Test-Quality/Cost Optimization Using Output-Deviation-Based Reordering of Test Patterns

Zhanglei Wang and Krishnendu Chakrabarty, *Senior Member, IEEE*

***Abstract*—At-speed functional testing, delay testing, and n-detection test sets are being used today to detect deep submicrometer defects. However, the resulting test data volumes are too high; the 2005 International Roadmap for Semiconductors predicts that test-application times will be 30 times larger in 2010 than they are today. In addition, many new types of defects cannot be accurately modeled using existing fault models. Therefore, there is a need to model the quality of test patterns such that they can be quickly assessed for defect screening. Test selection is required to choose the most effective pattern sequences from large test sets. Current industry practice for test selection is based on fault grading, which is computationally expensive and must also be repeated for every fault model. Moreover, although efficient methods exist today, for fault-oriented test generation, 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 simply taking all the patterns for each fault model. This paper presents the use of the output deviation as a surrogate coverage-metric for pattern modeling and test grading. A flexible, but general, probabilistic-fault model is used to generate a probability map for the circuit, which can subsequently be used for test-pattern reordering. The output deviations resulting from the probability map(s) are used as a coverage-metric to model test patterns; the higher the deviation, the better the quality of the test pattern. We show that, for the ISCAS benchmark circuits and as compared to other reordering methods, the proposed method provides "steeper" coverage curves for different fault models.**

***Index Terms*—Abort-on-first-fail, defect coverage, test-application time, test-pattern grading, test selection.**

I. Introduction

A NUMBER of fault models, such as stuck-at, transition delay, and shorts/opens, are typically used today to get high defect coverage [1], [2]. 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 2–5 times higher than that for stuck-at faults [3], and it has been demonstrated recently that test patterns for such sequence- and timing-dependent faults are more important for newer technologies [4]. 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 [5]. Although efficient methods exist today for fault-model-oriented test generation [6], [7], 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 2005 International Roadmap for Semiconductors (ITRS) document predicts that the test-data volume and test-application time for integrated circuits will be as much as 30 times larger in 2010 than they are today [8].

Test-pattern-reordering methods, which rank test patterns and place most effective test patterns at the top of the reordered test sequence, promise reductions in both testing time and test-data volume [9], [10]. Applying the most effective patterns first during volume ramp-up increases the likelihood of detecting manufacturing defects in less time. If highly effective test patterns are applied first in a reordered test set, defective chips will fail earlier, reducing test-application time in an abort-at-first-fail environment. Test-pattern reordering is also important for time-constrained and wafer-sort environments that have strict test data budgets: The reordered test set can be simply truncated to fit the test-data volume and test-time budgets. Moreover, the truncated test sets can be augmented with a small number of top-off patterns to reach mandated coverage levels for modeled faults.

This paper uses the "output deviation" [11] as a surrogate coverage-metric and a test-pattern-grading method for pattern reordering. A flexible, but general, probabilistic-fault model [12] is used to generate a probability map for the circuit, which can subsequently be used for pattern reordering. We show that such reordered test sets provide "steeper" coverage curves for different fault models as compared with other reordering methods.

It has been shown that n-detection test sets, where each stuck-at fault is targeted by $n > 1$ different patterns, are effective in detecting unmodeled defects [13]–[16]. As the number of unique detections for each fault increases, the defect coverage usually improves as compared with other test-generation methods. An advantage of this approach is that, even when n is very large, n-detection test sets can be generated using existing single stuck-at automatic test pattern generation (ATPG) tools with reasonable computation time. Therefore, in this paper, we choose a set of n-detect test patterns as a repository test set and apply the proposed pattern-reordering technique to it.

Manuscript received February 23, 2007; revised May 5, 2007 and June 25, 2007. This work was supported in part by the Semiconductor Research Corporation under Contract 1588. This paper was recommended by Associate Editor N. K. Jha.

Z. Wang is with Cisco Systems, Inc., San Jose, CA 95134 USA (e-mail: zhawang@cisco.com).

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

Digital Object Identifier 10.1109/TCAD.2007.907228

0278-0070/$25.00 © 2008 IEEE

The remainder of this paper is organized as follows. Section II discusses some related prior work. Section III introduces the probabilistic-fault model and the theory of output deviations. The design of experiments and experimental results are reported in Section VI. Section VII concludes this paper and outlines directions for future work.

II. Related Prior Work

In this section, we provide an overview of related work on defect screening. In [17], the MPG-D model was used to predict the "defective-part level" using information on the number of times that fault sites are observed and the probability of exciting undetected defects when those observations are made. The MPG-D model was shown to outperform the Williams–Brown model [18] that relies solely on measures of fault coverage. The DO-RE-ME test-generation technique was also used in improving defect coverage by maximizing the deterministic observation of defect sites in the circuit. Experimental results in [17] show that a DO-RE-ME test set achieves higher coverage for wired-AND/OR bridging faults and lower defect level for commercial production chips than a single stuck-at test set.

In [9], a test-pattern selection strategy based on integer linear programming (ILP) was proposed to select a subset of test patterns from a large n-detection test set. The weighted MPG-D model was used as the objective function for the ILP model to select patterns. However, an ILP-based method suffers from high computational complexity; it does not scale for large test sets. The experimental results reported in [9] also failed to demonstrate any significant improvement in fault coverage.

Adaptive test methods are discussed in [19]. These methods can increase test quality and decrease test cost based on predicted or measured defect and the parametric behavior of the silicon being tested. Predictive data can be obtained from other data sources or from a subset of patterns from the same test set. Such predictive data can also be used to select parameters for the probabilistic-fault model described in this paper, thereby resulting in more effective selection of test patterns.

In [20], an incremental ATPG flow was presented that targets different fault models in an n-detect manner. The initial fault list includes path-delay, transition, and stuck-at faults. First, LBIST is used to drop easy-to-detect faults. Since path-delay test patterns also detect many transition faults and stuck-at faults, n-detect path-delay fault ATPG is first performed for all the remaining faults not detected in the LBIST session. Similarly, transition-fault ATPG is then performed to detect transition and stuck-at faults not detected by the path-delay patterns. Finally, stuck-at fault ATPG is run, targeting all the remaining stuck-at faults. Using the incremental ATPG flow, up to 20% reduction in the total pattern count was reported in [20], as compared with the basic ATPG flow that simply combines all test patterns for the different fault models. Since the incremental ATPG flow first targets all delay faults and then targets remaining stuck-at faults, its test-generation time may be prohibitively long. Test generation for delay faults (particularly path-delay faults) is significantly more complex than that for stuck-at faults.

The proposed deviation-based test-pattern selection method can be combined with the ATPG flow in [20] for better test quality and reduced pattern count. For example, for each fault model, we can first generate a larger n'-detect set $(n' > n)$ from which high-deviation test patterns can be selected to form an n-detect set. Moreover, once the ATPG tool generates a test cube for a target fault, it usually randomly fills the don't-cares in it to get a test pattern. Therefore, we can obtain multiple test patterns from one test cube and select the patterns that have the highest output deviations.

Butler and Saxena [21] analyzed the impact of test ordering on the tester efficiency. They showed that it is useful to place a low-detection-latency test, such as a short burst of functional patterns, early in the tester program, even if the defect detection per pattern is lower than for some other test types, such as an I_{DDQ} test.

III. Probabilistic-Fault Model and Theory of Output Deviations

In this section, we develop the concept of output deviations in assessing test patterns.

A. Modeling of Gate Confidence Level (CL)

We first define the "CL" of a logic gate and show how it can be determined using transistor-level schematics.

The 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.

Definition 1: 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 follows: $R_i = (r_i^{(00,\ldots,00)} r_i^{(00,\ldots,01)} r_i^{(00,\ldots,10)}, \ldots, r_i^{(11,\ldots,11)})$, where each component of R_i denotes the probability that the gate output is correct for the corresponding input combination.

For example, $r_i^{(00)}$ is the probability that the output of a two-input G_i is correct under input 00. If $m = 2$ for a logic gate, we have $R_i = (r_i^{(00)} r_i^{(01)} r_i^{(10)} r_i^{(11)})$.

A more general definition of CL can be stated for a subcircuit (or "supergate") with m inputs and $k > 1$ outputs. The CL vector for this supergate is a set of k vectors, where each component of this set denotes the CL of the corresponding output. Alternatively, each component of R_i in Definition 1 can be viewed as a k-tuple corresponding to the k outputs.

The above gate-level CL vectors can be generated from simple transistor-level failure probabilities. Consider the two-input NAND and NOR gates shown in Fig. 1. Suppose each transistor can be stuck-open due to a defect, i.e., it cannot be switched on, with probability α. Similarly, suppose each transistor can be stuck-on due to a defect, i.e., it cannot be switched off, with probability β. Next, let us consider input combination $x_1x_2 = 00$. If only stuck-open faults are considered, the NAND gate produces the correct output for this combination with probability $1 - \alpha^2$, because the gate produces an incorrect output only if both p-transistors are stuck-open (the absence of a second pull-up path affects the pull-up time for this input combination, but this issue is ignored since worst-case pull-up

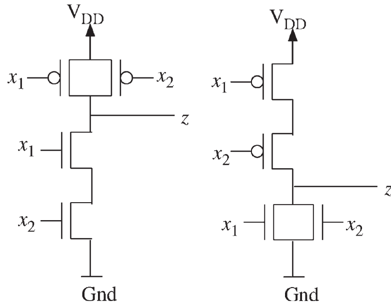


Fig. 1. Two-input NAND and NOR gates in static ratioless CMOS technology.

