wiring parameters," in *1972 Tohoku Branch Conf. Rec.* (IECE Japan), paper 2D-1.
[14] P. Silvester, "TEM wave properties of microstrip transmission lines," *Proc. Inst. Elec. Eng.*, vol. 115, pp. 43-48, Jan. 1968.
[15] M. Yagyu, T. Hauashi, H. Tanaka, and A. Masaki, "Analysis of pulse signal transmission on VLSI chips by scaling-up experiment," in *Trans. 1981 Tokyo Section Conv. IEE of Japan*, paper 196.
[16] T. Hayashi, M. Yagyu, H. Tanaka, A. Masaki, and T. Chiba, "Normally-on type GaAs FET logic with large load driving capability," in *1981 Nat. Conf. Rec. on Semiconductor Technology* (IECE, Japan), paper 117.
[17] A. W. Livingstone and P.J.T. Mellor, "Capacitor coupling of GaAs depletion-mode f.e.t.s.," *Proc. Inst. Elec. Eng.*, vol. 127, pt. I, pp. 297-300, Oct. 1980.

Concurrent Error Detection and Testing for Large PLA's

JAVAD KHAKBAZ AND EDWARD J. McCLUSKEY, FELLOW, IEEE

Abstract—**A system of checkers is designed for concurrent error detection in large PLA's. This system combines concurrent error detection with off-line functional testing of the PLA by using the same checker hardware for both purposes. The result is a significant saving in hardware cost. For a case example, the total hardware cost is estimated at about 37 percent of the original PLA area. The system is almost totally self-checking and, although the test patterns are not function-independent, their generation algorithm is simple. The total test time for the entire system is within the range of that of some recent PLA design schemes which were specifically aimed at simplifying off-line testing, but which have no provisions for concurrent error detection.**

Key Words—**programmable logic array, PLA, nonconcurrent PLA, concurrent error detection, testing, two-rail code checker, 1-out-of-*n* code checker.**

I. Introduction

The advantage of the programmable logic array (PLA) as a flexible and regular structure within a digital system is now well established, [12], [7]. For example, the use of the PLA as the driver of the control section of today's complex microprocessors has become a common practice. Such a PLA might have as many as 40 output lines, 30 input lines, and a few hundred product lines. A PLA of this size constitutes a significant portion of a digital system. This paper addresses the issue of error detection and testing of such large PLA's.

Manuscript received September 30, 1981; revised November 28, 1981. This work was supported in part by the US Army Electronics Research and Development Command under Contract DAAK-20-80-C-0266.

The authors are with the Center for Reliable Computing, Computer Systems Laboratory, Departments of Electrical Engineering and Computer Science, Stanford University, Stanford, CA 94305.

There are two modes of fault detection for any digital system. On one hand, we have off-line testing for either initial acceptance or for periodic testing of the system. In this mode, the normal operation of the system is interrupted, a selected set of inputs is applied, and the corresponding outputs are checked for possible error indication. In this paper we refer to this mode as simply *testing*. The second mode of fault detection, hereafter called *concurrent* or *on-line* error detection or checking, concerns monitoring the system under normal operation (i.e., with normal inputs).

Much work has been done in recent years on the problem of PLA fault detection: [14], [21], [6], [8], [10], [15], [18], [22], and [17]. However, most of this work has been concentrated on the question of testing, and not concurrent checking, of PLA's. For example, Fujiwara [8] suggests extra hardware to augment the PLA so that function-independent test of the PLA would be possible. This augmentation includes a set of *cascaded* XOR gates on the product lines. For a large PLA with a few hundred product lines, the delay corresponding to such an arrangement renders this extra circuitry useless for concurrent checking. In the present work, however, we integrate the two modes of testing so that any additional piece of hardware is used for both of these modes. A price is paid in terms of the generation of test patterns, as the test set for our design will be larger than that of, for example, Hong [10] or Fujiwara [8]. Further, the test set will not be function-independent, although the dependency is simple.

Section II of this paper lays the background, describes the fault model, and provides the definitions and assumptions that will be used in later sections. In Section III, the circuitry for concurrent error detection will be developed. In Section IV, we will show how the PLA can be tested using the detec-

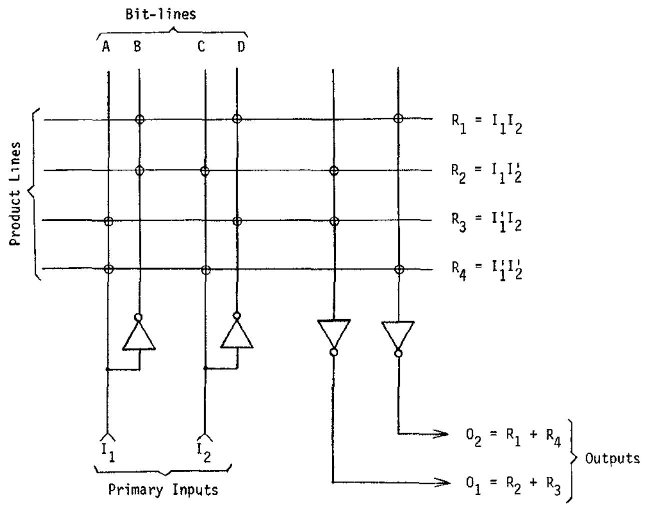

Fig. 1. A NOR-NOR PLA.

tion circuitry developed in Section III. Also, in Section IV we will address the question of testing the checkers. Finally, in Section V an example will be provided for measuring the cost of hardware incurred by our design. Following this example a comparison is made with three other proposed designs for testing PLA's.

