Short Papers

Enhancing Random-Pattern Coverage of Programmable Logic Arrays via Masking Technique

HIDEO FUJIWARA, FELLOW, IEEE

Abstract—**This paper presents a testable design of programmable logic arrays (PLA's) with high fault coverage for random test patterns. The proposed design is realized with low area overhead by adding two mask arrays to the AND and OR arrays of the PLA. To demonstrate the effect of the masking technique, an experiment was performed in which eight large PLA's were modified by adding various sizes of mask arrays, and then fault simulation with random patterns for those modified and unmodified PLA's was carried out to obtain random-pattern test coverage curves. It was found that fault coverage can be significantly enhanced via the proposed masking technique with very low area overhead.**

I. Introduction

Programmable logic arrays (PLA's) are very suitable to VLSI and have become a popular and effective tool for implementing logic functions because of their regular structure. On the other hand, the built-in self-test (BIST) approach using linear feedback shift registers (LFSR's) is currently being widely investigated as an attractive testing technique for VLSI circuits [1]. The major difficulty in random testing using LFSR-generated pseudorandom patterns is the low fault coverage for very high fan-in circuits such as PLA's. Hence, for BIST PLA's it is necessary either to employ deterministic (not random) test patterns or to augment a PLA to make it random-pattern-testable. The former includes BIST PLA designs with universal test patterns [2]-[6]. Although these BIST PLA's can achieve very high fault coverage, the area overhead is still high. For the latter approach, two random-pattern-testable designs of PLA's were proposed by Eichelberger and Lindbloom [7] and Ha and Reddy [5]. However, these methods also have high area overhead due to their extra circuitry for controlling a large number of product lines of PLA's.

In [9], we proposed an approach to designing a random-pattern-testable PLA with very low area overhead by adding a mask array only between the input decoder and the AND array. We estimated the number of random patterns necessary for achieving a given test confidence by analyzing the detection probability of stuck-at and crosspoint faults. In this paper, we shall propose an improved design based on the approach of [9] in which a mask array is added to the OR array in addition to the AND array of a PLA. To demonstrate the effect of the masking technique, an experiment was performed in which eight large PLA's were modified by adding various sizes of mask arrays, and then fault simulation with LFSR-generated pseudorandom patterns for those modified and unmodified PLA's was carried out to obtain fault coverage curves. We shall present the experimental results: real fault coverage curves for the eight large PLA's to demonstrate the effect of masking technique. It has been found that fault coverage can be significantly enhanced via the masking technique proposed in this paper while keeping the area overhead very low.

Manuscript received April 4, 1988; revised November 3, 1988. The review of this paper was arranged by Associate Editor V. K. Agarwal.

The author is with the Department of Computer Science, Meiji University, Kawasaki 214, Japan.

IEEE Log Number 8928247.

II. Random-Pattern-Testable PLA's

A PLA consists of three main sections: a decoder, an AND array, and an OR array. The decoder section usually consists of a collection of one-input or two-input decoders. Both the AND array and the OR array are used to implement multi-output combinational logic with sum-of-product forms. A PLA is typically implemented as a NOR–NOR array in nMOS technology.

Fig. 1 shows the design of random-pattern-testable PLA's proposed in this paper. The augmented PLA has additional circuitry consisting of two programmable mask arrays, called the *bit-mask* array and the *product-mask* array, which selectively mask the inputs of the AND array and the OR array, respectively. These mask arrays are programmed to increase the fault coverage of the PLA effectively, and are activated by extra inputs $t_1, t_2, \cdots, t_r$. These control inputs are all set to value zero during functional operation. Fig. 2 shows an example of the proposed PLA with mask decoders in nMOS technology.

The principle of masking is illustrated in Fig. 3 using an AND-OR equivalent circuit. In the figure, by controlling $u_1 = 0$, $u_2 = 1$, a_3 and a_4 are masked and only the inputs of a_1 and a_2 are applied to the AND gate. This causes the fan-in of the AND gate to decrease from 4 to 2. Here, the control lines u_1 and u_2 are called *mask-control lines*, the masked inputs a_3 and a_4 are called *mask*, and unmasked inputs such as a_1 and a_2 are called *window*. We consider the masking form which is illustrated in Fig. 4. The form is mask-disjoint; i.e., no pair of masks overlap each other and the union of all masks covers the fan-in of all AND and OR arrays.

