Generation of Compact Single-Detect Stuck-At Test Sets Targeting Unmodeled Defects

Xrysovalantis Kavousianos and Krishnendu Chakrabarty*

***Abstract*- We present 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 pattern repository, by using a new output deviation-based metric. Then it embeds complete coverage of stuck-at faults within these patterns, and also uses the proposed metric to further improve their unmodeled defect coverage. Simulation results are presented for ISCAS and IWLS benchmark circuits by using two surrogate fault models, the transition-delay and the bridging fault model, respectively, to measure defect coverage. The results show that the proposed method provides considerably higher transition-fault coverage and coverage ramp-up compared to another recently published method with similar test length. Moreover, in all cases, the proposed method either outperforms or is as effective as the competing approach in terms of bridging-fault coverage. In many cases, higher transition-fault coverage is obtained even than much larger *N*-detect test sets for several values of *N*. Finally, our 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*—Multi-detect testing, automatic-test-pattern-generation, defect-oriented testing.**

I. INTRODUCTION

Manufacturing test has became a major challenge in very-deep submicron (VDSM) process technologies. It is impossible to explicitly target every possible defect in VDSM while at the same time mandated product-quality levels must be ensured by screening all defective devices before they are shipped. To facilitate defect detection and manage test complexity, abstract fault models are used in practice to mimic the behavior of real defects. The most widely used fault model in industry today continues to be the single stuck-at fault model, as it offers a number of advantages—it is simple, it requires low computational effort for test generation, and test patterns for single stuck-at faults also detect many physical defects. However, even though coverage of all or a large percentage of single stuck-at faults is considered to be indispensable for any realistic test-development flow today, it does not always guarantee high defect coverage [13].

The inadequacy of the stuck-at model for achieving high defect coverage has led to the development of new fault models, such as transition-delay faults, bridging faults, transistor stuck-open faults and transistor stuck-on faults, etc. These fault models reflect the behavior of many realistic defects more accurately than the stuck-at fault model, at the cost however of increased complexity which hinders their widespread adoption. For example, the automatic test-pattern generation (ATPG) algorithms associated with these fault models require excessive computational effort. Moreover, these models lead to prohibitively high pattern counts, thereby leading to high test application times. Finally, many of these fault models require detailed layout information, which is available only at the latter stages of the chip design flow. As a result, test development becomes a significant contributor to delays in chip tape-out. Moreover, there are many new defect types, which cannot be accurately modeled using existing fault models [19].

An alternative approach that increases defect coverage, and at the same time benefits from the low complexity of simple fault models, is multi-detect testing, also referred to as *N*-detect testing. This method was proposed in [12], and since then it has been identified as a very effective test strategy [1]-[6], [8], [10], [11], [14]-[18]. The main idea of *N*-detect testing is to apply N ($N > 1$) 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. An important property of *N*-detect testing is that it employs conventional fault models (usually the stuck-at fault model), therefore it can be easily incorporated in existing ATPG algorithms and tools.

The major drawback of *N*-detect testing is that the size of the test set increases linearly with N in order to provide test patterns of high quality. This adversely affects both the test data volume and the test application time of the generated test sets. An alternative method was recently proposed in [7] that offers compact test sets with high coverage of unmodeled defects. Specifically, the method proposed in [7] 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. Thus, the test quality of the resulting vectors increases, while at the same time their volume remains low.

In this paper, 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. In contrast to [7], we do not embed multi-detection of stuck-at faults within single-detect stuck-at patterns. Instead, we embed single-detection of stuck-

*The work of K. Chakrabarty was supported in part by the Semiconductor Research Corporation under contract no. 1588. A preliminary version of a part of this paper was presented in DATE 2009.

X. Kavousianos is with the Dept. of Computer Science, University of Ioannina, Ioannina, 45110 Greece (corresponding author: +302651008870; fax: +302651008883; e-mail: kabousia@cs.uoi.gr).

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

at faults within a small number of the most-efficient test patterns selected from an N-detect pattern repository. These patterns are appropriately selected from the repository to guarantee high un-modeled defect coverage and in most of the cases they also detect the vast majority of the stuck-at faults. The detection of the remaining stuck-at faults is embedded within the 'X' values of the selected patterns and, if necessary, a few top-off patterns are generated.

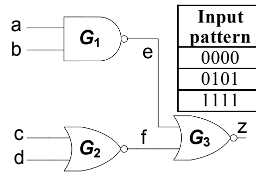
The proposed method utilizes a new test pattern-evaluation metric based on output deviations [21], to identify the most effective test patterns from the repository. Output deviations offer an effective probabilistic means to successfully identify the most effective test patterns, without being biased towards any particular fault model. As shown in [20], [22] unbiased testing provides higher test quality than a test method that is biased by a particular fault model. The proposed output deviation-based metric is more effective than the metric proposed in [21] because a) it selects test patterns in such a way as to achieve a weighted distribution of high deviation values at circuit outputs and b) it exploits the circuit structure to favor 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. Thus it is more effective than the metric proposed in an earlier version of this paper presented in [9], which generates different test sets to target each kind of these defects.

