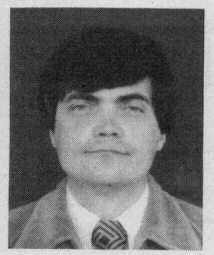
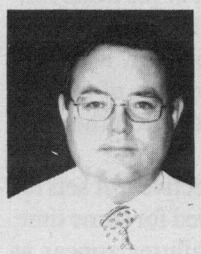


G. A. Jullien (M'70) was born in Wolverhampton, England on June 16, 1943. He received the B.Tech. degree from Loughborough University of Technology, Loughborough, in 1965, the M.Sc. degree from the University of Birmingham, Birmingham, in 1967 and the Ph.D. degree from Aston University in 1969, all in electrical engineering.

From 1961 until 1966, he worked for English Electric Computers at Kidgrove, England, first as a student apprentice and then as a data processing engineer. From 1967 until 1969 he was employed as a Research Assistant at Aston University in England. Since 1969 he has been in the Department of Electrical Engineering at the University of Windsor, Windsor, Ont., Canada and currently holds the rank of Professor. He is currently engaged in research in the areas of one- and two-dimensional digital signal processing, high-speed digital hardware and microprocessor systems. He also teaches courses on electronic circuits, microcomputer systems and digital signal processing. From 1975 until 1976 he was a visiting senior research engineer at the Central Research Laboratories of EMI Ltd., Hayes, Middlesex, England.

Dr. Jullien is a member of the Association of Professional Engineers of Ontario, and the American Society for Engineering Education.



William C. Miller was born in Toronto, Ont., Canada on September 20, 1937. He received the B.S.E. degree from the University of Michigan, Ann Arbor, and the M.A.Sc. and Ph.D. degrees from the University of Waterloo, Waterloo, Ont., all in electrical engineering.

Since 1968 he has been with the Department of Electrical Engineering at the University of Windsor, Windsor, Ont., where he is currently a Professor and member of the Signal Processing Laboratory Staff.

Dr. Miller is a registered Professional Engineer in the Province of Ontario.

Detection of Faults in Programmable Logic Arrays

JAMES E. SMITH, MEMBER, IEEE

Abstract—**A new fault model is proposed for the purpose of testing programmable logic arrays. It is shown that a test set for all detectable modeled faults detects a wide variety of other faults. A test generation method for single faults is then outlined. Included is a bound on the size of test sets which indicates that test sets are much smaller than would be required by exhaustive testing. Finally, it is shown that many interesting classes of multiple faults are also detected by the test sets.**

Index Terms—**Programmable logic arrays, fault detection, fault modeling, test generation.**

I. Introduction

PROGRAMMABLE logic arrays (PLA's) provide the logic designer with an economical way of realizing combinational switching functions [1]–[3]. The economy is achieved by manufacturing standard "blank" arrays, and, as a final step, "programming" the array to perform a particular set of functions. In some technologies programming is performed by using a custom mask for the final metalization step. Field programmable logic array (FPLA) technologies allow the user to program the array by blowing fusible links within the array.

Manuscript received August 9, 1977; revised September 1978.

The author is with the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, WI 53706.

As with any other logic circuit, PLA's must be tested to insure that they operate correctly. Essentially, three different testing schemes are possible. The first is to place special test circuitry on the array [3]. This special circuitry is then enabled by placing voltage levels on inputs and outputs which are beyond normal levels. This method is used to avoid the addition of pins for testing and checks the presence or absence of connections in the PLA. While this method allows the PLA to be tested quickly, a PLA tested in this way has not been tested under normal operating conditions. Furthermore, after such a PLA is placed in a system it may be difficult to apply the appropriate testing signals which require abnormal signal levels.

A second test method is to exhaustively apply all possible input vectors to the array and check to see if it responds correctly. Exhaustive testing more adequately reflects normal operating conditions, but it has the disadvantage of requiring rather sophisticated high-speed test equipment. While a manufacturer may have such equipment, a user in the field, who often needs to test a PLA, may not. In addition, as PLA technology advances and the number of inputs increases, exhaustive testing will become impractical even with high-speed equipment. Exhaustive testing can also be very difficult in a system environment where other logic typically separates the PLA from the "outside world." Here,

0018-9340/79/1100-0843$00.75

setting up test patterns and observing them can be very time consuming if testing has to be done exhaustively.

The third method is to select special test cases and to test the array for these cases only. This third method is the one considered in this paper. It overcomes most of the disadvantages of the other two methods, provided the size of the test set is relatively small. To be effective, however, a nonexhaustive test set must detect a large proportion of failures that are likely to occur, and it must be easy to select the set of tests to be used.

The problem of generating small and effective test sets for random (i.e., nonarray) logic has been studied for some time. In most of this paper it is assumed that failures appear as gate input or output lines stuck at logical values [4]. Using this assumption input sequences are generated which detect all (or most) such stuck-at faults [5].

The problem of generating test sets for PLA's differs from the problem generally considered on two counts. First, the circuit structure is different; PLA's essentially have only two levels of gates. The second and more important difference is that a more general fault model seems necessary because of the way PLA's are fabricated. The usual test generation techniques can be easily modified to take care of the first difference; however, the second requires additional study.

A brief description of PLA structures is presented in Section II, and the proposed fault model is discussed in Section III. The generation of tests which detect modeled faults is the topic of Section IV. Emphasis is placed on generating small test sets under a single fault assumption. Classes of multiple faults also detected are presented in Section V, and the paper concludes with Section VI.

II. The Structure of PLA's

Currently available PLA's typically consist of an input buffer and two arrays, the first of which effectively forms implicants (AND's) while the second forms logical sums (OR's) of the implicants. These are the AND array and OR array, respectively. Fig. 1(a) shows a two-level AND–OR network to be implemented with a PLA. Fig. 1(b) shows such an implementation using a fusible-link FPLA. Here, the AND array is implemented with diodes, and the OR array is implemented with bipolar transistors. An optional output inverter may be implemented in a PLA by using exclusive-OR gates as shown in Fig. 1(b). Other fabrication methods are possible, for example, using a MOSFET technology where the presence or absence of gate connections determine the function realized [1]. Typical PLA's and FPLA's currently available have 14–16 inputs, 48 or 96 product terms, and 8 outputs.