We shall consider two schemes of PLA's which are random-pattern-testable as follows:

Scheme 1: The augmented PLA with the bit-mask array but without the product-mask array.

Scheme 2: The augmented PLA with both the bit-mask and product-mask arrays.

We assume that random patterns are applied not only to the primary inputs of the PLA but also to the mask-control inputs in testing. Hence, in the augmented PLA, more than two mask-control lines may be active. Mask-disjoint form of masking is thus useful to this scheme. As illustrated in Fig. 3, the purpose of masking is to decrease the fan-in of AND and OR arrays in order to increase the fault detection probability. Those mask arrays should be programmed to enhance most effectively the fault coverage of the PLA. In the next section, we shall present a method of programming mask patterns for mask arrays.

III. Programming Mask Patterns

The most effective mask arrays that yield the maximum enhancement of fault detection probability could be obtained by considering the detailed connection information of both the AND and OR arrays. However, it is a difficult and time-consuming problem to obtain the optimum solution. Here we shall consider a simple method of generating mask patterns for each of the AND and OR arrays separately.

Let us consider a mask array and a masked array shown in Fig. 5. When the masked array is an AND array, inputs and outputs of the masked array correspond to bit lines and product lines of the PLA, respectively. When the masked array is an OR array, inputs and outputs of the masked array correspond to product lines and

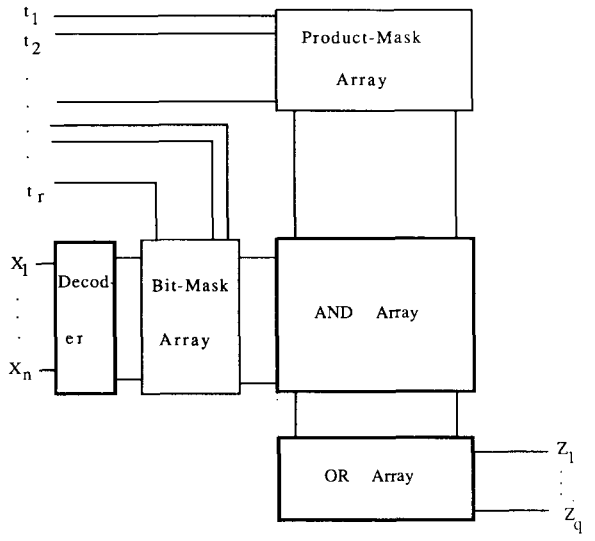

Fig. 1. The proposed PLA.

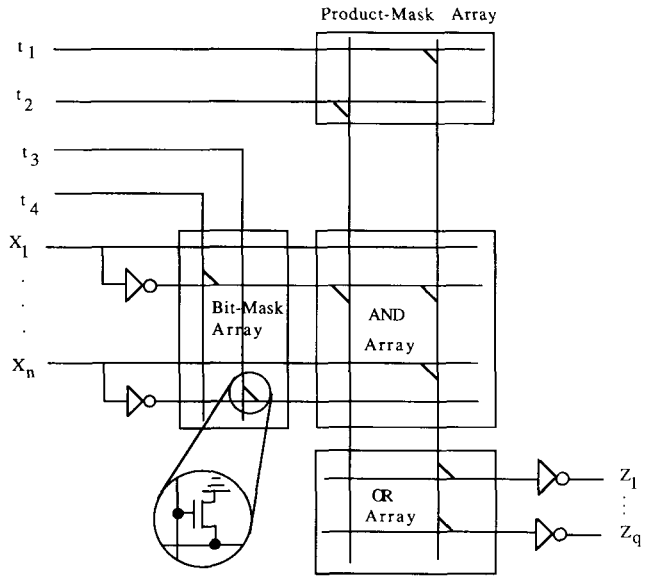

Fig. 2. Realization in nMOS.

