



A New Method for Testing Re-programmable PLAs¹

Charles E. Stroud²

Dept. of Electrical and Computer Engineering, University of North Carolina at Charlotte, Charlotte, NC 28223, U.S.A.

James R. Bailey

Dept. of Electrical Engineering, University of Kentucky, Lexington, KY 40506, U.S.A.

John M. Emmert²

Dept. of Electrical and Computer Engineering, University of North Carolina at Charlotte, Charlotte, NC 28223, U.S.A.

Abstract: We present a method for obtaining a minimal set of test configurations and their associated set of test patterns that completely tests re-programmable Programmable Logic Arrays (PLAs) including EEPROM, UV -EPROM, and SRAM based re-programmable PLAs typically found in Complex Programmable Logic Devices (CPLDs). The resultant set of test configurations and vectors detect all single and multiple stuck-at faults (including line and transistor faults) as well as all bridging faults in the PLA. Previously proposed test methods proposed for EEPROM based PLAs [1,2] require additional test hardware as well as a large number of test configurations and vectors for complete testing. Our approach requires no modification to the PLA and only two or four test configurations, depending on the ratio of PLA product terms to inputs.

Keywords: Electrical Erasable Programmable Logic Array testing, manufacturing test development, bridging faults

1. Introduction and Background

There has been a great deal of research and development in mask Programmable Logic Array (PLA) testing over the past 20 years [8-15], but there has been surprisingly little work reported in the area of testing re-programmable PLAs [1,2]. Re-programmable PLAs are a major component in most Complex Programmable Logic Device (CPLD) architectures [3-6] and newer CPLDs contain hundreds of large PLAs as their major programmable logic structures. Therefore, effective testing techniques for PLAs are necessary in order to ensure efficient testing and manufacturing costs as well as a fault-free product. Typical programming technologies include V-EPROM, EEPROM, flash, and SRAM based memory elements to store the configuration values that establish the logic functions performed by the PLA [3]. All re-programmable PLAs require multiple test configurations (and associated sets of test patterns) for complete testing and must be erased and re-programmed for each test configuration. The time required to erase and re-program these PLAs is substantial and may take as much as one second [2,3]. The test vector application time, on the other hand, is insignificant compared to the time needed to re-program the PLA [2]. Thus the primary goal in re-programmable PLA testing is to minimize the number of times the PLA must be reconfigured; in other words, minimize the number of test configurations.

A design-for-testability (DFT) technique has been proposed for EEPLAs which adds extra circuitry to the PLA in order to detect single and multiple stuck-at faults (including line and transistor faults in the PLA) as well as certain bridging faults on input lines, product term lines, and output lines [1,2]. This extra hardware consists of one transmission gate for each input bit line and each input bit-bar line as well as two test control inputs to the PLA. These transmission gates affect the performance and increase the area of the PLA. The DFT technique was first proposed in [1] along with a method for generating the test configurations and associated test vectors to detect all line stuck-at faults, cross-point faults, and wired-AND bridging faults [1]. In a EEPLA with n inputs, m outputs, and p product terms, a complete test requires $\max\{2np, mp\}$ test configurations.

¹ This effort was supported in part by a grant from Cypress Semiconductor, San Jose, CA.

² Formerly with Dept. of Electrical Engineering, University of Kentucky.

tions and one test vector associated with each test configuration. In [2], the testing time for this DFT technique was improved by reducing the number of test configurations while increasing the number of test vectors per test session. However, this improved approach still requires a large number of test configurations and test patterns. In addition, the number of test configurations varies with the size and dimensions of the EEPLA. Although both of these methods are applicable to any re-programmable PLA with a fully programmable OR-plane, the total testing time is long due to the large number of test configurations and the fact that reconfiguration time is much longer than test vector application time [2].

In this paper, we present a systematic approach to the development of a minimal set of test configurations and the associated set of test patterns for each configuration without the addition of any extra circuitry to the re-programmable PLA. As in [1] and [2], we will only consider re-programmable PLAs with a fully programmable OR-plane. The resultant test configurations and vectors detect all single and multiple stuck-at faults among the cross-points, input lines, product term lines, and output lines as well as wired-AND, wired-OR, and dominant bridging faults [8] among the input lines, product term lines, and output lines. In addition, faults in the programming circuitry are also detected. As will be shown, the number of test configurations and test vectors produced by this method is significantly less than that required by the methods proposed in [1] and [2]. We present the algorithm for generating test configurations and their associated sets of test patterns along with a brief discussion of the fault models considered. We then compare the testing time (in terms of the number of test configurations and test patterns) with that of [2] for various PLA sizes in Section III and conclude in Section IV. For the presentation, analysis, and comparison of the testing method, the variables given in Table 1 will be used.