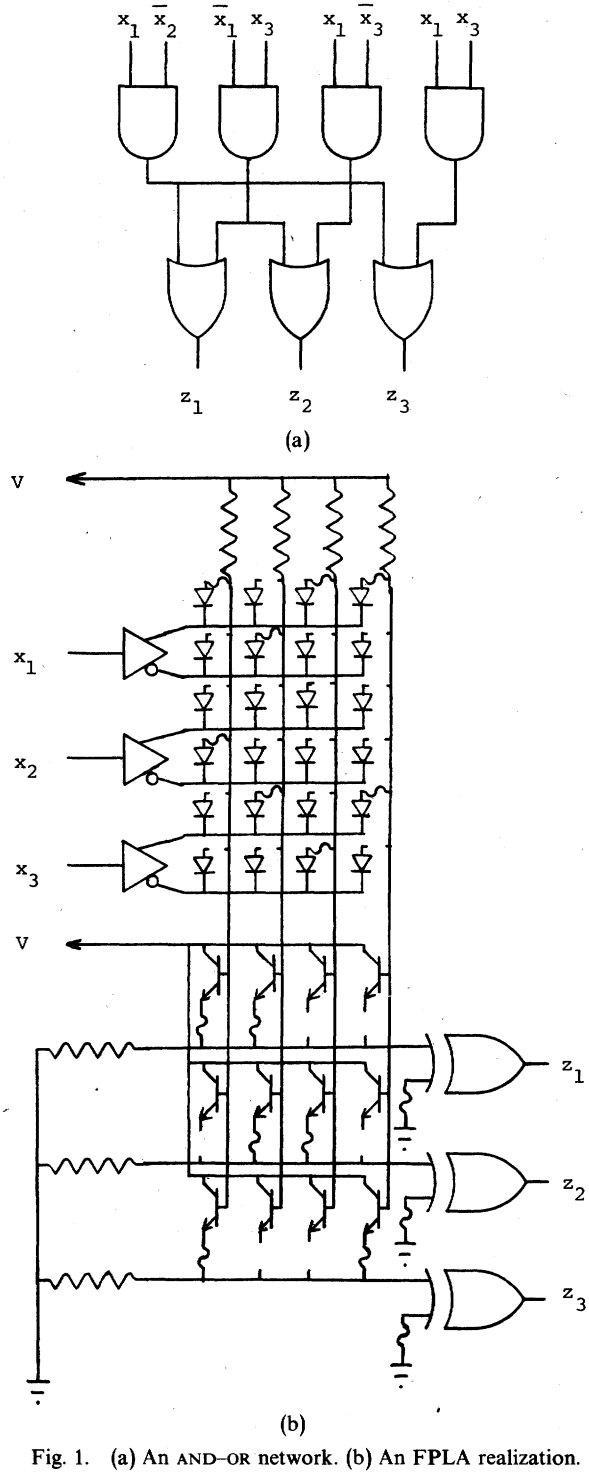


Fig. 1. (a) An AND–OR network. (b) An FPLA realization.

III. Modeling of Failures

In this section, a logical fault model is proposed for PLA's. A logical fault model allows us to avoid considering physical aspects of the circuits while generating tests and provides a large body of knowledge upon which we can build, namely the literature on stuck fault test generation.

The proposed fault model is similar to the one implicitly used in the first testing scheme mentioned earlier; that is, we test for incorrect logical connections in the AND and OR arrays. One justification for this is that a PLA is programmed by making or failing to make these connections. Hence, any programming errors would show up as an incorrect connection. Also, these "programming points" might be susceptible to failure while in use. For example, a fuse in an FPLA might blow under stressful use, or the metal fragments from an improperly blown fuse may migrate back together to form a connection [3].

Since AND gate inputs and OR gate inputs can be either

improperly connected or disconnected, modeled faults can be conveniently divided into four classes. First, if an input literal is disconnected from an AND gate, this causes the implicant to "grow" since the implicant becomes independent of some input variable. This growth of an implicant can be seen quite easily on a Karnaugh map [10]. These faults will be called *growth faults*. The set of all single growth faults will be denoted as G. If an AND gate becomes disconnected from an OR gate, this causes an implicant to disappear from the map of the function [10]. Hence, this set of single faults is the set of *disappearance faults* D. If an input literal becomes incorrectly connected to an AND gate, then the implicant "shrinks." These faults will be the set of *shrinkage faults* S. Finally, if an AND gate becomes incorrectly connected to an OR gate, then an implicant appears in the map of the affected function. Hence, these will be called *appearance faults* A.

To make test generation simpler, we use an assumption typically used for stuck-at faults in random logic—only one failure is present at a time. In order to make this assumption more palatable, it will later be shown that important classes of multiple faults are detected by any single fault test set.

Since logical connections will be tested using normal network paths, a test set designed for the proposed fault model will also detect other potential faults. To demonstrate this ability to "cover" other faults, we consider stuck type faults since this is consistent with some known failure mechanisms in semiconductor devices [6], [7], and because they have been widely studied.

In terms of an AND–OR equivalent network with optional output inverters, the faults are:

1) input inverters stuck-at-1 and 0,
2) AND gate inputs and outputs stuck-at-1 and 0,
3) OR gate inputs and outputs stuck-at-1 and 0,
4) output inverters stuck-at-1 and 0.

Using fundamental results concerning fault dominance and equivalence from [8], [9], the following theorem can be proved. A proof appears in Appendix I.

