A Seed-Selection Method to Increase Defect Coverage for LFSR-Reseeding-Based Test Compression*

Zhanglei Wang[†], Krishnendu Chakrabarty[†] and Michael Bienek[‡]

[†]ECE Dept., Duke University, Durham, NC
E-mail: {zw8,krish}@ee.duke.edu

[‡]Advanced Micro Devices, Austin, TX
E-mail: michael.bienek@amd.com

Abstract—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. 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 faults derived from selected seeds. These patterns achieve higher coverage for stuck-open and transition faults than patterns obtained using other methods.

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]. 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 [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 2005 ITRS document predicts that the test data volume and test application time for integrated circuits will be as much as thirty times larger in 2010 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 rely on the use of a linear decompressor such as an 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 don't-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]. 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, especially 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.

The choice of LFSR seeds for compression methods are either random, e.g. as an outcome of Gauss-Jordan Elimination [10], or designed for better seed compression [11]. To enhance the effectiveness of LFSR reseeding for unmodeled defects, new techniques are needed for pattern modeling and seed selection.

This paper uses the "output deviation" [12] as a surrogate coverage-metric for pattern grading for LFSR reseeding. A flexible, but general, probabilistic fault model [13] is used to generate 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.

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 [14]. As n increases, the defect coverage usually improves compared with other test generation methods. Therefore, in this paper, we choose n -detect test cubes as inputs to the LFSR-encoding procedure, and use the proposed seed selection technique to select effective seeds.

The remainder of this paper is organized as follows. Sections 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 the paper and outlines directions for future work.

II. PROBABILISTIC FAULT MODEL & 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 [12], [13].

The confidence level R_i of a gate G_i with m inputs and a single output is a vector with 2^m components, defined as: $R_i = (r_i^{(00...00)} r_i^{(00...01)} r_i^{(00...10)} \dots r_i^{(11...11)})$, where each

*This work was supported in part by the Semiconductor Research Corporation under contract number 1588.

Fig. 1. An illustrative example.

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

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

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

Likewise, the signal probabilities can be easily computed for other gate types. Figure 1 shows a circuit consisting of three gates G_1 , G_2 , and G_3 with different CLs. 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) 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 I are highlighted. Output deviations can be determined without explicit fault grading, hence the computation is feasible for large circuits and large test sets.

We use the following arbitrarily chosen set of CLs for our experiments. These vectors are defined separately for each gate type. For example, for a 2-input NAND gate, we use (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 non-controlling, 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 [12] 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 [12] 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-

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

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

(a) System of linear equations

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

(b) Gauss-Jordan elimination

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

(c) Solution space

Fig. 3. Example to illustrate the solution for a system of linear equations.

AND/OR bridging faults, and gate exhaustive metrics [15].

III. SEED SELECTION

Given the scan chain architecture for the circuit under test, LFSR feedback polynomial, and phase shifter (if any), 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 $A\vec{y} = \vec{z}$, where 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 1-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 $\vec{S}_j = H\vec{S}_{j-1}$, $j = 1, 2, \dots$, where H is the state transition matrix for the LFSR.

The j -th output of the 1-stage phase shifter shown in Fig. 2 can be represented as $O_j = \vec{P}^t \vec{S}_j = \vec{P}^t H^j \vec{S}_0$, $j = 1, 2, \dots$. 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 $O_1 = \vec{P}^t H \vec{S}_0 = (11101) \vec{S}_0 = y_1 + y_2 + y_3 + y_5$.

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 [10] can be used to transform a set of columns in 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.

