THE FIFTEENTH ANNUAL INTERNATIONAL SYMPOSIUM
ON FAULT-TOLERANT COMPUTING

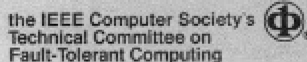
DIGEST OF PAPERS

FTCS 15

June 19-21, 1985
Horace H. Rackham Building
The University of Michigan
Ann Arbor, Michigan, USA

Sponsored by

the IEEE Computer Society's Technical Committee on Fault-Tolerant Computing



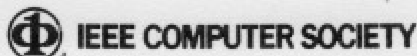
In cooperation with

Department of Electrical Engineering and Computer Science The University of Michigan

and

IFIP Working Group 10.4 on Reliable Computing and Fault-Tolerance

ISSN 0731-3071
ISBN 0-8186-0618-5
IEEE Catalog Number 85CH2143-6
Library of Congress Number 79-613257
IEEE Computer Society Order Number 618



THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

A LOW OVERHEAD, HIGH COVERAGE, BUILT-IN SELF-TEST PLA DESIGN

R. Treuer, H. Fujiwara* and V.K. Agarwal

Dept. Electrical Engineering, McGill University

3480 University St., Montreal, Canada H3A 2A7

Abstract

This paper presents a new design for a Built-In Self-Test PLA, that has a lower area overhead and higher multiple fault coverage (of all three types of faults: crosspoint, stuck and bridging) than any existing scheme. In addition to generating and using function independent test input patterns, this new scheme compresses the output responses into a function independent string of parity bits, whose fault-free expected values can be generated on-line with a simple circuit. The entire response to the test patterns can be effectively compressed into only TWO bits (representing the number of 0-to-0 and 1-to-1 transitions, with a fault free expected value of zero for both), and yet this scheme detects all single faults and more than $(1-2^{-(m+2n)})$ of all multiple faults, where m and n represent the number of product terms and input variables, respectively.

1. Introduction

Since the introduction of programmable logic arrays (PLAs) in 1975, many researchers have worked on the problem of testing such devices, largely because of the increasing use of PLAs in VLSI chips, such as in BELLMAC-32A [LS82] which has eight PLAs (the largest with 50 inputs, 67 outputs and 190 product terms).

Designs of easily testable PLAs [DM81, FK81, HO80, HM83, SKF83, YA81] take advantage of the PLA's regular structure to modify it in a manner which allows the use of simple input test patterns. The various approaches suggested provide different choices for the number of test patterns, delay per test, fault coverage, additional hardware, etc.[RT84]

Recently, in order to further simplify the testing of PLAs, there have been attempts to design a built-in self-test (BIST) PLA with high fault coverage and little extra hardware. The test patterns are generated (on chip) by part of the PLA decoder circuit, the output responses are compressed (on chip) into a small number of bits, and the comparison with the fault free compressed response is also done on chip.

In this paper, we provide the theory of a new approach to design BIST PLAs with the lowest overhead reported thus far, and the highest proven coverage which consists of all single faults, and almost all multiple faults of all types (crosspoint, stuck, bridging). A companion paper [TFA85] describes an nMOS implementation of our design.

2. Review

A PLA is characterized by three parameters: the number of inputs, n; the number of product terms, m; and the number of outputs, p. Each input "x" is "decoded" into 2 bit lines (x & ¬x), and each output is the inverse of a sum line. Despite the appellations "AND" and "OR" arrays, the actual operation done in each array depends on the technology. In nMOS for instance, each product line in the AND array performs a NOR operation on the bit lines "connected" to that product line. Each such "connection" (called a device) is a transistor whose gate is the bit line. Similarly, each sum line performs a NOR on those product lines which have devices on the sum line. It is well known that any sums of products boolean expressions can be implemented on such NOR-NOR PLAs.

The fault model of interest in PLAs consists of single and multiple presence or absence of erroneous devices in the two arrays ("crosspoint faults"), of various stuck faults on any of the lines, and of short circuits between any two neighboring lines ("bridging faults"). It is known [A80,OH79,RT83,S79] that a test set that detects all single crosspoint faults also detects most multiple crosspoint, stuck and bridging faults as

---

* Dr. H. Fujiwara was a Visiting Professor while this research was carried out. He is with the Dept. Electrical & Communications Eng., Meiji University, Japan. This research was supported by a fellowship to R. Treuer, and by a Strategic Grant from the Natural Sciences and Engineering Research Council of Canada.