Theorem 1: A test set that detects all detectable G, D, S, and A faults also detects all the detectable stuck-at faults listed above except in PLA's where either of the following conditions holds:

1) some AND gate(s) are redundant; i.e., they can be deleted from the array without affecting the functions realized;

2) if there exists an OR gate output which is normally 1 whenever any AND gate output in the array is 1.

From Theorem 1, we see that a test set for the assumed faults in G, D, S, and A not only detects incorrect connections at programming points, but it also detects all the stuck-at faults most of the time. In addition, improper connections to optional inverters are trivially detected since improperly inverted outputs can be detected by any input vector.

If stuck-at faults are considered to have a significant probability of occurrence, then one can check for the unusual circumstances under which some stuck-at faults may go undetected. Exception 1) in Theorem 1 can be checked by verifying that the disappearance of each implicant from at least one function is detected. If a redundant implicant is found, it can be removed by reprogramming the array, or a test can be generated to detect the simultaneous disappearance of all the redundant implicants that have an input literal in common.

L

C array	D array
$x_1x_2x_3$	$z_1z_2z_3$
10 ×	101
0 × 1	110
1 × 0	010
1 × 1	001

Fig. 2. Cubical specification of the network shown in Fig. 1.

Exception 2) can result in an undetectable stuck-at fault only if it causes some primary output line to be a 1 for all tests. This condition can be easily checked after a test set has been generated, and a test which causes the output to be 0 can be added to the test set.

IV. An Overview of a Test Generation Algorithm

We now consider the problem of generating a test set T for the fault set $F = G \cup D \cup S \cup A$ as described in the previous section. Much of the detail of the algorithm is omitted because many of the techniques used (e.g., path sensitization) have already been widely discussed in the literature.

A. Notation

In order to facilitate the description and computer implementation of an algorithm, cubical notation and operations are used [13]–[15]. The functions which a PLA realizes can be represented as arrays of cubes. Fig. 2 shows this representation for the function of Fig. 1. We use a vector $\boldsymbol{E}$ and pair of arrays $\boldsymbol{L} = (\boldsymbol{C}, \boldsymbol{D})$, where $\boldsymbol{C}$ and $\boldsymbol{D}$ have the same number of rows. Rows in the $\boldsymbol{C}$ array correspond to product terms and will be labeled $c^1, c^2, \cdots, c^n$. Columns correspond to input variables and will be labeled $c_1, c_2, \cdots, c_m$, and c_j^i represents the entry in the ith row and jth column. If $c_j^i = 1$ then x_j appears uncomplemented in the ith product term; if $c_j^i = 0$ then x_j appears complemented, and if $c_j^i = x$ then x_j does not appear in the product term.

The $\boldsymbol{D}$ array contains n rows corresponding to product terms and l columns corresponding to outputs. If $d_j^i = 1$ then the ith product term is used to form output j, and if $d_j^i = 0$ it does not contribute to output j.

When optional inverters can be connected to outputs of the OR array, the vector $\boldsymbol{E}$ indicates those outputs which are to be complemented. Component $e_i = 1$ if output i is to be complemented, and is a 0 otherwise. Whether optional output inversion is permitted will not affect the tests generated. It will only affect the output vectors the PLA produces under given tests.

In the sequel we will use the cubical subsuming relation $\sqsubseteq$ and the # and $\sqcup$ operations. Their descriptions can be found in Appendix II, as well as in [13], [15].

We also define an operation which allows us to change particular coordinates in the C and D arrays. $C \circ c_j^i \to \alpha$ is an array which is identical to C except that c_j^i is $\alpha, \alpha \in \{0, 1, X\}$. $D \circ d_j^i \to \beta$ is an array which is identical to D except that d_j^i is $\beta, \beta \in \{0, 1\}$.

B. An Algorithm

We are now ready to present a test generation algorithm. At the top level, the algorithm consists of seven steps.

Informally, the steps are:

1) input a description of the PLA to be tested;
2) select a fault $f \in F$ which has not yet been detected;
3) generate a test t for f and add it to T; if no test exists, output f and indicate that it is undetectable;
4) determine which other faults in F are detected by t;
5) if there are still faults which have not been considered, go to 2;
6) using the E vector which indicates inverted outputs, determine test results for the fault-free PLA;
7) output the test set T and the correct test results.

Optionally, if stuck-at faults are deemed significant, after 6) one might check for the exceptional conditions in Theorem 1 and generate any necessary additional tests.

For ease of computer storage and manipulation, it is reasonable to store the specification of a PLA as its L array and its E vector (if applicable) in a format similar to the one described earlier. Hence, 1) can be simplified if input is also done using this format.

2) and 5) can be implemented by simply listing all possible faults initially and "marking" them as they are detected or as they are discovered to be undetectable. The order in which the faults are considered can potentially affect the size of the final test set. We have found that, in general, the closer a fault site is to a primary output, the more tests it has. Hence, a useful heuristic for reducing the number of tests is to first consider faults with fewer tests, i.e., S and G faults, then to consider faults with more tests, i.e., D and A faults. We have found this to work well; after tests for G and S faults have been generated, a very large number (if not all) of the D and A faults are detected.

To generate a test for one of the faults in 3), path sensitization [4] is used. To automate this process for the two-level circuits being dealt with, the cubical notation and operators are used.

A modeled fault in a PLA realizing array $L = (C, D)$ results in a PLA that can be described as an array L'. For example, if input x_j becomes disconnected from the ith implicant, then $L' = (C \circ c_j^i \to x, D)$. Then, a test can be generated by examining the arrays L and L'.

If a member of G is present, say f, then the array of the faulty PLA is $L' = (C \circ c_j^i \to x, D)$ as explained above. Hence, we say $C' = C \circ c_j^i \to x$. The first phase of path sensitization dictates that inputs should be applied so that the faulty implicant (AND gate) yields a different output under f than its normal value. The set of input vectors that will do this is $c'^i \# c^i \sqcup c^i \# c'^i$. We observe that $c^i \sqsubseteq c'^i$; hence, $c^i \# c'^i = \phi$, and only $c'^i \# c^i$ is needed. This is just a single cube which is the same as c^i except that its jth coordinate is complemented.