II. The Model, Definitions, and Assumptions

A. The PLA

We assume the PLA is realized in NOR-NOR logic, as in most MOS implementations, [7], [12]. However, the proposed checking circuitry can be transformed readily to accommodate other implementations. An example of a NOR-NOR PLA is given in Fig. 1. We are interested mainly in large PLA's, as described in the Introduction.

Since this paper is concerned with the structural properties of PLA's, we assume that the primary (external) inputs to the PLA have a separate error-checking mechanism, such as a parity bit, and we exclude the primary PLA inputs from our discussion. If the PLA has n inputs, it need not necessarily see all 2^n possible input vectors during normal, fault-free operation. That is, if we conceive of a truth table corresponding to the PLA, there may be some possible input vectors for which no output is defined in the truth table. We call these the *don't-care* input vectors. If a don't-care input vector is applied to the PLA, the output would be undefined. Any input vector for which a corresponding output vector is defined is called a *normal input*.

Definition: A PLA is said to be *nonconcurrent* iff, under fault-free operation, any normal input vector selects exactly one product term.

We assume that the PLA is nonconcurrent. This property has been shown to be very desirable for testing purposes, [21], [18]. The remaining sections of this paper once again attest to this fact. Furthermore, the nonconcurrency property of the PLA is not necessarily as restrictive as it may appear to be. For example, a PLA has been used for the control section of a special-purpose microprocessor, [5]. This PLA has about 400 product terms, and, of its output lines, eight are used to

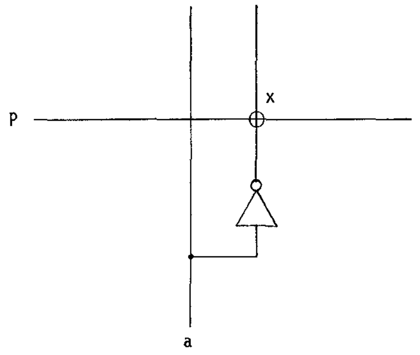

Fig. 2. Portion of the AND plane.

specify the state of the PLA and are fed back to the AND plane. This means that the machine has 256 states, or, on the average, it has less than two product terms per state. This implies a great degree of nonconcurrency that naturally exists in the PLA, as the states of the machine are mutually exclusive. It also implies a very easy and cheap search method for any possible concurrencies since concurrencies may occur only between the product lines of the *same* state. Then one can remove these occasional concurrencies by, for example, a method given in [21]. Such a process requires an increase in the size of the PLA, the degree of which depends on the number of concurrencies present.

Next consider a section of the AND plane of the (nonconcurrent) PLA, as shown in Fig. 2. P is a product term and a is an input literal. Then $P = M \cdot a$ where M is the product of some other input literals. Thus the input vector $I = M \cdot a$ is a normal input that selects product line P. We say that the device x in the AND plane (Fig. 2) is *irredundant* if input vector $J = M \cdot a'$ is also a normal input vector. Otherwise, we say that device x is *redundant*. A redundant device may be removed without affecting the function of the PLA or its nonconcurrency property. This is so because, by removing x, product line P will be selected by input M, irrespective of input literal a. But since x is redundant, J is not a normal input and it cannot occur in normal, fault-free operation. So M always occurs with $a = 1$ (never with $a = 0$). Thus device x may be removed without affecting the normal operation of the PLA. However, one may wish to have redundant devices for, say, a fault-tolerant design. Thus, in general, we may have redundant devices in the PLA.

Next, we state the definitions that will be used in the remainder of this paper. Consider a combinational circuit C with input code space S and output code space S' that implements a function Z. Let F be the set of faults that may occur in this system. Let $Z(s, f)$ denote the response of the circuit to input s in S in the presence of fault f in F. Let $Z(s, 0)$ denote the response of the system to input s in S when no fault is present. Then define the following [20]:

Definition: A combinational circuit C is said to be *fault-secure* with respect to input code space S and fault set F iff for all f in F and for all s in S either $Z(s, f) = Z(s, 0)$ or $Z(s, f)$ is not in S'.

Definition: A combinational circuit C is said to be *self-testing* with respect to input code space S and fault set F iff for all f in F, there is an s in S such that $Z(s, f)$ is not in S'.

Definition: A combinational circuit C is said to be *totally self-checking* (TSC) with respect to input code space S and fault set F iff it is both fault-secure and self-testing with respect to S and F.

Definition: A bit-line B_1 of a PLA is said to be *complementary* of another bit-line B_2, iff both B_1 and B_2 correspond to the same input literal.

Note that, under nonfaulty operation of the PLA, any bit line carries the complement logic value of its complementary bit line.

B. The Fault Model

There have been a number of studies conducted that were aimed at modeling physical failures in different technologies by logical faults, [19], [9]. A general conclusion from these works is that the classical stuck-at model is not sufficient for the present MOS technologies, [13]. Furthermore, these works exclusively deal with the so-called solid faults, and, to the best of our knowledge, there has not been any experimental result that has yielded a logic model for the transient or intermittent failures which, by their very nature, are not amenable to off-line testing.

Therefore, in this work, for both solid and nonsolid failures, we use the following logical fault models which have been used widely in recent works, e.g., [14], [21], [6], [8], [10], [17], [22]:

1) stuck-at-0 (sa0) and stuck-at-1 (sa1)
2) short between two adjacent parallel lines
3) extra/missing device at PLA crosspoints.

Further, we assume that a short between two lines always results in ANDing the logic values of the two lines.

