Generation of Compact Stuck-At Test Sets Targeting Unmodeled Defects

Xrysovalantis Kavousianos, *Member, IEEE,* and Krishnendu Chakrabarty, *Fellow, IEEE*

***Abstract*—This letter presents a new method to generate compact stuck-at test sets that offer high defect coverage. The proposed method first selects the most effective patterns from a large N-detect repository, by using a new output deviation-based metric. Then it embeds complete coverage of stuck-at faults within these patterns, and uses the proposed metric to further improve their defect coverage. Results show that the proposed method outperforms a recently proposed competing approach in terms of unmodeled defect coverage. In many cases, higher defect coverage is obtained even than much larger N-detect test sets for several values of N. Finally, results provide the insight that, instead of using N-detect testing with as large N as possible, it is more efficient to combine the output deviations metric with multi-detect testing to get high-quality, compact test sets.**

***Index Terms*—Defect-oriented testing, multi-detect testing.**

I. Introduction

The most widely used fault model is the single stuck-at fault model; it is simple, requires low computational effort for test generation, and test patterns for single stuck-at faults also detect many physical defects. However, it does not offer high defect coverage. This inadequacy has led to the development of new fault models which reflect the behavior of many realistic defects more accurately. A drawback of these models is that they lead to prohibitively high pattern counts, thereby leading to high test application times. Moreover, many new defects cannot be modeled using existing fault models [17].

An alternative approach that increases defect coverage, and benefits from the low complexity of simple fault models, is multi-detect testing, also referred to as N-detect testing [1]–[6], [8], [10]–[16]. The main idea of N-detect testing is to apply $N > 1$ different test patterns for each stuck-at fault. By detecting each stuck-at fault multiple times, with different test patterns each time, the probability that arbitrary defects are activated at the target fault site increases. The major drawback of N-detect testing is that the size of the test set increases linearly with N. An alternative method was proposed in [7] that exploits the unspecified values ("X") of single detect stuck-at test sets in order to embed multi-detection of stuck-at faults within these single-detect test sets.

In this letter, we propose a new method to generate high-quality compact test sets with test lengths similar to that of single-detect stuck-at test sets. The proposed method embeds single-detection of stuck-at faults within a small number of the most-efficient test patterns which are appropriately selected from an N-detect repository to guarantee high un-modeled defect coverage. In most of the cases these patterns also detect the vast majority of the stuck-at faults while the detection of the remaining stuck-at faults is embedded within their "X" values and with a few additional top-off patterns.

The proposed method utilizes a new output deviation-based metric to identify the most effective test patterns from the repository. Output deviations [18] offer an effective means to successfully identify the most effective test patterns, without being biased toward any particular fault model. The proposed metric is more effective than the metric proposed in [18] because: 1) it achieves a weighted distribution of high deviation values at circuit outputs, and 2) it favors those outputs which exhibit increased potential to detect defects. In addition, it evaluates test patterns for both timing-dependent and timing-independent unmodeled defects at the same time, and outperforms the metric proposed in [9], which generates different test sets to target each kind of these defects.

Simulations results for the ISCAS and IWLS benchmark circuits [20] show that, despite their compact size, the test sets generated by the proposed method provide significantly higher coverage of transition-delay faults and comparable coverage for bridging faults, when compared to the baseline single-detect test sets and the test sets obtained using [7]. Moreover, they offer higher coverage of transition-delay faults than larger N-detect test sets for several values of N. Finally, we show that instead of simply increasing the value of N for N-detection, which is currently common industry practice, a better approach is to combine N-detection with pattern selection based on output deviations.

II. Output-Deviation Based Metric

In this section, we present the proposed metric. Hereafter, a *test cube* is a pattern with 0, 1 and don't-care ("X") logic values, and a *test vector* is a test pattern without X values.