Simulations results for the ISCAS and IWLS benchmark circuits [23] 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, by analyzing the multi-detection profile of the proposed test-sets we show that simply increasing the value of N for N-detection, which is currently common industry practice, is not necessarily the best approach to enhance defect coverage. Instead, combining N-detection with pattern selection based on output deviations appears to be the most promising defect-screening strategy with low pattern counts.

II. Output Deviations

Output deviations are probability measures at primary outputs, as well as pseudo-outputs for full-scan designs (all denoted here as circuit outputs), that reflect the likelihood of error detection at these outputs. As shown in [21], test patterns with high deviations tend to be more effective for fault detection.

Output deviations are based on a probabilistic fault model, in which a probability map (referred to as 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 and a single output 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



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

R_1 (NOR) = (0.8 0.9 0.9 0.9)
R_2 (NAND)= (0.9 0.9 0.9 0.8)

Fig 1. An output-deviation calculation example.

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:

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

Likewise, the signal probabilities can be easily computed for other gate types. 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* $\Delta_{Gi,j}$ of G_i for t_j is defined as $p_{G_i,\bar{d}}$, 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 deviation values at the circuit outputs are indicative of the probability arbitrary defects to be detected at these 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.

Example 1. Fig. 1 shows a circuit consisting of three gates G_1, G_2, and G_3, with two different confidence level vectors (R_1 and R_2) assigned to the NOR and NAND gates (for the selection of the appropriate confidence-level vector the reader is referred to [21]). The first column of the Table in Fig. 1 presents three test patterns and their respective fault free value at output z. The next six columns present the signal probabilities computed at the internal circuit nodes using the aforementioned confidence level vectors. For each input pattern, the output-deviation is the probability that the output 'z' is complementary to the fault-free value, which is shown in bold in the two last columns of the Table. Note that the probability that an arbitrary defect causes an error at the output 'z' is higher when the last test pattern is applied (output deviation is equal to 0.396 in this case) compared to the other two patterns (output deviation is equal to 0.114 and 0.163, respectively). Therefore, the last test pattern (a, b, c, d) = (1, 1, 1, 1) is the most promising one for detecting defects as it provides the highest output deviation value. ■

III. Proposed Test-Generation Method

The flow of the proposed test generation method is shown in Fig. 2(a). In the following discussion, we explain each step in detail. A *test cube* is a test pattern that contains 0, 1 and don't-care ('X') logic values, and a *test vector* is a test pattern without X logic values.

Step 1.

In the first step, a repository of test cubes is generated, us-

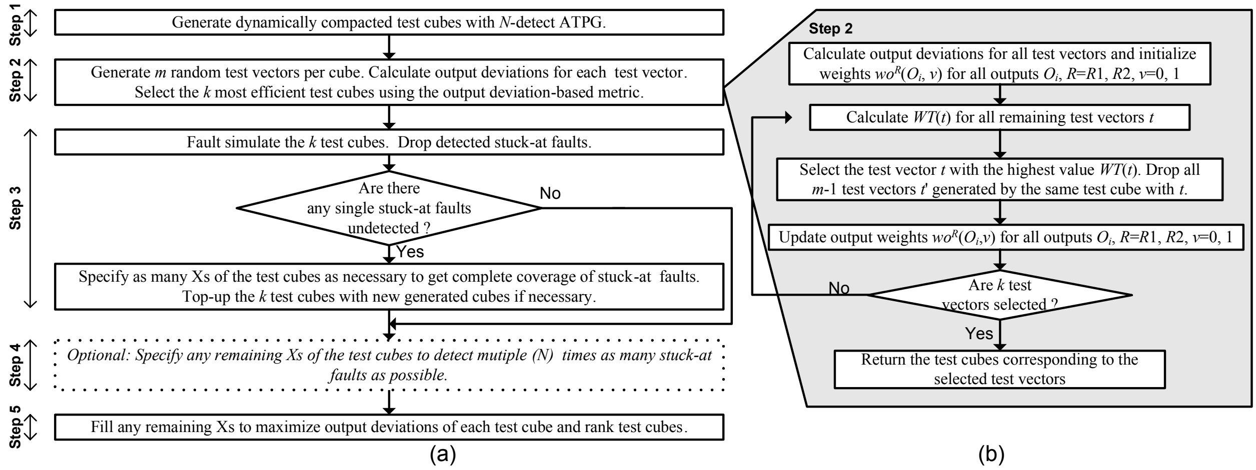


Fig. 2. Proposed test-generation flow.

ing 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 at the next step a subset of the most efficient (in terms of defect coverage) ones. This subset 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 effectively identified by the output deviation-based 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 the second step, the generated test cubes are evaluated according to the proposed output deviation-based metric. Since output deviations are not defined in [21] for test patterns containing Xs (i.e., test cubes), 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 an output deviation-based metric which will be described shortly. Note that the m random test vectors generated for each cube are used only for evaluating the respective test cube, and they are discarded afterwards. Eventually, the k most effective test vectors that correspond to k different test cubes are identified and the respective k test cubes are selected and form the basis for generating the test set (k and m are user-defined parameter).