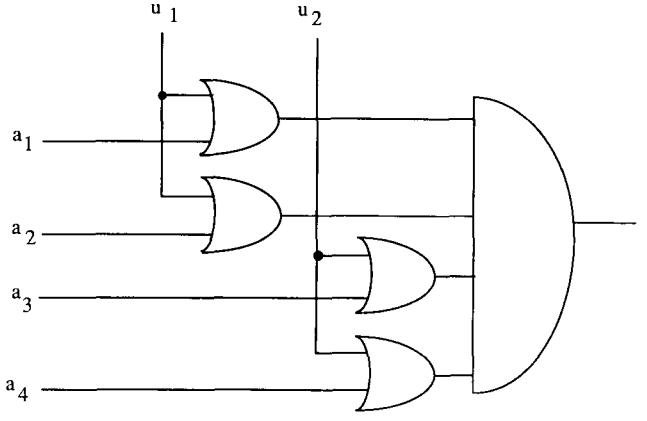

Fig. 3. Masking of AND gate.

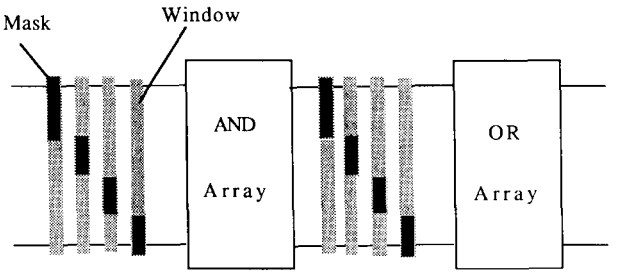

Fig. 4. Masking form of PLA.

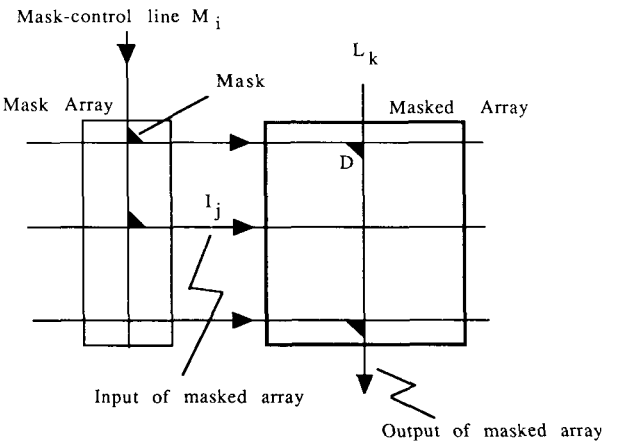

Fig. 5. Mask array and masked array.

outputs of the PLA, respectively. In Fig. 5, input I_j is *masked* by mask-control line M_i, and device D is also masked by M_i.

Suppose that there are μ mask-control lines, $M_1, M_2, \cdots, M_\mu$. The problem of programming a mask array for those mask-control lines is to generate a set of masked inputs for each mask-control line, i.e., to determine which input of the masked array is masked by M_i for each $i = 1, 2, \cdots, \mu$. Note that these sets of masked inputs are mutually disjoint. The fault detection probability of the masked array depends to a great extent on the size of windows or masks of the programmed mask array. The size of windows or masks is determined from active mask-control lines, which will be assumed to be selected equally likely by random test patterns. In order to obtain an equal effect of masking from each mask-control line, it would be better for each line L_k to be masked as uniformly as possible by all mask-control lines. From this viewpoint, we shall consider a method for programming a mask array which masks all devices in the masked array as uniformly as possible.

Let us consider the devices of line L_k in Fig. 5. Let m_{ik} and d_k be the number of devices masked by M_i and the number of all devices in L_k, respectively. The most uniform masking occurs when

$$m_{1k} = m_{2k} = \cdots = m_{\mu k} = \frac{d_k}{\mu}.$$

The difference of m_{ik} from d_k/μ represents a degree of inequality or lack of uniformity for m_{ik}. Hence the summation of those differences

$$\sum_{i=1}^{\mu} \left| m_{ik} - \frac{d_k}{\mu} \right|$$

represents a degree of total inequality of masking for line L_k. Normalizing this value by d_k, we have a measure D_k which can be used as an index of inequality of masking with respect to line L_k:

$$D_k = \sum_{i=1}^{\mu} \left| \frac{m_{ik}}{d_k} - \frac{1}{\mu} \right|.$$

Procedure for Mask Pattern Generation

Step 1: Calculate current degrees of inequality, D_j, for all lines L_j ($j = 1, 2, \cdots, \lambda$).
Step 2: Find the maximum of D_j ($j = 1, 2, \cdots, \lambda$). Let L_α be a line whose degree of inequality is maximum, i.e., $D_\alpha = \max\{D_j\}$.
Step 3: For the line L_α, find the minimum of $m_{i\alpha}$ ($i = 1, 2, \cdots, \mu$). Let M_β be the mask-control line such that $m_{\beta\alpha} = \min\{m_{i\alpha}\}$.
Step 4: Add a new mask point to the mask array with respect to mask-control line M_β so that m_β masks one of the unmasked inputs of line L_α to increment $m_{\beta\alpha}$. If there are more than two unmasked inputs, then select one whose fan-out is maximum, where the fan-out of an input is the number of lines to which the input fans out.
Step 5: If there remain unmasked inputs, then go to step 1. Otherwise, stop.

This procedure does not guarantee that the optimum solution, i.e., the most uniform mask array will be obtained, but it will produce a relatively uniform mask array using a measure of inequality of masking. Uniformity of masking is not our final objective; it is only a shortcut to obtain a mask array which will be expected to enhance the fault detection probability. Further, the correlation between the heuristic used in the above procedure and the actual mask effectiveness is difficult to show explicitly. Therefore, we would like to study actual PLA's so see how much the fault coverage would be enhanced after augmentation. In the next section, we shall show the experimental results.

IV. Experimental Results

To demonstrate the effect of the masking technique, an experiment was performed using eight large actual PLA's, most of which are control logic circuits. These PLA's were modified to PLA's embodying two types of schemes by adding various sizes of mask arrays. Then, fault simulation with LFSR-generated pseudorandom patterns for the modified and unmodified PLA's was carried out to obtain fault coverage curves. In [9], it was shown that the detection probability of crosspoint faults can be represented in the same way as that of stuck-at faults, and so we have considered here only single stuck-at faults in AND and OR arrays. In the following, we shall present the experimental results: area overhead for augmented PLA's and fault coverage enhancement of masking.

A. Area Overhead

Let us estimate the area overhead of two schemes of augmented PLA's. Let n, p, and m be the numbers of inputs, product lines, and outputs of the original PLA, respectively. Let μ and λ be the numbers of mask-control lines of bit-mask array and product-mask array, respectively.

We shall estimate the area by the number of transistor equivalents. Each area of the augmented PLA's of schemes 1 and 2 can be expressed as follows:

$2np$: area of the AND array,
mp: area of the OR array,
$2\mu n$: area of the bit-mask array,
λp: area of the product-mask array,
cn: area of the input-decoder of the original PLA,

where c is constant.

The area overhead of an augmented PLA is defined as follows:

$$\text{area overhead} = \frac{\text{extra area}}{\text{original area of the PLA}} \times 100\%.$$

TABLE I
Characteristics and Area Overhead of PLA's

Name	Inputs	Outputs	Product Lines	Scheme 1 (%)		Scheme 2 (%)	
				μ=2	μ=4	λ=μ=2	λ=μ=4
PLA 1	22	29	87	1.37	2.73	4.07	8.14
PLA 2	23	62	101	0.84	1.67	2.67	5.35
PLA 3	25	8	110	1.54	3.09	4.94	9.88
PLA 4	25	18	29	4.83	9.65	7.63	15.25
PLA 5	27	37	68	1.72	3.43	3.88	5.25
PLA 6	28	17	47	3.16	6.32	5.81	11.63
PLA 7	32	11	124	1.36	2.72	3.99	7.98
PLA 8	37	35	111	1.21	3.02	2.42	6.04

