Deviation-Based LFSR Reseeding for Test-Data Compression

Zhanglei Wang, Hongxia Fang, Krishnendu Chakrabarty, *Fellow, IEEE*, and Michael Bienek

Abstract—Linear feedback shift register (LFSR) reseeding forms the basis for many test-compression solutions. A seed can be computed for each test cube by solving a system of linear equations based on the feedback polynomial of the LFSR. Despite the availability of numerous LFSR-reseeding-based compression methods in the literature, relatively little is known about the effectiveness of these seeds for unmodeled defects, particularly since there are often several candidate seeds for a test cube. We use the recently proposed output deviation measure of the resulting patterns as a metric to select appropriate LFSR seeds. Experimental results are reported using test patterns for stuck-at and transition faults derived from selected seeds for the ISCAS-89 and the IWLS-05 benchmark circuits. These patterns achieve higher coverage for transition and stuck-open faults than patterns obtained using other seed-generation methods for LFSR reseeding. Given a pattern pair (p_1, p_2) for transition faults, we also examine the transition-fault coverage for launch on capture by using p_1 and p_2 to separately compute output deviations. Results show that p_1 tends to be better when there is a high proportion of do-not-care bits in the test cubes, while p_2 is a more appropriate choice when the transition-fault coverage is high.

Index Terms—Defect coverage, linear feedback shift register (LFSR) reseeding, output deviation, seed selection, test compression.

I. INTRODUCTION

A NUMBER of fault models such as stuck-at, transition delay, and shorts/opens are typically used today to achieve high defect coverage [1]. As a result, the test-data volumes for today's integrated circuits are prohibitively high. For example, the test-data volume for transition-delay faults is two to five times higher than that for stuck-at faults [2], and test patterns for such sequence- and timing-dependent faults are more important for newer technologies [3]. Moreover, due to shrinking process technologies, the physical limits of photolithographic processing, and new materials such as copper interconnects, many new types of manufacturing defects cannot be accurately modeled using known fault models [4]. Although efficient methods exist today for fault-model-oriented test generation [5], [6], there

is a lack of understanding on how best to combine the test sets thus obtained, i.e., derive the most effective union of the individual test sets, without increasing the test-data volume excessively by simply using all the patterns for each fault model. As a result, the 2007 International Technology Roadmap for Semiconductors final draft predicts that the test-data volume for integrated circuits will be as much as 38 times larger and the test-application time will be about 17 times larger in 2015 than it is today [7]. Test compression is therefore essential to reduce the test-data volume and testing time.

A category of test-compression techniques relies on the use of a linear decompressor such as a linear feedback shift register (LFSR) or an XOR network. These methods exploit the fact that test cubes for modeled faults in full-scan circuits contain very few specified (care) bits; a test cube with many do-not-care bits can be easily generated using an LFSR with a compact seed or by using a simple XOR "expander" network. Such compression techniques have been implemented in commercial tools such as TestKompress [8].

LFSR reseeding has long been recognized as an efficient test-compression technique [9]–[17]. A test cube, whose length equals the number of controllable inputs (primary inputs and scan cells) in the circuit, can be encoded with high probability into a compact seed of typical length $S_{\max} + 20$, where $S_{\max}$ equals the number of specified bits in the test cube [9]. A seed can be computed for each test cube by solving a system of linear equations based on the feedback polynomial of the LFSR. The solution space for the system of linear equations is quite large, particularly for test cubes with few specified bits. For example, if the LFSR has n stages and the number of care bits in the test cube is m , the number of solutions is at least 2^{n-m} . All patterns derived from the possible solutions (LFSR seeds) cover the original test cube. However, these patterns differ from each other in their ability to detect unmodeled defects. Therefore, it is important to select LFSR seeds that can be used to target a larger number of defects. Moreover, if more effective seeds are loaded first into the LFSR, a steeper ramp-up of fault coverage can be obtained for various fault models.

The choice of LFSR seeds for compression methods is either random, e.g., as an outcome of Gauss–Jordan elimination [18], or seed selection is designed for better seed compression [19]. These methods do not target the coverage of unmodeled faults. To enhance the effectiveness of LFSR reseeding for various defects and unmodeled faults, new techniques are needed for pattern modeling and seed selection.

This paper uses the "output deviation" [20] as a surrogate coverage metric for pattern grading in LFSR reseeding. A flexible, but general, probabilistic fault model is used to generate

Manuscript received May 16, 2008; revised August 16, 2008. Current version published January 21, 2009. This work was supported in part by the Semiconductor Research Corporation under Contract 1588. An earlier version of this paper was presented at the *Proceedings of IEEE European Test Symposium*, 2007, pp. 125–130. This paper was recommended by Associate Editor S. Hellebrand.

Z. Wang is with Cisco Systems, Inc., San Jose, CA 95134 USA.

H. Fang and K. Chakrabarty are with the Department Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: krish@ee.duke.edu).

M. Beinek is with the Advanced Micro Devices, Austin, TX 78735 USA.

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCAD.2008.2009166

a probability map for the circuit, which can subsequently be used for LFSR seed selection. Seeds for the LFSR are selected for a precomputed set of test cubes to ensure that the resulting test patterns provide high output deviations. We show that such patterns provide high coverage for different fault models.

We consider the large ISCAS-89 benchmark circuits, as well as several larger IWLS-05 benchmarks [21]. For the ISCAS-89 circuits, we provide n -detect stuck-at test cubes as inputs to the seed-generation procedure, since it has been shown that n -detection test sets, where each stuck-at fault is targeted by $n > 1$ different patterns, are effective for detecting unmodeled defects [22]. Test-data compression is particularly important for n -detection test sets, since these test sets contain more patterns and lead to higher test-data volume. For the IWLS benchmarks, we consider one-, three-, and five-detect stuck-at test cubes, as well as one-detect transition-fault test cubes, since the transition-fault model is popular in industry today.

The remainder of this paper is organized as follows. Section II provides an overview of the probabilistic fault model and the theory of output deviations. Section III presents the proposed seed-selection procedure. The design of experiments and experimental results are reported in Section IV. Section V concludes this paper and outlines directions for future work.

II. PROBABILISTIC FAULT MODEL AND OUTPUT DEVIATIONS

In this section, we review the general fault model used in this paper and the concept of output deviations. The *confidence level* (CL) of a single-output gate encompasses all the different input combinations of the gate, and for a given input combination, it provides the probability that the gate output is correct for the corresponding input combination. The probability that a gate output is correct can be different for the various input combinations [20].