0731-3071/85/0000/0112$01.00 © 1985 IEEE

well. In designing BIST PLAs, a major requirement is that the test set be independent of the functions realized by the PLA to permit the test generation circuitry to be universal. Another BIST PLA requirement is that fault coverage remain high even after the compression of output data into a few bits.

The table below compares several schemes:

BIST Scheme	Delay per Test	Output Response	Number of Test Patterns
[DM81]	O(1)	Dependent	O(n+m+p)
[HM83]	O(1)	Dependent	$O(2^n)$
[SKF83]	O(1)	Dependent	O(nm)
[YA81]	O(m)	Independent	O(n+m)
[FK81]	O(m)	Independent	O(n+m)
NEW	O(p)	Independent	O(nm)

The two most important criteria for comparing BIST PLA designs, (1) area overhead and (2) fault coverage, are not in the above table because the NEW scheme has a lower area overhead and a higher fault coverage than each of the other 5 schemes (due to limited space, this claim is proven elsewhere [T85]). Since minimizing area overhead and maximizing fault coverage are contradictory goals, it is natural to ask how the new scheme obtained improvements in both fault coverage and overhead. The answer: (1) less extra area was needed because a long test sequence was used and the compressed output is function independent, and (2) higher fault coverage was gained by exploiting all of the parity information.

3. New Built-in Self-test Approach

Figure 1 shows a PLA augmented by adding a shift register, 2 control lines, 1 product line, and 1 sum line. The shift register can disable all product lines but one, to allow the effect of a single product line on the outputs to be observed (e.g., to observe only product line "i", set S_i to 0, and all other S_j's to 1). When either of the control lines C_1 or C_2 is set to 1, then all ¬x's or x's, respectively, are forced to 0. The extra product line is used to make the number of devices (i.e., transistors) and non-devices (i.e., lack of a transistor) on each bit line odd. If the original number of product lines is odd, then 1 extra line (total m is even) suffices; otherwise 2 extra lines are needed. Likewise, the extra sum line forces every product line (OR array part only) to have an odd number of devices.

The PLA's area is increased in two more places not indicated in Figure 1. (1.) We augment the input

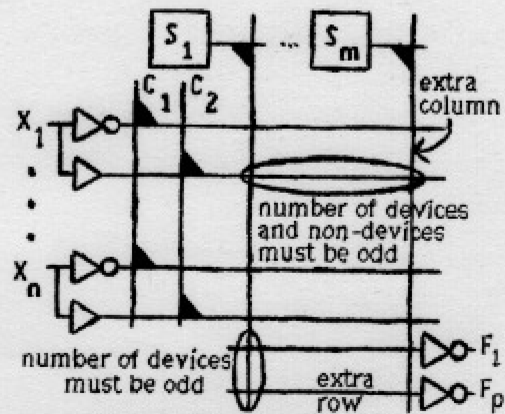


Figure 1: Augmented PLA

decoder so that it can directly generate the function independent test input sequence. (2.) A cascade of exclusive-NOR gates put after the output decoders determines the parity of the output vector. Unlike the schemes of [FK81,HO80,YA81], we do not use a second parity circuit on the product lines, which not only saves area but also reduces delay per test.

The augmented PLA is tested with these patterns:

	$x_1 \dots x_{i-1}$	x_i	$x_{i+1} \dots x_n$	c_1	c_2	$s_1 \dots s_{j-1}$	s_j	$s_{j+1} \dots s_m$
I^1	0 ... 0	0	0 ... 0	1	0	1 ... 1	1	1 ... 1
I^2_j	0 ... 0	0	0 ... 0	1	0	1 ... 1	0	1 ... 1
I^3_j	1 ... 1	1	1 ... 1	0	1	1 ... 1	0	1 ... 1
I^4_{ij}	0 ... 0	1	0 ... 0	1	0	1 ... 1	0	1 ... 1
I^5_{ij}	1 ... 1	0	1 ... 1	0	1	1 ... 1	0	1 ... 1