TABLE I
CLs OF THE NAND AND NOR GATES EXPRESSED IN TERMS OF TRANSISTOR STUCK-OPEN AND STUCK-ON PROBABILITIES

(a)

x_1x_2	$R_1^{NAND}(\tilde{x_1},\tilde{x_2},\alpha)$	$R_2^{NAND}(\tilde{x_1},\tilde{x_2},\beta)$
00	$1-\alpha^2$	$1-\beta^2$
01	$1-\alpha$	$1-\beta$
10	$1-\alpha$	$1-\beta$
11	$(1-\alpha)^2$	$(1-\beta)^2$

(b)

x_1x_2	$R_1^{NOR}(\tilde{x_1},\tilde{x_2},\alpha)$	$R_2^{NOR}(\tilde{x_1},\tilde{x_2},\beta)$
00	$(1-\alpha)^2$	$(1-\beta)^2$
01	$1-\alpha$	$1-\beta$
10	$1-\alpha$	$1-\beta$
11	$1-\alpha^2$	$1-\beta^2$

TABLE II
RELATIONSHIP BETWEEN STUCK-AT FAULTS AND THE CL VECTORS

Stuck-at fault in two-input NAND gate	Corresponding CL vector for the NAND gate
Input x_1 s-a-0	(1 1 1 0)
Input x_1 s-a-1	(1 0 1 1)
Input x_2 s-a-0	(1 1 1 0)
Input x_2 s-a-1	(1 1 0 1)
Output z s-a-0	(0 0 0 1)
Output z s-a-1	(1 1 1 0)

times consider only one pull-up path). Likewise, if we only consider stuck-on faults, the probability that the NAND gate produces the correct output for input 00 is $1-\beta^2$.

Table I presents the CL vectors of the NAND and NOR gates for stuck-open and stuck-on failure modes ($R_1^{\mathrm{NAND}}(\tilde{x_1},\tilde{x_2},\alpha)$ and $R_2^{\mathrm{NAND}}(\tilde{x_1},\tilde{x_2},\beta)$, respectively) for different input combinations. Note that $\tilde{x_1}$ and $\tilde{x_2}$ refer to input combinations, e.g., $R_1^{\mathrm{NAND}}(0,0,\alpha)$ refers to the probability that input 00 produces the correct output when the stuck-open probability for a transistor is α. If the stuck-open or stuck-on probabilities for the different transistors are unequal, e.g., due to different device dimensions, the actual values can be easily used to calculate the CLs in Table I (we assume that the gate output is in error if both pull-up and pull-down paths are active).

Note that classical faults, e.g., single stuck-at faults, can also be expressed in terms of CL vectors. Consider a two-input NAND gate with inputs ordered as x_1 and x_2 and output z. Stuck-at faults on x_1, x_2, and z can be modeled as shown in Table II.

Let $R(\tilde{x_1},\tilde{x_2},\alpha,\beta)$ be the gate reliability for input combination $(\tilde{x_1},\tilde{x_2})$ when any number of stuck-open and stuck-on faults can simultaneously occur. It can be easily shown that $R(\tilde{x_1},\tilde{x_2},\alpha,\beta) = R_1(\tilde{x_1},\tilde{x_2},\alpha)\cdot R_2(\tilde{x_1},\tilde{x_2},\beta)$. The result follows from the fact that, for any input combination, the set of transistors that affects the output under stuck-on conditions is disjoint from the set of transistors that affects the output under stuck-open conditions.

In this paper, we limit ourselves to basic gates with one or two inputs. Extension of the theory of output deviations to gates with larger fan-out is straightforward. In this paper, gates with larger fan-out are first expanded to a network of basic gates.

The gate-level CL vectors can also be generated in other ways, e.g., using layout information, inductive-fault analysis [22], and failure-data analysis. Our objective here is not to develop new techniques for determining CL vectors but rather to use these as inputs for the computation of output deviation. Therefore, in this paper, we use two arbitrarily chosen sets of CL vectors for our experiments. These vectors are defined separately for each gate type. For example, for a two-input NAND gate, we use as follows:

1) low CL: $R^{\mathrm{NAND2}} = (0.8^{(00)}, 0.8^{(01)}, 0.8^{(10)}, 0.7^{(11)})$;
2) high CL: $R^{\mathrm{NAND2}} = (0.95^{(00)}, 0.95^{(01)}, 0.95^{(10)}, 0.85^{(11)})$.

The above CL vectors 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 other input combinations. Since both sets of CL vectors yield similar results for test-pattern ordering, we only report results obtained using the "high CL" set.

B. Computation of Signal Probabilities

Next, we associate signal probabilities $p_{i,0}$ and $p_{i,1}$ with each line i in the circuit, where $p_{i,0}$ and $p_{i,1}$ are the probabilities for line i to be at logic 0 and 1, respectively. Obviously, we have $p_{i,0}+p_{i,1}=1$. The calculation of the signal probabilities is along the same lines as introduced in [23] and used later in [24]. To reduce the amount of computation as in [23] and [24], signal correlations due to reconvergent fan-out are not considered here.

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

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

The above definition of the signal probabilities can be easily extended to the case of more than two inputs. It can also be easily verified that $p_{i,0}+p_{i,1} = p_{j,0}p_{k,0}+p_{j,0}p_{k,1}+p_{j,1}p_{k,0}+p_{j,1}p_{k,1}=1$.

Let G be a gate with two inputs j and k, controlling value c, and inversion value v. Let $\overline{c}$ be the complement of the controlling value c. The signal probabilities for the output i of

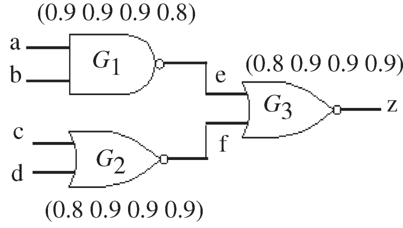


Fig. 2. Illustrative example.

TABLE III
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

such a gate can be easily expressed as follows:

$$p_{i,c\oplus v} = p_{j,c}p_{k,c}r_i^{(cc)} + p_{j,c}p_{k,\bar{c}}r_i^{(c\bar{c})} + p_{j,\bar{c}}p_{k,c}r_i^{(\bar{c}c)} + p_{j,\bar{c}}p_{k,\bar{c}}\left(1 - r_i^{(\bar{c}\bar{c})}\right)$$

$$p_{i,\bar{c}\oplus v} = p_{j,c}p_{k,c}\left(1 - r_i^{(cc)}\right) + p_{j,c}p_{k,\bar{c}}\left(1 - r_i^{(c\bar{c})}\right) + p_{j,\bar{c}}p_{k,c}\left(1 - r_i^{(\bar{c}c)}\right) + p_{j,\bar{c}}p_{k,\bar{c}}r_i^{(\bar{c}\bar{c})}.$$

Next, consider a gate without a controlling value, e.g., XOR and XNOR. For the XOR gate, the signal probabilities can be expressed as follows (the formulas for the XNOR gate are similar):

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

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

Fig. 2 shows a simple circuit consisting of three gates G_1, G_2, and G_3 with CL vectors (0.9 0.9 0.9 0.8), (0.8 0.9 0.9 0.9), and (0.8 0.9 0.9 0.9), respectively. For the three different deterministic input vectors 0000, 0101, and 1111, the signal probabilities are determined and presented in Table III. The fault-free values at the output z are also listed in the first column of Table III.

C. Output Deviations, Fault Model, and Deviation-Based Test Patterns

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, $d \in \{0,1\}$. 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 III are highlighted. Output deviations can be determined without explicit fault grading; hence, the computation is feasible for large circuits and large test sets.

Next, we formally define the probabilistic-fault model for a combinational circuit $\mathcal{C}$.

Definition 2: A combinational circuit $\mathcal{C}$ is defined as $\mathcal{C} = \{\mathcal{G}, \mathcal{PI}, \mathcal{Z}, \mathcal{R}\}$, where $\mathcal{G} = \{G_1, G_2, \ldots, G_N\}$ is the set of logic gates in $\mathcal{C}$, $\mathcal{PI}$ and $\mathcal{Z}$ are the sets of primary inputs and outputs, respectively, and $\mathcal{R} = \{R_1, R_2, \ldots, R_N\}$ is the set of CL vectors of the gates in $\mathcal{G}$.

Definition 3: A probabilistic-fault model $\mathcal{F}$ for circuit $\mathcal{C}$ is defined as follows: 1) Each gate G_i can fail independently of other gates and 2) the fault behavior of $\mathcal{C}$ is defined by R_i.

Under this fault model, the expected output values of the circuit in response to an input pattern is no longer deterministic. Rather, it is given by the signal probabilities at POs. Note that the circuit behavior is assumed to be deterministic after manufacturing; the probabilistic-fault model is only used during test development for pattern grading.