In order to effectively identify the test vectors offering the highest unmodeled defect coverage, 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 detecting timing-dependent defects, we assume that each stuck-at test vector is applied at the circuit using the Launch-On-Capture (LOC) technique (also referred to as broadside testing) which is the most commonly used technique in industry (other two-pattern testing techniques can be straightforwardly used). According to LOC technique, the second response (denoted hereafter as $R2$) of each stuck-at test vector is used to detect timing-dependent defects. The proposed metric evaluates both responses $R1$, $R2$ of each test vector at the same time, and thus selects the most effective test vectors for detecting both types of 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 NO observable outputs, and the set L of stuck-at test vectors. Each test vector $t \in L$ is applied on the circuit and two responses are captured in the scan chain: the immediate response t^{R1} and the second response t^{R2} which, as mentioned earlier, is generated according to the LOC technique. Let $t_{O_i}^{R1}$, $t_{O_i}^{R2}$ be the fault free values at output O_i, ($i \in [1, NO]$) at the first and second response of test vector t respectively, and let $D(t_{O_i}^{R1})$, $D(t_{O_i}^{R2})$ be the deviation value at this output at the first and second response of test vector t respectively. At first the metric calculates the maximum deviation value $MD^R(O_i, v)$ at output O_i when the fault free logic value is equal to v at response R (hereafter R will refer to either $R1$ or $R2$ response) using the following formula :

$$MD^R(O_i, v) = max\{D(t_{O_i}^R): t \in L,\ t_{O_i}^R = v\} \quad \text{for } R = R1, R2,\ i \in [1, NO],\ v = 0, 1 \tag{1}$$

Note that the above 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 $R1$, $R2$. For example, $MD^{R2}(O_i, 0)$ is the maximum among the deviation values at the output O_i, provided by any test vector in set L producing fault free logic value 0 at this output at the second response $R2$. Calculating four different maximum deviation values for each output reflects the fact that a) different defects are observable at the two responses $R1$, $R2$, at each output and b) different defects are usually observable at the same output and the same response (first $R1$ or second $R2$) by patterns that produce different fault free logic values at this output. Thus, in total, 4xNO maximum deviation values are calculated.

The maximum deviation values are used to determine the boundary between those output deviation values that are considered high and those that are not. This step is important as only the high output-deviation values are considered for selecting the most efficient test cubes (the remaining deviation values are discarded). According to the proposed metric, any deviation value $D(t_{O_i}^R)$ is high if it is in the range $(1 - Thr)\cdot MD^R(O_i, t_{O_i}^R) \leq D(t_{O_i}^R) \leq MD^R(O_i, t_{O_i}^R)$ where $0 \leq Thr \ll 1$. *Thr* is a real-valued quantity used as the threshold value between low and high output deviation values. By setting *Thr* to 0, only test vectors that offer the maximum deviation values are identified as being effective for detecting defects. Thus the screening process is made strict by discarding all test vectors that offer output deviation values lower than the maximum ones. This strategy adversely impacts the effectiveness of the selection process as it may exclude test vectors that offer near-maximum output deviation values and which may be very effective in detecting defects. To avoid this problem, the value of *Thr* should be set above the value of 0. However, a large value of *Thr* (i.e., a value close to 1) should also be avoided as it may degrade the quality of the resulting test set by selecting test vectors with low output deviation values, which are thus less effective for detecting defects. Therefore a value higher but close to 0 is the most appropriate one for parameter *Thr*. We experimentally verified that a value of *Thr* in the range [0.005, 0.01] provides high-quality vectors, and thus for the experiments reported here, we set $Thr = 0.005$ (higher values of *Thr* can be also used).

The second objective of the metric is to evaluate each test vector by considering also the volume of defects that can be potentially detected at each of the circuit outputs which have high deviation values. To achieve a good measure of this volume, the circuit structure is taken into account by using a very simple observation. The volume of defects that can be detected at a circuit output which is driven by a large logic cone is likely to be higher than the corresponding volume at the circuit output which is driven by a small logic cone. We can therefore measure the volume of defects that can be potentially detected at any circuit output as the number of lines in the logic cone driving this output. Of course, this measure is not an actual measure of defects, but it is used to compare different test vectors and select the most effective one. For example, among two vectors which provide high deviation values at two different outputs, we favor the vector with high deviation value at the output driven by the largest logic cone. 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. Then, for each test vector $t \in L$, a weight $WT^R(t)$ is calculated for each of the two responses $R1$, $R2$ as the sum of the weights $wo^R(O_i, t_{O_i}^R)$ of all outputs O_i, $1 \leq i \leq NO$, with high deviation values (note that $t_{O_i}^R = 0$ or 1). Thus, for $R = R1$ and $R = R2$ we have (we set $Thr' = 1\text{-}Thr$):

$$WT^R(t) = \sum_{\substack{i:t_{O_i}^R=0,\\ D(t_{O_i}^R)\geq Thr'\cdot MD^R(O_i,0)}} wo^R(O_i,0) + \sum_{\substack{i:t_{O_i}^R=1,\\ D(t_{O_i}^R)\geq Thr'\cdot MD^R(O_i,1)}} wo^R(O_i,1)$$