We apply these test patterns in the following order, called the "Universal Test Sequence":

$$I^1 \bullet \prod_{j=1}^{m} (I^2_j) \bullet \prod_{i=1}^{n} \prod_{j=1}^{m} (I^4_{ij}) \bullet \prod_{j=1}^{m} (I^3_j) \bullet \prod_{i=1}^{n} \prod_{j=1}^{m} (I^5_{ij})$$

Every new output vector is compressed into a single parity bit, and then this bit is exclusive-ORed with a bit representing the cumulative parity of all previous output vectors, to obtain a new cumulative parity bit. By examining the cumulative parity bit at only 2n + 2m + 1 specific times, we can detect all single faults and almost all multiple faults.

Definition: Let the term Cumulative Parity Comparison refer to the comparison scheme shown in figure 2. The asterisks indicate the (2n + 2m +1) times when the cumulative parity bit is compared to its expected value. Note that the Cumulative Parity Comparison scheme is function independent (i.e., valid for all PLAs). The expected parity bits of the Output Vectors were derived as follows (using Figure 3):

Figure 2: Cumul. Parity Comparison scheme

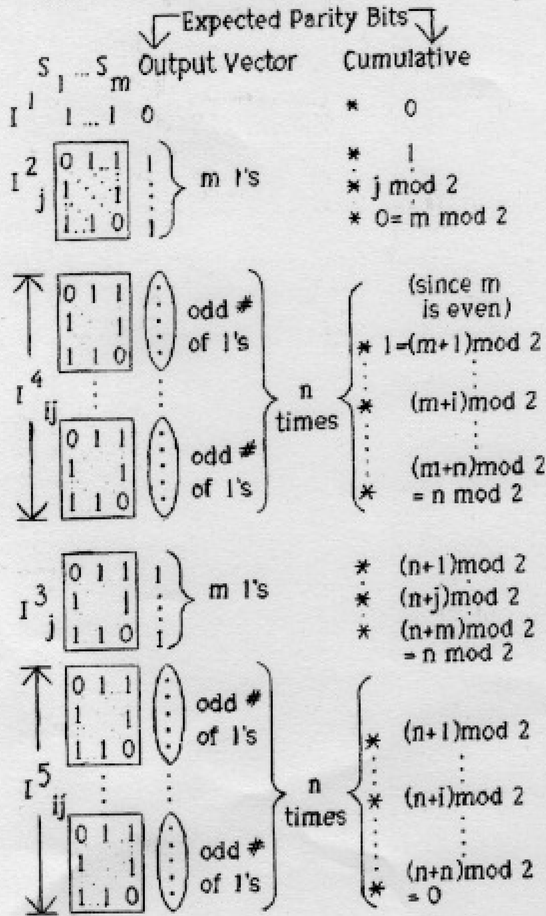
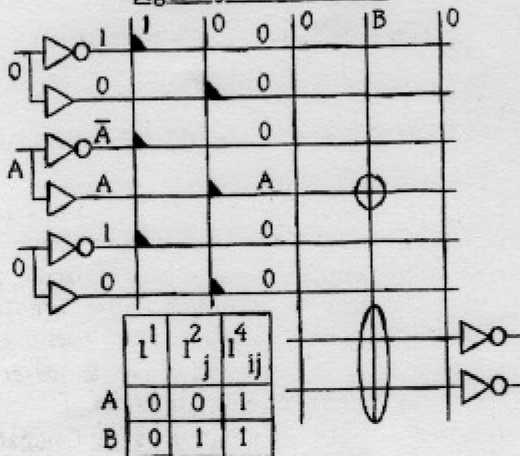


Figure 3: Test Patterns



I^1: If the PLA is fault-free, then the output vector is a string of all zeroes, whose parity is zero.

I^2_j: If the PLA is fault-free, then each of the m output vectors has an odd number of 1's because each product line in the OR array has an odd number of devices. Thus, the parity of each output vector is 1. (Similarly for I^3_j.)