Next, the error must be propagated to an output. It can be propagated to the kth output only if $d_k^i = 1$. Hence, such a k is chosen. Now let Γ_k^i be an array consisting of all those c^p such that $d_k^p = 1$, and $p \neq i$. These represent all the implicants besides the faulty one which are used to form output k.

To propagate the error to output k, a test must cause all these implicants to be 0. Hence, the set of tests for the fault in question is defined by the array

$$(c'^i \# c^i) \# \# \Gamma_k^i, \tag{1}$$

and one of the tests defined by the array should be chosen.

If this array is empty, however, another value of k must be chosen where $d_k^i = 1$. When all such k have been exhausted, it must be concluded that no test exists.

If a member of S is to be detected, $L' = (C \circ c_j^i \to 0, D)$ or $L' = (C \circ c_j^i \to 1, D)$ depending on whether x_j or x_j, respectively, is connected to c^i. The complete set of tests through output k is then

$$(c^i \# c'^i) \# \# \Gamma_k^i$$

due to a similar argument as before.

If a member of D is to be detected, say the ith implicant disappears from the jth output, then

$$L' = (C, D \circ d_j^i \to 0).$$

The set of possible tests is defined by

$$c'^i \# \Gamma_j^i.$$

If a member of A occurs, then

$$L' = (C, D \circ d_j^i \to 1),$$

and the set of tests is defined as

$$c^i \# \Gamma_j^i.$$

Step 4) of the algorithm outlined above is essentially the opposite of 3), but the concept of path sensitization is still quite useful. An efficient method for determining the faults detected by a test t is to first construct an array $L = (\hat{C}, \hat{D})$ where $\hat{C}$ includes all those rows of C which cover t, and $\hat{D}$ includes the rows of D which correspond to rows in $\hat{C}$. After forming $\hat{L}$, the columns of $\hat{D}$ can be examined to determine which outputs are on sensitized paths. For example, a column of all 0's indicates that some OR gate has all 0 inputs. Hence, sensitized paths for A and G faults pass through it.

If the jth output column is all 0's, then the appearance of any implicant in $\hat{C}$ as an input to the jth OR gate is detected. Hence, all A faults involving implicants in $\hat{C}$ and the jth OR gate are detected by the test t.

By comparing the test vector with implicants *not* in the $\hat{C}$ list, G faults also detected can be determined. The test vector t and implicant c^k have a *single 1–0 conflict* (*single 0–1 conflict*) if t has a 1(0) in some position where c^k has a 0(1) and they have the same value in every other position except where x appears in c^k. For example $t = 011101$ and $c^k = 00x1x1$ have a single 1–0 conflict in the second position. If t and c^k have a single 1–0 (0–1) conflict then c^k is

not in $\hat{C}$, but if the input variable in the disagreeing position is disconnected (i.e., there is a growth fault), then the output of the AND gate realizing c^k goes from 0 to 1. Hence, if d^k has a 1 in some column where $\hat{D}$ is all zeros then the fault is sensitized to the corresponding output and is detected.

A column in $\hat{D}$ with only one 1 indicates that a path is sensitized through the OR gate corresponding to that column. S and D faults are potentially detected over this path. The D fault involving the implicant that yields the single 1 is detected because if this implicant should disappear, the OR gate output would then change from 1 to 0.

To determine detectable S faults, let c^k be an implicant where d^k contains the single 1 in an output column of $\hat{D}$. If a position in c^k contains an x and the same position in t is a 0(1), then when x_j is the input variable for this position the S fault corresponding to $x_j(\overline{x_j})$, being erroneously connected, is detected.

Step 6) of the algorithm, if it is required, can either be performed as indicated, or it can be performed simultaneously with 3).

C. Discussion

The algorithm given here is not the only one, of course. Many variations are possible; two of them will now be briefly discussed.

In order to simplify the implementation of the algorithm, it may be desirable to omit 4) and 5), and simply generate a test for every fault. This will lead to larger test sets, but, as we shall see, the resulting test set would still be relatively small. The advantage is that it would eliminate much of the bookkeeping required by the algorithm.

Some of the computation required by the sharp operation involving the Γ_k^i can be eliminated [e.g., (1)]. This is because a single test for a fault is needed rather than all of them. Hence, in evaluating (1), one can sharp ($c'^i \# c^i$) with the first member of Γ_k^i and stop when one cube is produced. Then this cube can be sharped with the next member of Γ_k^i, and again only the first cube is taken. One continues through the remainder of Γ_k^i in this manner. If no cube is produced by one of these sharps, it is necessary to backtrack and look for a second cube formed by the previous sharp. If this ends in failure, more backtracking may be necessary. In the worst case, all of the work required by a complete sharping will be necessary. In many cases, however, the partial sharping with backtracking will be successful, thus lowering the average case computational complexity. The disadvantage of this method is that additional bookkeeping and programming are required by the backtrack procedure.

D. Example

We will now generate a test set for a PLA whose L array and E vector are shown below.

	C	D	E
L:	$x_1x_2x_3x_4$	$z_1z_2z_3$	$z_1z_2z_3$
	1 × 0 1	0 1 0	0 0 1
	× 1 0 1	1 0 1	
	1 0 × ×	0 1 1	
	0 0 × 1	1 1 0	

We will denote a fault by the effects it has on the L array. For example, a G fault will simply be denoted as $c_j^i \to x$ when L' resulting from the fault is $L' = (C \circ c_j^i \to x, D)$.