TABLE II
Fault Coverage with Pseudorandom Patterns

Name	Original PLA			Scheme 1						Scheme 2					
				μ=2			μ=4			λ=μ=2			λ=μ=4		
	500	5K	50K	500	5K	50K	500	5K	50K	500	5K	50K	500	5K	50K
PLA 1	51.2	87.6	96.8	72.4	94.3	99.1	82.6	92.0	98.1	74.9	95.9	99.1	83.8	98.3	99.9
PLA 2	39.3	43.9	54.6	52.4	72.7	79.2	65.6	80.3	85.4	54.6	74.3	88.9	59.5	84.8	95.4
PLA 3	48.8	80.0	87.2	60.4	88.4	99.7	59.2	92.4	99.5	42.9	88.5	99.3	62.7	97.6	100
PLA 4	69.9	87.3	100	95.8	100	100	92.7	100	100	77.6	100	100	97.5	100	100
PLA 5	72.7	95.6	100	69.4	96.9	100	79.2	98.7	99.9	64.3	96.5	100	92.4	99.9	100
PLA 6	63.9	93.6	99.7	79.8	93.7	98.6	86.8	96.1	98.2	85.3	96.9	98.6	90.7	98.5	99.9
PLA 7	40.7	67.9	87.4	46.6	71.5	84.9	51.7	70.2	83.2	49.9	78.8	92.2	62.7	89.7	99.4
PLA 8	66.8	96.2	98.5	69.1	89.2	98.7	74.6	95.8	99.6	71.3	93.3	99.2	83.4	99.4	100

Hence, the area overhead of the PLA of scheme 1 is calculated by

$$\text{area overhead of scheme 1} = \frac{2\mu n}{2np + mp + cn} \times 100\%$$

and that of Scheme 2 is calculated by

$$\text{area overhead of scheme 2} = \frac{2\mu n + \lambda p}{2np + mp + cn} \times 100\%.$$

Table I shows the characteristics of eight large PLA's and the area overheads of the augmented PLA's of schemes 1 and 2, where $c = 4$ is assumed. The fifth through eighth columns show the area overhead of schemes 1 and 2.

B. Fault Coverage Enhancement

In Table II we have presented the results on fault coverage with 500, 5000, and 50 000 pseudorandom patterns for each PLA. Among the eight PLA's, three original PLA's, #2, #3, and #7, are pseudorandom-pattern resistant; i.e., fault coverages for those PLA's are low after fault simulation even with 50 000 pseudorandom patterns, as shown in Table II. After modification, scheme 1 achieved high fault coverage for PLA #3 but not for PLA's #2 and #7. On the contrary, scheme 2 was able to achieve very high fault coverage for all PLA's, especially in the case where $\mu = \lambda = 4$.

Fig. 6 shows fault-coverage curves with unmodified and four modified schemes for PLA #2. The unmodified PLA had a 43.9 percent fault coverage with 5000 patterns and a 54.6 percent coverage with 50 000 patterns. The modified PLA of scheme 1 could not reach a 90 percent fault coverage after 50 000 patterns. The

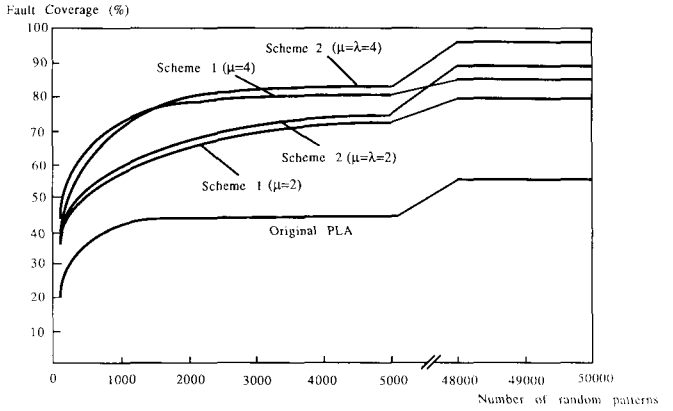

Fig. 6. Random-pattern test coverage curves for PLA 2.

modified PLA of scheme 2 achieved a 95.4 percent fault coverage with 50 000 patterns when $\mu = \lambda = 4$. Therefore scheme 2 with $\mu = \lambda = 4$ is the best among them for achieving a high fault coverage.

V. Conclusion

We have proposed a design of random-pattern testable PLA's. The proposed design is realized with very low area overhead: 0.84 percent through 15.25 percent for eight benchmark PLA's. We have also presented experimental results to show that the fault coverage can be significantly enhanced; for example, a 54.6 percent fault coverage with 50 000 pseudorandom patterns of an original PLA can be enhanced to 95.4 percent after modification of the PLA with 5.35 percent additional logic. The experimental results show that the proposed approach achieved almost 100 percent fault coverage in pseudorandom testing with very low area overhead for all eight benchmark PLA's.