I^4_{ij}: Assume the PLA to be fault-free. For simplicity, consider the partial test pattern $\Pi_j (I^4_{ij})$, with i being constant, which activates the crosspoints on bit line x_i only. If the crosspoint (x_i,j) has a device, then the product line j is pulled down to 0, which produces an output of all zeroes, and thus a parity bit of zero. If the crosspoint (x_i,j) has NO device, then the product line j stays at 1, which produces an output with an odd number of ones, and thus a parity bit of 1. Recall from Figure 1 that the number of non-devices of each bit line is odd, therefore an odd number of 1's parity bits are created for each bit line. (Similarly for I^5_{ij}.)

Interestingly, the cumul. parity bit sequence of length (2n+2m+1) is simply an alternating sequence of 0's and 1's (this sequence can be further compressed into only TWO bits: the first bit giving the number of 0-to-0 transitions, and the second the number of 1-to-1 transitions; for a fault free PLA, both bits have the value 0). Hence the fault free cumul. parity bit sequence can be generated on-line by a simple circuit, and thus does not have to be stored in (2n+2m+1) bits. Implementation details are in [TFA85].

4. Fault Coverage

This section proves the following theorem.

•Theorem 1 All single and almost all multiple crosspoint, stuck and bridging faults are detected by the Cumulative Parity Comparison scheme when the Universal Test Sequence

$$I^1 \bullet \prod_{j=1}^{m} (I^2_j) \bullet \prod_{i=1}^{n} \prod_{j=1}^{m} (I^4_{ij}) \bullet \prod_{j=1}^{m} (I^3_j) \bullet \prod_{i=1}^{n} \prod_{j=1}^{m} (I^5_{ij})$$

is applied. Fewer than $2^{-(m+2n)}$ of the multiple faults remain undetectable by this function independent scheme. The proof is presented by a sequence of 10 lemmas and a second theorem. The first 5 lemmas deal with crosspoint faults.

•Lemma 1 All single and almost all of the multiple crosspoint faults in the AND array can be detected by the test patterns $\prod_{i=1}^{n} \prod_{j=1}^{m} (I^4_{ij})$ and $\prod_{i=1}^{n} \prod_{j=1}^{m} (I^5_{ij})$.

Proof A crosspoint fault at (x_i,j) inverts the expected value of the product line j, hence the parity of the output is also inverted. I^4_{ij} and I^5_{ij} activate all m crosspoints of a bit line to produce one new

cumulative parity bit; thus, if the total number of faults on a bit line is even, the cumulative parity bit indicates no fault since the numbers of devices and non-devices for the line remain odd. Thus, the only multiple faults that are not detected are those which have an even number of faults (or zero faults) on each bit line. The total number of possible multiple crosspoint faults in the AND array is $2^{(2n)(m)}$. The number of possible even faults (including zero faults) for a single bit line is $2^{(m-1)}$. Thus, the fraction of undetectable faults in the AND array is: $(2^{m-1})^{2n} / 2^{2nm} = 2^{-2n}$.

(A way to improve the fault coverage indicated in Lemma 1 is by increasing the number of comparisons of cumul. parity bits during I^4_{ij} and I^5_{ij} from n times to nm times; but this has the serious disadvantage of rendering the cumul. parity bit comparison scheme function dependent, which requires very much extra hardware to store the expected cumul. parity bits for the individual crosspoints of the AND array.)

•Lemma 2 All single and almost all of the multiple crosspoint faults in the OR array can be detected by applying either $\prod_{j=1}^{m} (I^2_j)$ or $\prod_{j=1}^{m} (I^3_j)$.

Proof If the product line segment circled in Figure 3 contains a single crosspoint fault, then the parity of the output is changed. As in Lemma 1, undetectable multiple faults have an even (or zero) number of faults per product line. The fraction of undetectable faults in the OR array is: $(2^{p-1})^m / 2^{pm} = 2^{-m}$.

(The fault coverage of Lemma 2 could be improved by adding a second shift register so that the crosspoints in the OR array can be individually activated (i.e.: equivalent to replacing $\prod_j(I^2_j)$ and $\prod_j(I^3_j)$ by $\prod_j\prod_k(I^2_{jk})$ and $\prod_j\prod_k(I^3_{jk})$ where k varies from 1 to p). In addition to the extra shift register area, this proposal also makes the output response comparisons function dependent, thereby requiring an impractical amount of extra hardware to store the expected output for each crosspoint in the OR array.)