Following the heuristic of first generating tests for faults close to the inputs, we begin by considering G faults. The first such fault is ${}_1c^1 \to x$, and $C = C \circ {}_1c^1 \to x$. Then $c'^1 \# c^1 = 0X01$. By examining d^1, we see that an error can only be propagated to z_2. Both c^3 and c^4 contribute to z_2, so a test must be included in

$$0 \times 01 \# \begin{vmatrix} 1 & 0 & \times & \times \\ 0 & 0 & \times & 1 \end{vmatrix}.$$

Hence, $t_1 = 0101$ is the only test. Now, we must determine other faults detected by t_1. To do this we form $\hat{L} = (\hat{C}, \hat{D})$:

$$\hat{L} = \frac{\hat{C}}{\times 101} \; \frac{\hat{D}}{101}.$$

The single 1 in column 1 of $\hat{D}$ indicates that some S or D faults may be detected. This is the case since $d_1^2 \to 0$ is detected (disappearance of implicant $x101$), and because t_1 has a 0 in position 1 and c^2 has an x, $c_1^2 \to 1$ is detected (an S fault). The single 1 in column 3 of $\hat{D}$ leads us to deduce that $d_3^2 \to 0$ is detected.

All 0's in column 2 indicates that some G and A faults may be detected. The appearance of d^2 is detected, i.e., $d_2^2 \to 1$, and c_1 and c_4 have a single 1–0 conflict in position 2, so the G fault $c_2^4 \to x$ is detected.

Using the same method, we generate $t_2 = 1111$ for the fault $c_3^1 \to x$. $\hat{L}$ is empty, so we consider all output columns to be all 0's, and some S faults are detected, in particular $c_3^2 \to x$ and $c_2^3 \to x$. For $c_4^1 \to x$, $t_3 = 1100$ is a test; it also detects $c_4^2 \to x$. The next G fault to be considered is $c_2^2 \to x$, and $t_4 = 1001$ is a test. t_4 also detects $d_1^1 \to 1$, $d_1^3 \to 1$, $d_3^3 \to 0$, $c_1^4 \to x$, $c_3^3 \to 1$, and $c_4^3 \to 0$. $t_5 = 0000$ is the test found for $c_1^3 \to x$. t_5 also detects $c_4^4 \to x$.

The S faults are the next to be considered. $c_2^1 \to 0$ yields the test $t_6 = 1101$; also detected are the faults $c_1^2 \to 0$ and $d_2^1 \to 0$. The fault $c_3^3 \to 0$ is detected by $t_7 = 1010$; $c_4^3 \to 1$ and $d_2^3 \to 0$ are also detected. Finally, $t_8 = 0001$ and $t_9 = 0011$ are needed to detect $c_3^4 \to 1$ and $c_3^4 \to 0$. Both of these tests also detect $d_1^4 \to 0$, $d_2^4 \to 0$, and $d_3^4 \to 1$. $c_2^1 \to 1$ is found to be undetectable.

Next, we turn to D and A faults, and find they have all been detected except for $d_3^1 \to 1$ which is undetectable.

The test set and correct outputs under the tests which are determined using the E vector are:

Test	Correct Output
$x_1x_2x_3x_4$	$z_1z_2z_3$
0 1 0 1	1 0 0
1 1 1 1	0 0 1
1 1 0 0	0 0 1
1 0 0 1	0 1 0
0 0 0 0	0 0 1
1 1 0 1	1 1 0
1 0 1 0	0 1 0
0 0 0 1	1 1 1
0 0 1 1	1 1 1

E. The Size of Test Sets

A bound on the size of a test set T generated for the proposed fault model will now be derived. Let m be the number of PLA inputs, n the number of product terms, and l the number of outputs. The largest test set that can be generated is only as large as the number of faults. The maximum number of G faults $\max|G|$ is $m \cdot n$. The maximum number of S faults $\max|S|$ is $2m \cdot n$, $\max|D| = l \cdot n$, and $\max|A| = l \cdot n$. However, if an implicant can grow in some coordinate it cannot shrink in the same coordinate, and vice versa. Therefore, $\max|G \cup S| = 2m \cdot n$. Furthermore, if an implicant can disappear, it cannot appear, and vice versa. Hence, $\max|D \cup A| = l \cdot n$. Therefore, $\max|F| \leq \max|G \cup S| + \max|D \cup A| = (2m + l) \cdot n$, and the largest necessary test set T is such that

$$|T| \leq (2m + l) \cdot n.$$

For a PLA with 16 inputs, 48 product terms, and 8 outputs (Signetics 82S100, for example), $|T| \leq 1920$. Exhaustive testing, on the other hand, would require $2^{16} \simeq 65\,000$ tests. The savings is substantial. Since most tests detect several previously undetected faults, and some faults are undetectable, the size of a test set is frequently a small fraction of $(2m + l) \cdot n$. In fact, for a realistic PLA, a test set that is less than $\frac{1}{4}$ of the bound is common. The savings of tests over an exhaustive test set can be measured in orders of magnitude for PLA's of a realistic size.

It should be pointed out, however, that in exchange for the considerable savings in testing time, one gets less testing coverage than with exhaustive testing. That is, some nonmodeled fault may slip by undetected with the nonexhaustive test set proposed here. For example, some stuck-at faults may not be detected under certain unusual circumstances (see Theorem 1). Nevertheless, the coverage provided by this test set is good, and provides a practical alternative to exhaustive testing.

V. Detection of Multiple Faults

Thus far, we have discussed the detection of the faults in F (all single faults of the four types described earlier). In practice, this single fault assumption might not be valid. However, it does seem justifiable to assume that if multiple failures occur, they will not be randomly distributed among the four types of failures. For example, if several fuses in an FPLA are not blown then only members of S and A are present. Furthermore, under normal operation, independent failures of some types may be more prevalent than others, with the likelihood of a failure of a given type being determined by the technology used. In this section we will discuss some interesting classes of multiple faults that are detected by a test set generated for F. We will let T denote any such single fault test set.

First, we consider multiple faults that are combinations of single faults that are all of the same type. Such faults are of interest because some technologies may be much more likely to have some fault type than the others; in such a situation a multiple fault would be likely to have all its components of the same type.

Theorem 2: T detects any detectable combination of A faults.