Output deviations [18] are probability measures at primary outputs and pseudo-outputs that reflect the likelihood of error detection at these outputs. They are based on a probabilistic fault model, in which a probability map, the confidence-level vector, is assigned to every gate in the circuit. Signal probabilities $p_{i,0}$ and $p_{i,1}$ are associated with each line i for every input pattern, where $p_{i,0}$ and $p_{i,1}$ are the probabilities for line i to be at logic 0 and 1, respectively. The confidence level R_i of a gate G_i with m inputs is a vector with 2^m components, defined as $R_i = (r_i^{0\ldots00} r_i^{0\ldots01} \ldots r_i^{1\ldots11})$, where each component of R_i denotes the probability that the gate output is correct for the corresponding input combination. For example, let y be the output of a NAND gate G_i, with inputs a and b. We have

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

Manuscript received July 31, 2010; revised October 30, 2010; accepted December 9, 2010. Date of current version April 20, 2011. The work of K. Chakrabarty was supported in part by Semiconductor Research Corporation, under Contract 1588. A preliminary version of this paper was presented at DATE 2009. This paper was recommended by Associate Editor A. Ivanov.

X. Kavousianos is with the Department of Computer Science, University of Ioannina, Ioannina 45110, Greece (e-mail: kabousia@cs.uoi.gr).

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

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.2010.2101750

For any logic gate G_i in a circuit, let its fault-free output value for any given input pattern t_j be d, with $d \in \{0, 1\}$. The *output deviation* $\Delta G_{i,j}$ of G_i for t_j is defined as $p_{G_i,\overline{d}}$, where $\overline{d}$ 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 pattern. Deviation values are indicative of the probability arbitrary defects to be detected at circuit outputs. Output deviations can be determined without explicit fault grading; hence the computation (linear in the number of gates) is feasible for large circuits and large test sets.

The proposed metric evaluates each test vector according to its potential to detect both timing-independent and timing-dependent defects. Timing-independent defects are detected by the immediate response (denoted hereafter as $R1$) of each stuck-at test vector. For timing-dependent defects, we assume that each stuck-at test vector is applied at the circuit using the launch-on-capture (LOC) technique. According to the LOC technique, the second response (denoted as $R2$) of each stuck-at test vector is used to detect timing-dependent defects.

The first objective of the metric is to identify all vectors that offer high output deviation values at the first and/or second response. Let us consider a circuit with Q observable outputs, and the set L of stuck-at test vectors. Each test vector $t \in L$ is applied at the circuit and two responses are captured in the scan chain: the immediate response t^{R1} and the second response t^{R2} which is generated according to the LOC technique. Let $t^{R1}_{O_i}$, $t^{R2}_{O_i}$ be the fault free values, and $D(t^{R1}_{O_i})$, $D(t^{R2}_{O_i})$ be the deviation values at output O_i, $(i \in [1, Q])$ at the first ($R1$) and second ($R2$) response of t, respectively. The metric calculates the maximum deviation value $MD^R(O_i, v)$ at O_i when the fault free logic value is equal to v at response R (hereafter R will be either $R1$ or $R2$ or both) using the formula

$$MD^R(O_i, v) = max\{D(t^R_{O_i}) : t \in L, t^R_{O_i} = v\}$$
$$\text{for} \quad R = R1, R2 \quad i \in [1, Q] \quad v = 0, 1. \tag{1}$$

This formula calculates four maximum deviation values for each output O_i, which correspond to both fault free logic values 0, 1 at this output for both responses $R1$, $R2$. Calculating four maximum deviation values for each output reflects the fact that: 1) different defects are observable at the responses $R1$, $R2$, at each output, and 2) different defects are usually observable at the same output and the same response by patterns that produce different fault free logic values at this output. Thus, 4xQ maximum deviation values are calculated.

The maximum deviation values are used to establish the boundary between high and low output deviation values (low output-deviation values are discarded). Specifically, any value $D(t^R_{O_i})$ is high if $D(t^R_{O_i}) \geq Thr{\cdot}MD^R(O_i, t^R_{O_i})$ where $0 << Thr \leq 1$. Thr is a real-valued quantity used as the threshold value between low and high output deviation values. The value of Thr should be set close to 1 in order to select only vectors that offer close to maximum output deviation values. We verified that a value of Thr in the range [0.99, 0.995] provides high-quality vectors, and thus we set $Thr = 0.995$.

