requirements of sequential switching circuits," Research Lab. of Electronics, M.I.T., Cambridge, Tech. Rept. 293, April 1955.

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

5. R.J. Smith, J.H. Tracey, and G.K. Maki, "Computer generated next-state equations in asynchronous sequential circuits," 1968 Proc. 20th Ann. Southwestern IEEE Conf., April 1968.

6. 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.

7. 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.

8. G. Saucier, "Encoding of asynchronous sequential networks," IEEE Trans. on Electronic Computers, vol. EC-16, pp. 365-369, June 1967.

9. B. Hazeltine, "Encoding of asynchronous sequential circuits," IEEE Trans. on Electronic Computers, vol. EC-14, pp. 727-729, October 1965.

10. A.O. Lindstrum, Jr., Abstract Algebra. California: Holden-Day, 1967, pp. 46-57.

11. G. Birkhoff and S. MacLane, A Survey of Modern Algebra. New York: MacMillan Company, 1962, pp. 142-157.

12. W.W. Peterson, Error-Correcting Codes. Massachusetts: MIT Press, 1961, pp. 30-46.

VITA

Gary Keith Maki was born in Marquette, Michigan, on July 25, 1943. He graduated from W.G. Mather High School at Munising, Michigan, in June of 1961. A Bachelor of Science Degree in Electrical Engineering was received from Michigan Technological University in June, 1965. The Master of Science Degree in Electrical Engineering was received from the University of Missouri at Rolla in May of 1968.

During the summer months of 1966 and 1967, the author was employed by the Naval Weapons Center, China Lake, California, as an Electronic Engineer. From September, 1965, to June, 1968, the author was on the staff of the Electrical Engineering Department at the University of Missouri at Rolla.

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

He is married to the former Alice Kesti of Munising, Michigan.