The CL R_i of a gate G_i with m inputs and a single output is a vector with 2^m components, defined as $R_i = (r_i^{(00,\dots,00)}, r_i^{(00,\dots,01)}, r_i^{(00,\dots,10)}, \dots, r_i^{(11,\dots,11)})$, where each component of R_i denotes the probability that the gate output is correct for the corresponding input combination.

We associate signal probabilities $p_{i,0}$ and $p_{i,1}$ with each line i , where $p_{i,0}$ and $p_{i,1}$ are the probabilities for line i to be at logic 0 and 1, respectively. To reduce the amount of computation, signal correlations due to reconvergent fan-out are not considered.

Let i be the output of a two-input gate G , and let j and k be its input lines. If G is a NAND gate, we have

$$\begin{aligned} p_{i,0} &= p_{j,1}p_{k,1}r_i^{(11)} + p_{j,0}p_{k,0} \left(1 - r_i^{(00)}\right) \\ &\quad + p_{j,0}p_{k,1} \left(1 - r_i^{(01)}\right) + p_{j,1}p_{k,0} \left(1 - r_i^{(10)}\right) \\ p_{i,1} &= p_{j,0}p_{k,0}r_i^{(00)} + p_{j,0}p_{k,1}r_i^{(01)} \\ &\quad + p_{j,1}p_{k,0}r_i^{(10)} + p_{j,1}p_{k,1} \left(1 - r_i^{(11)}\right). \end{aligned}$$

Likewise, the signal probabilities can be easily computed for other gate types. Fig. 1 shows a circuit consisting of three gates

Fig. 1. Example to illustrate CLs.

TABLE I
SIGNAL PROBABILITIES FOR DIFFERENT INPUT COMBINATIONS

Input pattern, z	$p_{e,0}$	$p_{e,1}$	$p_{f,0}$	$p_{f,1}$	$p_{z,0}$	$p_{z,1}$
0000, 0	0.1	0.9	0.2	0.8	0.886	0.114
0101, 0	0.1	0.9	0.9	0.1	0.837	0.163
1111, 1	0.8	0.2	0.9	0.1	0.396	0.604

G_1 , G_2 , and G_3 with different CLs. For this circuit, Table I lists the signal probabilities for three different input vectors. The fault-free values at the output z are also listed in the first column of Table I.

For any logic gate [or primary output (PO)] g in a circuit, let its fault-free output value for any given input pattern t_j be d , where $d \in \{0, 1\}$. The *output deviation* $\Delta_{g,j}$ of g for input pattern t_j is defined as $p_{g,\bar{d}}$, where $\bar{d}$ is the complement of d . Intuitively, the deviation for an input pattern is a measure of the likelihood that the gate output is incorrect for that input pattern. The output deviations for the three patterns in Table I are highlighted. Output deviations can be determined without explicit fault grading; hence, the computation is feasible for large circuits and test sets.

We use the following arbitrarily chosen set of CLs for our experiments. It has been shown in prior work [20] that the pattern-grading results are relatively insensitive to small variations in the values of CL. These vectors are defined separately for each gate type. For example, for a two-input NAND gate, we use the following: 1) low CL: $R^{nand2} = (0.8^{(00)}, 0.8^{(01)}, 0.8^{(10)}, 0.7^{(11)})$ and 2) high CL: $R^{nand2} = (0.95^{(00)}, 0.95^{(01)}, 0.95^{(10)}, 0.85^{(11)})$. These CLs are chosen to reflect the fact that, when both inputs are noncontrolling, the probability for the gate to produce the correct output is lower than for other input combinations. Since both sets of CLs yield similar results, we only report results obtained using the “high CL” values.

The output deviation metric was used in [23] to select effective test patterns from a large repository n -detect test set. Test selection is important for time-constrained and wafer-sort environments that have strict budgets on test-data volume and test time. If highly effective test patterns are applied first, defective chips will fail earlier, reducing test-application time in an abort-at-first-fail environment. Experimental results in [23] show that, for the same test length, test patterns selected using output deviations are consistently more effective than patterns selected using other methods, in terms of the coverage for resistive shorts, wired-AND/OR bridging faults, and gate exhaustive metrics [24].

III. SEED SELECTION

Given the scan-chain architecture for the circuit under test (CUT), LFSR feedback polynomial, and phase shifter (if any),

Fig. 2. (a) LFSR and phase shifter. (b) State transition matrix of the LFSR.

Figure 3(a) shows a system of linear equations represented as a matrix equation:

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Figure 3(b) shows the Gauss-Jordan elimination process, where the matrix is partitioned into 'Pivots' and 'Free Variables'.

$$\left(\begin{array}{ccccc|c} \text{Pivots} & \text{Free Variables} & & & \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{array} \right)$$

Figure 3(c) shows the solution space as a linear combination of free variables:

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = y_4 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + y_5 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

Fig. 3. Example to illustrate the solution for a system of linear equations. (a) System of linear equations. (b) Gauss-Jordan elimination. (c) Solution space.

the operation of the LFSR and phase shifter can be symbolically simulated to determine a system of linear equations for each test cube. The resulting system of linear equations has the form $\mathbf{A}\vec{y} = \vec{z}$, where $\mathbf{A}$ is a matrix that can be derived from the LFSR feedback polynomial and the phase shifter, $\vec{z}$ is a column vector corresponding to the specified bits in the test cube, and the solution for the vector $\vec{y}$ is the seed.

Fig. 2 shows an external-XOR LFSR [1] with the feedback polynomial $x^5 + x^3 + x + 1$, and a one-stage phase shifter. The state of the LFSR can be represented using a vector $\vec{S} = (s_1, s_2, \dots, s_N)^t$, where N is the size of the LFSR and $s_1(s_N)$ corresponds to the leftmost (rightmost) stage. The j th state of the LFSR is derived recursively as $\vec{S}_j = \mathbf{H}\vec{S}_{j-1}$, with $j = 1, 2, \dots$, where $\mathbf{H}$ is the state transition matrix for the LFSR.

The j th output of the one-stage phase shifter shown in Fig. 2 can be represented as $O_j = \vec{P}^t \vec{S}_j = \vec{P}^t \mathbf{H}^j \vec{S}_0$, where $j = 1, 2, \dots$. Vector $\vec{P}$ represents the operation of the phase shifter. If stage j of the LFSR is connected to the XOR gate, the j th row in $\vec{P}$ is set to “1.” For the phase shifter in Fig. 2, we have $\vec{P} = (10100)^t$. For example, the second output of the phase shifter is $O_1 = \vec{P}^t \mathbf{H} \vec{S}_0 = (11101) \vec{S}_0 = y_1 + y_2 + y_3 + y_5$.