Proof: A test for any A fault causes the output of the affected OR gate to be a 0 normally, with the output of the appearing implicant being a 1. Consequently, if the implicant appears, the OR gate output changes from 0 to 1. If any additional A faults should occur, they cannot possibly cause the OR gate output to change back to a 0. Hence, any combination of A faults including a single detectable A fault must be detected by T.

Now consider a combination of A faults that are detectable, but which contains no detectable single A fault. A test t for the combination must cause at least one OR gate output to be a 0 normally and a 1 under fault. Let OR_1 be such a gate, and let A_1 be the subset of A that appears as an implicant of OR_1 under the fault. Then the test t must also detect the combination of faults A_1. Now, let A_1' be the subset of A_1 that have implicants that are 1 under the test t. The set of faults A_1' must be nonempty, and is detected by T also. It should also be clear that any subset of A_1' is also detected by T, but this contradicts the assumption that no single member of the original combination was detectable. Hence, there cannot be such a detectable combination. □

Theorem 3: T detects any detectable combination of S faults.

The proof of Theorem 3 is similar to the one for Theorem 2 although slightly more complex, and has consequently been omitted. □

Theorems for G and D faults similar to Theorems 2 and 3 do not hold. This is because two or more undetectable faults of these types may be detectable in combination [11]. Any combination of G or D faults containing at least one detectable single fault is detected by T, however.

In order to guarantee the detection of multiple G and D faults we need to make some assumptions about the detectability of single faults in PLA's. When stuck-at faults are considered in general combinational circuits, it is frequently assumed that all stuck-at faults are detectable; that is, that circuits are irredundant. An undetectable stuck-at fault corresponds to a line (or lines) which can be removed from the circuit without changing the function realized by the circuit.

When considering the more general fault model we are using, undetectable G and D faults can be eliminated by removing redundant connections as before; conversely, undetectable S and A faults can be eliminated by creating connections. These two methods of removing undetectable faults work against each other. That is, breaking a connection to eliminate an undetectable G or D fault creates an undetectable S or A fault; creating a connection to eliminate an undetectable S or A fault introduces an undetectable G or D fault. Hence, it may be impossible to have a PLA where all faults are detectable. Instead, we define two weaker types of irredundancy.

Definition 1: A circuit is ***G–D irredundant*** if all G and D faults are detectable.

Definition 2: A circuit is *S–A irredundant* if all *S* and *A* faults are detectable.

Most PLA's are programmed by either making connections or by breaking connections. Hence a PLA can be made *G–D* irredundant by making as few connections as are necessary to realize the desired function or by breaking as many as possible. Conversely, a PLA can be made *S–A* irredundant by making as many connections as possible or by breaking as few as are necessary.

When programming PLA's, common errors are the making of too many connections or too few. For example, several fuses might not be blown in an FPLA. Such programming errors appear as multiple *G* and *D* faults or as multiple *S* and *A* faults. Also, the failure mechanisms which lead to *G* faults and *D* faults are very similar, while those that lead to *S* faults and *A* faults are very similar. Hence, it follows that multiple faults containing only *G* and *D* components or faults containing only *S* and *A* faults should be more common than other mixtures of faults, and we should be interested in the detection of multiple *G* and *D* failures and of multiple *S* and *A* failures by our single fault test set *T*.

Theorem 4: In a *G–D* irredundant circuit, any detectable combination of *G* and *D* faults is detected by *T*.

Theorem 5: In an *S–A* irredundant circuit, any detectable combination of *S* and *A* faults is detected by *T*.

The proof to Theorem 4 follows immediately from the well-known result due to Schertz and Metze [12] for stuck-at faults in two-level circuits. That is, in an irredundant two-level circuit, a single stuck-at test set is a multiple stuck-at test set. The proof to Theorem 5 is very similar to the proof of Theorem 4.

As a consequence of Theorems 4 and 5 and the discussion preceding them, we see that the test set we have derived also has value as far as verifying the programming of a PLA. Furthermore, if multiple *G* and *D* faults are possible due to multiple programming errors, we should program the PLA to be *G–D* irredundant to make it easier to verify programming, i.e., the test set *T* can be used. And a similar statement holds if *S* and *A* faults result from programming errors.

VI. Summary and Conclusions

In this paper a fault model is formulated to aid in the detection of failures in PLA's. In some respects, it is a generalization of the standard stuck-at model, and existing test generation techniques are easily modified to include it. We believe the fault model is realistic, and at the same time it results in a relatively small number of tests. Test sets generated for the assumed fault model are also shown to detect large classes of multiple faults. This capability makes the test set useful for verifying the programming of a PLA.

Another interesting area of research is the location of faults in a PLA. While recent technological advances have made fault location in random logic less important than it once was, in FPLA's a fault may be "removed" by reprogramming the PLA, either by using unused portions of the PLA, or by correcting programming errors. Therefore, if a fault can be located, the PLA may still be useable.

After this paper was submitted for publication, two other papers that discuss the same topic [16], [17] have appeared. These papers consider test generation for IBM PLA's [2], which differ from the usual PLA structure in that they have a more general input decoder structure. These papers propose essentially the same fault model as the one here, and show that this fault model yields tests that also detect a wide variety of short circuit faults. This serves as further evidence that the fault model proposed here yields tests that detect a wide variety of possible faults.

Appendix I
Analysis of Stock-At Fault Coverage

In this appendix we prove Theorem 1, which demonstrates the stuck-at fault coverage of a test set generated for *G*, *D*, *S*, and *A* faults.

Definition: Two logical faults in a PLA are *equivalent* if the output pattern of the PLA is the same for both faults under all possible input patterns.

Lemma 1: In a two-level AND/OR PLA with optional output inverters:

1) for any AND gate, the input stuck-at-0 and the output stuck-at-0 faults are equivalent.

2) For any OR gate, the input stuck-at-1 and the output stuck-at-1 are equivalent.

3) For any inverter the output stuck-at-1 (0) is equivalent to the input stuck-at-0 (1).