Table 1. Variables used in testing method presentation, analysis and comparison.

Variable	Representation
n	number of input lines
m	number of output lines
p	number of product term lines
N_{prog}	total number of test configurations
t_{prog}	time required to program the PLA
T_{vec}	total number of test vectors for all test configurations
t_{vec}	time required to apply a single test vector

2. Test Configurations and Vectors

To illustrate our test development, we will consider an 8-input (N0-7, with both bit and bit-bar lines for each input), 8-product term (P0-7), 4-output PLA (M0-3). This PLA can be completely tested for stuck-at faults and bridging faults with only two test configurations. The first test configuration is illustrated in Figure 1 where each product term line is programmed with a unique input bit ANDed together with the other 7 input bit-bars. Note that in the test configurations illustrated in all figures a '1' in the AND-plane implies the bit-line cross-point is active and the bit-bar-line cross-point is inactive while a '0' in the AND-plane implies the bit-line cross-point is inactive and the bit-bar-line cross-point is active. Similarly, a '1' in the or-plane implies the cross-point is active while a '0' implies the cross-point is inactive. The configuration pattern in the AND-plane indicates that one input bit and all other input bit-bars are active in a given product term line. If the number of product terms is greater than the number of inputs ($p > n$), then we would begin to repeat the cross-point pattern such that product term line $n+1$ would have the same pattern as product term line 1. The OR-plane is configured by assigning the first product term line to the first output line, the second product term line to the second output line, and so on. If the number of product term lines is greater than the number of output lines ($p > m$), then we begin to repeat the cross-point pattern in the OR-

plane such that product term line $m+1$ is assigned the same pattern as product term line 1 as illustrated in Figure 1. In Figure 1, only two product term lines are assigned for each output and spaced such that each product term line is active for only one output line. The complete set of test vectors for the first test configuration are also illustrated in Figure 1. This set of test vectors includes the all 0s pattern, walking a 1 through a field of 0s, and all combinations of two 1s in a field of 0s. As a result, there is a total of $(n^2+n+2)/2$ test vectors in the set where n equals the number of inputs to the PLA. For our example PLA in Figure 1 this would be total of 37 test vectors.

The second test configuration, illustrated in Figure 2, is simply the inverse of the first test configuration. Every activated cross-point from the first test configuration is deactivated during the second and vice versa. Similarly, the set of test vectors for the second test configuration is the inverse of the test patterns for the first test configuration. There are a total of $(n^2+n+2)/2$ test vectors in the set consisting of the all 1s pattern, walking a 0 through a field of 1s, and all combinations of two 0s in a field of 1s. One can simply invert the test vectors for the first test configuration in order to obtain the test vectors for the second in the same way as one can invert the first test configuration to obtain the second test configuration.

Stuck-at fault models we consider include any lines stuck-at-1 (sa1) and stuck-at-0 (sa0), cross-point transistors stuck-on and stuck-off, and configuration memory bits sa1 and sa0. These lines include all input lines (both bit and bit-bar), all product term lines, and all output lines. From investigating actual PLA implementations in CPLDs for bridging fault considerations, we find that the physical layout of the PLA is planar. All bit and bit-bar lines are in one plane, product term lines are in another plane, and output lines are either in a third plane or else in the same plane as the input lines. Therefore, when considering adjacency of these lines for bridging faults we can safely assume that a given product term line i , for example, is only adjacent to product term lines $i-1$ and $i+1$. Similarly, output line j is adjacent only to output lines $j-1$ and $j+1$. This assumption holds to a certain extent with input lines in that the bit and bit-bar lines for input k will only be adjacent to the bit and bit-bar lines for inputs $k-1$ and $k+1$. However, the physical layout lines may either repeat (i.e., bit $k-1$, bit-bar $k-1$, bit k , bit-bar k , bit $k+1$, bit-bar $k+1$, etc.) or alternate (i.e., bit $k-1$, bit-bar $k-1$, bit-bar k , bit k , bit $k+1$, bit-bar $k+1$, etc.). We consider both of these layouts for bridging faults. In addition, we consider bridging faults between any input line (bit or bit-bar) and any product term line, any input line and any output line, as well as any product term line and any output line. Using the two test configurations and their associated set of test vectors, the following faults are detected in our example PLA (as verified through fault simulation):