For the test cube 101xxxx (the leftmost bit “1” is loaded into the first scan cell that is next to the scan out pin), we can obtain a system of linear equations, as shown in Fig. 3(a). Gauss-Jordan elimination [18] can be used to transform a set of columns in $\mathbf{A}$ into an identity matrix (these columns are referred to as *pivots*), while the remaining columns are *free variables*, as shown in Fig. 3(b). The set of solutions for the pivots can be represented as a linear combination of the free variables, as shown in Fig. 3(c). To obtain multiple seeds for each test cube, we first make random assignments to free variables and then compute pivots.

Inputs: T, N_s

- 1: **for all** test cube t_p ,
- $p = 1, 2, \dots, |T|$ **do**
- 2: **for** $i = 1$ **to** N_s **do**
- 3: generate a new seed $s_{p,i}$;
- 4: expand seed $s_{p,i}$ into a test pattern $v_{p,i}$;
- 5: add $v_{p,i}, s_{p,i}$ to sets V, S ;
- 6: **end for**
- 7: **end for**
- 8: $V' = \text{Sort_Pattern}(V)$;
- 9: **while** $V' \neq \emptyset$ **do**
- 10: select the top-most pattern, namely $t_{p,k}$;
- 11: select seed $s_{p,k}$;
- 12: remove patterns $t_{p,i}$ from $V', i = 1, 2, \dots, N_s$;
- 13: **end while**

(a)

Inputs: T -test set
Output: T' -sorted test set

- 1: sort test patterns by deviations
- at each PO, obtain X ;
- 2: **while** $|T'| < |T|$ **do**
- 3: **for** $j = 1$ **to** M **do**
- 4: **for** $i = 1$ **to** $|T|$ **do**
- 5: **if** $X[i][j] \notin T'$ **then**
- 6: add $X[i][j]$ to T' ;
- 7: **break**;
- 8: **end if**
- 9: **end for**
- 10: **end while**
- 11: **end while**

(b)

Fig. 4. Procedures. (a) LFSR reseeding with seed selection, Procedure 1. (b) Sort test patterns according to output deviations, Procedure 2.

The high-level flow of LFSR seed selection is shown in Fig. 4(a). Given a set of test cubes T , we generate N_s seeds for each test cube (lines 2–6). N_s is a user-defined parameter that restricts the search space of seeds. Each seed is expanded into a fully specified test pattern (line 4). All seeds and their corresponding patterns are stored in two sets S and V , respectively. In line 8, test patterns in V are sorted, and a new sorted test set V' is obtained. To obtain a seed set, in lines 9–13, we sequentially select the topmost pattern in V' and its corresponding seed. Once a test pattern is selected (line 10), all other test patterns that are derived from the same test cube in line 4 are removed from V' (line 12). Procedure 1 in Fig. 4(a) is generic. A different sort method in procedure `Sort_Pattern` leads to a different seed set.

In this paper, we use output deviation as a metric for seed selection, i.e., test patterns in V are sorted according to their output deviations, such that test patterns with high deviations can be selected earlier than test patterns with low deviations. As shown in Fig. 4(b), for each PO, all test patterns in T are sorted in descending order based on their output deviations. The result is stored in a matrix X with M columns and $|T|$ rows, where M is the number of POs (line 1). The element in the i th row and the j th column of X is the test pattern that has the i th highest output deviation at the j th PO. Fig. 5 shows this sorting procedure. From matrix X , a new ordered test set T' can be obtained. For the example, in Fig. 5, the resulting sorted test set is $T' = \{t_1, t_3, t_5, t_2, t_4, t_6\}$.

To evaluate the effectiveness of a set of seeds for defect screening and for getting steeper fault-coverage ramp-up, we

Pattern ID	Output deviations			Port A	Port B	Port C
	Port A	Port B	Port C			
t_1	0.9	0.2	0.7	t_1	t_3	t_5
t_2	0.8	0.3	0.4	t_2	t_4	t_3
t_3	0.6	0.7	0.8	t_5	t_5	t_1
t_4	0.2	0.6	0.5	t_3	t_2	t_4
t_5	0.7	0.4	0.9	t_6	t_1	t_2
t_6	0.3	0.1	0.2	t_4	t_6	t_6

(a)

(b)

Fig. 5. Example for the sorting of test patterns based on output deviations. (a) Test patterns and output deviations. (b) Outcome of the sorting procedure (matrix X).

Fig. 6. Applying a test pattern pair using LOS and LOC schemes. (a) p_0 is the output of the LFSR. (b) LOS: p_2 is obtained by shifting p_1 one clock cycle. (c) LOC: p_2 is obtained by capturing the response (shown as “?”) for p_1 .

consider the fault coverage obtained using the test patterns that are derived from the seeds for different fault models, including stuck-at, transition, and stuck-open faults. To detect sequence-dependent stuck-open and transition faults, we use the launch-on-shift (LOS) and launch-on-capture (LOC) schemes to apply a test pattern pair (p_1, p_2) to the CUT. These methods are also referred to as skewed-load and broadside testings [1], respectively. Fig. 6 shows an example. Given a test cube from an n -detect stuck-at test set or a one-detect LOC transition-fault test set, a seed is computed, and a “high-deviation” test pattern p_0 is derived from the seed.

We first consider the case when p_0 is derived from a stuck-at test cube. In the LOS scheme, p_2 is obtained by shifting p_1 one clock cycle. Hence, as shown in Fig. 6(b), p_1 is obtained by shifting the first $L - 1$ bits of p_0 (L is the length of the scan chain), and p_2 is identical to p_0 . In the LOC scheme, as shown in Fig. 6(c), p_2 is obtained by capturing the responses of p_1 , and p_1 is identical to p_0 . For both the LOS and LOC schemes, output deviations are first computed directly using the test patterns in V (Fig. 4(a), line 8). The resulting seed set is referred to as S_{dev1} . For the LOC scheme, we also generate another seed set in the following manner: First, obtain the responses of the patterns in V , and then, compute output deviations using these responses. The resulting seed set is referred to as S_{dev2} . We also assume that the combinational primary inputs remain unchanged for p_1 and p_2 .

Next, we consider the case when p_0 is derived from a LOC transition-fault test cube. In this case, p_2 is obtained by capturing the responses of p_1 . We use p_2 to calculate deviations, and the resulting seed set is referred to as S_{dev3} .