The second objective of the metric is to evaluate the volume of defects that can be detected at each circuit output. This volume is likely to be higher at a circuit output which is driven by a large logic cone than the corresponding volume at the circuit output which is driven by a small logic cone. We can measure the volume of defects that can be detected at any circuit output as the number of lines in the logic cone driving this output. To this end, we consider a weight $wo^R(O_i, v)$ for every (R, i, v)-tuple $(v = 0, 1)$ which is initially set equal to the number of lines in the logic cone driving output O_i. In accordance with the calculation of four different maximum deviation values at each output we also assume four different weights for each output, one for each pair of response, error-free logic value (as it will be apparent shortly, during the selection of test vectors these weights are independently adjusted according to the output-deviation values of the selected test vectors to reflect the potential of each output to detect defects). Then, for each vector $t \in L$, a weight $WT^R(t)$ is calculated for both responses $R1$, $R2$ as the sum of the weights $wo^R(O_i, t^R_{O_i})$ of all outputs O_i, $1 \leq i \leq Q$, with high deviation values (note that $t^R_{O_i} = 0$ or 1). Thus

$$WT^R(t) = \sum_{i \in [1, Q]:\ D(t^R_{O_i}) \geq Thr \cdot MD^R(O_i, t^R_{O_i})} wo^R(O_i, t^R_{O_i}) \quad R = R1, R2.$$

Each output O_i with high deviation value at response R when t is applied contributes to the weight of t proportionally to its potential to observe defects as it is represented by the value $wo^R(O_i, t^R_{O_i})$. The weight of test vector t is calculated as

$$WT(t) = WT^{R1}(t) + WT^{R2}(t). \tag{2}$$

Among the test vectors in set L, the one with the highest value WT is identified as the most effective one.

The final objective of the metric is to select test vectors in such a way as to provide a weighted distribution of high deviation values at all outputs. Note that outputs with increased observability for test vector t are expected to detect many defects at their logic cones when t is applied. Thus, if t is selected, these outputs are expected to offer less defect detection during the application of the test vectors following, regardless of the potential of these vectors to detect defects. Thus, every time a test vector t is selected that provides high deviation value at output O_i at response R, the weight $wo^R(O_i, t^R_{O_i})$ is divided by a constant factor DF (note that $t^R_{O_i} = 0$ or 1). In this way, the selected test vectors offer high deviation value at all outputs in a weighted fashion (i.e., outputs of large logic cones are still favored compared to outputs of small logic cores). The value of parameter DF determines how fast the selection process begins to select test vectors with high deviation values at the outputs of smaller cones too (the higher is the value of DF, the sooner such test vectors are selected). We verified that a value of DF in the range [2, 10] guarantees the selection of test vectors with high deviation values at all outputs. We have chosen the value of $DF = 8$.

The proposed output deviation-based metric is more efficient than the metric proposed in [18] as it considers the structure of the circuit and also offers a weighted distribution of high deviation values at all outputs. In addition, it is more efficient than the metric proposed in [9] as it evaluates both responses $R1$, $R2$ of each test cube at the same time and thus enables the generation of compact test sets with high coverage of both timing-dependent and timing-independent defects.

Finally, we note that another output-deviation based metric was proposed in [19] but this targets only small delay defects.

III. Proposed Test-Generation Method

Step 1) In the first step, a repository of test cubes is generated, using N-detect ATPG with as high a value of N as is computationally feasible (N is a user-defined parameter). The purpose of this step is to generate a pool of highly efficient test cubes in order to select the most efficient (in terms of defect coverage) ones. This set will become the basis for generating the single detect stuck-at test set. N-detection ATPG offers large volumes of test cubes among which test cubes that are very effective for detecting defects exist and which can be identified by the proposed metric. During ATPG, the Xs of the test cubes are left unspecified in order to be exploited at later steps, and the dynamic-compaction option is turned on in order to limit the size of the repository.