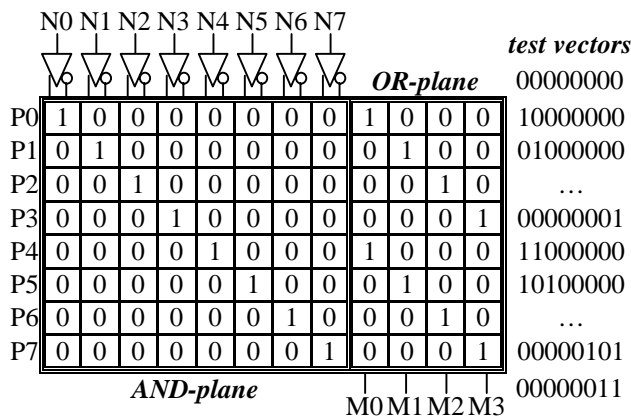


Figure 1. Test Configuration # 1 and Test Vectors

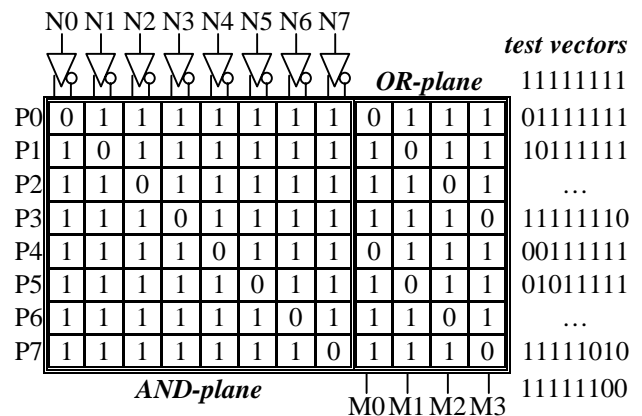


Figure 2. Test Configuration #2 and Test Vectors

- ◆ 100% of single stuck-at faults including line and transistor faults
- ◆ 100% of wired-AND, wired-OR, and dominant bridging faults [7] between:
 - input lines (any combination of bit/bit-bar, bit/bit, and bit-bar/bit-bar lines)
 - product term lines
 - output lines
 - input and product term lines
 - input and output lines
 - product term and output lines
- ◆ 100% of multiple stuck-at faults (verified via fault simulation of several different sets of 1000 randomly sampled fault groups containing from 2 to 10 randomly sampled faults per group)

Test configurations for the AND-plane (similar to these two test configurations) are commonly used by some CPLD manufacturers for testing PLAs within the CPLD. The typical test patterns only include the all 0s test pattern, walking a 1 through a field of 0s, the all 1s test pattern, and walking a 0 through a field of 1s. However, the test vectors consisting of every combination of two 1s in a field of 0s and vice versa is the key to detecting many additional stuck-at faults in the PLA without having to reprogram it many times as was done in [1] and [2]. Walking a 1 through a field of 0s detects all wired-AND and wired-OR bridging faults in the PLA. However, the key to detecting all dominant bridging faults is in the configuration of the PLA. By alternating active cross-points with respect to the physical layout of the PLA, the walking 1 vectors will detect all appropriate bridging faults.

These two test configurations and their associated set of test patterns can completely test any full programmable OR-plane based re-programmable PLA where $n \geq p$. However, for the case of $p > n$, two additional test configurations are needed for complete testing of the OR-plane. The first two configurations (shown in Figures 1 and 2) completely test the programmable AND-plane for all stuck-at faults and bridging faults, regardless of the number of product term lines, p . In addition, these two test configurations test most of the faults in the OR-plane even when $p > n$. However, the repeated product terms result in a failure to detect some faults including some product term lines sa0 at the output line cross-points as well as some output line cross-points stuck-on. The third test configuration detects the remaining output line cross-points stuck-on. This test configuration is obtained by simply activating all bit cross-points in the AND-plane (to turn on all product term lines regardless of the inputs) while deactivating all cross-points in the OR-plane. There is only one test vector for this configuration