Table II shows the various seed sets obtained using pattern pairs (p_1, p_2) . For the case of stuck-at test cubes, only S_{dev1}

TABLE II
VARIOUS SEED SETS OBTAINED USING PATTERN PAIRS (p_1, p_2)

Fault model for precomputed test cubes	Test application method	Pattern used to generate seeds	Seed set
Stuck-at	LOS	p_2	S_{dev1}
	LOC	p_1	S_{dev1}
		p_2	S_{dev2}
Transition	LOC	p_2	S_{dev3}

is generated in the LOS test-application method. For the LOC test-application method, both S_{dev1} and S_{dev2} are generated. We obtain S_{dev1} when p_1 is used to generate seeds and obtain S_{dev2} when p_2 is used. For the case of transition-fault test cubes, only S_{dev3} is obtained.

For comparison purposes, we also generate three other seed sets, referred to as S_{rand} , S_{greedy} , and S_{comp} , respectively. The seed set S_{rand} is obtained by randomly generating one seed for each test cube. The seed set S_{greedy} is generated by sorting test patterns in V by the number of hard stuck-at faults that they detect. Hard faults are the ones that are not detected by a predefined number (we use 128, 256, or 1024 for the different benchmarks in our experiments) of pseudorandom patterns. The detection count is obtained using fault simulation without fault dropping.

The seed set S_{comp} is used as a baseline to evaluate the impact on storage requirement for the deviation-based seed-selection method. In [19], a Huffman-encoding-based compression method is proposed. The LFSR is divided into multiple blocks, with each block having k stages (the last block may have less stages if the k cannot divide the LFSR size N). Consequently, each N -bit LFSR seed is divided into multiple k -bit segments. Each segment can be viewed as a k -bit symbol, and these symbols are compressed using Huffman encoding. To improve the encoding efficiency, when generating seeds for S_{comp} , the system of linear equations is solved to ensure that the most significant bits of the symbols are biased toward zero. Hence, seeds in S_{comp} have more “0” bits than seeds in other seed sets, which may result in lower defect coverage. To evaluate the impact of the seed-selection method on the effectiveness of seed compression, we used the method from [19] to compress S_{dev1} , S_{dev2} , S_{dev3} , and S_{comp} .

IV. EXPERIMENTAL RESULTS

To evaluate the effectiveness of our seed-selection procedures, experiments are carried out on the ISCAS-89 and the IWLS-05 benchmark circuits; the latter set contains larger circuits than the ISCAS-89 benchmarks. The experiments for the ISCAS-89 circuits are carried out using the n -detect stuck-at test cubes, while the experiments for the IWLS-05 benchmarks are carried out using both n -detect stuck-at and one-detect transition-fault test cubes.

A. Results for ISCAS-89 Circuits

1) *Experimental Setup*: All experiments for ISCAS-89 were performed on 64-b Linux servers with 4-GB memory each. The n -detect stuck-at test cubes were generated using proprietary tools from industry; we also used these tools to carry out fault

simulation for various fault models. The deviation calculation, seed selection, and other related programs were coded in C++.

To obtain the transition-fault coverage, we used LOC fault simulators. The LOC transition-fault simulator assumes the following two constraints: 1) Primary inputs remain unchanged during the two capture cycles and 2) POs are not probed to detect faults. The reason is that, for at-speed testing, low-speed testers usually cannot drive or probe pins at the functional clock frequency, which is much faster than the scan-test frequency.

2) *Results for Five-Detect Stuck-at Test Cubes:* For each ISCAS-89 circuit, we first obtain an n -detect set of test cubes as described in Fig. 1 of [25]. During the automatic test pattern generation (ATPG) process, after a new set of test cubes is generated, the ATPG tool randomly fills do-not-cares in these test cubes and performs fault simulation to drop faults that are fortuitously detected by these patterns and have been detected n times. Next, a procedure called *Xize* is invoked to identify as many bits as possible that can be relaxed to do-not-cares without any loss of n -detect fault coverage. Due to space limits, we only report results obtained with $n = 5$. Other results have been posted on the Web [26].

From the n -detect test cubes, the seed sets described in Section III are generated. For each circuit, the following three cases are considered corresponding to different fault models and test-application schemes:

- 1) stuck-at faults;
- 2) stuck-open faults detected using the LOS scheme (the industry tool reports fault coverage for stuck-open faults only for LOS);
- 3) transition faults detected using the LOC scheme. We do not consider LOS here because LOC is more practical and commonly used in industry.

The fault coverage for these fault models is obtained using patterns derived from the different seed sets.

Fig. 7 shows the fault coverage obtained for various test lengths for different fault models for two ISCAS-89 circuits. Results for other ISCAS circuits are available in [27].

For stuck-open and transition faults, the deviation-based approach yields the best results. The seeds selected using the proposed approach provide higher defect coverage with a smaller number of patterns. For stuck-at faults, the results for the various methods are comparable. This is expected because the precomputed test sets are tailored for stuck-at coverage.

It can be seen from cases 2 and 3 that, as the number of test patterns applied to the CUT increases, the difference in the fault coverage for the different seed sets decreases, i.e., the fault coverage tends to converge to a maximum level. This is because, by first applying test patterns derived from effective LFSR seeds, we can cover most easy-to-detect stuck-open and transition faults using the first few test patterns (40%–50% for most benchmark circuits). In case 3, the fault coverage for the complete set of patterns is relatively low, because the second test pattern p_2 is not directly generated by the n -detect ATPG tool. These observations suggest that, in order to obtain high defect coverage with a small number of test patterns, we should only apply a subset of the n -detect test patterns (with seeds determined using output deviations) and then add top-off ATPG

patterns targeting the remaining hard-to-detect faults. The high coverage provided by a small number of high-deviation patterns (for different fault models) leads to a reduction in the test-data volume that is required to predict high defect coverage.

Finally, we compare the storage requirements for S_{dev1} (S_{dev2}) and S_{comp} , as shown in Table III. The compression results for S_{dev2} and S_{dev1} are identical for all the benchmark circuits. In Table III, columns named “Vol.” denote the resulting seed volume in bits after compression, and columns named “Comp%” indicate the corresponding compression percentage relative to the set of seeds. As can be seen, the test-data volume for the proposed deviation-based seed-selection technique is only slightly higher than the test-data volume for [19] while achieving significantly higher defect coverage. It is also noteworthy that the seed volume reported in Table III only includes the seed data stored on the tester. For large industrial designs, the area overhead of the on-chip Huffman decoder may be prohibitively large, rendering the compression scheme of [19] infeasible.

3) *Selection of p_1 or p_2 in the Calculation of Output Deviations:* For the six ISCAS-89 circuits, we have separately evaluated the LOC transition-fault coverage for patterns derived from S_{dev1} and S_{dev2} . (Recall that S_{dev1} is the seed set selected using the deviations for p_1 , and S_{dev2} is the seed set selected using the deviations for p_2 in the LOC scheme.) The following discussion explains how an appropriate choice (p_1 or p_2) can be made based on the properties of the test set.