Consider the simple circuit shown in Fig. 2. According to $\mathcal{F}$, this circuit can fail in a number of ways, each of which is termed a fault event. Table IV lists the various fault events $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_7$ and the event $\mathcal{E}_0$, corresponding to the fault-free case. It also lists the probability associated with each fault event and the corresponding circuit output, for input pattern $abcd = 0000$. Only the events $\mathcal{E}_1$, $\mathcal{E}_4$, $\mathcal{E}_5$, and $\mathcal{E}_6$ are detected by the given input pattern. Let $\mathcal{E}$ be the event that the pattern 0000 detects a fault in the circuit. It can be easily seen that

TABLE IV
FAULT EVENTS FOR THE CIRCUIT OF FIG. 2 UNDER INPUT PATTERN $abcd = 0000$

Fault event	Fault event description	Event probability	Output value
$\mathcal{E}_0$	G_1, G_2, G_3 fault-free	$.9 \times .8 \times .9 = .648$	0
$\mathcal{E}_1$	G_1, G_2 fault-free, G_3 faulty	$.9 \times .8 \times .1 = .072$	1
$\mathcal{E}_2$	G_1, G_3 fault-free, G_2 faulty	$.9 \times .2 \times .9 = .162$	0
$\mathcal{E}_3$	G_2, G_3 fault-free, G_1 faulty	$.1 \times .8 \times .9 = .072$	0
$\mathcal{E}_4$	G_1, G_2 faulty, G_3 fault-free	$.1 \times .2 \times .8 = .016$	1
$\mathcal{E}_5$	G_1, G_3 faulty, G_2 fault-free	$.1 \times .8 \times .1 = .008$	1
$\mathcal{E}_6$	G_2, G_3 faulty, G_1 fault-free	$.9 \times .2 \times .1 = .018$	1
$\mathcal{E}_7$	G_2, G_3, G_1 faulty	$.1 \times .2 \times .2 = .004$	0

$$P[\mathcal{E}] = P[\mathcal{E}_1 \bigcup \mathcal{E}_4 \bigcup \mathcal{E}_5 \bigcup \mathcal{E}_6] = P[\mathcal{E}_1] + P[\mathcal{E}_4] + P[\mathcal{E}_5] + P[\mathcal{E}_6] = 0.114$$

since the fault events are mutually exclusive. Note from Table III that Δ_z for input pattern 0000 is also 0.114.

The above example shows that the probability that t_j will produce an observable error at z_i for fault model $\mathcal{F}$ is directly proportional to $\Delta_{z_i,j}$. The goal of testing is to apply those vectors to $\mathcal{C}$ that produce large deviations at the output. Therefore, the concept of output deviations offers a promising method for modeling pattern quality. From now on, we only consider output deviations at outputs and use the terms "output deviation" and "deviation" interchangeably.

Note that the deviations at circuit outputs reflect the observability of errors at these nodes. The more the deviation,

```
1: Perform single-detect fault simulation with single-detect pattern set
   T_1 for all faults;
2: Save all faults detected by single-detect fault simulation with pattern
   set T_1 in list F;
3: Set the number of detections n;
4: for K = 1 to (n-1) do
5:     Perform multiple-detect fault simulation with pattern sets T_1 to
       T_K for faults in list F;
6:     Save faults detected K times in F_K;
7:     Target faults in F_K and perform single-detect ATPG to increase
       the number of detections by one;
8:     Save the patterns to T_{K+1};
9: end for
```

Fig. 3. n-detect ATPG algorithm.

```
Require: T: test set
Ensure: T': reordered test sequence
 1: reorder test patterns by deviations at each PO and PPO, 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] ∉ T' then
 6:         add X[i][j] to T';
 7:         break;
 8:       end if
 9:     end for
10:   end for
11: end while
```

Fig. 4. Reorder test patterns according to output deviations.

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

(a)

Port A	Port B	Port C
t_1	t_3	t_5
t_2	t_4	t_3
t_5	t_5	t_1
t_3	t_2	t_4
t_6	t_1	t_2
t_4	t_6	t_6

(b)

Fig. 5. Example of reordering test patterns based on output deviations.

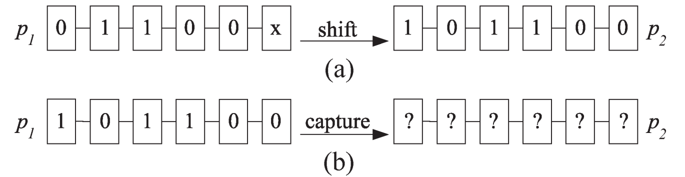


Fig. 6. Applying a test-pattern pair using LOS and LOC schemes.

the higher is the likelihood that an error is observed at the corresponding output.

D. Recent Work on Applications of Output Deviations

The concept of output deviations was first introduced in [12]. To show that output deviations can be used as a simple criterion to rank test patterns in terms of their effectiveness, preliminary experiments were conducted in which a set of random test patterns are split into two sets, referred to as $T_{\rm high}$ and $T_{\rm low}$, respectively, according to their output deviations. For any given pattern, if there exists one output pin whose deviation is larger than a predefined threshold, this pattern is put into $T_{\rm high}$ ($T_{\rm low}$, otherwise). Experimental results for the International Symposium on Circuits And Systems (ISCAS) benchmark circuits show that $T_{\rm high}$ leads to higher fault coverage for single stuck-at and bridging faults.

The output deviation metric was used in [11] to select effective test patterns from a large repository n-detect test set. If highly effective test patterns are applied first in a reordered test set, defective chips will fail earlier, reducing test-application time in an abort-at-first-fail environment. Experimental results in [11] 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 fault coverage for resistive shorts, wired-AND and wired-OR bridging faults, and several gate-exhaustive metrics [25].

More recently, it has been shown that output deviations can be used to select seeds for test compression based on linear-feedback shift-register (LFSR) reseeding [26]. Compared to seeds selected using other methods, these seeds provide higher coverage for a variety of fault models.

IV. Test-Pattern Reordering

In this section, we reorder the test patterns in a large n-detect test set, referred to as $T_{\rm orig}$, such that the most effective patterns appear in the front of the reordered test sequence. In this section, we use the term "test sequence" to denote a reordered test set.

The n-detect test set $T_{\rm orig}$ is generated using the procedure described in Fig. 3 (derived from [27, Fig. 1]), which uses n iterations to generate an n-detect test set. In each iteration i, first, a new set of test cubes is generated for the faults that are not detected i times, $i = 1, 2, \ldots, n$. Next, the ATPG tool performs fault simulation to update the detection count for each fault. Faults that have been detected n times are dropped. After n iterations, an n-detect set of test patterns is obtained. Once the n-detect test cubes are generated, they are randomly filled to obtain a set of n-detect test patterns.

We propose to use output deviation as a metric to reorder test patterns, such that test patterns with high deviations can be selected earlier than test patterns with low deviations. As shown in Fig. 4, for each PO and pseudo-PO, all test patterns in T are reordered 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 ith row and the jth column of X is the test pattern that has the ith highest output deviation at the jth PO. Fig. 5 shows this reordering procedure. From matrix X, a new ordered test set T' can be obtained. For the example in Fig. 5, the resulting reordered test sequence is $T' = \{t_1, t_3, t_5, t_2, t_4, t_6\}$.

To evaluate the effectiveness of the reordered test sequence for defect screening, we consider its fault coverage for different fault models, including stuck-at, stuck-open, and transition faults. To detect sequence-dependent transition and stuck-open 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 circuit under test (CUT). These methods are also referred to as skewed-load and broadside-testing [1], respectively.

Fig. 6 provides an example. Let the test pattern from the reordered test sequence be p_0. In the LOS scheme, p_2 is obtained

by shifting p_1 by one clock cycle. Since errors are captured by p_2, we should apply p_0 as the second test pattern for improved fault coverage and use p_2 to compute output deviations. Hence, as shown in Fig. 6(a), 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(b), p_1 is identical to p_0, and p_2 is obtained by capturing the responses of p_1.

For the LOS scheme, output deviations are computed directly using p_2. The resulting reordered test sequence is referred to as S_{dev1}. For the LOC scheme, we first obtain the responses of the test patterns in T_{orig} and, then, compute output deviations using these responses. We also assume that the combinational primary inputs remain unchanged for p_1 and p_2. The resulting reordered test sequence is referred to as S_{dev2}.

Besides S_{dev1} and S_{dev2}, for comparison purposes, we also generate three other seed sets, referred to as S_{rand}, S_{greedy}, and S_{inc}. The set S_{rand} is obtained by randomly ordering T_{orig}. S_{greedy} is generated by reordering test patterns in T_{orig} by the number of hard stuck-at faults that they detect. Hard faults are ones that are not detected by a certain number (ranges from 128 to 256 for the different benchmarks in this paper) of pseudorandom test patterns. The detection count is obtained using fault simulation without fault dropping. The set S_{inc} is generated as described in [10], whereby patterns are reordered to provide the steepest curve for stuck-at faults.

V. Analysis of Signal Correlation

The computation of signal probabilities in this paper does not account for signal correlations due to reconvergent fanouts, i.e., all signals are assumed to be mutually independent. This assumption is motivated by practical considerations; however, it can introduce errors in the values determined for the signal probabilities and output deviations. Accurate signal probabilities can be computed using probabilistic-transfer matrices (PTMs) and tensor products [28]. However, the computation method in [28] is expensive, since all correlated signals must be tensored. In this section, we use PTMs to compute accurate output deviations for some small circuits, and we show that the error is negligible if we ignore signal correlations.

In the PTM framework, the probabilistic behavior of a gate is described by a matrix $\boldsymbol{M}$, referred to as PTM, where the (j,k)th entry represents the probability that output signals $O = o_0, o_1, \ldots, o_n$ have value k, given that input signals $I = i_0, i_1, \ldots, i_m$ have value j. This is denoted as $p(k|j)$. Here, the row and column indexes j and k are bit vectors, whose entries represent the values of the signals that form the input and output. For instance, $p(1,1|1,0)$ represents the probability that the two output variables $\{o_\phi, o_1\}$ have value $\{1,1\}$ given that the two input variables $\{i_0, i_1\}$ have value $\{1,0\}$.

A fault-free circuit has an ideal transfer matrix, i.e., the correct value of the output occurs with probability of one. Similarly, an input vector v is a row vector representing the joint probability distribution of the input signals. The ith entry of v, denoted by $v(i)$, gives the probability that the input signals have values represented by the bit vector i. The output probability distribution, after input vector v is evaluated on gate g with PTM $\boldsymbol{P_g}$, is given by $v \times \boldsymbol{P_g}$.