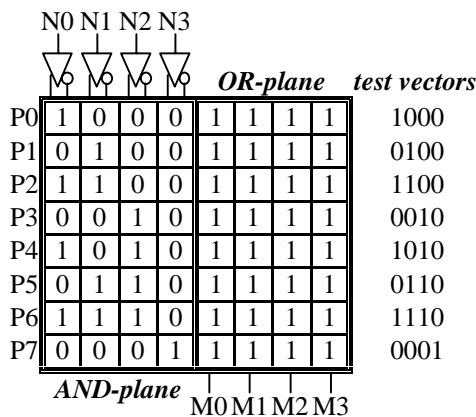


Figure 3. Test Configuration #4 and Test Vectors

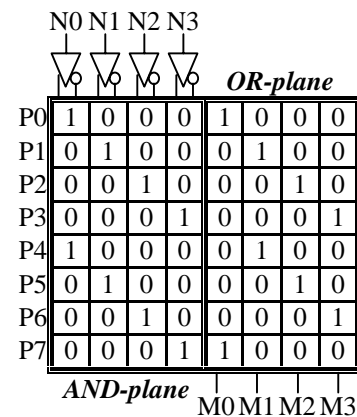


Figure 4. Modified Test Configuration #1

consisting of all don't cares while an output response of all 0s can be expected for the fault-free case. The fourth test configuration detects the product term lines sa_0 at the output line cross-points. This test configuration is obtained by assigning the binary value associated with each product term line as the configured product term in the AND-plane. For example, the first product term line 000...001, the second product term line 2 would be assigned 000...010, and so on as illustrated in Figure 3. Since there are 2^n-1 unique non-zero binary values available for product term assignment, this covers all possible sizes for a PLA due to the fact that a structure with 2^n product terms is a ROM (and not a PLA), where n is the number of inputs. For this test configuration, all cross-points are activated in the OR-plane. The set of test vectors consists of the binary count values from 1 to p , where p is the number of product term lines.

A special case occurs when $p > n$ and p is an integral multiple of n and m . In this case the first test configuration is modified so that when the OR-plane pattern is repeated, the activated cross-points are shifted by one (each time they are repeated) as illustrated in Figure 4. Here the objective is to avoid identical product terms being assigned to the same output line. The second test configuration for this special case is again the inverse of this modified first test configuration. The sets of test vectors for the first two test configurations remain unchanged along with the third and fourth test configurations and their sets of test vectors as described above.

3. Testing Analysis and Comparison for PLAs

In this section we consider the various sizes of PLAs that were used for analysis in [2] and compare our results with those reported in [2]. Recall that the method proposed in [2] produced fewer test configurations and vectors than the method proposed in [1]. Total testing time for completely testing a re-programmable PLA is given by [2]:

$$T_{test} = N_{prog} t_{prog} + T_{vec} t_{vec}$$

The total testing time for completely testing a re-programmable PLA with our method is given by:

$$\begin{aligned} T_{test} &= 2t_{prog} + (n^2 + n + 2)t_{vec} && \text{for } n \geq p \\ T_{test} &= 4t_{prog} + (n^2 + n + p + 3)t_{vec} && \text{for } p > n \end{aligned}$$

Therefore, testing time with our approach is primarily a function of the PLA architecture, specifically whether $p > n$ or $n \geq p$. This is due to the fact that configuration time is much greater than test vector application time [2]. As a secondary order effect, testing time is a function of the number of inputs to the PLA, n . Total testing time for the methods proposed in [1] and [2] are dependent on the size of the PLA with respect to the number of inputs, n , as well as the number of product terms, p , and the number of outputs, m . Six cases were considered in [2] for varying sizes of n , p , and m . In Table 2, we summarize those six cases with example values of n , p , and m in order to compare the total testing time associated with our approach to that of [2]. It should be noted that Case 4 is the best-case scenario for the method proposed in [2], in terms of the minimum number of test configurations required. We specifically chose this case as an example to show that the method in [2] does not always require a large number of test configurations. However, even in this case, there is a significant reduction in the number of test configurations using our approach. In all cases, the number of test configurations required by our method is significantly less than the number required by the method proposed in [2] which required from 9 to 512 times as many test configurations as our approach. The number of test vectors required by our method is less than that required by the method proposed in [2] in most cases. However, in the two cases (Case 4 and Case 6) where fewer test vectors were required by the method in [2], the larger number of test configurations and the time to reconfigure the PLA result in longer total test times than with our approach.