4) If an optional inverter is attached to an OR gate output, the input of the inverter stuck-at-1 (0) is equivalent to the OR gate output stuck-at-1 (0).

Proof: Immediate from results on fault equivalence in [8], [9]. □

Now, using the equivalence above and the transitivity of the equivalence relation, we see that each single stuck-at fault is equivalent to at least one of the following:

1) AND gate inputs stuck-at-1;
2) OR gate inputs stuck-at-0;
3) AND gate outputs stuck-at-0;
4) OR gate outputs stuck-at-1;
5) AND gate outputs stuck-at-1;
6) OR gate outputs stuck-at-0;
7) input inverter outputs stuck-at-1;
8) input inverter outputs stuck-at-0.

Lemma 2:

1) Each *G* fault is equivalent to some AND gate input stuck-at-1 and vice versa.

2) Each *D* fault is equivalent to some OR gate input stuck-at-0 and vice versa.

Proof: See [10], [18]. □

In the following Lemmas, recall that an implicant in a PLA is redundant if it can be completely removed from the PLA without changing any of the logic functions realized by the PLA. Let *T* be any test set that detects all detectable *G*, *D*, *S*, and *A* faults.

Lemma 3: *T* detects AND gate outputs stuck-at-0 when there are no redundant implicants in the PLA.

Proof: Consider an AND gate output that contributes

irredundantly to output function z_i and let input j of the OR gate, whose output is z_i, be connected to the affected AND gate. Then a test for input j stuck-at-0 must be in T, and the AND gate output is normally 1 when the test is applied. If the AND gate output becomes stuck-at-0 then input j goes from 1 to 0 as does the output z_i, thus, indicating the presence of the fault. □

Lemma 4: T detects all OR gate outputs stuck-at-1 unless an OR gate output is 1 whenever any AND gate output is 1.

Proof: For some input t, let some OR gate output be 0 and some AND gate output be 1. Then t detects the appearance of the implicant realized by the AND gate at the OR gate inputs. Furthermore any test for this appearance fault must place a 1 on the AND gate output and a 0 on the OR gate output. Some such test must be in T, and must also detect the OR gate output stuck-at-1. The lemma follows immediately. □

Lemma 5: T detects all AND gate outputs stuck-at-1 unless an OR gate output is 1 whenever any AND gate output is 1.

Proof: If no OR gate output is 1 whenever any AND gate output is 1 then all OR gate outputs stuck-at-1 are detected according to Lemma 4. If an AND gate feeds OR gate i, then the test for the output of gate i stuck-at-1 also detects the AND gate output stuck-at-1. □

Lemma 6: T detects all OR gate outputs stuck-at-0 if there are no redundant implicants in the PLA.

Proof: If there are no redundant implicants then the disappearance of each implicant from some output function must be detectable. A test for a D fault must put a 1 on the output of the implicant being tested, which places a 1 on all the OR gate outputs fed by the implicant. Since each OR gate is fed by at least one implicant, its output must be 1 for any test involving a disappearance of the implicant (regardless of the output function from which it disappears). Hence, each OR gate output has a 1 for some test, leading to the detection of its output stuck-at-0. □

Lemma 7: T detects all detectable input inverter outputs stuck-at-1.

Proof: If an input inverter stuck-at-1 fault is detectable, there must be some input pattern which sensitizes a path or multiple paths (which may happen since there may be reconvergent fanout) to some output. Consider an AND gate on such a path (or on any one of the paths if multiple paths are sensitized). Say input j of the AND gate is fed by the inverter. Then the test for the inverter stuck-at-1 detects the AND gate input j stuck-at-1 as well, and input j stuck-at-1 must, therefore, be detectable. Now, consider any test for input j stuck-at-1 (equivalently, a growth fault, see Lemma 2). This test must also detect the inverter stuck-at-1 since it implies a sensitized path from the inverter. □

Lemma 8: T detects all detectable input inverter outputs stuck-at-0 if there are no redundant implicants in the PLA.

Proof: If there are no redundant implicants, each AND gate output stuck-at-0 is detectable through some output (Lemma 3). A test for an AND gate output stuck-at-0 also sensitizes a path for an inverter output stuck-at-0, and if the inverter fault is detectable, it must feed at least one AND gate. □

Theorem 1: A test set that detects all detectable G, D, S, and A faults also detects all the detectable stuck-at faults at inverter, AND gate, and OR gate inputs and outputs except in PLA's where either of the following conditions hold:

1) some implicant(s) are redundant; i.e., they can be deleted from the array without affecting the functions realized;

2) if some OR gate output is normally a 1 whenever any implicant in the array is 1.

Proof: Immediate from the preceding lemmas. □

Appendix II
Cubical Relations and Operations

The representation of logic functions in terms of cubes is discussed in Section IVA, and most textbooks in logic design use a notation of this type. Several cubical operations and relations are also used, but they are not as common and are defined in this appendix.

Let a and b be two cubes or n-tuples of elements $a_i, b_i \in \{0, 1, x\}$. Then a *subsumes* b, $a \sqsubseteq b$ if $a_i \sqsubseteq b_i = \varepsilon$ for all i where the table below defines the coordinate subsuming relationship ($a_i \sqsubseteq b_i$).