In this work we base our design on the single-fault assumption. It has been shown that a set of test patterns that detects all single faults in a PLA also detects most of the multiple faults, [1]. However, for concurrent error detection, where we have no control over the inputs, we need to be more careful about multiple-fault situations. Therefore, our approach will be that of detecting *all single faults* and as many multiple faults as possible. To this end, we will distribute checkers throughout the PLA, as described in Section II; although there will be significant overlaps between the sets of errors detected by these checkers.

For modeling the multiple fault situations, we assume that, at any instance of time, any circuit element type may display at most one failure mode. For example, we assume that we may not have a sa1 on one product line and, at the same time, a short between two other product lines. Or, we may not have a missing device at one crosspoint and, simultaneously, an extra device at a different crosspoint. We refer to this assumption as *the single-mode fault assumption*. This is a reasonable assumption because one expects that the cause of the failures, be it a fabrication problem, extreme environmental condition, or otherwise, may affect similar circuit elements in similar manners.

C. Summary

We assume that the PLA has the following properties:

1) it is large
2) it is nonconcurrent
3) it is NOR–NOR implemented
4) it has single-input decoders at the input.

Further, we make the following assumptions about the faults that may occur:

1) possible faults are stuck-at, extra/missing device, and short between two adjacent parallel lines;
2) a short fault AND's the lines involved;
3) single fault is assumed, but detects as many multiple faults as possible;
4) for multiple-fault case, make the single-mode fault assumption;
5) primary PLA inputs are excluded from this work.

III. Concurrent Error Detection

We will divide the PLA into the AND and OR planes, [12], and will consider them separately for concurrent error detection.

A. The AND Plane

The AND plane consists of

1) the input inverters
2) the bit lines
3) the crosspoints on the bit lines
4) the product lines.

A fault in an input inverter can be modeled as a stuck-at fault at its outputs, which is equivalent to a stuck-at fault on the corresponding bit line. Thus we will ignore the input inverters for fault analysis.

Lemma: For any normal input to the PLA, any number of faults on *one* of the following circuit element groups either cause no error, or desensitize the (only) sensitized product line, or sensitize one or more *extra* product lines. The element groups are

1) the bit lines
2) the crosspoints in the AND plane
3) the product lines.

Proof: We just give the proof for the bit lines. The proofs for the other two cases are similar. There are three possibilities:

1) The faults are all sa1's. An examination of Fig. 1, and the fact that the PLA is NOR–NOR, reveals that these faults can at most desensitize the (only) selected product line.

2) The faults are all sa0's. Again Fig. 1 reveals that these faults do not affect the already-selected product line. They, however, may cause activation of more product lines.

3) The faults are all shorts. If shorted lines carry the same logic value, no change would occur. If two shorted lines carry opposite logic values, they will both become 0, which would put us back in case 2). Q.E.D.

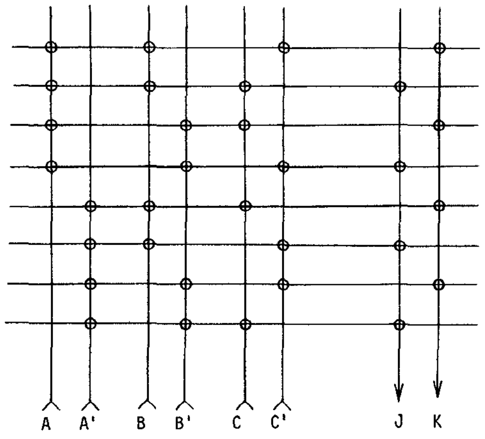

Fig. 3. A TSC two-rail checker.

The conclusion from this Lemma is that, if there are r product lines, a 1-out-of-r code checker on the product lines detects all single and many multiple faults in the AND plane of the PLA. There are several implementations of a TSC 1-out-of-n checker: [2], [16], [11]. The approach in [2] and [16] is that first the 1-out-of-r code is translated into a k-out-of-$2k$ code, and then a TSC checker for the resulting k-out-of-$2k$ code is designed. In [11], the 1-out-of-r code is first translated into a two-rail code, and then a TSC two-rail checker is used to check the output of this translator. In [11] it is shown that for some values of r, the first method is more efficient, and for the others, the second method results in a better design. Both approaches yield a TSC 1-out-of-r code checker. Here we denote the code translator by L and the checker of the translated code by $C1$. We will choose the design method in accordance with the result of Khakbaz [11].

Another structural regularity of the PLA that can be utilized is that the bit lines form a two-rail code. Thus we can put a two-rail code checker, called $C2$, on the bit lines. The general approach for building a TSC two-rail checker for n pairs (A_i, A_i') is to generate a parity tree with two output lines, [4], [20]. The first output line is the parity of, say, the A_i lines. The other output line is simply the complement of the first. Thus the output of the two-rail checker is a 1-out-of-2 code. A PLA implementation of such a tree is described in [21], and an example of it is shown in Fig. 3 (for three input pairs). The checkers of the type shown in Fig. 3 can be used to make a two-rail checker tree with many input pairs. For example, Fig. 4 shows a TSC two-rail checker with nine input pairs.

B. The OR Plane

The OR plane consists of the following circuit elements:

1) the output lines
2) the crosspoints on the output lines.

As before, we model the failures of the output inverters by stuck-at faults at the corresponding output lines.