The weight of test vector t is finally calculated as

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

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

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 in order to offer observability to as many defects as possible. Note that by using formula (2) the selection is biased towards test vectors that increase the deviations at a subset of outputs (the ones with the highest weights which are driven by the largest cones), which may adversely affect the observability for circuit nodes which do not belong in the logic cones driving these outputs. Moreover, 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, in order to enhance the effectiveness of the selection process, every time a test vector t is selected that provides high deviation value at output O_i at response R, the weight $wo^R(O_i, t_{O_i}^R)$ is divided by a constant factor *DF* (note that $t_{O_i}^R = 0$ or 1). In this way the selection process is enforced to gradually select test vectors which offer high deviation value at all outputs but in a weighted fashion (i.e. outputs of large logic cones are still favored compared to outputs of small logic cores). The value of *DF* is a parameter which 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 experimentally that a value of *DF* in the range [2, 10] is sufficient to guarantee the selection of test vectors which provide high deviation values at all outputs. We have chosen the value of $DF = 8$ in the experiments reported in Section IV. Higher values of *DF* can be also used.

Note that the proposed output deviation-based metric is more efficient than the metric proposed in [21] as it combines the notion of output deviations with structural information of the circuit and also attempts to select patterns which offer a weighted distribution of high deviation values at all outputs. In addition, it is more efficient than both the metric proposed in [21] and the metric proposed in an earlier version of the paper [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.

The flow for the selection process of step 2 is highlighted in Fig. 2(b). At each iteration the test vector t with the highest weight $WT(t)$ is identified and the corresponding test cube is selected. Subsequently, 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).

Let us illustrate the selection of test cubes using the proposed output deviation-based metric with an illustrative example.

	First Response (R1)										Second Response (R2)										Maximum Deviations			
	$t = t_1$		$t = t_2$		$t = t_3$		$t = t_4$		$t = t_5$		$t = t_1$		$t = t_2$		$t = t_3$		$t = t_4$		$t = t_5$		$MD^{R1}(O_i, t^{R1}_{Oi})$		$MD^{R2}(O_i, t^{R2}_{Oi})$	
	t^{R1}_{Oi}	$D(t^{R1}_{Oi})$	t^{R1}_{Oi}	$D(t^{R1}_{Oi})$	t^{R1}_{Oi}	$D(t^{R1}_{Oi})$	t^{R1}_{Oi}	$D(t^{R1}_{Oi})$	t^{R1}_{Oi}	$D(t^{R1}_{Oi})$	t^{R2}_{Oi}	$D(t^{R2}_{Oi})$	t^{R2}_{Oi}	$D(t^{R2}_{Oi})$	t^{R2}_{Oi}	$D(t^{R2}_{Oi})$	t^{R2}_{Oi}	$D(t^{R2}_{Oi})$	t^{R2}_{Oi}	$D(t^{R2}_{Oi})$	$t^{R1}_{Oi} = 0$	$t^{R1}_{Oi} = 1$	$t^{R2}_{Oi} = 0$	$t^{R2}_{Oi} = 1$
O_1	0	**0.6**	1	0.2	1	**0.4**	0	0.3	1	**0.41**	0	0.4	1	0.2	0	**0.5**	1	0.3	1	**0.68**	0.6	0.41	0.5	0.68
O_2	1	**0.5**	0	**0.5**	1	0.3	1	0.1	0	0.2	1	**0.63**	0	**0.33**	1	**0.62**	1	0.2	0	**0.32**	0.5	0.5	0.33	0.63
O_3	0	0.3	1	0.2	0	0.1	1	**0.4**	0	**0.41**	1	**0.61**	0	0.35	1	**0.6**	0	**0.51**	1	**0.62**	0.41	0.4	0.51	0.62

(a)

Initial Output Weights

	$wo^{R1}(O_i,t^{R1}_{Oi})$		$wo^{R2}(O_i,t^{R2}_{Oi})$	
	$t^{R1}_{Oi} = 0$	$t^{R1}_{Oi} = 1$	$t^{R2}_{Oi} = 0$	$t^{R2}_{Oi} = 1$
O_1	28	28	28	28
O_2	54	54	54	54
O_3	32	32	32	32

$WT^{R1}(t_1) = 28 + 54 + 0 = 82$

$WT^{R2}(t_1) = 0 + 54 + 32 = 86$

(b)