Step 2) In this step, the generated test cubes are evaluated using the proposed metric. Since output deviations are not defined in [18] for test cubes (i.e., test patterns containing Xs), we evaluate each test cube according to its potential to yield test vectors with high output deviations values if its Xs are replaced randomly by logic values 0, 1. To this end, we generate m random test vectors per cube by specifying its Xs in m different random ways (m is a predetermined constant). Next, all test vectors are inserted in a set L and they are evaluated using the output deviation-based metric. Note that the m random test vectors generated for each cube are used only for evaluating the respective test cube, and they are discarded afterward. Eventually, the k most effective test vectors that correspond to k different test cubes are identified and the respective k cubes are selected and form the basis for the test set (k and m are user-defined parameter). Specifically, for selecting each of the k test cubes the test vector t with the highest weight $WT(t)$ is identified and the corresponding test cube is selected. The rest $m-1$ test vectors corresponding to the selected test cube are dropped from set L. This is iteratively applied k times (i.e., until k test cubes are selected).

Step 3) The next step ensures that the selected test cubes achieve complete coverage of stuck-at faults. We perform stuck-at fault simulation with the selected cubes and we drop every stuck-at fault the first time it is detected. Then, we specify the "X" values of the selected cubes in order to detect as many undetected stuck-at faults as possible. If necessary, we generate additional top-off test cubes using a new ATPG step with the dynamic compaction option turned on.

Step 4) This step is optional and is motivated by the method proposed in [7]. As many of the remaining Xs as possible are specified in order to achieve multiple detections of as many stuck-at faults as possible, as suggested in [7]. This further improves the defect coverage of the test sets in many cases.

Step 5) At this step, all remaining Xs are exploited to maximize the effectiveness of test patterns according to the proposed metric (note that even if Step 4 is applied prior to Step 5, many Xs still remain in the test cubes as Step 4 cannot exploit all Xs for improving the multi-detection of test cubes). To this end, a similar process with the selection of test cubes from the repository described in Step 2 is applied. First, for each test cube, m random test vectors are generated by filling the unspecified bits in m different random ways. Then, the output deviations for all test vectors at both responses $R = R1, R2$ as well as the $MD^R(O_i, v)$ for $v = 0, 1, i \in [1, Q]$ values are calculated. The high output deviation values are identified and the weights $wo^R(O_i,v)$ for $i \in [1, Q]$, $v = 0, 1$ and $R = R1, R2$ are initialized to the volume of lines in the logic cone of output O_i. Then, an iterative process selects the test vectors with the highest weights WT calculated using (2) and updates the weights as shown in Section II. When a test vector is selected, the remaining $m - 1$ vectors generated by the same cubes are discarded. This terminates when one vector has been selected for each cube.

TABLE I
Benchmarks and Test-Set Sizes

	Circuit	# Inp.	# Scan Cells	# Gates	Reg_SD/ Emb_ND	Prop SD/ND	Pure_ND ($N = 10$)
ISCAS'89	s5378	35	179	3114	130	140	436
	s9234	36	211	4636	150	159	1126
	s13207	62	638	6837	269	290	869
	s15850	77	534	7949	137	143	678
	s38417	28	1636	21 K	106	111	560
	s38584	38	1426	23 K	164	170	1049
IWLS'05	sytemcaes	258	670	17 K	211	220	621
	tv80	13	359	13 K	640	672	3577
	usb_funct	112	1746	23 K	129	123	964
	ac97_ctrl	54	2199	24 K	53	60	393
	mem_ctrl	116	1078	22 K	577	608	2680
	pci_bridge32	159	3358	38 K	203	215	1610
	ethernet	93	10 544	136 K	1110	1117	7491

IV. Simulation Results

The test-generation flow, excluding ATPG and fault simulation, was implemented in C. Commercial tools were used for all ATPG-related and fault simulation steps. We used the largest ISCAS'89 and a subset of IWLS'05 benchmark circuits [20]. The basic characteristics of these circuits are shown in Table I. The total CPU time (including all steps) of the proposed method varies from a few minutes for small circuits to 2.5 h for the largest circuit, namely "*Ethernet*." The most time-consuming part of our method is related to the selection of the test patterns from the N-detect repository in Step 2 (see Section II). Specifically, for the "*Ethernet*" benchmark circuit this step takes 1.9 h to finish.