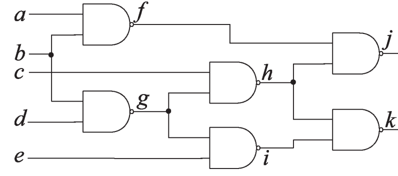


Fig. 7. Small ISCAS-85 circuit c17.

By definition, the PTM for a single gate is equivalent to its CL vector. For example, the PTM for a two-input NAND gate with the CL vector $R = (r^{(00)}, r^{(01)}, r^{(10)}, r^{(11)})$ can be written as

$$\boldsymbol{M} = \begin{bmatrix} 1 - r^{(00)} & r^{(00)} \\ 1 - r^{(01)} & r^{(01)} \\ 1 - r^{(10)} & r^{(10)} \\ r^{(11)} & 1 - r^{(11)} \end{bmatrix}.$$

If the input pins for this gate (j and k) are independent, then the input vector is $v = (p_{j,0}p_{k,0}, p_{j,0}p_{k,1}, p_{j,1}p_{k,0}, p_{j,1}p_{k,1})$. The output vector for this NAND gate can be easily obtained and is identical to the signal probabilities as described in Section III-B. Hence, for a single gate, if the input signals are mutually independent, the computation of signal probabilities is equivalent to PTM multiplication.

We can therefore derive output deviations from the output vector of a circuit. Since the PTM framework accounts for signal correlation using tensor products and the output vector denotes the joint-output probability distribution, the output deviation thus obtained is more accurate; we refer to it as OD_{PTM}. Output deviations defined in Section III-C are less accurate, and they are referred to as OD_{prob} in this section.

If we denote the CPU times for OD_{PTM} and OD_{prob} computation as t_{PTM} and t_{prob}, respectively, then $t_{\text{PTM}} = t_{\text{prob}} + t_{\text{corr}}$, where t_{corr} is the CPU time needed to handle correlated signals. As shown in [28, Fig. 4], because matrix multiplications, tensor products, and gate-input/output permutations must be done for correlated signals, t_{corr} grows quadratically with the number of correlated signals.

For the small ISCAS-85 circuit c17 in Fig. 7, Table V lists the computation of output vectors and OD_{prob} using the high- and low-CL vectors. We assume that all six NAND gates have identical CL vectors (and, hence, PTMs). Column "Error" reports the error defined as $\text{OD}_{\text{prob}} - \text{OD}_{\text{PTM}}$. Take the first row of Table V(a) as an example. Under the input pattern $\{abdce\} = \{00011\}$, the correct output response is $\{jk\} = \{11\}$; the joint probabilities for the output being 00, 01, 10, and 11 is 0.0528, 0.1416, 0.0504, and 0.7552, respectively; the OD_{PTM} at port k is $0.0528 + 0.0504 = 0.1032$; the error at k is $0.0789 - 0.1032 = -0.0243$.

In Table V(a) and (b), the error is noticeable only for eight out of the 32 input combinations. Even in these eight cases, the percentage error is quite low. Since the output deviations are being used here as a metric for classifying test patterns,

TABLE V
DIFFERENCE BETWEEN OD_{PTM} AND OD_{prob} FOR c17. (a) USING THE HIGH CL. (b) USING THE LOW CL

(a)

Input Pattern		Output	Output Vector (using PTM)				OD_{PTM}		OD_{prob}		Error		Error%	
ID	**{*abdce*}**	**{*jk*}**	**00**	**01**	**10**	**11**	**Port *j***	**Port *k***	**Port *j***	**Port *k***	**Port *j***	**Port *k***	**Port *j***	**Port *k***
1	00000	00	0.6234	0.1486	0.1486	0.0794	0.2280	0.2280	0.2280	0.2280	0.0000	0.0000	0.0000	0.0000
2	00001	01	0.1556	0.6164	0.0388	0.1892	0.2280	0.1944	0.2280	0.1944	0.0000	0.0000	0.0000	0.0000
3	00010	11	0.1267	0.0677	0.0677	0.7379	0.1944	0.1944	0.1944	0.1944	0.0000	0.0000	0.0000	0.0000
4	00011	11	0.0528	0.1416	0.0504	0.7552	0.1944	0.1032	0.1944	0.0789	0.0000	-0.0243	0.00%	-23.57%
5	00100	00	0.6234	0.1486	0.1486	0.0794	0.2280	0.2280	0.2280	0.2280	0.0000	0.0000	0.00%	0.00%
6	00101	01	0.1556	0.6164	0.0388	0.1892	0.2280	0.1944	0.2280	0.1944	0.0000	0.0000	0.00%	0.00%
7	00110	11	0.1267	0.0677	0.0677	0.7379	0.1944	0.1944	0.1944	0.1944	0.0000	0.0000	0.00%	0.00%
8	00111	11	0.0528	0.1416	0.0504	0.7552	0.1944	0.1032	0.1944	0.0789	0.0000	-0.0243	0.00%	-23.57%
9	01000	00	0.6234	0.1486	0.1486	0.0794	0.2280	0.2280	0.2280	0.2280	0.0000	0.0000	0.00%	0.00%
10	01001	01	0.1556	0.6164	0.0388	0.1892	0.2280	0.1944	0.2280	0.1944	0.0000	0.0000	0.00%	0.00%
11	01010	11	0.1267	0.0677	0.0677	0.7379	0.1944	0.1944	0.1944	0.1944	0.0000	0.0000	0.00%	0.00%
12	01011	11	0.0528	0.1416	0.0504	0.7552	0.1944	0.1032	0.1944	0.0789	0.0000	-0.0243	0.00%	-23.57%
13	01100	00	0.6234	0.1486	0.1486	0.0794	0.2280	0.2280	0.2280	0.2280	0.0000	0.0000	0.00%	0.00%
14	01101	00	0.5495	0.2225	0.1313	0.0967	0.2280	0.3192	0.2280	0.3192	0.0000	0.0000	0.00%	0.00%
15	01110	00	0.5450	0.1358	0.1358	0.1834	0.3192	0.3192	0.3192	0.3192	0.0000	0.0000	0.00%	0.00%
16	01111	00	0.5333	0.1475	0.1331	0.1861	0.3192	0.3336	0.3192	0.3989	0.0000	0.0653	0.00%	19.57%
17	10000	00	0.6234	0.1486	0.1486	0.0794	0.2280	0.2280	0.2280	0.2280	0.0000	0.0000	0.00%	0.00%
18	10001	01	0.1556	0.6164	0.0388	0.1892	0.2280	0.1944	0.2280	0.1944	0.0000	0.0000	0.00%	0.00%
19	10010	11	0.1267	0.0677	0.0677	0.7379	0.1944	0.1944	0.1944	0.1944	0.0000	0.0000	0.00%	0.00%
20	10011	11	0.0528	0.1416	0.0504	0.7552	0.1944	0.1032	0.1944	0.0789	0.0000	-0.0243	0.00%	-23.57%
21	10100	00	0.6234	0.1486	0.1486	0.0794	0.2280	0.2280	0.2280	0.2280	0.0000	0.0000	0.00%	0.00%
22	10101	01	0.1556	0.6164	0.0388	0.1892	0.2280	0.1944	0.2280	0.1944	0.0000	0.0000	0.00%	0.00%
23	10110	11	0.1267	0.0677	0.0677	0.7379	0.1944	0.1944	0.1944	0.1944	0.0000	0.0000	0.00%	0.00%
24	10111	11	0.0528	0.1416	0.0504	0.7552	0.1944	0.1032	0.1944	0.0789	0.0000	-0.0243	0.00%	-23.57%
25	11000	10	0.1309	0.0331	0.6411	0.1949	0.1640	0.2280	0.1640	0.2280	0.0000	0.0000	0.00%	0.00%
26	11001	11	0.0327	0.1313	0.1617	0.6743	0.1640	0.1944	0.1640	0.1944	0.0000	0.0000	0.00%	0.00%
27	11010	11	0.0282	0.0446	0.1662	0.7610	0.0728	0.1944	0.0728	0.1944	0.0000	0.0000	0.00%	0.00%
28	11011	11	0.0127	0.0601	0.0905	0.8367	0.0728	0.1032	0.0728	0.0789	0.0000	-0.0243	0.00%	-23.57%
29	11100	10	0.1309	0.0331	0.6411	0.1949	0.1640	0.2280	0.1640	0.2280	0.0000	0.0000	0.00%	0.00%
30	11101	10	0.1154	0.0486	0.5654	0.2706	0.1640	0.3192	0.1640	0.3192	0.0000	0.0000	0.00%	0.00%
31	11110	10	0.1147	0.0349	0.5661	0.2843	0.1496	0.3192	0.1496	0.3192	0.0000	0.0000	0.00%	0.00%
32	11111	10	0.1123	0.0373	0.5541	0.2963	0.1496	0.3336	0.1496	0.3989	0.0000	0.0653	0.00%	19.57%

(b)