Initial Test Vector Weights

	$WT^{R1}(t)$	$WT^{R2}(t)$	$WT(t)$
$t = t_1$	82	86	168
$t = t_2$	54	54	108
$t = t_3$	28	114	142
$t = t_4$	32	32	64
$t = t_5$	60	114	**174**

(c)

Selection of t_5

Updated Output Weights

	$wo^{R1}(O_i,t^{R1}_{Oi})$		$wo^{R2}(O_i,t^{R2}_{Oi})$	
	$t^{R1}_{Oi} = 0$	$t^{R1}_{Oi} = 1$	$t^{R2}_{Oi} = 0$	$t^{R2}_{Oi} = 1$
O_1	28	14	28	14
O_2	54	54	27	54
O_3	16	32	32	16

$WT^{R1}(t_1) = 28 + 54 + 0 = 82$

$WT^{R2}(t_1) = 0 + 54 + 16 = 70$

(d)

New Test Vector Weights

	$WT^{R1}(t)$	$WT^{R2}(t)$	$WT(t)$
$t = t_1$	82	70	152
$t = t_2$	54	27	81
$t = t_3$	14	98	112
$t = t_4$	32	32	64

(e)

Fig. 3. Example of proposed metric calculation

Example 2. Fig. 3 presents the selection of test cubes using the proposed output deviation-based metric. We consider a hypothetical circuit with $NO = 3$ observable outputs O_1, O_2 and O_3, which are driven by logic cones consisting of 28, 54 and 32 lines respectively. We assume that the N-detect test set generated for this circuit at step 1 consists of 5 test cubes $c_1, \ldots, c_5$ among which two must be selected (i.e., $k = 2$). For simplicity we assume that $m = 1$, i.e., the Xs of each test cube are randomly filled only once, generating thus set L which consists of 5 test vectors, $L=\{ t_1, \ldots, t_5 \}$ (note that test vector t_i corresponds to test cube c_i). The table of Fig. 3a presents the fault free response $t^R_{O_i}$ and the output deviation value $D\left(t^R_{O_i}\right)$ of each test vector at each output O_i for the first response $R = R1$ (columns 2-6) as well as the second response $R = R2$ (columns 7-11). The maximum output deviation values $MD^R(O_i, t^R_{O_i})$ for each fault free logic value $t^R_{O_i} = 0, 1$ at both responses are shown in the rightmost part of this table. The output deviation values which are near-maximum (we assume that $Thr = 0.1$) are shown boldfaced for each test vector at columns 2-11. Fig. 3b presents the output weights which are initially set equal to the volume of lines at the logic cones which drive outputs O_1, O_2 and O_3 (note that initially the weight of each output is the same for both fault free logic values at both responses $R1$, $R2$). Fig. 3c presents the weights of test vectors $t_1, \ldots, t_5$ which are calculated using output weights $wo^R(O_i, t^R_{O_i})$. For example, the weight $WT^{R1}(t_1)$ is calculated as follows: at the first response $R1$, test vector t_1 exhibits near-maximum deviation values at outputs O_1 for fault free logic value $t^{R1}_{O_1} = 0$, and O_2 for fault free logic value $t^{R1}_{O_2} = 1$ (see Fig. 3a). The weight $WT^{R1}(t_1)$ is calculated as the sum of the respective output weights shown in Fig. 3b: $wo^{R1}(O_1, 0) = 28$, $wo^{R1}(O_2, 1) = 54$. Thus $WT^{R1}(t_1) = 54 + 28 = 82$. In the same way we calculate $WT^{R2}(t_1) = 86$. According to equation (2) we have $WT(t_1) = WT^{R1}(t_1) + WT^{R2}(t_1) = 168$. The weights $WT(t)$ for the rest test vectors are calculated in the same way (see Fig. 3c). Test vector t_5 has the largest weight and thus test cube c_5 is selected and test vector t_5 is removed from set L. After the selection of test vector t_5, the weights of the outputs which correspond to the near-maximum output deviation values of test vector t_5 (shown underlined at Fig 3d) are divided by DF (we assume that $DF = 2$ in this example). Using the updated output weights $wo^R(O_i, t^R_{O_i})$, we calculate the weights for the remaining test vectors $t_1, \ldots, t_4$ (see Fig. 3e) in the same way as it was described previously. Based on the new test vector weights, t_1 has the highest weight and thus test cube c_1 is selected. Thus, among the five test cubes $c_1, \ldots, c_5$, cubes c_5 and c_1 are selected. ■

Step 3.

The purpose of the next step is to guarantee that the selected test cubes achieve complete coverage of single stuck-at faults. Therefore, we perform stuck-at fault simulation with the selected test cubes and we drop every stuck-at fault that is detected at least once. Then, we specify the 'X' values of the selected test cubes in order to detect as many undetected stuck-at faults as possible, and if necessary, we generate additional top-off test cubes. Note that the additional test cubes are not selected from the repository but they are generated using a new ATPG step with the dynamic compaction option turned on, in order to minimize their volume.