Under the single-fault assumption in the OR plane, at most one output line can be altered. Thus for single-fault detection in the OR plane one output parity line suffices. Moreover,

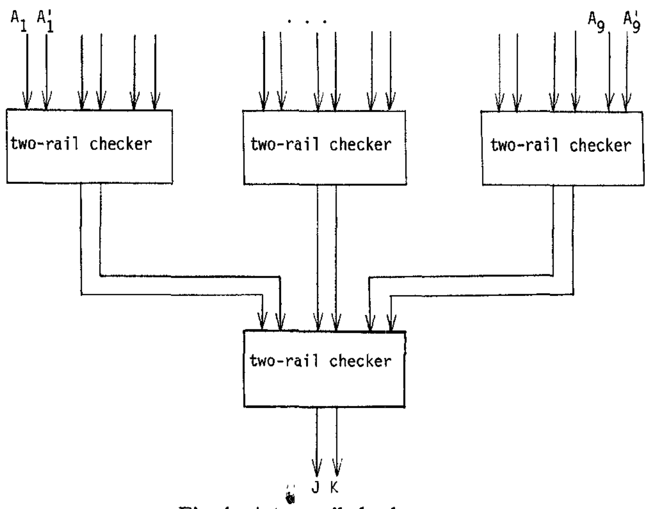

Fig. 4. A two-rail checker tree.

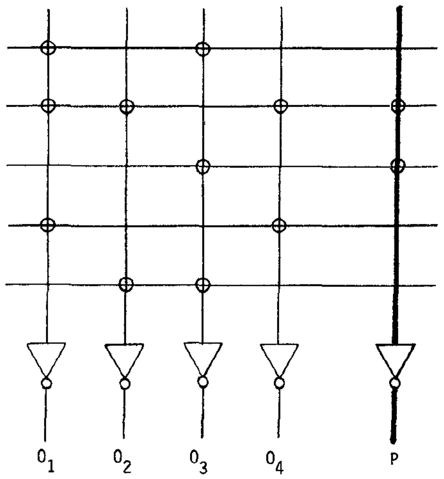

Fig. 5. Even output parity for a nonconcurrent PLA.

since the PLA is nonconcurrent, the generation of such output parity would be trivial. To generate the even (odd) output parity, we just add one output line and put devices on the crosspoints with those product lines which have odd (even) number of devices on them. An example is shown in Fig. 5. Then a parity tree can concurrently check for any single error on the $m + 1$ output lines.

For better error detection, other encoding schemes may be used for the output lines. For example, we may add one parity line for every three output lines. Any such encoding scheme requires some redundant output lines (denoted by D), and an output code checker $C3$ on the resulting output lines. For example, for the case of a single output parity, D consists of a single output line and $C3$ is a parity tree.

C. Summary

We have suggested the following set of checkers for concurrent (on-line) error checking of the PLA:

1) A TSC 1-out-of-r checker on the product lines. This consists of a code translator L and a checker $C1$ to check the output code of L.

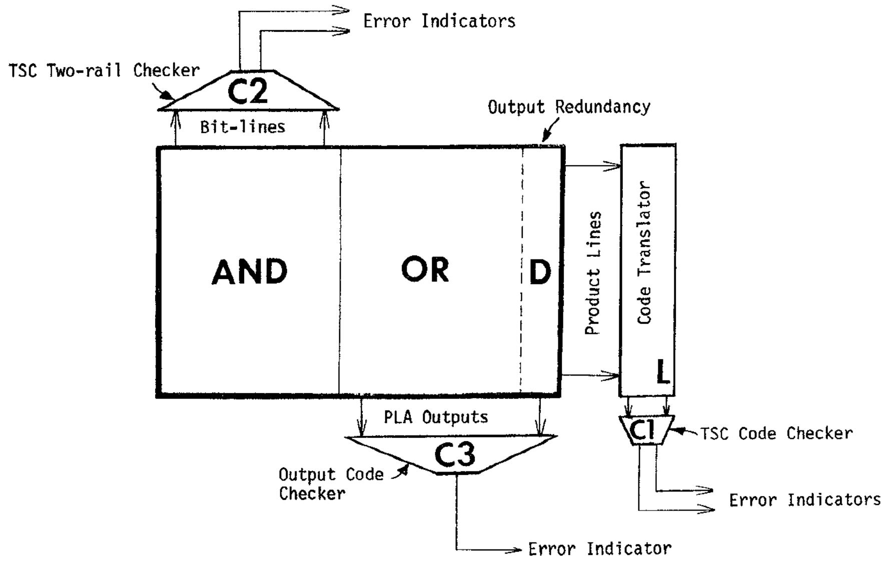

Fig. 6. The complete system diagram.

2) A two-rail code checker $C2$ for the bit lines.

3) An output encoding scheme (e.g., single parity) which requires redundant output lines D and a corresponding output code checker $C3$.

Fig. 6 shows a complete diagram of the system.

IV. Testing

In this section we will consider the question of test pattern generation for the system of Fig. 6. Each subcircuit will be considered separately. The fault model is described in Section I. We assume that we have full external control over only the primary PLA inputs. This is a realistic limitation since direct external access to points within the PLA or the checkers, or even to the connections between the PLA and the checkers, generally requires extra hardware and/or extra pins on the chip. We will show that even with this limitation, the entire system of Fig. 6 is testable, except the internal parts of $C3$ and the redundant devices in the AND plane of the PLA.

Notation: Let us number the product lines, from top to bottom, 1 through r. Let $I_0(p)$ be the input vector that selects product line p, with all don't-care input literals set to 0. Similarly, let $I_1(p)$ be the input vector that selects product line p, with all don't-care input literals set to 1. Also denote the all-0 input vector by $I0$ and the all-1 input vector by $I1$.