We carried out a set of experiments using one-detect stuck-at test cubes. Experiments were performed using two flows. In the first flow, test cubes were generated using a proprietary (i.e., in-house) tool from an IC design house. In the second flow, test cubes were generated using the commercially available TetraMAX tool from Synopsys. Table IV compares the results obtained using these two flows. The second and fifth columns ($x\%$) list the average percentage of do-not-care bits in a test cube. The third and sixth column (p_1/p_2) indicates which seed-generation method (based on p_1 or p_2) yields higher transition-fault coverage. The fourth column ($fc\%$) shows the transition-fault coverage provided by the patterns derived from S_{dev2} in the second flow.

From Table IV, we can see that two factors ($x\%$ and $fc\%$) determine whether p_1 or p_2 should be used to calculate deviations. Among these two factors, $x\%$ appears to be more important. We make the following key observations.

- 1) When $x\%$ is low (e.g., $x\% < 93\%$ for these six benchmarks), the seeds in S_{dev2} are better than those in S_{dev1} . In other words, p_2 should be used for calculating deviations.
- 2) When $x\%$ is high (e.g., $x\% > 99\%$ for these six benchmarks), the seeds in S_{dev1} are better than those in S_{dev2} . In other words, p_1 should be used for calculating deviations.
- 3) When $x\%$ is moderately high (e.g., $93\% < x\% < 99\%$ in Table IV) and $fc\%$ is low ($fc\% < 33\%$ in Table IV), it is more appropriate to compute deviations using p_1 .
- 4) When $x\%$ is moderately high (e.g., $93\% < x\% < 99\%$ in Table IV) and $fc\%$ is high ($fc\% > 60\%$ in Table IV), p_2 should be used for computing deviations.

Fig. 7. Fault coverage for s5378 and s9234 (five-detect stuck-at test cubes): (1) stuck-at faults; (2) stuck-open faults (LOS); and (3) transition faults (LOC). (a) s5378. (b) s9234.

TABLE III
COMPARISON OF COMPRESSION RESULTS

Circuit	Seed Size (bits)	Block Size	S_{dev1}, S_{dev2}		S_{comp}	
			Vol.	Comp%	Vol.	Comp%
s5378	158	32	37504	58.73	31391	65.45
s9234	158	32	58938	61.10	49011	67.65
s13207	247	64	62592	79.95	35272	88.70
s15850	274	64	44134	78.29	34917	82.82
s38417	661	64	96220	78.52	94994	78.80
s38584	520	64	94624	77.69	83293	80.37

TABLE IV
SELECTION OF p_1 OR p_2 IN THE CALCULATION OF DEVIATIONS

Circuit	Flow1 (Proprietary ATPG tool)		Flow2 (TetraMAX)		
	$x\%$	p_1 or p_2 ?	$fc\%$	$x\%$	p_1 or p_2 ?
s5378	72.56	p_2	75.67	96.07	p_2
s9234	74.60	p_2	32.64	95.38	p_1
s13207	92.87	p_2	60.96	98.91	p_2
s15850	86.93	p_2	26.57	98.81	p_1
s38417	82.28	p_2	84.53	99.33	p_1
s38584	86.55	p_2	16.70	99.50	p_1

Fig. 8. Transition-fault coverage (LOC, p_1) using three-detect stuck-at test cubes. (a) tv80. (b) mem_ctrl.

A low value of $x\%$ implies that seed selection based on p_1 has less degree of freedom, which adversely affects the quality of the resulting patterns. On the other hand, a large value of $x\%$ implies that seed selection using p_1 has more degree of freedom, which leads to high-quality seeds. Where $x\%$ is small (e.g., $x\% < 93\%$ in Table IV) or very high (e.g., $x\% > 99\%$ in Table IV), it plays a dominant role in determining whether p_1 or p_2 is more efficient for deviation calculation. The impact of $fc\%$ can be ignored in these cases. For example, in the first flow, the use of p_2 consistently leads to better results than the use of p_1 because the value of $x\%$ is rather low ($x\% < 93\%$). However, for the second flow, for $s38417$ and $s38584$, p_1 is preferred to p_2 since the value of $x\%$ is very high ($x\% > 99\%$) for both of them.

When the value of $x\%$ is neither very large or very small (e.g., between 93% and 99% in Table IV), $fc\%$ must also be considered. A relatively high value of $fc\%$ ($fc\% > 60\%$ here) implies that it is easy to activate the transition faults. When the transition faults can be easily activated, p_2 becomes more important since it is used to capture the fault effects; hence, p_2 should be used to calculate deviations. In the second flow, p_2 is better than p_1 for $s5378$ and $s13207$ since the following conditions are satisfied: The value of $x\%$ is between 93% and 99%, and the value of $fc\%$ is more than 60%. On the other hand, low $fc\%$ ($fc\% < 33\%$ in Table IV) indicates that it is difficult to activate the transition faults. In this case, p_1 becomes more important (e.g., as shown by the results for $s9234$ and $s15850$ in the second flow). The aforementioned

Fig. 9. Transition-fault coverage (LOC, p_1) using three-detect stuck-at test cubes. (a) ac97_ctrl. (b) DMA.

Fig. 10. Transition-fault coverage (LOC, p_1) using three-detect stuck-at test cubes: pci_bridge32.

qualitative discussion can serve as a guideline for deviation calculation.

B. Results for IWLS-05 Benchmarks

In order to evaluate the effectiveness of the proposed seed-selection method for larger circuits, we carried out experiments using the IWLS-05 benchmarks.

The organization of this section is as follows. We first briefly describe the IWLS-05 benchmarks in Section IV-B1 and the experimental setup in Section IV-B2. Figs. 8–13 in Section IV-B3 show the experimental results for n -detect ($n = 1, 3$) stuck-at test cubes. Sections IV-B4 and IV-B5 discuss the impacts of

Fig. 11. Transition-fault coverage (LOC, p_1) using one-detect stuck-at test cubes. (a) tv80. (b) mem_ctrl.

Fig. 12. Transition-fault coverage (LOC, p_1) using one-detect stuck-at test cubes. (a) ac97_ctrl. (b) DMA.

Fig. 13. Transition-fault coverage (LOC, p_1) using one-detect stuck-at test cubes. (a) pci_bridge32. (b) Ethernet.

varying LFSR size and number of random patterns on the effectiveness of the proposed method, respectively. Section IV-B6 presents the experimental results for one-detect LOC transition-fault test cubes. We consider transition-fault test cubes because it is a common practice in industry to first apply transition-fault patterns and then apply additional top-off stuck-at patterns during manufacturing testing.