Step 4.

This step is optional. Any remaining Xs are specified in order to achieve multiple detections of as many stuck-at faults as possible, as suggested in [7]. This step is motivated by the

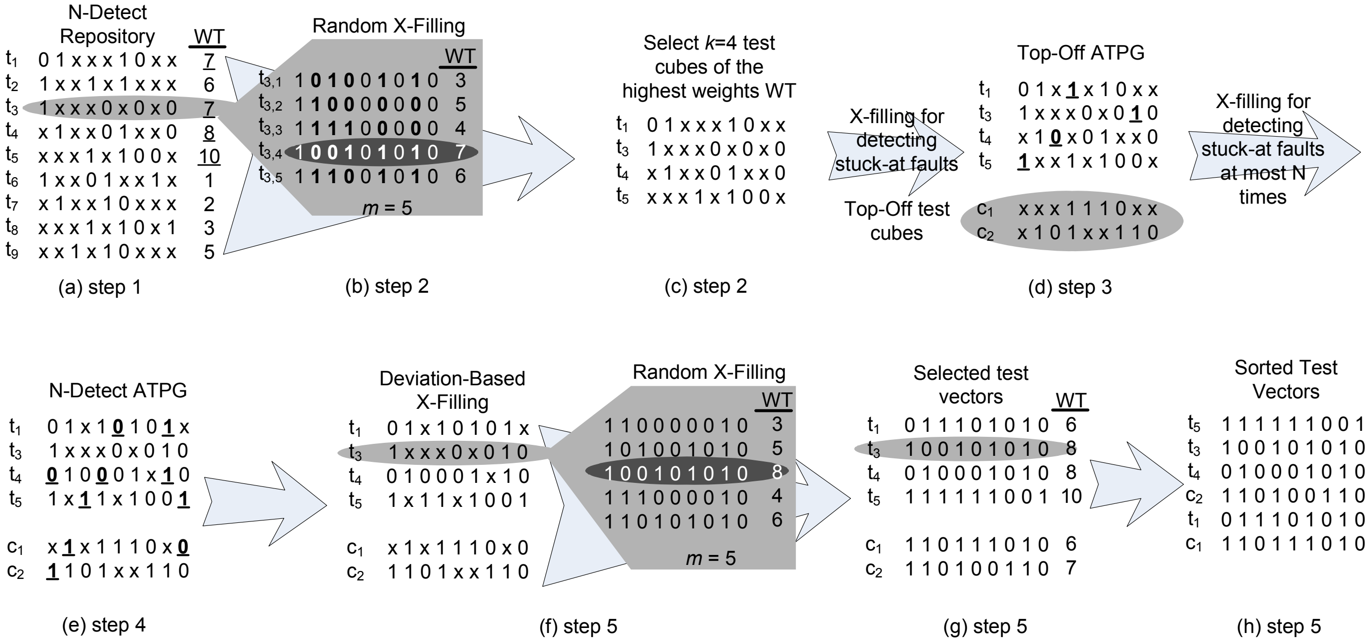


Fig. 5. Example of the proposed test-generation flow.

effectiveness of the method proposed in [7]. As shown in section IV, this step further improves the defect coverage of the generated test sets in many cases.

Step 5.

At this step, all remaining Xs, are specified in such a way as to maximize the effectiveness of test patterns according to the proposed metric. To this end, a similar process with the selection of test cubes from the repository described in step 2 is applied. The flowchart of this process is shown in Fig. 4. At 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 $R1$, $R2$ as well as the $MD^R(O_i, v)$ for $v = 0, 1$, $i \in [1, NO]$ values are calculated. Subsequently, the high output deviation values are identified (the rest ones are discarded) and the weights $wo^R(O_i, v)$ for $i \in [1, NO]$, $v = 0, 1$ and $R = R1, R2$ are initialized again to the volume of lines in the logic cone of output O_i. Then, an iterative process is applied, which selects at each iteration the test vector with the highest weight WT calculated using formula (2). When a test vector is selected, the remaining $m-1$ vectors generated by the same test cubes are discarded. This process terminates when one test vector has been selected for each test cube.

The following example illustrates the whole test generation process.