The quality of the proposed method, with respect to defect coverage, was evaluated using two surrogate fault models—the transition-delay and the bridging fault model. These fault models are not targeted by the generated stuck-at test sets, but instead, they are used as a means to evaluate the effectiveness

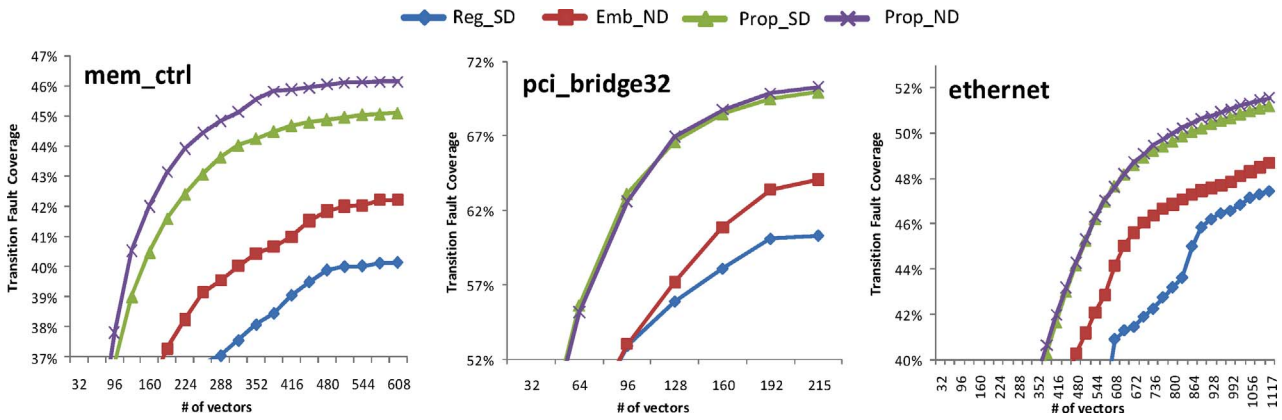

Fig. 1. Transition-fault coverage ramp-up.

of the proposed method for detecting un-modeled defects. We used the LOC technique for detecting timing-dependent defects. Thus, transition faults are detected by the response to the second vector for each vector pair (consisting of each stuck-at test and its response). For detecting bridging faults, we used the immediate response to every stuck-at test vector.

We compare the proposed test-generation method with traditional single-detect and N-detect stuck-at ATPG as well as with the embedded multi-detect ATPG method proposed in [7]. The following test sets were compared with each other.

1) ***Reg_SD***: traditional (regular) single-detect stuck-at test set, with the Xs specified randomly.
2) ***Pure_ND***: traditional N-detect stuck-at test set with the Xs specified randomly.
3) ***Emb_ND***: single-detect stuck-at test set, with the Xs filled in such a way as to embed multi-detection (up to N) of stuck-at faults. The approach of [7] was implemented for this purpose.
4) ***Prop_SD***: compact single-detect stuck-at test set generated by the proposed flow, with Xs specified exclusively to maximize output deviation values (Step 4 is omitted).
5) ***Prop_ND***: compact single-detect stuck-at test set generated by the proposed test-generation flow, with the Xs specified in order to detect first multiple (up to N) times as many stuck-at faults as possible (Step 4 is applied) and then the remaining Xs are specified in order to maximize output deviation values.

We run various experiments to study the effect of parameters N, k (k is the volume of test cubes selected from the N-detect repository) and m (m is the volume of random fillings applied at each test cube). An extensive analysis of these experiments can be found in [21]. From these experiments we concluded that the values of N and m should be as large as possible, while the value of k should be the largest one that complies with the test data volume constraints of the design. For the rest of the experiments, we assume the following values for these parameters: $N = 10$, $m = 10$, and $k = 30\%, 50\%$. We also assume the value $N = 10$ for the Emb_ND method in order to ensure a fair comparison.