•Lemma 3 All single and almost all multiple extra device faults in the input decoder are detected when $\prod_{i=1}^{n} \prod_{j=1}^{m} (I^4_{ij})$ and $\prod_{i=1}^{n} \prod_{j=1}^{m} (I^5_{ij})$ are applied.

Proof An extra device at the intersection of C_1 and x_i causes the bit line x_i to be constantly 0 while I^4_{ij} is applied. In the fault free case, $\prod_j(I^4_{ij})$ with i constant, causes an odd number of product lines to have the value 1, but with the fault, all product lines (an even number) have the value 1. The odd number too many 1's produced by the product lines, causes an odd number times an odd number too many 1's at the outputs, and thus a wrong parity. Similarly, I^5_{ij} detects when the extra device is at $(C_2, \neg x_i)$.

•Lemma 4 All single and almost all multiple missing device faults in the input decoder are detected when either $\prod_{j=1}^{m} (I^2_j)$ or $\prod_{j=1}^{m} (I^3_j)$ is applied.

Proof When the device at $(C_1, \neg x_i)$ is missing, the bit line $\neg x_i$ has the value 1. Hence, an odd number too few product lines have the value 1, which causes the output parity to be wrong an odd number of times. Similarly for a missing device at (C_2, x_i).

•Lemma 5 Undetectable multiple crosspoint faults in the OR array cannot mask detectable multiple crosspoint faults in the AND array.

Proof The ability of the I^4_{ij} and I^5_{ij} test patterns to detect AND array faults hinges upon the assumption that the OR array part of every product line has an odd number of devices. Undetectable OR array faults cause an even number of devices in each product line to appear or disappear, hence the total number of devices per product line remains odd, and therefore no AND array faults are masked.

•Theorem 2 Almost all multiple crosspoint faults are detected by the Cumulative Parity Comparison scheme. More exactly, only 2^{-2n} of the multiple faults in the AND array are undetectable, and only 2^{-m} of the multiple faults in the OR array are undetectable.

Proof Lemmas 1 to 5, inclusive.

Single and multiple stuck faults are considered in the following two lemmas.

•Lemma 6 All single stuck faults are detected by the Cumulative Parity Comparison scheme. Further, almost all single stuck faults are indistinguishable at the output from detectable multiple crosspoint faults, and therefore can be detected by the same test patterns.

Proof There are six cases to consider:

sum line stuck	(a.) at 0	(b.) at 1
product line stuck	(c.) at 0	(d.) at 1
bit line stuck	(e.) at 0	(f.) at 1

(a.) sum line s-a-0: This fault is obviously detected by I^1. It is also the only stuck fault which is not output indistinguishable from some multiple crosspoint fault.

(b.) sum line s-a-1: This fault is indistinguishable from

having all devices on the sum line missing, which is a multiple crosspoint fault detectable (since any product line has at most one fault) by I^2_j (or I^3_j).

(c.) product line s-a-0: Since the s-a-0 product line has no effect upon the sum lines, this fault is indistinguishable from having all the devices in the OR array part of the product line missing. This multiple crosspoint fault is detectable (since all devices missing, means an odd number of devices missing) by I^2_j (or I^3_j).

(d.) product line s-a-1: This fault is indistinguishable from having all devices in the AND array part of the product line missing, which is a multiple crosspoint fault detectable (since any bit line has at most one fault) by I^4_{ij} and I^5_{ij}.

(e.) bit line s-a-0: Since the s-a-0 bit line has no effect upon the product lines, this fault is indistinguishable from having all the devices on the bit line missing. This multiple crosspoint fault is detectable (since all devices missing, means an odd number missing) by I^4_{ij} and I^5_{ij}.

(f.) bit line s-a-1: Since all the product lines which the s-a-1 bit line connects with are forced to 0, this fault is equivalent to having all the affected product lines s-a-0, and thus can be detected by I^2_j (or I^3_j).