Pin Requirement: We assume that the outputs of $C1$, $C2$, and $C3$ of Fig. 6 are directly connected to external pins. But as mentioned earlier, and as will be discussed in the following, since, in general, the inputs of $C3$ are not controllable from the PLA inputs, a TSC implementation of $C3$ may not be necessary. Thus one output line could suffice for $C3$. Therefore, the system of Fig. 6 requires *5 extra pins.*

A. Testing the AND Plane

A portion of the AND plane is shown in Fig. 7. Table I specifies the input patterns required for testing various kinds of faults in the AND plane. Note that since device x is irredundant, by the argument given earlier, the input vector "M with

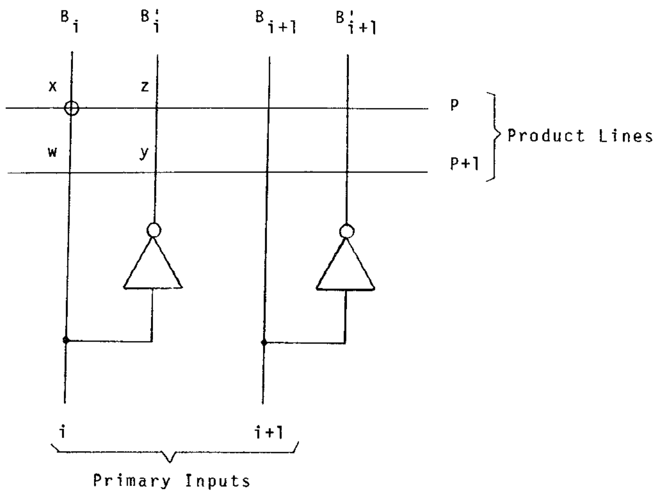

Fig. 7. A portion of the AND plane.

$i = 1$" is also a normal input and thus it selects a product line q. With x missing, this input also selects p, and hence the product lines no longer form a 1-out-of-r code.

B. Testing the OR Plane

A portion of the OR plane is shown in Fig. 8. Table II lists the PLA input patterns that test for single faults in the OR plane.

C. Testing $(L, C1)$

Either of the two implementations of the 1-out-of-r checker $(L, C1)$, one in [2] and the other in [11], is completely tested by applying $I_0(p), p = 1, 2, \cdots, r$; see [11].

D. Testing $C2$

The general two-rail checker tree organization is exemplified in Fig. 4. Any such implementation of $C2$ can be exhaustively tested by 2^t input patterns, where t is the number of input pairs to the largest block of such a tree, [3]. For example, the following is the list of the (PLA) input patterns that are needed

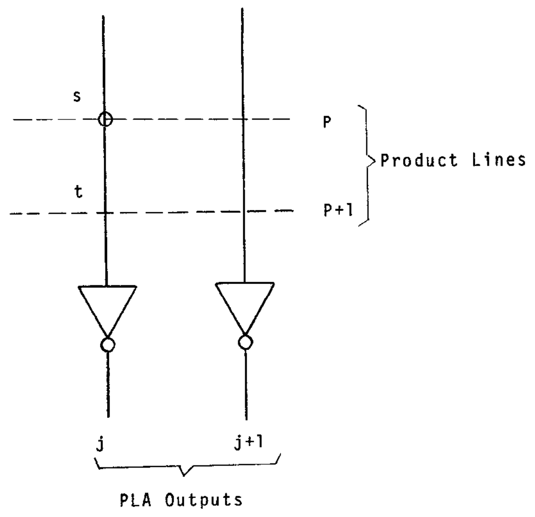

Fig. 8. A portion of the OR plane.

TABLE I
TEST PATTERN FOR THE AND PLANE INPUT

Fault	Test Pattern	Error Indicator
B_i sa0	I1	C2
B_i sa1	I0	C2
B_i short to B_i'	any input	C2
B_i' short to B_{i+1}	I0	C2
p sa1	$I_0(p+1)$	C1
p sa0	$I_0(p)$	C1
p short to p+1	$I_0(p)$	C1
extra device at z	$I_0(p)$	C1
extra device at w	$I_1(p+1)$ (*)	C1
extra device at y	$I_0(p+1)$ (*)	C1
missing device at x	M with i=1 (**)	C1

(*) Note that input literal i is a don't-care for product line p+1.

(**) Here product line p equals M.i', and device x is irredundant.

TABLE II
INPUT TEST PATTERN FOR THE OR PLANE

Fault	Test Input	Error Indicator
j sa0	$I_0(p)$	C3
j sa1	$I_0(p+1)$	C3
j short to j+1	$I_0(p)$	C3
missing device s	$I_0(p)$	C3
extra device at t	$I_0(p+1)$	C3

for testing $C2$, for a PLA with nine input lines, and for $C2$ implemented as in Fig. 4.

000 000 000
001 011 011
011 001 101
010 010 110

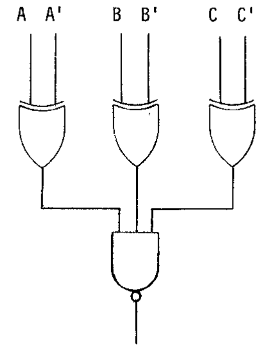

Fig. 9. A simple two-rail checker.

101 101 001
100 110 010
110 100 100
111 111 111

We denote such set of test patterns by $T(C2)$.