Input Pattern		Output	Output Vector (using PTM)				OD_{PTM}		OD_{prob}		Error		Error%	
ID	**{*abdce*}**	**{*jk*}**	**00**	**01**	**10**	**11**	**Port *j***	**Port *k***	**Port *j***	**Port *k***	**Port *j***	**Port *k***	**Port *j***	**Port *k***
1	00000	00	0.2960	0.2240	0.2240	0.2560	0.4800	0.4800	0.4800	0.4800	0.0000	0.0000	0.0000	0.0000
2	00001	01	0.2000	0.3200	0.1600	0.3200	0.4800	0.3600	0.4800	0.3600	0.0000	0.0000	0.0000	0.0000
3	00010	11	0.1680	0.1920	0.1920	0.4480	0.3600	0.3600	0.3600	0.3600	0.0000	0.0000	0.0000	0.0000
4	00011	11	0.1320	0.2280	0.1680	0.4720	0.3600	0.3000	0.3600	0.2800	0.0000	-0.0200	0.00%	-6.67%
5	00100	00	0.2960	0.2240	0.2240	0.2560	0.4800	0.4800	0.4800	0.4800	0.0000	0.0000	0.00%	0.00%
6	00101	01	0.2000	0.3200	0.1600	0.3200	0.4800	0.3600	0.4800	0.3600	0.0000	0.0000	0.00%	0.00%
7	00110	11	0.1680	0.1920	0.1920	0.4480	0.3600	0.3600	0.3600	0.3600	0.0000	0.0000	0.00%	0.00%
8	00111	11	0.1320	0.2280	0.1680	0.4720	0.3600	0.3000	0.3600	0.2800	0.0000	-0.0200	0.00%	-6.67%
9	01000	00	0.2960	0.2240	0.2240	0.2560	0.4800	0.4800	0.4800	0.4800	0.0000	0.0000	0.00%	0.00%
10	01001	01	0.2000	0.3200	0.1600	0.3200	0.4800	0.3600	0.4800	0.3600	0.0000	0.0000	0.00%	0.00%
11	01010	11	0.1680	0.1920	0.1920	0.4480	0.3600	0.3600	0.3600	0.3600	0.0000	0.0000	0.00%	0.00%
12	01011	11	0.1320	0.2280	0.1680	0.4720	0.3600	0.3000	0.3600	0.2800	0.0000	-0.0200	0.00%	-6.67%
13	01100	00	0.2960	0.2240	0.2240	0.2560	0.4800	0.4800	0.4800	0.4800	0.0000	0.0000	0.00%	0.00%
14	01101	00	0.2600	0.2600	0.2000	0.2800	0.4800	0.5400	0.4800	0.5400	0.0000	0.0000	0.00%	0.00%
15	01110	00	0.2480	0.2120	0.2120	0.3280	0.5400	0.5400	0.5400	0.5400	0.0000	0.0000	0.00%	0.00%
16	01111	00	0.2345	0.2255	0.2030	0.3370	0.5400	0.5625	0.5400	0.5888	0.0000	0.0263	0.00%	4.67%
17	10000	00	0.2960	0.2240	0.2240	0.2560	0.4800	0.4800	0.4800	0.4800	0.0000	0.0000	0.00%	0.00%
18	10001	01	0.2000	0.3200	0.1600	0.3200	0.4800	0.3600	0.4800	0.3600	0.0000	0.0000	0.00%	0.00%
19	10010	11	0.1680	0.1920	0.1920	0.4480	0.3600	0.3600	0.3600	0.3600	0.0000	0.0000	0.00%	0.00%
20	10011	11	0.1320	0.2280	0.1680	0.4720	0.3600	0.3000	0.3600	0.2800	0.0000	-0.0200	0.00%	-6.67%
21	10100	00	0.2960	0.2240	0.2240	0.2560	0.4800	0.4800	0.4800	0.4800	0.0000	0.0000	0.00%	0.00%
22	10101	01	0.2000	0.3200	0.1600	0.3200	0.4800	0.3600	0.4800	0.3600	0.0000	0.0000	0.00%	0.00%
23	10110	11	0.1680	0.1920	0.1920	0.4480	0.3600	0.3600	0.3600	0.3600	0.0000	0.0000	0.00%	0.00%
24	10111	11	0.1320	0.2280	0.1680	0.4720	0.3600	0.3000	0.3600	0.2800	0.0000	-0.0200	0.00%	-6.67%
25	11000	10	0.1760	0.1440	0.3440	0.3360	0.3200	0.4800	0.3200	0.4800	0.0000	0.0000	0.00%	0.00%
26	11001	11	0.1200	0.2000	0.2400	0.4400	0.3200	0.3600	0.3200	0.3600	0.0000	0.0000	0.00%	0.00%
27	11010	11	0.1080	0.1520	0.2520	0.4880	0.2600	0.3600	0.2600	0.3600	0.0000	0.0000	0.00%	0.00%
28	11011	11	0.0870	0.1730	0.2130	0.5270	0.2600	0.3000	0.2600	0.2800	0.0000	-0.0200	0.00%	-6.67%
29	11100	10	0.1760	0.1440	0.3440	0.3360	0.3200	0.4800	0.3200	0.4800	0.0000	0.0000	0.00%	0.00%
30	11101	10	0.1550	0.1650	0.3050	0.3750	0.3200	0.5400	0.3200	0.5400	0.0000	0.0000	0.00%	0.00%
31	11110	10	0.1505	0.1470	0.3095	0.3930	0.2975	0.5400	0.2975	0.5400	0.0000	0.0000	0.00%	0.00%
32	11111	10	0.1426	0.1549	0.2949	0.4076	0.2975	0.5625	0.2975	0.5888	0.0000	0.0263	0.00%	4.67%

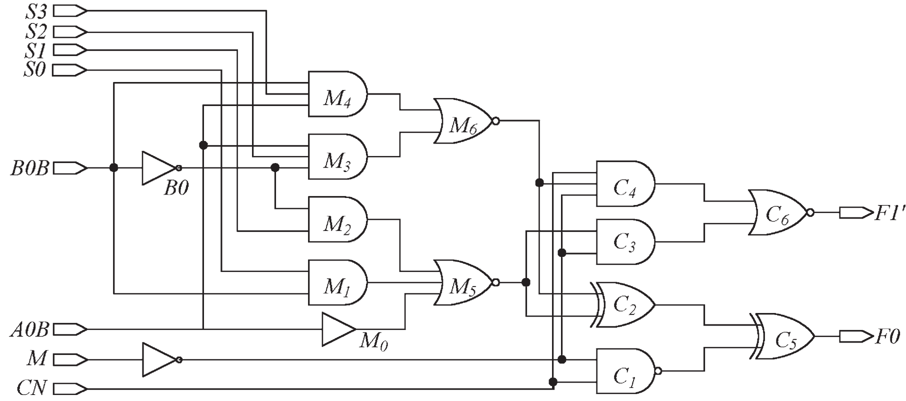


Fig. 8. Fragment of the 74181 ALU.

such errors are acceptable if pattern modeling based on output deviations leads to effective pattern selection.

As shown in Table V, the error at port j is always zero. This is because the input cone of j has only one fan-out stem, and this stem is at the primary input b. Since signal probabilities at primary inputs only take values of zero and one, the accuracy is not affected by the assumption that fan-out branches at a primary input are independent. The input cone of k, however, has a fan-out stem that is not at a primary input. Hence, the error at k is not always zero for c17.

Table V also shows that the low-CL set results in smaller error between $\mathrm{OD_{prob}}$ and $\mathrm{OD_{PTM}}$. Since both sets of CL vectors yield similar results for test-pattern ordering, in the remaining part of this section, we only use the low-CL values to compute $\mathrm{OD_{prob}}$ and $\mathrm{OD_{PTM}}$.

We also considered a fragment of the 74181 ALU [29], as shown in Fig. 8. The fragment that we take only involves the inputs S_0–S_3, M, CN, and the least significant bits of A and B, i.e., $A0B$ and $B0B$. Fig. 9(a) and (b) shows $\mathrm{OD_{prob}}$ and $\mathrm{OD_{PTM}}$ for all the 256 possible input patterns at output ports F_0 and F_1', respectively. We conclude that, although the differences between $\mathrm{OD_{prob}}$ and $\mathrm{OD_{PTM}}$ are noticeable, the curve for $\mathrm{OD_{prob}}$ approximately tracks the shape of the curve for $\mathrm{OD_{PTM}}$. Fig. 9 implies that, if a sufficiently large number of patterns are selected, $\mathrm{OD_{prob}}$ will yield similar results as that obtained using $\mathrm{OD_{PTM}}$. Since we are only interested in the reordering of a large number of test patterns, such differences are acceptable.

To justify the above conclusion, we use the test-pattern-reordering procedure described in Section IV to reorder all the 256 patterns of the fragment, using both $\mathrm{OD_{PTM}}$ and $\mathrm{OD_{prob}}$ as the metric, to obtain two reordered test sequences. If only the first k patterns in each reordered sequence are considered, we can find $c(k)$ common test patterns that belong to both sequences. Table VI lists some pairs of k and $c(k)$. As shown, the number of common test patterns $c(k)$ is sufficiently high to show that we can use $\mathrm{OD_{prob}}$ as a practical alternative to $\mathrm{OD_{PTM}}$ for test-pattern reordering.

To compute $\mathrm{OD_{PTM}}$ for the circuit shown in Fig. 8, we must tensor the input signals and PTMs for all the logic gates, except the three one-input gates (two inverters and one buffer). For example, for gate M_6, its input signals M_3 and M_4 are correlated with M_0, M_1, and M_2. These five signals must be stored in a tensored form, i.e., a 1×32 vector. The PTM for M_6 should also be tensored with an 8×8 identity matrix, resulting in a 32×16 matrix. The output of M_6 is therefore a 1×16 vector that stores signals M_0, M_1, M_2, and M_6. Hence, the computation of $\mathrm{OD_{PTM}}$ is much expensive than the computation of $\mathrm{OD_{prob}}$, and it also requires more memory.