Example 3. Fig. 5 presents an example of the test generation process for a hypothetical circuit. Fig. 5a presents the generated N-detect repository consisting of 9 test cubes $t_1 \ldots t_9$, and the corresponding weights WT calculated using the proposed output-deviation based metric during the application of step 2 (see Example 2). Note that in order to calculate weights WT, the unspecified bits of each test cube are randomly filled $m = 5$ times as it is shown in Fig. 5b (the boldfaced bits correspond to the randomly filled unspecified bits). The maximum weight achieved for any of the m random fillings for each test cube is considered as the weight of the test cube. For example, in the case of t_3 the maximum weight is achieved by the fourth random filling, i.e., $t_{3,4}$ and thus $WT(t_3) = 7$. The $k = 4$ test cubes with the highest weights (i.e., t_1, t_3, t_4, t_5) are selected as shown in Fig. 5c. Fig. 5d presents the application of step 3 on these test cubes. The underlined bits of t_1, t_3, t_4 and t_5 correspond to unspecified bits which are specified by the ATPG process to detect any stuck-at faults which are not already detected by these test cubes. Additionally, test cubes c_1, c_2 are generated in order to detect the remaining undetected stuck-at faults which are not detected by test cubes t_1, t_3, t_4 and t_5. Subsequently, the remaining Xs are exploited in order to detect N-times as many stuck-at faults as possible according to [7]. This process is shown in Fig. 5e (the Xs specified for this purpose are shown

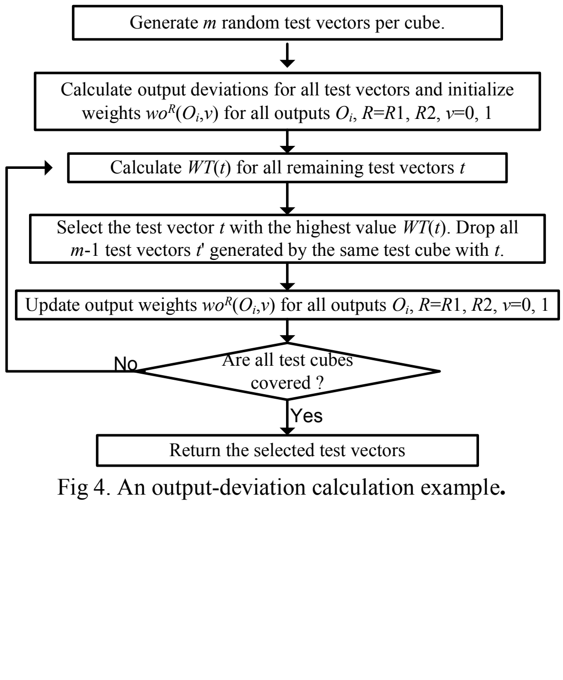


Fig 4. An output-deviation calculation example.

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	21K	106	111	560
	s38584	38	1426	23K	164	170	1049
IWLS'05	**sytemcaes**	258	670	17K	211	220	621
	tv80	13	359	13K	640	672	3577
	usb_funct	112	1746	23K	129	123	964
	ac97_ctrl	54	2199	24K	53	60	393
	mem_ctrl	116	1078	22K	577	608	2680
	pci_bridge32	159	3358	38K	203	215	1610
	ethernet	93	10544	136K	1110	1117	7491

boldfaced and underlined). For each of the 6 test cubes generated at the previous steps m = 5 candidate test vectors are generated by randomly filling the remaining unspecified bits of each test cube. Then, the proposed output-deviation based metric is used to calculate a weight WT for each candidate test vector as it is shown in Fig. 5f. The best candidate vector (i.e., the one with the highest weight WT) is selected for each test cube as it is shown in Fig. 5g (note that the best candidate for test cube t_3 is highlighted in both Fig. 5f, 5g). Finally, the selected test vectors are sorted in descending order of weights, as shown in Fig. 5h. ■

IV. SIMULATION RESULTS

In this section, we evaluate the defect coverage of the proposed test-generation method. The test-generation flow (Fig. 2), excluding ATPG and fault simulation, was implemented in C. Commercial tools were used for all ATPG-related and fault simulation steps. For the experiments we used the largest ISCAS'89 and a subset of IWLS'05 benchmark circuits [23]. The basic characteristics of these circuits are shown in the first four columns of Table I. The first column presents the name of each circuit, the second and the third columns present the number of primary inputs and scan cells respectively, and the fourth column presents the number of gates of the combinational part of each circuit (note that the gates comprising the scan cells are not included in the gate count).

The quality of the proposed method, with respect to defect coverage, was evaluated using two surrogate fault models—the transition-delay fault model and the bridging fault model. Note that 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 of the proposed test generation method for detecting un-modeled defects. As we mentioned in Section III 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 the response to the stuck-at test). For detecting bridging faults, we used the immediate response to every stuck-at test vector.

In order to highlight the benefits of the proposed test-generation method, we compare it with traditional single-detect stuck-at ATPG, N-detect stuck-at ATPG as well as with the embedded multi-detect ATPG method proposed in [7]. For each experiment, the following four test sets were compared with each other:

Reg_SD: traditional (regular) dynamically compacted single-detect stuck-at test set, with the Xs specified randomly.

Pure_ND: traditional dynamically compacted N-detect stuck-at test sets.

Emb_ND: dynamically compacted single-detect stuck-at test set, with the Xs filled in such a way as to embed multi-detection of stuck-at faults (test cubes detect stuck-at faults as many times up to N as possible). The approach of [7] was implemented for this purpose.