Table 2. Comparison of test time results for our method and the method in [2]

Case: PLA size	Example PLA size	Our Method	Method in [2]
Case 1: $m \leq n < p$	$n=16, p=64$ and $m=8$	$T_{test} = 4t_{prog} + 339t_{vec}$	$T_{test} = 72t_{prog} + 2304t_{vec}$
Case 2: $n \leq m < p$	$n=8, p=64$ and $m=16$	$T_{test} = 4t_{prog} + 139t_{vec}$	$T_{test} = 68t_{prog} + 1075t_{vec}$
Case 3: $n < p \leq m$	$n=16, p=32$ and $m=40$	$T_{test} = 4t_{prog} + 307t_{vec}$	$T_{test} = 41t_{prog} + 1289t_{vec}$
Case 4: $m \leq p \leq n$	$n=32, p=16$ and $m=8$	$T_{test} = 2t_{prog} + 1058t_{vec}$	$T_{test} = 18t_{prog} + 960t_{vec}$
Case 5: $p \leq n \leq m$	$n=19, p=16$ and $m=40$	$T_{test} = 2t_{prog} + 382t_{vec}$	$T_{test} = 42t_{prog} + 2663t_{vec}$
Case 6: $p \leq m \leq n$	$n=32, p=16$ and $m=20$	$T_{test} = 2t_{prog} + 1058t_{vec}$	$T_{test} = 1024t_{prog} + 1024t_{vec}$

4. Conclusions

We have presented a method for completely testing re-programmable PLAs with fully programmable OR-planes without any extra circuitry or modifications to the PLA itself. This method minimizes the number of test configurations required by adding test vectors to detect faults that previously required additional test configurations. The number of test configurations is, for the most part, independent of the size of the PLA thus making this method practical for any size PLA structure. Depending on the architecture of the PLA, either two or four test configurations will be needed. This greatly reduces the amount of time needed to test a programmable PLA since the number of test configurations dominates testing time. The number of test vectors needed is primarily dependent on the number of inputs to the PLA. By selecting the appropriate configurations for the AND-plane and the OR-plane, complete bridging fault coverage (including wired-AND, wired-OR and dominant bridging fault models) is achieved within this set of test configurations and vectors. All single and multiple stuck-at faults including line and transistor faults are also detected. This technique improves on the approaches proposed in [1,2] since testing time is reduced without the addition of any DFT hardware that would impact the PLA performance or size.

References

- [1] R. Rajsuman, "A New Testing Method for EEPLA", *IEEE Trans. on CAD*, Vol. 13, No. 7, pp. 935-939, 1994.
- [2] A. Munshi, F. Meyer, and F. Lombardi, "A New Method for Testing EEPLA's", *Proc. IEEE International Symp. on Defect and Fault Tolerance in VLSI Systems*, pp. 146-154, 1998.
- [3] K. Skahill, *VHDL for Programmable Logic*, Addison Wesley, 1996.
- [4] J. Oldfield and R. Dorf, *Field Programmable Gate Arrays*, Wiley Interscience, 1995.
- [5] *Cypress Data Book CD-ROM*, Cypress Semiconductor Corp., 1997.
- [6] *MACH Family Data Book*, Advanced Micro Devices, 1992.
- [7] D. Chess and T. Larrabee, "Bridge Fault Simulation Strategies for CMOS Integrated Circuits", *Proc. ACM/IEEE Design Automation Conf.*, pp. 458-462, 1993.
- [8] V. Agarwal, "Multiple Fault Detection in Programmable Logic Arrays", *IEEE Trans. On Computers*, Vol. C-29, No. 6, pp. 518-522, 1980.
- [9] P. Bose and J. Abraham, "Test Generation for Programmable Logic Arrays", *Proc. 19th Design Automation Conf.*, pp. 574-580, 1982.
- [10] C. Cha, "A Testing Strategy for PLAs", *Proc. ACM/IEEE Design Automation Conf.*, pp. 326-331, 1978.
- [11] E. Eichelberger and E. Lindbloom, "A Heuristic Test-Pattern Generator for Programmable Logic Arrays", *IBM J. of Research and Development*, Vol. 24, No. 1, pp. 15-22, 1980.
- [12] H. Fujiwara, "A New PLA Design for Universal Testability", *IEEE Trans. on Computers*, Vol. C-33, No. 8, pp. 745-750, 1984.
- [13] D. Ostapko and S. Hong, "Fault Analysis and Test Generation for Programmable Logic Arrays", *IEEE Trans. on Computers*, Vol. C-28, No. 9, pp. 617-627, 1979.
- [14] D. Pradhan and K. Son, "The Effect of Undetectable Faults in PLAs and a Design for Testability", *Proc. IEEE International Test Conf.*, pp. 359-367, 1980.
- [15] J. Smith, "Detection of Faults in Programmable Logic Arrays", *IEEE Trans. on Computers*, Vol. C-28, No. 11, pp. 848-853, 1979.