•Lemma 7 Almost all multiple stuck faults can be detected by the Cumulative Parity Comparison scheme. More exactly, fewer than $2^{-(m+2n)}$ of multiple stuck faults are undetectable.

Proof The three fault types: product line s-a-0, bit line s-a-0 and bit line s-a-1, when combined into multiple stuck faults always remain detectable because the multiple crosspoint faults, which are output indistinguishable from these multiple stuck faults, are detectable since neither bit lines nor the OR array part of product lines can have an even number of faults (recall from cases c, e and f: "since all devices missing, means an odd number missing"). When sum line s-a-1 and product line s-a-1 are part of multiple stuck faults, then about $2^{-(m+2n)}$ of them are output indistinguishable from undetectable multiple crosspoint faults (recall from cases b and d: "since any product/bit line has at most one fault"). When sum line s-a-0 faults are part of multiple stuck faults, I^1 detects only an odd number of such faulty sum lines. To detect an even number of sum line s-a-0 faults, the following observation is used: when I^2_j (or I^3_j) is applied then each s-a-0 sum line behaves as if it had an extra device at the crosspoint with the currently active product line. Hence, in the context of I^2_j (or I^3_j) the sum line s-a-0 fault is output indistinguishable from a multiple extra device fault where each faulty sum line has m devices (i.e., a device at each of its crosspoints); only 2^{-m} of such OR array multiple crosspoint faults cannot be detected by I^2_j (or I^3_j).

The last three lemmas consider bridging faults.

•Lemma 8 All single and multiple product line bridging faults are detected by the Cumulative Parity Comparison scheme.

Proof Since a bridging fault (in nMOS) causes both lines to be 0 when at least one of them is 0, bridged product lines would remain at 0 when I^2_j or I^3_j try to give them a value of 1, and thus a fault revealing parity bit would be produced.

•Lemma 9 All single and almost all multiple sum line bridging faults are detected by the Cumulative Parity Comparison scheme.

Proof If two sum lines A and B are bridged, then their common bridged value V is the logical AND of the values of A and B. While the test sequence I^2_j (or I^3_j) is being applied, V has the value 1 only when neither line A nor B has a device on the currently active product line. Hence, this bridging fault is output indistinguishable from a multiple crosspoint fault where both A and B have extra devices everywhere except along the product lines where both A and B were originally without devices, as shown in figure 4. Therefore, I^2_j (or I^3_j) allows us to detect all single bridging faults, and almost all multiple bridging faults (about 2^{-m} of them remain undetectable).

•Lemma 10 All single and multiple bit line bridging faults are detected by the Cumulative Parity Comparison scheme.

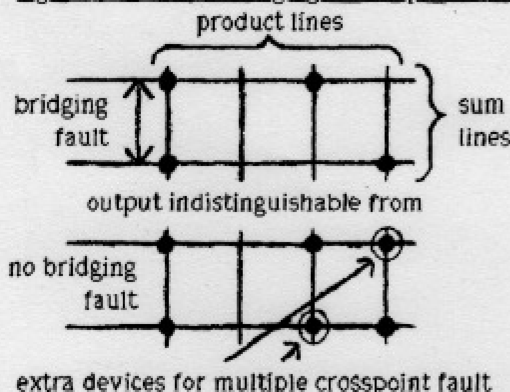
Proof Since a bridging fault causes both lines to be 0 when at least one of them is 0, bridged bit lines would remain at 0 when I^4_{ij} and I^5_{ij} try to give them a value of 1. A fault free bit line produces an odd number of 1's parity bits, while a bridged bit line produces m (an even number) 1's parity bits, and hence a fault revealing cumulative parity bit.

These lemmas complete the proof of Theorem 1. The proven fault coverage of other BIST schemes does not encompass both single and multiple faults of all three types, as considered above.

5. Conclusion

A new approach to design low overhead and high fault coverage PLAs has been presented. The fault coverage is proven to consist of all single faults and

Figure 4: sum line bridging fault equivalence



almost all multiple faults of crosspoint, stuck and bridging types. A companion paper [TFA85] describes the implementation details of this approach and compares it with the other schemes. Due to the regular nature of the cumulative parity scheme used in our approach, the additional overhead requirements are quite small compared to all existing schemes. Table 1 lists the percentage overhead for various PLAs. For large PLAs, the overhead can be as small as 15% of the total PLA area, which may be acceptable in such cases. This overhead includes all extra testing circuitry, and is based on actual nMOS layouts [TFA85].