The sizes of the test sets generated by the Reg_SD, Emb_ND, $Pure\_ND$, $Prop\_SD$, and $Prop\_ND$ methods are shown in the last three columns of Table I. Column 5 presents

TABLE II
TRANSITION: BRIDGING FAULT COVERAGE (%)

	Transition Fault Coverage				Random Bridging Faults			
Circuit	RegSD	Emb ND	Prop SD	Prop ND	Reg SD	Emb ND	Prop SD	Prop ND
s5378	58.5	61.6	65.2	**65.3**	93.6	95.1	94.6	**95.4**
s9234	39.7	42.9	48.5	**48.9**	86.1	87.4	86.5	**87.9**
s13207	61.1	63.1	**68.4**	67.0	92.1	94.0	93.7	**94.6**
s15850	50.9	49.7	**54.6**	52.9	93.1	94.1	93.6	**94.4**
s38417	77.1	78.5	**80.4**	80.3	96.7	97.6	97.0	**97.8**
s38584	60.9	61.7	**62.4**	62.1	89.4	90.5	89.8	**90.7**
sytemcaes	65.6	64.2	**72.8**	71.4	95.4	95.8	95.7	**95.9**
tv80	53.0	55.2	**60.5**	59.7	88.9	89.5	89.4	**89.6**
usb_funct	60.0	63.6	64.6	**65.9**	94.7	**96.3**	95.1	96.1
ac97_ctrl	42.8	43.9	47.2	**47.6**	96.4	97.2	96.8	**97.7**
mem_ctrl	40.1	42.2	45.1	**46.2**	74.3	75.0	75.1	**75.5**
pci_bridge32	59.7	64.1	70.0	**70.3**	95.7	96.8	96.3	**97.0**
ethernet	47.4	48.7	51.2	**51.6**	90.5	**91.5**	90.8	91.3

the number of test vectors in Reg_SD and Emb_ND (the pattern counts are the same), column 6 presents the (identical) number of test vectors in $Prop\_SD$ and $Prop\_ND$, and finally, column 7 lists the number of the test cubes in the 10-detect pattern repositories (also denoted as $Pure\_ND$). The size of the test sets generated by the proposed method is almost the same as the size of the test sets generated by the other methods and significantly smaller than the size of the repositories used. Note that the test-set sizes for the proposed method can be reduced even further using smaller values of k.

Next, we compare the four test sets with respect to the coverage achieved for transition-delay faults. The results are shown in the second to the fifth column of Table II. As expected, in most of the cases, Emb_ND, $Prop\_SD$, and $Prop\_ND$ provide significantly higher transition-fault coverage than the baseline Reg_SD test set. Moreover, both the proposed test sets, $Prop\_SD$ and $Prop\_ND$, provide higher coverage than Emb_ND. In more than half of the cases, the highest coverage is provided by the $Prop\_ND$ test sets.

In Fig. 1, we present the transition fault coverage ramp-up for these methods for selected benchmark circuits (the respective charts for the rest of the circuits can be found in [21]). In each of the charts the x-axis presents the volume of test vectors applied and the y-axis the transition fault coverage. It is clear that both the proposed methods provide high ramp-up and thus they offer reduced test application time in an abort-at-first-fail environment.

Next, we show that a high degree of multi-detection is not always necessary for high defect coverage. We present results

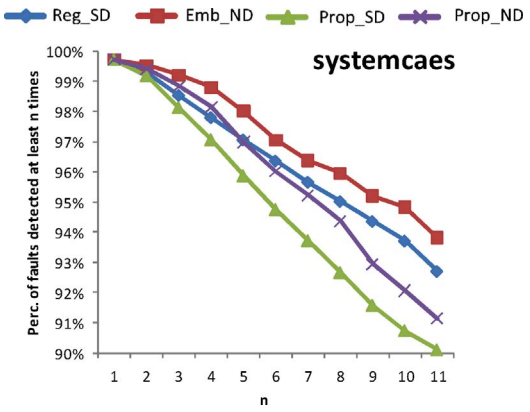

Fig. 2. Multi-detection results for the different test sets.