1) *IWLS-05 Benchmarks*: We used eight designs from the recently released set of IWLS-05 benchmarks. Table V shows the characteristics of the selected benchmarks, including their sizes, and the number of stuck-at faults.

2) *Experimental Setup*: All experiments for the IWLS-05 benchmarks were performed on 64-b Linux servers with 4-GB memory each. The n -detect stuck-at and one-detect LOC

TABLE V
CHARACTERISTICS OF IWLS-05 BENCHMARKS

Benchmark name	No. of gates	No. of scan cells	No. of stuck-at faults
tv80	13978	404	73562
mem_ctrl	25380	1350	124686
ac97_ctrl	33004	2302	146136
DMA	49960	3131	230338
pci_bridge32	52163	3720	231050
wb_conmax	73783	3316	366720
des_perf	167592	9105	773570
ethernet	176075	10752	811084

TABLE VI
CHARACTERISTICS OF n -DETECT STUCK-AT TEST CUBES

Benchmark name	1-detect		3-detect	
	No. of test cubes	Fault coverage	No. of test cubes	Fault coverage
tv80	7008	97.36	24800	97.32
mem_ctrl	10649	79.24	32768	79.24
ac97_ctrl	20146	99.66	60268	99.66
DMA	26782	84.14	89408	84.12
pci_bridge32	31312	99.38	93982	99.38
wb_conmax	57681	95.12	173173	95.12
des_perf	76001	99.83	226532	99.83
ethernet	119636	99.85	358194	99.85

Fig. 14. Transition-fault coverage (LOC, p_1) using one-detect stuck-at test cubes with varying LFSR size for (a) tv80 and (b) ac97_ctrl.

Fig. 15. Transition-fault coverage (LOC, p_1) using one-detect stuck-at test cubes for ac97_ctrl with (a) 64 and (b) 256 random patterns.

transition test cubes were generated using the TetraMAX tool from Synopsys, which was also used to run fault simulations for different fault models. The Synopsys Verilog Compiler was used to run Verilog simulations and compute the output deviations for benchmarks. All other programs were implemented in C++.

As before, we obtained the transition-fault coverage under the LOC test-application method. The constraints used in the LOC scheme are the same as that for the ISCAS-89 circuits.

3) *Experiments Based on n -Detect Stuck-at Test Cubes:* For each benchmark, we first obtained the n -detect stuck-at test cubes using the TetraMAX tool from Synopsys. Table VI presents the details of the n -detect ($n = 1, 3$) test cubes that we generated.

If the number of stuck-at test cubes is more than 50 000 for a benchmark, we only use the first 50 000 test cubes. We do this because of the following two reasons.

- 1) For various seed-generation methods, there is only a slight difference in cumulative coverage after 50 000 patterns; thus, it is sufficient to select only the first 50 000 seeds to evaluate pattern quality.
- 2) For each benchmark, due to limited memory, several rounds are needed to complete the seed-selection procedure. In each round, we sort $K \cdot N_s$ (recall that N_s is the

Fig. 16. Transition-fault coverage (LOC, p_1) using one-detect stuck-at test cubes for pci_bridge32 with (a) 64 and (b) 256 random patterns.

number of seeds generated for each test cube) patterns and obtain K seeds that lead to K high-deviation patterns. In our experiments, K is set to be less than 10 000. Therefore, the 50 000 seeds selected for the 50 000 test cubes are nearly identical to the first 50 000 seeds obtained if the complete set of test cubes is used.

The seed sets described in Section III are next generated from the n -detect test cubes. For each benchmark, we report the cumulative fault coverage obtained for transition faults using the LOC method. The stuck-at fault coverage is nearly identical for all the seed sets; therefore, it is not shown here. The choice of either p_1 or p_2 for deviation calculation is based on the arguments presented in Section IV-A. The cumulative fault coverage for each case is obtained using the patterns derived from the different seed sets.

Figs. 8–13 show the fault coverage obtained for different fault models. Figs. 8–10 are based on the three-detect stuck-at test cubes, and Figs. 11–13 are based on the one-detect stuck-at test cubes. For the three largest benchmarks wb_conmax, des_perf, and ethernet, we first dropped the easy-to-detect faults by using either 256 or 10 000 pseudorandom patterns and then generated one-detect stuck-at test cubes for the remaining faults. The result for ethernet is shown in Fig. 13(b). The

TABLE VII
ONE-DETECT LOC TRANSITION-FAULT TEST CUBES

Benchmark name	No. of faults	No. of 1-detect transition-fault test cubes	Fault coverage
tv80	71942	18387	96.35
mem_ctrl	119274	25258	68.09
ac97_ctrl	136924	38127	99.61
DMA	217806	50593	80.30
pci_bridge32	216166	61962	99.38
wb_conmax	353452	116549	95.02
des_perf	737146	186100	99.85
ethernet	768072	251728	99.73

Fig. 17. Fault coverage for tv80 (one-detect transition test cubes). (a) Stuck-at faults. (b) Transition faults.

results for wb_conmax and des_perf are not shown since we obtained similar LOC transition-fault coverage for them by using different seed-generation methods. For LOC-based testing for transition faults, the deviation-based approach clearly outperforms the other methods for most circuits. Moreover, the seeds selected using the proposed approach provide higher coverage with a smaller number of patterns.

Another observation is that the first few test patterns can cover most easy-to-detect transition faults if these test patterns are derived from effective LFSR seeds, which implies that, in order to obtain high defect coverage with a small number of test patterns, we should only apply a subset of the n -detect test patterns (with seeds determined by output deviations) and then add top-off ATPG patterns targeting the remaining hard-to-detect faults. The high coverage provided by a small number of high-deviation patterns (for different fault models) reduces the test-data volume required to achieve high defect coverage.

Fig. 18. Fault coverage for mem_ctrl (one-detect transition test cubes). (a) Stuck-at faults. (b) Transition faults.

Fig. 19. Fault coverage for DMA (one-detect transition test cubes). (a) Stuck-at faults. (b) Transition faults.

4) *Impact of Varying LFSR Size:* We next study the impact of LFSR size on the effectiveness of the proposed method. Intuitively, increasing the LFSR size will provide more freedom for seed selection and therefore improve the fault coverage. We repeat the transition-fault-coverage experiments using one-detect stuck-at test cubes on tv80 and ac97_ctrl (refer to Figs. 11 and 12) with varying LFSR size, as shown in Fig. 14. It can be seen that there is no direct correlation between the LFSR size and the final fault coverage after seed selection. This is because we are using a static LFSR reseeding technique that requires that the LFSR must be large enough to encode the test pattern with the most care bits. If the majority of the test cubes consist of much fewer care bits, increasing the LFSR size will not be beneficial since only N_s seeds are generated per test cube. If a continuous-flow LFSR reseeding technique is utilized, however, increasing the LFSR size might yield higher fault coverage.