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

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) Procedure 1

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 for
11: end while

```

(b) Procedure 2

Fig. 4. Procedures: (a) LFSR reseeding with seed selection, (b) Sort 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) Test patterns and output deviations.

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) Outcome of the sorting procedure (Matrix X).

Fig. 5. An example of sorting test patterns based on output deviations.

each test cube (Lines 2-6). 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 top most 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 primary output (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 illustrates 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, we consider the fault coverage obtained using the test patterns that are derived from the seeds for different fault models, including stuck-at faults, stuck-open faults, 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 CUT. These methods are also referred to as skewed-load and broadside testing [1], respectively. Fig. 6 provides an example. Given a test cube

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

from the n -detect test set, a seed is computed and a test pattern p_0 is derived from the seed.

In the LOS scheme, p_2 is obtained by shifting p_1 one clock cycle. Since errors are captured by p_2 , we should apply the n -detect test cubes as the second test pattern for improved fault coverage, and use p_2 for seed selection. 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 the LOS scheme, output deviations are 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 first obtain the responses of the patterns in V , 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 seed set is referred to as S_{dev2} .

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

S_{comp} is used as a baseline to evaluate the impact on storage requirement for the deviation-based seed selection method. In [11], 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 0. 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 [11] to compress S_{dev1} , S_{dev2} , and S_{comp} .

IV. EXPERIMENTAL RESULTS

In this section, we evaluate the effectiveness of the proposed seed-selection procedure for the ISCAS-89 circuits. For each benchmark, we first obtain an n -detect set of test cubes as

described in Figure 1 of [16]. During the ATPG process, after a new set of test cubes is generated, the ATPG tool randomly fills don't-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 don't-cares without any loss of n -detect fault coverage. Due to space limits, we only report results obtained with $n = 5$.

From the n -detect test cubes, the seed sets described in Section III are generated. For each circuit, four 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; (4) transition faults detected using the LOC scheme. The fault coverage for these fault models are obtained using patterns derived from the different seed sets.

The LOC transition fault simulator assumes two constraints: (1) primary inputs remain unchanged during the two capture cycles, and (2) primary outputs 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. The LOS fault simulators do not assume these constraints. We use LOS and LOC fault simulators, and an ATPG tool from industry, as provided by our partners.

Figures 7-8 show the fault coverage obtained for various test lengths for different fault models for several ISCAS-89 circuits. For stuck-open and transition faults, the deviation-based approach appears to be the most promising. 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 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 (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 requirement for S_{dev1} (S_{dev2}) and S_{comp} , as shown in Table II. The compression results for S_{dev2} and S_{dev1} are identical for all the benchmark circuits. In Table II, 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

TABLE II. 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

than the test data volume for [11], while achieving significantly higher defect coverage. It is also noteworthy that the seed volume reported in Table II 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 [11] infeasible.

V. CONCLUSIONS

We have proposed the use of output deviations as a metric to select effective LFSR seeds. These seeds lead to high-deviation patterns, which provide high defect coverage than patterns derived from other seeds. We have also shown that the patterns generated using the proposed method have negligible impact on test data volume. To evaluate defect coverage, we have determined the coverage of stuck-open and transition faults provided by a set of n -detect stuck-at patterns. In ongoing work, we are studying the problem of selecting LFSR polynomials to increase the defect coverage.

REFERENCES

- [1] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing*. Kluwer Academic Publishers, Norwell, MA, 2000.
- [2] B. Keller *et al.*, "An economic analysis and ROI model for nanometer test," in *Proc. ITC*, 2004, pp. 518-524.
- [3] F.-F. Ferhani and E. J. McCluskey, "Classifying bad chips and ordering test sets," in *Proc. ITC*, 2006.
- [4] B. Vermeulen *et al.*, "Trends in testing integrated circuits," in *Proc. ITC*, 2004, pp. 688-697.
- [5] H. Cox and J. Rajski, "On necessary and nonconflicting assignments in algorithmic test pattern generation," *IEEE Trans. CAD*, vol. 13, pp. 215-230, 1994.
- [6] K. Yang, K.-T. Cheng, and L.-C. Wang, "TranGen: a SAT-based ATPG for path-oriented transition faults," in *Proc. Asia and South Pacific Design Automation Conf.*, 2004, pp. 92-97.
- [7] Semiconductor Industry Association. (2005) International Technology Roadmap for Semiconductors (ITRS), <http://www.itrs.net/Common/2005ITRS/Home2005.htm>.
- [8] J. Rajski *et al.*, "Embedded deterministic test," *IEEE Trans. CAD*, vol. 23, pp. 776-792, May 2004.
- [9] B. Koenemann, "LFSR-coded test patterns for scan design," in *Proc. European Test Conf.*, 1991, pp. 237-242.
- [10] C. Cullen, *Linear Algebra with Applications*. Addison-Wesley, 1997.
- [11] C. Krishna and N. A. Toubia, "Reducing test data volume using LFSR reseeding with seed compression," in *Proc. ITC*, 2002, pp. 321-330.
- [12] 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.
- [13] Z. Wang, K. Chakrabarty, and M. Goessel, "Test set enrichment using a probabilistic fault model and the theory of output deviations," in *Proc. DATE Conf.*, 2006.
- [14] 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.
- [15] K. Y. Cho, S. Mitra, and E. J. McCluskey, "Gate exhaustive testing," in *Proc. ITC*, 2005, pp. 771-777.
- [16] Y. Huang, "On N -detect pattern set optimization," in *Proc. Symp. Quality Electronic Design*, 2006.

Fig. 7. Fault coverage for s5378 and s9234: (1) stuck-at faults; (2) stuck-open faults (LOS); (3) transition faults (LOS); (4) transition faults (LOC).

Case (1) stuck-at faults

Case (1) stuck-at faults

Case (2) stuck-open faults (LOS)

Case (2) stuck-open faults (LOS)

Case (3) transition faults (LOS)

Case (3) transition faults (LOS)

Case (4) transition faults (LOC)

(a) s15850

Case (4) transition faults (LOC)

(b) s38417

Fig. 8. Fault coverage for s15850 and s38417: (1) stuck-at faults; (2) stuck-open faults (LOS); (3) transition faults (LOS); (4) transition faults (LOC).