Our approach provides better fault coverage than all known BIST schemes. In terms of the area overhead, a comparison (see Table 1) can be made only with [HJA84], where the overhead is calculated with respect to actual layouts. All other schemes report overhead in terms of the number of additional logic gates, which does not reflect the actual area overhead. However, by simply comparing the various components used in each scheme, it easily follows that the new scheme uses less area.

References

[A80] V.K. Agarwal, "Multiple Fault Detection in Programmable Logic Arrays", 1980, IEEE Trans. Computers, vol. C-29, pp. 518-522.

[DM81] W. Daehn and J. Mucha, "A Hardware Approach to Self-Testing of Large Programmable Logic Arrays", 1981, IEEE Trans. Computers, vol. C-30, pp. 829-833.

[F84] H. Fujiwara, "A New PLA Design for Universal Testability", 1984, IEEE Trans. Computers, vol. C-33, pp. 745-750.

[FK81] H. Fujiwara and K. Kinoshita, "A Design of Programmable Logic Arrays with Universal Tests", 1981, IEEE Trans. Computers, vol. C-30, pp. 823-828.

[HJA84] K.A. Hua, J.-Y. Jou and J.A. Abraham, "Built-In Tests for VLSI Finite-State Machines", 1984, FTCS-14, pp. 292-297.

[HM83] S.Z. Hassan and E.J. McCluskey, "Testing PLAs Using Multiple Parallel Signature Analyzers", 1983, FTCS-13, pp. 422-425.

[HO80] S.J. Hong and D.L. Ostapko, "FITPLA: A Programmable Logic Array for Function Independent Testing", 1980, FTCS-10, pp. 131-136.

[LS82] H.-F.S. Law and M. Shoji, "PLA Design for the BELLMAC-32A Microprocessor", 1982, Proc. Int. Conf. on Circ. and Comp., pp. 161-164.

[OH79] D.L. Ostapko and S.J. Hong, "Fault Analysis and Test Generation for Programmable Logic Arrays", 1979, IEEE Trans. Comput., vol. C-28, pp. 617-626.

[RT83] J. Rajski and J. Tyszer, "Combinational Approach to Multiple Contact Faults Coverage in Programmable Logic Arrays",1983, to appear in IEEE Trans. Computers.

[RT84] J. Rajski and J. Tyszer, "Easily Testable PLA Design", 1984, EUROMICRO, pp. 139-146.

[S79] J. Smith, "Detection of Faults in Programmable Logic Arrays", 1979, IEEE Trans. Computers, vol C-28, pp. 845-853.

[SKF83] K.K. Saluja, K. Kinoshita and H. Fujiwara, "An Easily Testable Design of Programmable Logic Arrays for Multiple Faults", 1983, IEEE Trans. Comput., vol C-32, pp. 1038-1046.

[T85] R. Treuer, "A New Design of Built-In Self Testing Programmable Logic Arrays with High Fault Coverage and Low Overhead", master's thesis, Dept. Electrical Eng., McGill Univ., Montreal, March 1985.

[TFA85] R. Treuer, H. Fujiwara and V.K. Agarwal, "Implementing a Built-In Self-Test PLA Design", IEEE Design and Test of Computers Magazine, April 1985.

[YA81] S. Yajima and T. Aramaki, "Autonomously Testable Programmable Logic Arrays", 1981, FTCS-11, pp. 41-43.

Table 1: Overhead Comparison

PLA Size			Percentage of Total Area	
n	m	p	[TFA85]scheme	[HJA84]scheme
60	200	60	15.2%	19.1%
50	190	67	15.8%	20.0%
54	134	61	17.7%	21.6%
27	181	54	19.9%	25.5%
30	153	37	22.0%	27.9%
30	120	27	24.6%	30.6%
24	44	13	35.6%	41.1%
25	42	12	36.0%	41.3%
12	58	21	36.2%	43.3%