5) *Impact of Varying Number of Random Patterns:* In the aforementioned experiments, pseudorandom patterns are first applied to drop easy-to-detect faults. To study the impacts of using different numbers of random patterns on the effectiveness of the proposed method, we repeat the transition-fault-coverage experiments using one-detect stuck-at test cubes on ac97_ctrl and pci_bridge32 (refer to Figs. 12 and 13) with 64 and 256 random patterns, as shown in Figs. 15 and 16. It can be seen from these figures that the proposed technique consistently

outperforms other seed-selection techniques regardless of the number of random patterns. Moreover, when more random patterns are applied, the number of ATPG test cubes and the final fault coverage after seed selection both decrease. More ATPG patterns help to improve fault coverage and provide more freedom for seed selection.

6) *Experiments Based on One-Detect LOC Transition Test Cubes:* Since it is a common practice in industry to apply transition-fault test patterns using LOC, we also carried out experiments on one-detect LOC transition-fault test cubes. Table VII shows the fault coverage obtained using transition-fault test patterns.

As in the case of stuck-at test cubes, seeds are generated for the one-detect LOC transition-fault test cubes and subsequently expanded to test patterns. For the LOC transition-fault test cubes, both p_1 and p_2 are generated using TetraMAX. Since p_2 is directly generated by TetraMAX, we use p_2 to calculate the deviation for each pattern. Seeds are selected based on the obtained deviation data, with the selected seed set denoted by S_{dev3} (Table II). Test patterns are derived from the four seed sets obtained from different methods. For these pattern sets, we compare the cumulative LOC transition-fault coverage and stuck-at coverage.

Figs. 17–20 show the cumulative stuck-at and transition-fault coverage (LOC scheme) for the different seed sets. We only show the cumulative coverage of the first 5000 or 10000

Fig. 20. Fault coverage for pci_bridge32 (one-detect transition test cubes). (a) Stuck-at faults. (b) Transition faults.

patterns since the coverage difference is not obvious for the different seed sets after 5000 or 10 000 patterns. We can see that, for most of the benchmarks, S_{dev3} provides the best fault coverage among all the seed sets, for both stuck-at fault and transition-fault models. This demonstrates the effectiveness of the proposed seed-selection method for transition test cubes.

Finally, we note that, for the IWLS-05 benchmark circuits, the seeds obtained using [19] can be compressed more effectively than the seeds obtained using the proposed method. The seeds derived using [19] have very long runs of 0s. Nevertheless, the difference is not particularly significant because the length of the seeds for these circuits is already much smaller (43 times on average) than the test-pattern lengths.

V. CONCLUSION

We have proposed the use of output deviations as a surrogate coverage metric to select effective seeds for LFSR-reseeding-based test compression. The goal is to select seeds that are expanded to patterns with high output deviations. Experimental results for the ISCAS-89 and IWLS-05 benchmark circuits show that the selected seeds lead to high-deviation patterns, which provide higher defect coverage than patterns derived from other seeds. Moreover, faster fault-coverage ramp-up is obtained using these LFSR seeds.

Our experiments are based on both n -detect stuck-at test cubes and one-detect LOC transition test cubes. For the case of stuck-at test cubes, in order to evaluate defect coverage, we have determined the coverage for transition (stuck-open) faults provided by a set of n -detect stuck-at patterns. For the case of transition test cubes, in order to evaluate defect coverage, we have determined the coverage of stuck-at faults provided by a set of one-detect transition patterns. We have also shown that the patterns generated using the proposed method have negligible impact on test-data volume. We have also highlighted the factors that influence the choice of p_1 and p_2 for the calculation of deviations. This choice is related to the fraction of do-not-care bits and the transition-fault coverage provided by the test cubes.

REFERENCES

- [1] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing*. Norwell, MA: Kluwer, 2000.
- [2] B. Keller, M. Tegethoff, T. Bartenstein, and V. Chickermane, "An economic analysis and ROI model for nanometer test," in *Proc. Int. Test Conf.*, 2004, pp. 518–524.
- [3] F.-F. Ferhani and E. J. McCluskey, "Classifying bad chips and ordering test sets," in *Proc. Int. Test Conf.*, 2006, pp. 1–10.
- [4] B. Vermeulen, C. Hora, B. Kruseman, E. J. Marinissen, and R. van Rijnsing, "Trends in testing integrated circuits," in *Proc. Int. Test Conf.*, 2004, pp. 688–697.
- [5] H. Cox and J. Rajski, "On necessary and nonconflicting assignments in algorithmic test pattern generation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, no. 4, pp. 515–530, Apr. 1994.
- [6] K. Yang, K.-T. Cheng, and L.-C. Wang, "TranGen: A SAT-based ATPG for path-oriented transition faults," in *Proc. Asia South Pacific Des. Autom. Conf.*, 2004, pp. 92–97.
- [7] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors (ITRS)*, 2007. [Online]. Available: <http://www.itrs.net/Links/2007ITRS/Home2007.htm>
- [8] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 5, pp. 776–792, May 2004.
- [9] B. Koenemann, "LFSR-coded test patterns for scan design," in *Proc. Eur. Test Conf.*, 1991, pp. 237–242.
- [10] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers," *IEEE Trans. Comput.*, vol. 44, no. 2, pp. 223–233, Feb. 1995.
- [11] S. Hellebrand, H.-G. Liang, and H. J. Wunderlich, "A mixed mode BIST scheme based on reseeding of folding counters," in *Proc. Int. Test Conf.*, 2000, pp. 778–784.
- [12] C. V. Krishna, A. Jas, and N. A. Touba, "Achieving high encoding efficiency with partial dynamic LFSR reseeding," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 9, no. 4, pp. 500–516, Oct. 2004.
- [13] J. Rajski, J. Tyszer, and N. Zacharia, "Test data decompression for multiple scan designs with boundary scan," *IEEE Trans. Comput.*, vol. 47, no. 11, pp. 1188–1200, Nov. 1998.
- [14] E. Kalligeros, X. Kavousianos, and D. Nikolos, "Multiphase BIST: A new reseeding technique for high test-data compression," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 10, pp. 1429–1446, Oct. 2004.
- [15] H.-S. Kim and S. Kang, "Increasing encoding efficiency of LFSR reseeding-based test compression," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 5, pp. 913–917, May 2006.
- [16] A. A. Al-Yamani and E. J. McCluskey, "Seed encoding with LFSRs and cellular automata," in *Proc. Des. Autom. Conf.*, 2003, pp. 560–565.
- [17] A. A. Al-Yamani, S. Mitra, and E. J. McCluskey, "BIST reseeding with very few seeds," in *Proc. IEEE VLSI Test Symp.*, 2003, pp. 69–74.
- [18] C. G. Cullen, *Linear Algebra With Applications*. Reading, MA: Addison-Wesley, 1997.