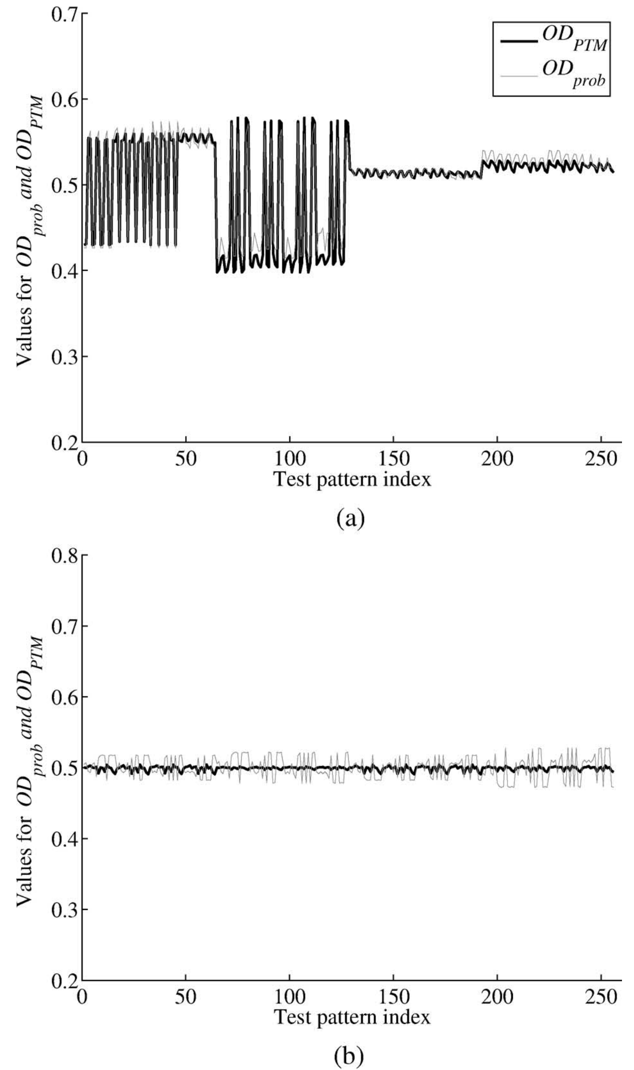


Fig. 9. $\mathrm{OD_{prob}}$ and $\mathrm{OD_{PTM}}$ for the fragment of 74181, as shown in Fig. 8. (a) At port F_1'. (b) At port F_0.

TABLE VI
COMMON TEST PATTERNS SELECTED USING $\mathrm{OD}_{\mathrm{PTM}}$ AND $\mathrm{OD}_{\mathrm{prob}}$

k	$c(k)$	$c(k)/k$ (%)
50	28	56.00
75	42	56.00
100	70	70.00
125	96	76.80
150	123	82.00
175	139	79.43
200	171	85.50
256	256	100

VI. EXPERIMENTAL RESULTS

In this section, we evaluate the effectiveness of the proposed pattern-reordering method for the ISCAS-89 circuits. For each benchmark, we first obtain an n-detect set of test patterns and, then, reorder this test set using different reordering methods.

We do not run ATPG separately for different fault models. ATPG is only done once for n-detection of stuck-at faults. We use different reordering methods to rank these n-detect test patterns. Next, we run fault simulation for the four fault models using the reordered test patterns.

The program to compute output deviations was implemented using C++. On a 64-b Linux server with a 2.4G AMD Opteron 250 processor and 4-GB memory, it takes approximately 15 s to compute the deviations for the 851 n-detect patterns and reorder them for the full-scan version of s38584. This program is an unoptimized prototype. Event-driven techniques can be used to decrease computation time; for consecutive patterns, only the gates whose outputs are changed need to be evaluated.

The fault simulators that we use are not industrial-strength; they are unoptimized academic prototypes that simply use event-driven single-pattern-parallel-fault simulation, hence the fault-simulation times are high. For the 851 five-detect patterns for s38584, the stuck-open fault simulation takes 22 s, and the transition-fault simulation takes 16.9 s. Nevertheless, repeated use of different fault simulators for various fault models is time-consuming for large circuits, even if industry-strength tools are used.

For each circuit, four different cases are considered corresponding to four fault models: 1) stuck-at faults; 2) stuck-open faults detected using the LOS scheme; 3) transition faults detected using the LOS scheme; and 4) transition faults detected using the LOC scheme. The fault coverage for these fault models are obtained using the different reordered test sequences.

The LOC transition-fault simulator assumes two constraints: 1) primary inputs remain unchanged during the two capture cycles and 2) POs are not probed to detect faults. This is because, for at-speed testing, low-speed testers usually cannot drive or probe pins at the functional frequency, which is much faster than the test frequency [30]. The LOS fault simulators do not assume these constraints.

To eliminate any bias in the comparison of different methods for test-set reordering, we use two arbitrarily chosen sets of CL vectors for our experiments. These vectors are defined separately for each gate type. For example, for a two-input NAND gate, we use as follows: 1) "low-CL" vector: $R^{\mathrm{NAND2}} = (0.8^{(00)}, 0.8^{(01)}, 0.8^{(10)}, 0.7^{(11)})$ and 2) "high-CL" vector: $R^{\mathrm{NAND2}} = (0.95^{(00)}, 0.95^{(01)}, 0.95^{(10)}, 0.85^{(11)})$. Fig. 10(a) and (b) shows the coverage for stuck-open (LOS) and transition (LOS) faults obtained using test sequences that are reordered using both high and low CLs. As shown, both sets of CL vectors yield similar results. This is because both CL vectors 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 other input combinations. This observation also shows that the deviation-based reordering technique is less sensitive to the absolute values of CL vectors, as long as the CL vectors can reflect basic attributes of the layout and/or process. It is also expected that more accurate CL vectors that carry more information can yield better results. How to derive those CL vectors is an important topic for ongoing research. In this section, we only report results obtained using the "high-CL" set.

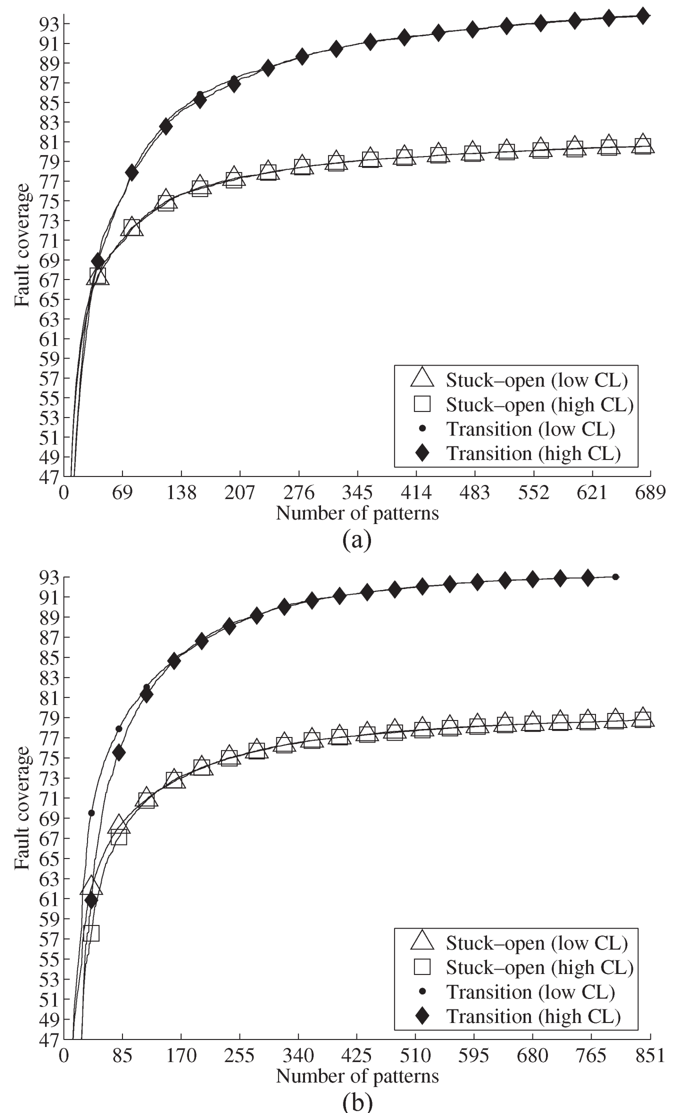


Fig. 10. Coverage for stuck-open and transition faults obtained using high and low CLs, for s38417 and s38584.

Figs. 11–13 show the fault coverage obtained for various test lengths for different fault models for several ISCAS-89