	$a \sqsubset b_i$	b_i = 0	1	x
a_i	0	ε	ϕ	ε
	1	ϕ	ε	ε
	x	ϕ	ϕ	ε

Alternatively, b is said to *cover* a. Example:

$$01x \sqsubset x1x.$$

Given a list of cubes $\boldsymbol{C}$, we *absorb* $\boldsymbol{C}$ if we remove all cubes that subsume one or more other cubes in $\boldsymbol{C}$. If $\boldsymbol{B} = \{b^1, b^2, \cdots\}$ and $\boldsymbol{C} = \{c^1, c^2, \cdots\}$ are sets of cubes of the same number of variables, the *union* of these arrays $\boldsymbol{B} \sqcup \boldsymbol{C}$ is the absorbed set $\boldsymbol{B} \cup \boldsymbol{C}$. Example:

$$\begin{matrix} 1x0 \\ x01 \end{matrix} \sqcup \begin{matrix} 110 \\ x0x \end{matrix} = \begin{matrix} 1x0 \\ x0x \end{matrix}.$$

The *sharp operator* # is used to find the vertices in one cube (or list of cubes), but not in a second cube (or list of cubes). Given two cubes a and b:

$$a \# b = \begin{cases} a & \text{if } a_i \# b_i = \phi \text{ for some } i \\ \phi & \text{if } a_i \# b_i = \varepsilon \text{ for all } i \\ \bigsqcup\limits_i (a_1 a_2, \cdots, \overline{b_i}, \cdots, a_n) & \text{otherwise} \end{cases}$$

where the union is over all i where $a_i \# b_i = \alpha_i \in \{0, 1\}$.

Using the coordinate sharp table:

	$a_i \# b_i$	b_i = 0	1	x
a_i	0	ε	ϕ	ε
	1	ϕ	ε	ε
	x	1	0	ε

Examples:

$$x10 \# x01 = x10, \quad x10 \# xx0 = \phi,$$

$$x1x \# 111 = 01x, \quad x10.$$

The sharp operator can be extended to an array # a cube, a cube # an array, and an array # an array:

$$C \# b = \{\{c^1 \# b\} \quad \{c^2 \# b\} \quad \cdots\}$$

$$b \# C = \{\cdots \{\{b \# c^1 \# \}c^2 \# \} \cdots\}$$

$$B \# C = \{\cdots \{\{B \# c^1 \# \}c^2\} \cdots\}.$$

Acknowledgment

The author would like to thank Prof. C. R. Kime and Prof. D. L. Dietmeyer for their careful reading of the manuscript and their many helpful suggestions. The author would also like to thank N. Godiwala for several interesting discussions during the early stages of this research.

References

[1] W. N. Carr and J. P. Mize, *MOS/LSI Design and Applications*, Texas Instruments Electronics Series. New York: McGraw-Hill, 1972, pp. 229–258.

[2] H. Fleisher and L. I. Maissel, "An introduction to array logic," *IBM J. Res. Develop.*, vol. 19, pp. 98–109, Mar. 1975.

[3] *Signetics Field Programmable Logic Arrays*. Sunnyvale, CA: Signetics, Mar. 1976.

[4] M. A. Breuer and A. D. Friedman, *Diagnosis and Reliable Design of Digital Systems*. Woodland Hills, CA: Comput. Sci., 1976.

[5] J. P. Roth *et. al.*, "Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits," *IEEE Trans. Electron. Comp.*, vol. EC-16, pp. 567–580, Oct. 1967.

[6] J. Hlavicka and E. Kottels, "Fault model for TTL circuits," *Digital Processes*, vol. 2, pp. 169–180, Autumn 1976.

[7] G. R. Case, "Analysis of actual fault mechanisms in CMOS logic gates," in *13th Design Auto. Conf. Proc.*, 1976, pp. 265–270.

[8] D. R. Schertz and G. Metze, "A new representation for faults in combinational digital circuits," *IEEE Trans. Comput.*, vol. C-21, pp. 858–866, Aug. 1972.

[9] F. W. Clegg and E. J. McCluskey, "Fault equivalence in combinational logic networks," *IEEE Trans. Comp.*, vol. C-20, pp. 1286–1293, Nov. 1971.

[10] M. R. Paige, "Generation of diagnostic tests using prime implicants," Coord. Sci. Lab., Univ. Illinois, Urbana, IL, Rep. R-414, May 1969.

[11] A. D. Friedman, "Fault detection in redundant circuits," *IEEE Trans. Electron. Comp.*, vol. EC-16, pp. 99–100, Feb. 1967.

[12] D. R. Schertz and G. Metze, "On the design of multiple fault diagnosable networks," *IEEE Trans. Comput.*, vol. C-20, pp. 1361–1364, Nov. 1971.

[13] J. P. Roth, "Algebraic topological methods for the synthesis of switching systems I," *Trans. Amer. Math. Soc.*, vol. 88, July 1958.

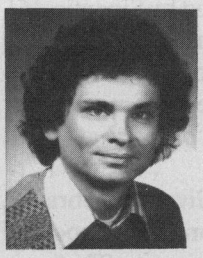
[14] D. L. Dietmeyer, *Logic Design of Digital Systems*. Boston, MA: Allyn and Bacon, 1971.

[15] M. A. Breuer, "Logic synthesis," in *Design Automation of Digital Systems*, vol. 1, M. A. Breuer, Ed. Englewood Cliffs, NJ: Prentice-Hall, 1972, ch. 2.

[16] D. L. Ostapko and S. J. Hong, "Fault analysis and test generation for programmable logic arrays," in *8th Fault Tolerant Comput. Symp. Proc.*, 1978, pp. 83–89.

[17] C. W. Cha, "A Testing Strategy for PLA's," in *15th Design Auto. Conf. Proc.*, 1978, pp. 326–331.

[18] I. Kohavi and Z. Kohavi, "Detection of multiple faults in combinational logic networks," *IEEE Trans. Comput.*, vol. C-21, pp. 556–568, June 1972.



James E. Smith (S'74–M'76) received the B.S. degree in electrical engineering and computer science and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana–Champaign, in 1972, 1974, and 1976, respectively.

From 1972 to 1976 he was a Research Assistant at the Coordinated Science Laboratory, University of Illinois, working in computer arithmetic and fault-tolerant computing. In 1976 he joined the faculty of the Department of Electrical and Computer Engineering, University of Wisconsin, Madison, where he is an Assistant Professor. He spent the summer of 1978 at the IBM T. J. Watson Research Center, Yorktown Heights, NY. His current research interests are in fault-tolerant computing and parallel computation.

Dr. Smith is a member of the Association for Computing and Sigma Xi.