- [19] C. V. Krishna and N. A. Touba, "Reducing test data volume using LFSR reseeding with seed compression," in *Proc. Int. Test Conf.*, 2002, pp. 321–330.
- [20] Z. Wang and K. Chakrabarty, "Test-quality/cost optimization using output-deviation-based reordering of test patterns," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 2, pp. 352–365, Feb. 2008.
- [21] [Online]. Available: <http://www.iwls.org/iwls2005/benchmarks.html>
- [22] I. Polian, I. Pomeranz, and B. Becker, "Exact computation of maximally dominating faults and its application to n-detection tests," in *Proc. Asian Test Symp.*, 2002, pp. 9–14.
- [23] Z. Wang and K. Chakrabarty, "An efficient test pattern selection method for improving defect coverage with reduced test data volume and test application time," in *Proc. Asian Test Symp.*, 2006, pp. 333–338.
- [24] K. Y. Cho, S. Mitra, and E. J. McCluskey, "Gate exhaustive testing," in *Proc. Int. Test Conf.*, 2005, pp. 771–777.
- [25] Y. Huang, "On N-detect pattern set optimization," in *Proc. Int. Symp. Quality Electron. Des.*, 2006, pp. 445–450.
- [26] [Online]. Available: <http://www.ee.duke.edu/~hf12/>
- [27] Z. Wang, K. Chakrabarty, and M. Bienek, "A seed-selection method to increase defect coverage for LFSR-reseeding-based test compression," in *Proc. Eur. Test Symp.*, 2007, pp. 125–130.
- [28] C. V. Krishna and N. A. Touba, "Adjustable width linear combinational scan vector decompression," in *Proc. Int. Conf. Comput.-Aided Des.*, 2003, pp. 863–866.
- [29] E. H. Volkerink and S. Mitra, "Efficient seed utilization for reseeding based compression," in *Proc. IEEE VLSI Test Symp.*, 2003, pp. 232–237.

Zhanglei Wang received the B.Eng. degree from Tsinghua University, Beijing, China, in 2001 and the M.S.E. and Ph.D. degrees in computer and electrical engineering from Duke University, Durham, NC, in 2004 and 2007, respectively.

He is currently a Hardware Engineer with Cisco Systems, Inc., San Jose, CA. His research interests include test compression, test pattern grading, test generation, high-speed test, and system-level test and diagnosis.

Hongxia Fang received the B.S. degree in mathematics from Nankai University, Tianjin, China, in 2002 and the M.S. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2005. She is currently working toward the Ph.D. degree in electrical and computer engineering in the Department Electrical and Computer Engineering, Duke University, Durham, NC.

Her current research is focused on VLSI testing, including test compression, test selection, and register-transfer-level testing.

Krishnendu Chakrabarty (S'92–M'96–SM'01–F'08) received the B.Tech. degree from the Indian Institute of Technology, Kharagpur, India, in 1990 and the M.S.E. and Ph.D. degrees in computer science and engineering from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively.

He is currently a Professor of electrical and computer engineering with the Department Electrical and Computer Engineering, Duke University, Durham, NC. He has contributed 15 invited chapters to book volumes, published 290 papers in archival journals, refereed conference proceedings, and delivered over 120 keynote, plenary, and invited talks. He has authored five books—*Microelectrofluidic Systems: Modeling and Simulation* (CRC Press, 2002), *Test Resource Partitioning for System-on-a-Chip* (Kluwer, 2002), *Scalable Infrastructure for Distributed Sensor Networks* (Springer, 2005), *Digital Microfluidics Biochips: Synthesis, Testing, and Reconfiguration Techniques* (CRC Press, 2006), and *Adaptive Cooling of Integrated Circuits using Digital Microfluidics* (Artech House, April 2007)—and edited the book volumes *SOC (System-on-a-Chip) Testing for Plug and Play Test Automation* (Kluwer, 2002) and *Design Automation Methods and Tools for Microfluidics-Based Biochips* (Springer, 2006). He is the Editor of *Journal of Electronic Testing: Theory and Applications (JETTA)*. He is the holder of a U.S. patent in built-in self-test and is the Coinventor of two pending U.S. patents on sensor networks and digital microfluidics, respectively. His current research projects include testing and design-for-testability of system-on-chip integrated circuits; digital microfluidic biochips; nanotechnology circuits and systems based on DNA self-assembly; and delay-tolerant wireless networks.

Prof. Chakrabarty is a recipient of the National Science Foundation Early Faculty (CAREER) award (in 1999) and the Office of Naval Research Young Investigator award (in 2001). He is a recipient of the best paper awards at the 2007 IEEE International Conference on Very Large Scale Integration (VLSI) Design, the 2005 IEEE International Conference on Computer Design, and the 2001 IEEE Design, Automation and Test in Europe Conference. He is also a recipient of the Humboldt Research Fellowship, awarded by the Alexander von Humboldt Foundation, Germany, in 2004 and the Mercator Visiting Professorship for 2000–2002, awarded by the Deutsche Forschungsgemeinschaft, Germany. He served as a Distinguished Visitor of the IEEE Computer Society for 2005–2007 and as a Distinguished Lecturer of the IEEE Circuits and Systems Society for 2006–2007. Since 2008, he has been serving as an Association for Computing Machinery (ACM) Distinguished Speaker. He is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, IEEE TRANSACTIONS ON VLSI SYSTEMS, IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS, and ACM *Journal on Emerging Technologies in Computing Systems* and the Editor of the IEEE Design and Test of Computers. He recently completed his term as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM—PART I (in 2006–2007). He is a Distinguished Engineer of ACM, and a member of Sigma Xi. He received a Meritorious Service Award from the IEEE Computer Society in 2008. He is a recipient of the Duke University's 2008 Dean's Award for Excellence in Mentoring.

Michael Bienek received the B.S.E.E. degree from Texas A&M University, College Station, in 1977.

He is with Advanced Micro Devices, Austin, TX, leading the Strategic Product Engineering Team, where he is responsible for defining the test roadmap for future products. He is a member of the Technical Advisory Board for the Computer-Aided Design and Test Sciences thrust of the Global Research Collaboration and serves as an Industrial Liaison to multiple projects. He also contributes to the International Technology Roadmap for Semiconductors roadmap and is holder of several patents related to design for testability.