E. Testing C3

Since, in general, we do not have full control over the inputs of $C3$ (outputs of the PLA) from the inputs of the PLA (Fig. 6), it may not be possible to test $C3$ fully. The general approach to this problem is to use a simple implementation of $C3$. For example, if a two-rail output encoding is used for the PLA, we may use the implementation of Fig. 9 rather than that of Figs. 3 or 4. This implementation of a two-rail checker is not totally self-checking. However, since it is simple and it contains less circuitry than the TSC implementations, the possibility of a fault occurring in it is smaller.

F. Summary

In this section we have shown that the system of Fig. 6 can be tested for all single faults, except the missing redundant devices in the AND plane and the internal faults of $C3$. The test patterns are applied to the primary inputs of the PLA and the error indicators are the outputs of $C1$ (two lines), $C2$ (two lines, and $C3$ (one line). Let $I_0(p)$, $I_1(p)$, $I1$, $I0$, and $T(C2)$ be defined as before. Also, let $J(x)$ be the test pattern needed for testing missing device x in the AND plane, as described in Table I. Then Table III summarizes the complete testing scheme for the system of Fig. 6.

V. AN EXAMPLE

Consider a large PLA with 25 inputs, 300 product terms, and 40 output lines. Furthermore, assume that there are 6000 devices in the AND plane. We would like to get an estimate of the cost in area for implementing our design. To this end, we use the standards in [12, p. 103] for area calculations. For simplicity, we separately calculate the cost incurred by circuits $C1, C2, C3, L$, and D. Then, to estimate the total cost, we will add these numbers together. Thus the details of spacings, interconnections, and the actual layouts are ignored. Further, we assume PLA implementations for $C1$, $C2$, and $C3$. This usually is not the most compact implementation; thus the cost estimates for these circuits are upper limits for these values.

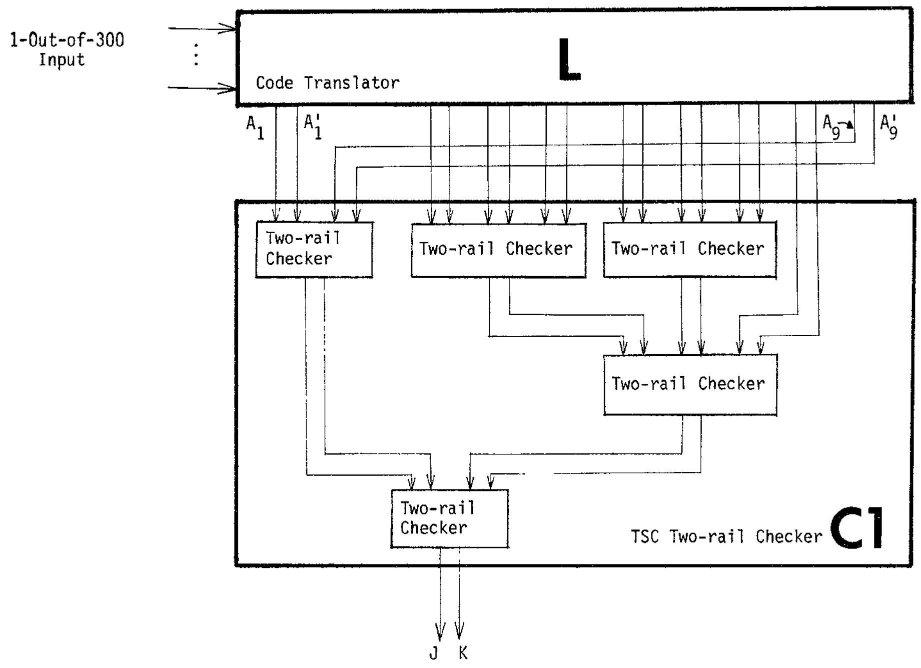

Fig. 10. A TSC 1-out-of-300 code checker.

TABLE III
COMPLETE TEST SET FOR THE SYSTEM OF FIG. 6

Test Input	Error Indicator	Faults Detected	Number of Tests
I0, I1	C2	single faults on bit-lines	2
$I_o(p)$	C1, C3	faults on product lines faults in the OR plane some extra devices in the AND plane, faults in L, faults in C1	r
$I_I(p)$	C1	other extra devices in the AND plane	r
J(x)	C1	missing devices in the AND plane	d
T(C2)	C2	faults in C2	2^t

r : number of product lines;
d : number of devices in the AND plane;
t : number of inputs to the biggest block in the tree implementation of two-rail checker C2.

TABLE IV
AREA COST

Circuit	Area
Original PLA	27,000
L	5,400
C1	264
C2 (Fig. 11)	1,152
D	300
C3 (Fig. 12)	2,740

The unit of area is immaterial, since the purpose here is a relative comparison.

We assume a single output parity for the PLA, i.e., D consists of only one line. Finally, the 1-out-of-r checker (L, $C1$) for $r = 300$ is implemented using the method of Khakbaz [11]. Such an implementation of $C1$ is shown in Fig. 10. Table IV shows the estimate of the areas of these circuits. From Table IV we conclude that the cost of the excess circuitry is about *37 percent of the original PLA area.*

Next consider the question of error-detection delay for this example. Here we define detection delay as follows:

Definition: The delay between the time the output of the PLA is ready and the time that error indicators can be sampled safely is called *error-detection delay*, or simply, delay.

To calculate the delay of this circuit, we assume that each of the PLA input decoders, the AND plane, and the OR plane has one gate delay. Note that the PLA's used for the two-rail checkers have no input decoders. Also, we assume that each XOR gate has two gate delays. With these assumptions, for the above example, we have

delay of $C1$ (Fig. 10): 6 gate delays:
delay of $C2$ (Fig. 11): 5 gate delays:
delay of $C3$ (Fig. 12): 9 gate delays.