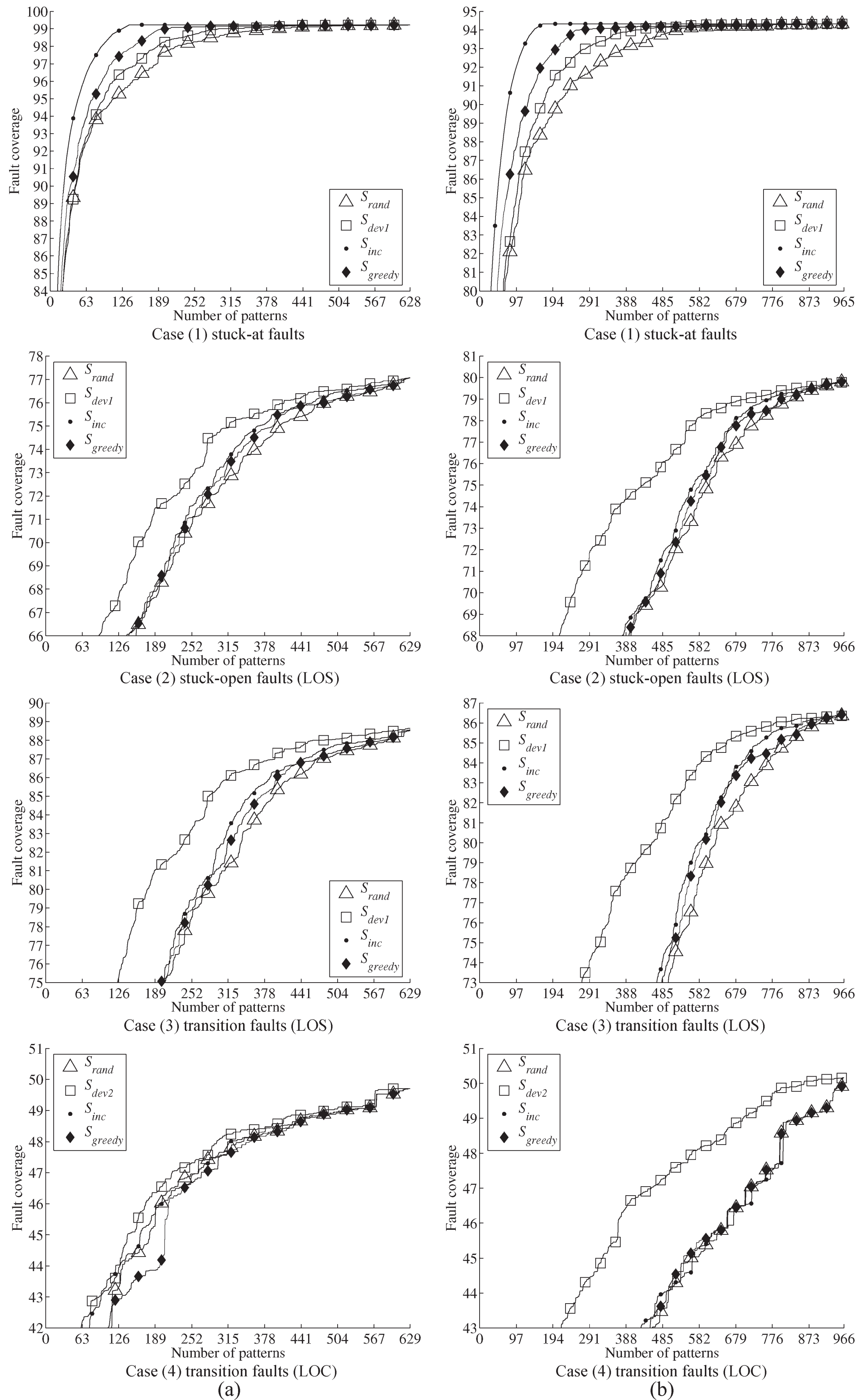


Fig. 11. Fault coverage for (a) s5378 and (b) s9234 (five-detect).

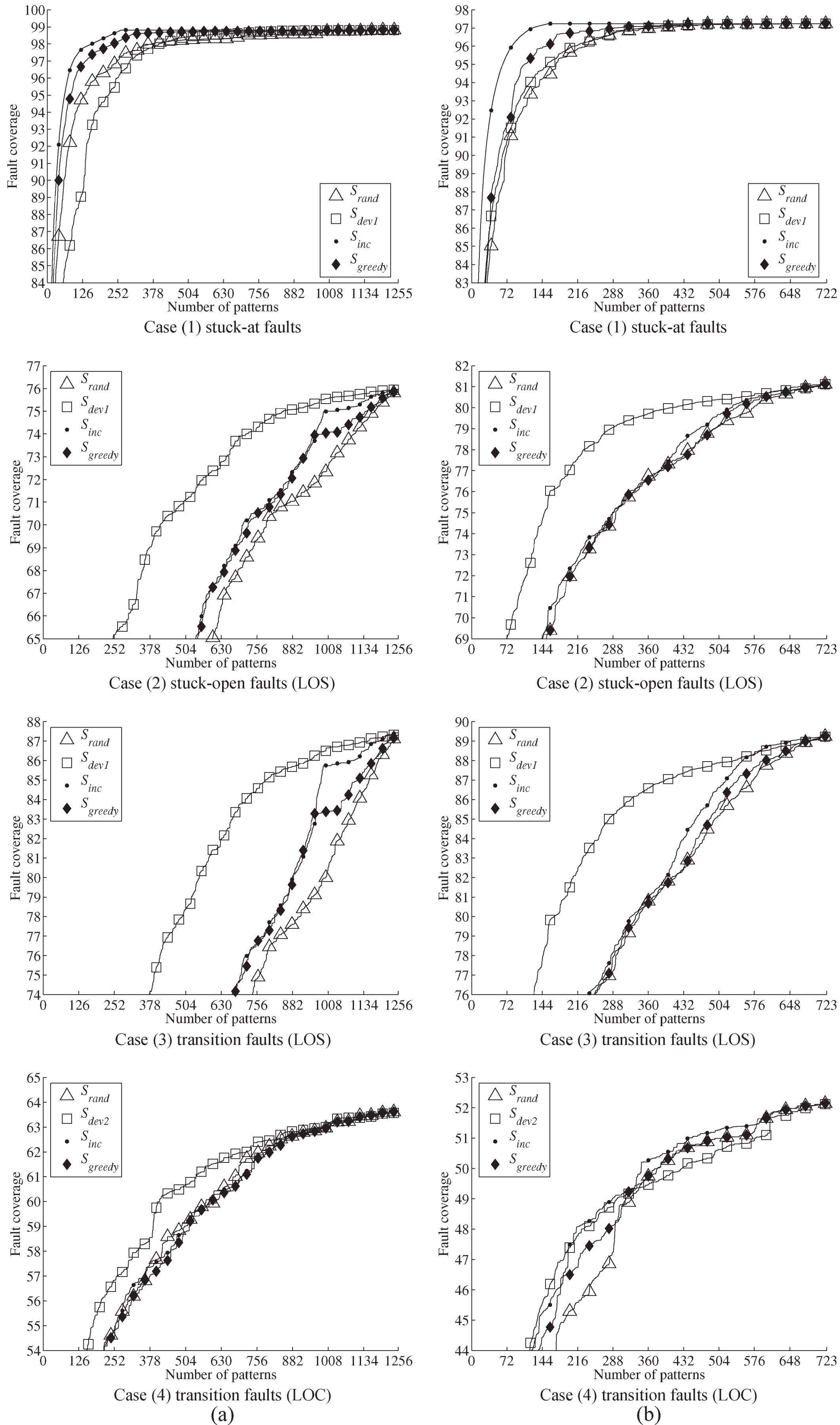


Fig. 12. Fault coverage for (a) s13207 and (b) s15850 (five-detect).

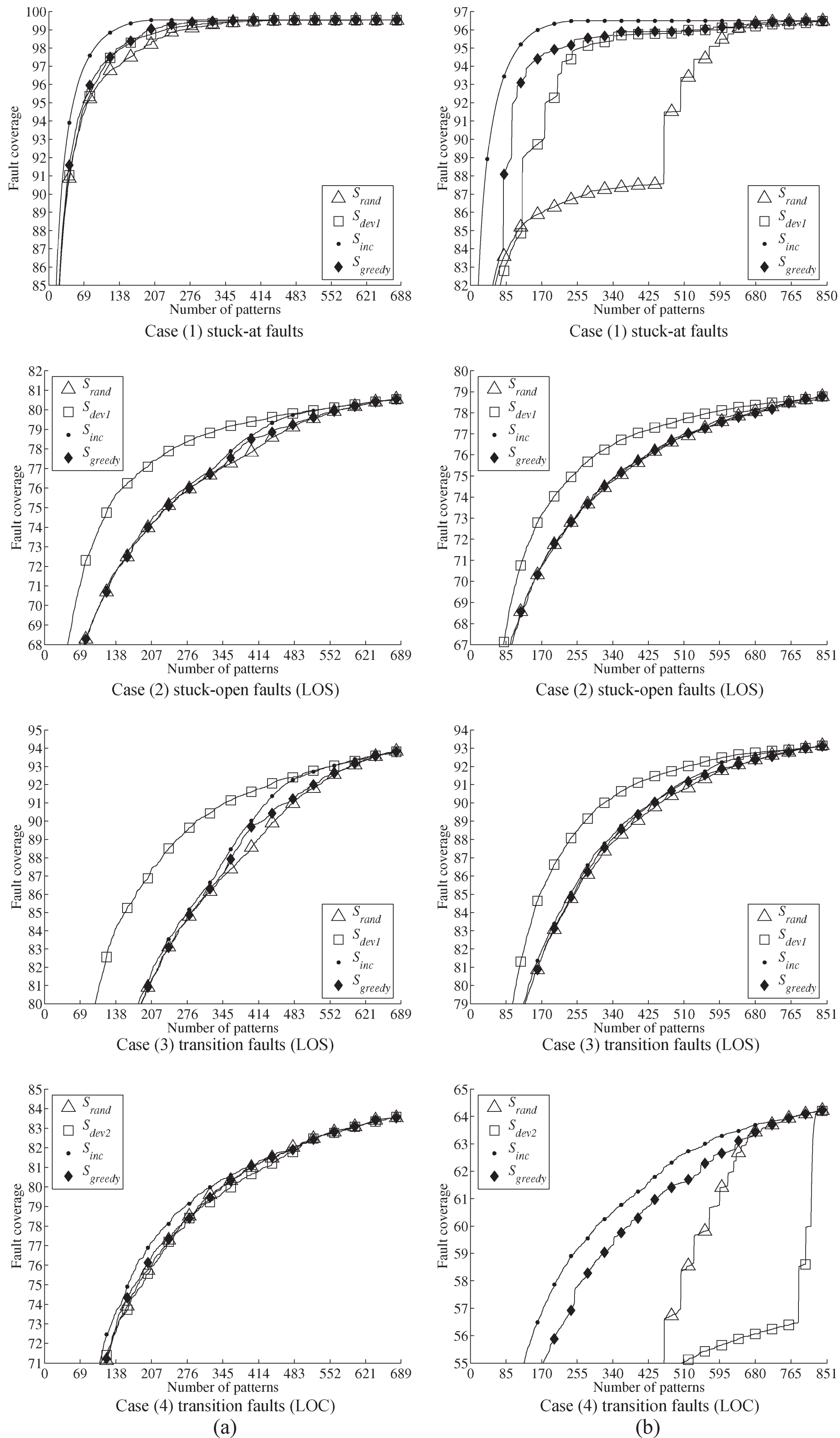


Fig. 13. Fault coverage for (a) s38417 and (b) s38584 (five-detect).

benchmark circuits, using five-detect test cubes. For stuck-open and transition faults, the deviation-based reordering approach appears to be the most promising. The patterns selected using the proposed approach provide higher defect coverage with a smaller number of test patterns. For stuck-at faults, $S_{\rm inc}$ outperforms other reordering methods. This is expected because $S_{\rm inc}$ is tailored to obtain the steepest stuck-at fault coverage.