TABLE III
SMALL TEST SETS ARE MORE EFFECTIVE THAN N-DETECT TEST SETS

Circuit	N^*: Threshold on N	Test-Set Size N^*-Detect	Test-Set Size Proposed	Reduction of Size (%)
s5378	6	318	140	56.0%
s9234	5	585	159	72.8%
s13207	6	608	290	52.3%
s15850	5	368	143	61.1%
s38417	3	208	111	46.6%
s38584	2	259	170	34.4%
sytemcaes	6	441	220	50.1%
tv80	3	1324	672	49.2%
usb_funct	2	219	123	43.8%
ac97_ctrl	3	130	60	53.8%
mem_ctrl	8	2285	608	73.4%
pci_bridge32	3	534	215	59.7%
ethernet	2	2102	1117	46.9%

for systemcaes benchmark circuit which is a representative case (the other benchmarks exhibit similar behavior). Each curve in Fig. 2 presents the percentage of stuck-at faults detected n times or more for $n = 1, 2 \ldots 11$. Note that the test set of the proposed method provides less multi-detection than the two baseline methods, yet it provides higher transition-fault coverage. The test set of the proposed *Prop_SD* method offers less multi-detection than the *Emb_ND* method but higher transition fault coverage at the same time. We therefore conclude that generating patterns with high deviations allows us to get high defect coverage with a smaller value of N than would be possible by using N-detect testing alone. Hence, a combination of output deviations and multi-detection offers the most promising solution.

Next, we compare the four test sets using the bridging fault model. 100 K pairs of lines were selected randomly and four bridging faults were simulated for each pair by considering both lines as aggressors and victims, as well as both AND, OR bridging faults. Columns 6–9 of Table II show the random bridging fault coverage (the best results are boldfaced). In most of the cases, the *Prop_ND* test set provides the best results. In very few cases *Emb_ND* provides marginally higher bridging fault coverage than the *Prop_ND* case.

Finally, we determine a threshold N^* on N, such that for all $N < N^*$, either *Prop_SD* or *Prop_ND* test set offers higher transition-fault coverage than an N^*-detect (*Pure_ND*) test set (note that all test sets provide complete coverage of detectable stuck-at faults). Table III presents the results. Columns 2 and 3 present the value of N^* as well as the corresponding size of *Pure_ND* test set. The last two columns present the test set size of the proposed method and the test set size reduction compared to the N^* detect test set, respectively. The results in Table III demonstrate that, for most benchmarks, the proposed method leads to much smaller but more effective test sets than several pure N-detect test sets. This supports our finding that N-detect ATPG in conjunction with the proposed output deviation-based method offers the most promising solution for generating test sets of high defect coverage.

V. CONCLUSION

We presented a new method to generate stuck-at test sets with high un-modeled defect coverage. Results show that compact test sets can be generated offering higher coverage of unmodeled defects compared to other methods. The effectiveness of the proposed method is attributed to the combination of multi-detect ATPG and pattern selection based on output deviations; therefore, this method serves as a promising alternative to N-detect ATPG with large N.