Since the PLA itself has 3 gate delays, for each test pattern applied, the tester must wait at least 12 gate delays before it samples the error indicators. From Table III we conclude that $2^t + 2r + d + 2$ test patterns are required. For our example, this number is 6618. Therefore, the total test time is, at least, 6618 * 12 or about 79 000 gate delays.

It would be interesting to compare the characteristics of our design with those of some other proposed schemes. In particular, the designs of Hong [10], Fujiwara [8], and Yajima [22] have been selected for this purpose, since, in our opinion, they represent the most novel and the state-of-the-art ideas in PLA testing. Table V gives a general comparison of these works, denoted by H, F, and Y, respectively. The present work is denoted by K. In the calculations that resulted in this table, in addition to the assumptions made earlier, we have assumed that one gate delay is required for entering a bit into a shift register.

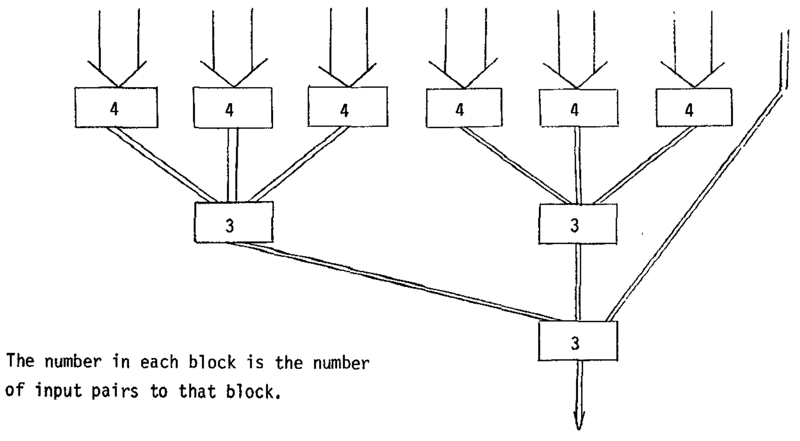
4
4
4
4
4
4
3
3
3
The number in each block is the number of input pairs to that block.

Fig. 11. A TSC two-rail checker tree with 25 input pairs.

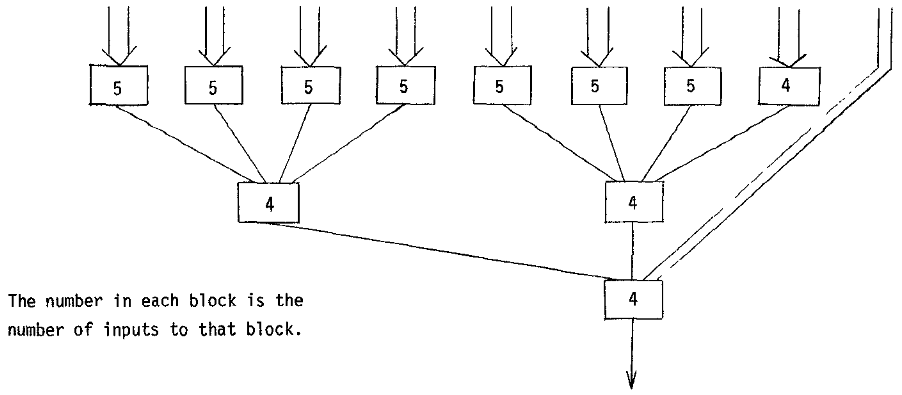
5
5
5
5
5
5
5
4
4
4
4
The number in each block is the number of inputs to that block.

Fig. 12. Parity tree with 41 inputs.

TABLE V
COMPARISON WITH OTHER METHODS

	Concurrent Detection	Function Indep.	No. Test Patterns	Delay per Test Response	No. Extra Pins
K	x		$2r+d+2^t+2$	12	5
H		x	$3n+2r+5$	$7 + 2\lceil\log(r)\rceil$	$7 + \lceil\log(n+1)\rceil$
Y		x	$n+2r+8$	$2m + 7$	3
F		x	$2n+3r$	$2r + 3$	4

n : number of PLA inputs;
m : number of PLA outputs;
r : number of PLA product terms;
d : number of devices in the AND plane;
t : max. number of inputs to a block in C2 tree.

TABLE VI
A COMPARISON, NUMERICAL VALUES

	Number of Test Patterns	Total Test Time in Gate Delays	Number of Extra Pins	Area Cost (Percent of PLA)
K	6,618	79,000	5	37
H	680	16,500	12	21
Y	633	56,000	3	27
F	950	573,000	4	21

To get numerical values, again consider the PLA of the above example. To get an estimate of the cost in area, we assume every block of combinational circuits is PLA implemented. Table VI shows the result.

Note that, of these designs, only this one has combined concurrent error checking with off-line testing. In fact, all of the extra circuitry of Fig. 6 is used for both modes of testing. Thus on the average, over 20-percent savings in area is made over the other systems, as the other designs still need a complete set of circuitry for concurrent error detection. The price for this saving is in terms of the number of test patterns required, which, for our case, is an order of magnitude larger than for the other three cases. However, the total test time for our method is within the same range as the total test time for the other methods, since the response time to each test pattern is much smaller for our design than for the others.

VI. CONCLUSION

A system of checkers is designed for concurrent error detection in large PLA. This system combines concurrent error detection with off-line testing by using the same circuits for both modes of testing. This results in a significant saving in hardware cost.