It can be seen from all cases that, by first applying test patterns selected by the deviation-based method, 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 4, 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 (reordered by the deviation-based method) and, then, add top-off ATPG patterns targeting the remaining hard-to-detect faults.

For the coverage for stuck-open faults and transition faults, test sets reordered using the deviation-based method ($S_{\rm dev1}$ and $S_{\rm dev2}$) outperform test sets reordered by the other methods. For stuck-at fault coverage $S_{\rm inc}$, which is specifically tuned for stuck-at fault coverage, outperforms $S_{\rm dev1}$ and $S_{\rm dev2}$. However, $S_{\rm inc}$ is only optimized for one fault model. Since our goal is to improve the defect coverage, it is inefficient to consider only one fault model for pattern ordering.

VII. Conclusion

We have presented a probabilistic fault model as a technique to grade test patterns for defects that cannot be modeled using compact fault models. We have shown that output deviations offer an effective surrogate coverage metric to model the quality of test patterns. Pattern ranking based on output deviations offers a useful method for test-pattern reordering during high-volume and time-constrained production testing. Experimental results show that test sequences reordered using output deviations are consistently more effective than sequences returned by other methods. We have evaluated pattern grading using the fault coverage for stuck-open and transition faults.

Acknowledgment

The authors would like to thank V. Iyengar of IBM, S. Chakravarty of LSI Logic, Prof. M. Goessel of the University of Potsdam, and S. Wang and W. Wei of NEC Laboratories for their helpful discussions.

References

[1] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing*. Norwell, MA: Kluwer, 2000.

[2] N. Jha and S. Gupta, *Testing of Digital Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[3] B. Keller *et al.*, "An economic analysis and ROI model for nanometer test," in *Proc. Int. Test Conf.*, 2004, pp. 518–524.

[4] F.-F. Ferhani and E. J. McCluskey, "Classifying bad chips and ordering test sets," in *Proc. Int. Test Conf.*, 2006, pp. 1–10.

[5] B. Vermeulen *et al.*, "Trends in testing integrated circuits," in *Proc. Int. Test Conf.*, 2004, pp. 688–697.

[6] 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. 215–230, Apr. 1994.

[7] 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.

[8] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors (ITRS)*, 2005. [Online]. Available: http://www.itrs.net/Common/2005ITRS/Home2005.htm

[9] Y. Tian, M. Mercer, W. Shi, and M. Grimaila, "An optimal test pattern selection method to improve the defect coverage," in *Proc. Int. Test Conf.*, 2005, pp. 762–770.

[10] X. Lin, J. Rajski, I. Pomeranz, and S. Reddy, "On static test compaction and test pattern ordering for scan designs," in *Proc. Int. Test Conf.*, 2001, pp. 1088–1097.

[11] 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.

[12] Z. Wang, K. Chakrabarty, and M. Goessel, "Test set enrichment using a probabilistic fault model and the theory of output deviations," in *Proc. Des., Autom. Test Eur. Conf.*, 2006, pp. 1275–1280.

[13] S. C. Ma, P. Franco, and E. J. McCluskey, "An experimental chip to evaluate test techniques experiment results," in *Proc. Int. Test Conf.*, 1995, pp. 663–672.

[14] S. M. Reddy, I. Pomeranz, and S. Kajihara, "On the effects of test compaction on defect coverage," in *Proc. IEEE VLSI Test Symp.*, 1996, pp. 430–435.

[15] I. Polian, I. Pomeranz, and B. Becker, "Exact computation of maximally dominating faults and its applications to n-detection tests," in *Proc. Asian Test Symp.*, 2002, pp. 9–14.

[16] C. Tseng, S. Mitra, S. Davidson, and E. J. McCluskey, "An evaluation of pseudo random testing for detecting real defects," in *Proc. IEEE VLSI Test Symp.*, 2001, pp. 404–409.

[17] J. Dworak, J. Wicker, S. Lee, M. Grimaila, R. Mercer, K. Butler, B. Stewart, and L. Wang, "Defect-oriented testing and defective-part-level prediction," *IEEE Des. Test Comput.*, vol. 18, no. 1, pp. 31–41, Jan./Feb. 2001.

[18] T. Williams and N. Brown, "Defect level as a function of fault coverage," *IEEE Trans. Comput.*, vol. C-30, no. 12, pp. 987–988, Dec. 1981.

[19] R. Madge, B. Benware, R. Turakhia, R. Daasch, C. Schuermyer, and J. Ruffler, "In search of the optimum test set—Adaptive test methods for maximum defect coverage and lowest test cost," in *Proc. Int. Test Conf.*, 2004, pp. 203–212.

[20] X. Gu, C. Wang, A. Lee, B. Eklow, K.-H. Tsai, J. Tofte, M. Kassab, and J. Rajski, "Realizing high test quality goals with smart test resource usage," in *Proc. Int. Test Conf.*, 2004, pp. 525–533.

[21] K. Butler and J. Saxena, "An empirical study on the effects of test type ordering on overall test efficiency," in *Proc. Int. Test Conf.*, 2000, pp. 408–416.

[22] F. J. Ferguson and J. P. Shen, "A CMOS fault extractor for inductive fault analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 7, no. 11, pp. 1181–1194, Nov. 1988.

[23] K. P. Parker and E. J. McCluskey, "Probabilistic treatment of general combinational networks," *IEEE Trans. Comput.*, vol. C-24, no. 6, pp. 668–670, Jun. 1975.

[24] S. C. Seth, L. Pan, and V. D. Agrawal, "PREDICT—Probabilistic estimation of digital circuit testability," in *Proc. Int. Symp. Fault-Tolerant Comput.*, 1985, pp. 220–225.

[25] K. Y. Cho, S. Mitra, and E. J. McCluskey, "Gate exhaustive testing," in *Proc. Int. Test Conf.*, 2005, pp. 771–777.

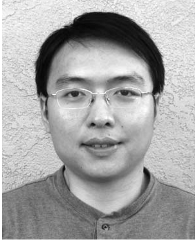
[26] Z. Wang, K. Chakrabarty, and M. Bienek, "A seed-selection method to increase defect coverage for LFSR-reseeding-based test compression," in *Proc. Eur. Test Symp.*, 2007, pp. 125–130.

[27] Y. Huang, "On N-detect pattern set optimization," in *Proc. Symp. Quality Electron. Des.*, 2006, pp. 445–450.

[28] S. Krishnaswamy, I. L. Markov, and J. P. Hayes, "Logic circuit testing for transient faults," in *Proc. Eur. Test Symp.*, 2005, pp. 102–107.

[29] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE Des. Test Comput.*, vol. 16, no. 3, pp. 72–80, Jul.–Sep. 1999.

[30] M. Nelms, K. Gorman, and D. Anand, "Generating at-speed array fail maps with low-speed ATE," in *Proc. IEEE VLSI Test Symp.*, 2004, pp. 87–92.



Zhanglei Wang received the B.Eng. degree in computer and electrical engineering 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 Design Engineer with Cisco Systems, Inc., San Jose, CA. His research interests include test compression, test pattern grading, and test generation.

Krishnendu Chakrabarty (SM'07) received the B.Tech. degree in computer science and engineering 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 of Electrical and Computer Engineering, Duke University, Durham, NC. His current research projects include testing and design-for-testability of system-on-chip integrated circuits, microfluidics biochips, microfluidic-based chip cooling, and wireless-sensor networks. He has authored five books, such as *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 Reconfigutaion Techniques* (CRC Press, 2006), and *Adaptive Cooling of Integrated Circuits using Digital Microfluidics* (Artech, 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 has contributed over a dozen invited chapters to book volumes and published 250 papers in archival journals and refereed conference proceedings. He is the holder of a U.S. patent in built-in self-test and is a coinventor of a pending U.S. patent on sensor networks.

Prof. Chakrabarty was the recipient of the National Science Foundation Early Faculty (CAREER) Award and the Office of Naval Research Young Investigator Award. He was also the recipient of 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 was also the recipient of the Humboldt Research Fellowship, which was awarded by the Alexander von Humboldt Foundation, Germany, and the Mercator Visiting Professorship, which was awarded by the Deutsche Forschungsgemeinschaft, Germany. He is a Distinguished Visitor of the IEEE Computer Society from 2005 to 2007 and a Distinguished Lecturer of the IEEE Circuits and Systems Society from 2006 to 2007. He is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM I, IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS, and *ACM Journal on Emerging Technologies in Computing Systems*, an Editor of the IEEE DESIGN AND TEST OF COMPUTERS, and an Editor of *Journal of Electronic Testing: Theory and Applications (JETTA)*. He serves as a Subject Area Editor for the *International Journal of Distributed Sensor Networks*. He is a Senior Member of the Association for Computing Machinery and a member of Sigma Xi. He served as Program Chair for the 2005 IEEE Asian Test Symposium and Program Chair for the CAD, Design, and Test Conference for the 2007 IEEE Symposium on Design, Integration, Test, and Packaging of Microelectromechanical System/MOEMS.