REFERENCES

[1] M. Amyeen, S. Venkataraman, A. Ojha, and S. Lee, "Evaluation of the quality of n-detect scan ATPG pattern on a processor," in *Proc. Int. Test Conf.*, 2004, pp. 669–678.
[2] B. Benware, C. Schuermyer, N. Tamarapalli, K.-H. Tsai, S. Ranganathan, R. Madge, J. Rajski, and P. Krishnamurthy, "Impact of multiple-detect test patterns on product quality," in *Proc. Int. Test Conf.*, 2003, pp. 1031–1040.
[3] R. Blanton, K. Dwarakanath, and A. Shah, "Analyzing the effectiveness of multiple-detect test sets," in *Proc. Int. Test Conf.*, 2003, pp. 876–885.
[4] E. J. McCluskey and C.-W. Tseng, "Stuck-fault tests versus actual defects," in *Proc. Int. Test Conf.*, 2000, pp. 336–343.
[5] J. Dworak, J. D. Wicker, S. Lee, M. R. Grimaila, M. R. Mercer, K. M. Butler, B. Stewart, and L.-C. Wang, "Defect-oriented testing and defect-part-level prediction," *IEEE Design Test Comput.*, vol. 18, no. 1, pp. 31–41, Jan.–Feb. 2001.
[6] P. Franco, W. D. Farwell, R. L. Stokes, and E. J. McCluskey, "An experimental chip to evaluate test techniques chip and experiment design," in *Proc. Int. Test Conf.*, 1995, pp. 653–662.
[7] J. Geuzebroek, E. J. Marinissen, A. Majhi, A. Glowatz, and F. Hapke, "Embedded multi-detect ATPG and its effect on the detection of unmodeled defects," in *Proc. IEEE Int. Test Conf.*, Oct. 2007, pp. 1–10.
[8] M. R. Grimaila, S. Lee, J. Dworak, K. M. Butler, B. Stewart, H. Balachandran, B. Houchins, V. Mathur, J. Park, L.-C. Wang, and M. R. Mercer, "REDO-random excitation and deterministic observation-first commercial experiment," in *Proc. IEEE VLSI Test Symp.*, Apr. 1999, pp. 268–274.
[9] X. Kavousianos and K. Chakrabarty, "Generation of compact test sets with high defect coverage," in *Proc. DATE*, 2009, pp. 1130–1135.
[10] S. Lee, B. Cobb, J. Dworak, M. Grimaila, and M. Mercer, "A new ATPG algorithm to limit test set size and achieve multiple detections of all faults," in *Proc. Design Autom. Test Eur.*, 2002, pp. 92–99.
[11] Y.-T. Lin, O. Poku, N. Bhatti, and R. Blanton, "Physically-aware n-detect test pattern selection," in *Proc. Design Autom. Test Eur.*, 2008, pp. 634–639.
[12] J. Nelson, J. Brown, R. Desineni, and R. Blanton, "Multiple-detect ATPG based on physical neighborhoods," in *Proc. Design Autom. Conf.*, 2006, pp. 1099–1102.
[13] I. Pomeranz and S. M. Reddy, "A measure of quality for n-detection test sets," *IEEE Trans. Comput.*, vol. 53, no. 11, pp. 1497–1503, Nov. 2004.
[14] I. Pomeranz and S. M. Reddy, "Worst-case and average case analysis of n-detection test sets," in *Proc. Design Autom. Test Eur.*, 2005, pp. 444–449.
[15] H. Tang, G. Chen, S. M. Reddy, W. Chen, J. Rajski, and I. Pomeranz, "Defect aware test patterns," in *Proc. Design Autom. Test Eur.*, 2005, pp. 450–455.
[16] S. Venkataraman, S. Sivaraj, E. Amyeen, S. Lee, A. Ojha, and R. Guo, "An experimental study of n-detect scan ATPG patterns on a processor," in *Proc. VLSI Test Symp.*, 2004, pp. 23–28.
[17] B. Vermeulen, C. Hora, B. Kruseman, E. Marinissen, and R. Rijsinge, "Trends in testing integrated circuits," in *Proc. Int. Test Conf.*, 2004, pp. 688–697.
[18] Z. Wang and K. Chakrabarty, "Test-quality/cost optimization using output-deviation-based reordering of test patterns," *IEEE Trans. Comput.-Aided Design*, vol. 27, no. 2, pp. 352–365, Feb. 2008.
[19] M. Yilmaz, K. Chakrabarty, and M. Tehranipoor, "Test-pattern selection for screening small-delay defects in very-deep submicrometer integrated circuits," *IEEE Trans. Comput.-Aided Design*, vol. 29, no. 5, pp. 760–773, May 2010.
[20] *IWLS'05* [Online]. Available: http://www.iwls.org/iwls2005/benchmarks.html
[21] X. Kavousianos and K. Chakrabarty, "Generation of compact single-detect stuck-at test sets targeting unmodeled defects," Dept. Electric. Comput. Eng., Duke Univ., Durham, NC, Tech. Rep. ECE-2010-02 [Online]. Available: http://dukespace.lib.duke.edu/dspace/handle/10161/2846