Acknowledgment

The author would like to thank Prof. T. Sasao of Kyushu Institute of Technology and T. Yoshimura of the NEC Corporation for their kind offer of benchmark PLA's. Thanks are also due to O. Fujisawa and K. Hikone for their assistance in obtaining the experimental results of this work.

References

[1] H. Fujiwara, *Logic Testing and Design for Testability*. Cambridge, MA: The MIT Press, 1985.

[2] K. A. Hua, J.-Y. Jou, and J. A. Abraham, "Built-in tests for VLSI finite-state machines," in *Proc. 14th Int. Symp. Fault-Tolerant Computing*, 1984, pp. 292–297.

[3] R. Treuer, H. Fujiwara, and V. K. Agarwal, "Implementing a built-in self-test PLA design," *IEEE Design and Test of Computers*, vol. 2, no. 2, pp. 37–48, Apr. 1986.

[4] C-Y. Liu, K. K. Saluja, and J. S. Upadhyaya, "BIST-PLA: A built-in self-test design of large programmable logic arrays," in *Proc. 24th Design Automat. Conf.* 1987, pp. 385–391.

[5] D. S. Ha and S. M. Reddy, "On BIST PLA's," in *Proc. 1987 Int. Test Conf.*, 1987, pp. 342–351.

[6] D. L. Liu and E. J. McCluskey, "Design of large embedded CMOS PLA's for built-in self-test," in *Proc. 1987 IEEE Int. Conf. Computer Design*, 1987, pp. 678–681.

[7] E. M. Eichelberger and E. Lindbloom, "Random pattern coverage and diagnosis for LSSD logic self-test," *IBM J. Res. Develop.*, vol. 27, pp. 265–272, May 1983.

[8] D. S. Ha and S. M. Reddy, "On the design of random pattern testable PLA's," in *Proc. 1986 Int. Test Conf.*, 1986, pp. 688–695.

[9] H. Fujiwara, "A design of programmable logic arrays with random-pattern-testability," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 5–10, Jan. 1988.

Schwarz–Christoffel Transformation for the Simulation of Two-Dimensional Capacitance

Ç. K. KOÇ AND P. F. ORDUNG

***Abstract*—An inherent problem in the use of simulators for the determination of capacitance in VLSI circuits is the verification of the reliability of the simulation. The problem is due to the numerical approximations made in order to achieve a versatile simulation. The Schwarz–Christoffel transformation provides theoretically exact simulation of a limited class of problems consisting of two odd shaped conductors embedded in a uniform dielectric. We propose that the Schwarz–Christoffel technique can be used to calibrate simulators designed for more general problems.**

I. Introduction

Estimating parasitic capacitance of VLSI buses is a crucial step in computing circuit delay, particularly as packing density increases and conductor spacings decrease [5]. A great deal of effort has been spent in designing simulators which take the cross section geometry of the conductor–dielectric system as input, and compute the capacitance per unit length between the metal lines [2], [3].

The numerical methods to estimate the capacitance are usually CPU intensive, and thus prohibitive for VLSI layouts. Another solution is to find geometry-dependent approximate formulas which require short run times and a modest amount of memory [1]. An inherent problem in using simulators to estimate the parasitic capacitance is the difficulty of determining the reliability of the predicted capacitance values. The reason for this is that all methods in one way or another involve approximations which are difficult to evaluate.

The Schwarz–Christoffel conformal mapping technique, on the other hand, does not require *a priori* approximations. As long as the computations can be performed free of round-off error, the computed value would be the exact capacitance of the conductor–dielectric system. The Schwarz–Christoffel mapping technique provides theoretically exact estimates for a limited class of problems, which can be used to check the reliability of simulators designed for more general problems.

II. Schwarz–Christoffel Transformation

In this section, we show the application of the Schwarz–Christoffel conformal mapping technique for a class of capacitance problems. We consider a pair of conductors which exhibit symmetry

Manuscript received April 29, 1988; revised December 14, 1988, and February 20, 1989. This work was supported by the University of California and Rockwell International under the MICRO Grant UC-86-033/C7CJ-242025. The review of this paper was arranged by Associate Editor D. Rose.

Ç. K. Koç is with the Department of Electrical Engineering, University of Houston, Houston, TX 77204.

P. F. Ordung is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106.

IEEE Log Number 8928243.