An example for our proposed error-detection design is provided. For this example, the cost in area is estimated at 37 percent of the area of the original PLA. Concurrent error indication lags the PLA outputs by 9 gate delays. To test the entire system, we need to apply 6618 test patterns, each requiring 12 gate delays before its corresponding error indicators may be sampled. The whole system thus takes about 79 000 gate delays for a complete off-line test. This design requires at most 5 extra pins.

A comparison is made with three other existing designs. The result is that our system is better in terms of area cost, while it requires a bigger test set.

Acknowledgment

The authors wish to thank all the members of the Center for Reliable Computing for their helpful comments and criticisms.

References

[1] V. K. Agarwal, "Multiple fault detection in programmable logic arrays," in *Proc. 9th Annu. Symp. Fault-Tolerant Computing* (FTCS-9), pp. 227-234 (Madison, WI, June 20-22, 1979).

[2] D. A. Anderson and G. Metze, "Design of totally self-checking check circuits for *m*-out-of-*n* codes," *IEEE Trans. Comput.*, vol. C-22, no. 3, pp. 263-269, Mar. 1973.

[3] D. C. Bossen, D. L. Ostapko, and A. M. Patel, "Optimum test patterns for parity networks," in *Proc. American Federation of Information Processing Societies 1970 Fall Joint Computer Conf.*, vol. 37, pp. 63-68 (Houston, TX, Nov. 17-19, 1970).

[4] W. C. Carter and P. R. Schneider, "Design of dynamically checked computer," in *Proc. 4th Congress International Federation of Information Processing Societies*, vol. 2, pp. 878-883 (Edinburgh, Scotland, August 5-10, 1968).

[5] J. H. Clark, "The geometry engine: A VLSI geometry system for graphics," to be published in *SIGGRAPH 82*; also a private conversation, Computer Systems Laboratory, Stanford University, Stanford, CA, 1981.

[6] E. B. Eichelberger and E. Lindbloom, "A heuristic test-pattern generation for programmable logic arrays," *IBM J. Res. Develop.*, vol. 24, no. 1, pp. 15-22, Jan. 1980.

[7] H. Fleisher and L. I. Maissel, "An introduction to array logic," *IBM J. Res. Develop.*, vol. 19, no. 2, pp. 98-109, Mar. 1975.

[8] H. Fujiwara and K. Kinoshita, "A design of programmable logic arrays with universal tests," *IEEE Trans. Comput.*, vol. C-30, no. 11, pp. 823-828, Nov. 1981.

[9] J. Galiay and Y. Crouzet, "Physical vs. logical fault models in MOS LSI circuits, impact on their testability," in *Proc. 9th Annu. Symp. Fault-Tolerant Computing* (FTCS-9), pp. 195-202 (Madison, WI, June 20-22, 1979).

[10] S. J. Hong and D. L. Ostapko, "FITPLA: A programmable logic array for function-independent testing," in *Proc. 10th Annu. Symp. Fault-Tolerant Computing* (FTCS-10), pp. 131-136 (Kyoto, Japan, Oct. 1-3, 1980).

[11] J. Khakbaz, "Totally-self-checking checker for 1-out-of-*n* using two-rail codes," to be published in *IEEE Trans. Comput.* (Special Issue on Fault-Tolerant Computing), vol. C-31, no. 7, July 1982.

[12] C. Mead and L. Conway, *Introduction to VLSI Systems.* Reading, MA: Addison-Wesley, 1980.

[13] V. V. Nickel, "VLSI–The inadequacy of the stuck at fault model," in *Dig. 1980 Test Conf.*, pp. 378-381 (Philadelphia, PA, Nov. 11-13, 1980).

[14] D. L. Ostapko and S. J. Hong, "Fault analysis and test generation for programmable logic arrays," *IEEE Trans. Comput.*, vol. C-28, no. 9, pp. 617-626, Sept. 1979.

[15] D. K. Pradhan and K. Son, "The effects of untestable faults in PLAs and a design for testability," in *Dig. 1980 Test Conf.*, pp. 359-367 (Philadelphia, PA, Nov. 11-13, 1980).

[16] S. M. Reddy, "A note on self-checking checkers," *IEEE Trans. Comput.*, vol. C-23, no. 10, pp. 1100-1102, Oct. 1974.

[17] K. K. Saluja *et al.*, "A multiple fault testable design of programmable logic arrays," in *Proc. 11th Annu. Symp. Fault-Tolerant Computing* (FTCS-11), pp. 44-46 (Portland, ME, June 24-26, 1981).

[18] K. Son and D. K. Pradhan, "Design of programmable logic arrays for testability," in *Dig. 1980 Test Conf.*, pp. 163-166 (Philadelphia, PA, Nov. 11-13, 1980).

[19] R. L. Wadsack, "Technology dependent logic faults," in *Proc. COMPCON Spring 78*, pp. 124-129 (San Francisco, CA, Feb. 28-Mar. 2, 1978).

[20] J. Wakerly, *Error Detecting Codes, Self-Checking Circuits and Applications.* New York, NY: Elsevier-North-Holland, 1978.

[21] S. L. Wang and A. Avizienis, "The design of totally-self-checking circuits using programmable logic arrays," in *Proc. 9th Annu. Symp. Fault-Tolerant Computing* (FTCS-9), pp. 173-180 (Madison, WI, June 20-22, 1979).

[22] S. Yajima and T. Aramaki, "Autonomously testable programmable logic arrays," in *Proc. 11th Annu. Symp. Fault-Tolerant Computing* (FTCS-11), pp. 41-43 (Portland, ME, June 24-26, 1981).