Prop_SD: compact single-detect stuck-at test set generated by the proposed test-generation flow, with the Xs specified exclusively to maximize output deviation values (Step 4 is omitted).

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 the output deviation values.

At the first experiment we study the effect of parameter k on the defect coverage of the proposed *Prop_SD* method (we remind that k is the volume of test cubes selected from the N-detect repository at the second step of the test generation flow in order to form the basis for the generated test set). Hereafter we consider the parameter k as a percentage of the volume of test cubes of the traditional single-detect stuck-at test sets (*Reg_SD*). For this experiment we vary the value of k between 10% and 90% of the size of *Reg_SD* test set and we set N = 10 and m = 10. In Fig. 6 we present the transition fault coverage (left y-axis) as well as the size in number of test vectors (right y-axis) of the test set generated by applying the *Prop_SD* method for the representative benchmark circuit *systemcaes* (the rest of the benchmark circuits exhibit similar behavior). The x-axis presents the percentage of k used in each case. Note that for comparison reasons, we also include the transition fault coverage and the test set size for the *Reg_SD* test set (leftmost case in Fig. 6) and the *Pure_ND* test set for N = 10 (rightmost case in Fig. 6). The rest of the results correspond to the *Prop_SD* method. It is obvious that even for small values of k the proposed method provides much higher transition fault coverage than the *Reg_SD* method. Moreover, as k increases the transition fault coverage of the test set generated using proposed method increases considerably, compared to the *Reg_SD* case and approaches the transition fault coverage

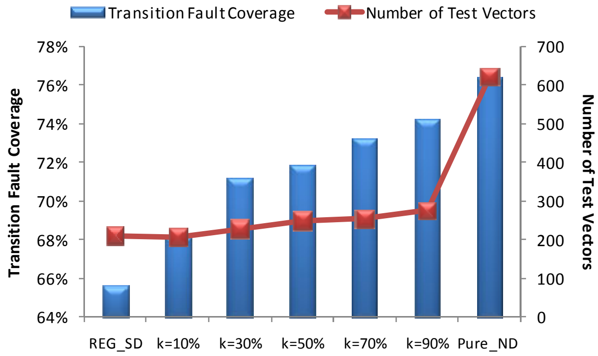


Fig 6. Effect of parameter k on *Prop_SD* case.

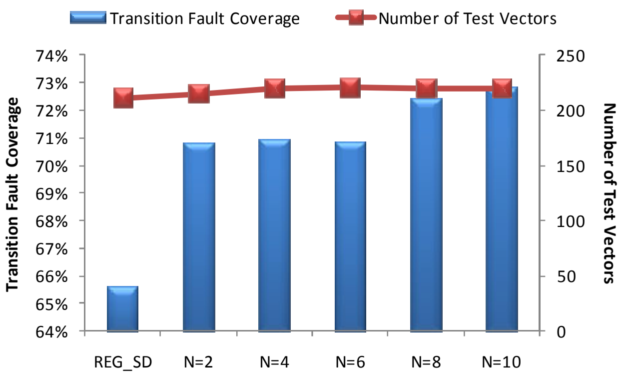


Fig 7. Effect of parameter *N*.

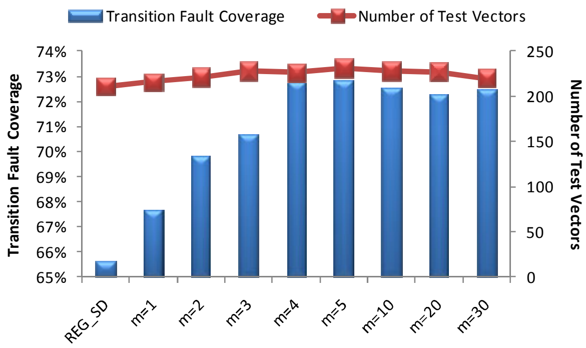


Fig 8. Effect of parameter *m* on Prop_SD case.

achieved by the large *N*-detect repository (*Pure_ND* method reported as the rightmost case in Fig. 6). When the value of *k* is small the size of the *Prop_SD* test set is similar or even smaller (see the case of *k* = 10% in Fig. 6) to the size of *Reg_SD* case. As *k* increases the size of the generated test set increases too, because more test patterns are selected from the *N*-detect repository. However, the size of the test set increases very slowly as *k* increases, because the additional test patterns selected from the repository tend to detect most of the undetected stuck-at faults at step 3, and thus they are counterbalanced by the reduced number of top-off test patterns generated for providing complete coverage of stuck-at faults. Moreover, the proposed *Prop_SD* method approaches the defect coverage offered by *Pure_ND* at a significantly lower volume of test vectors (note the large difference between the test set size of the *Prop_SD* method for all values of *k*, and the size of the *Pure_ND* test set reported at Fig. 6). Thus we conclude that the proposed method effectively identifies the most efficient test patterns from the *N*-detect repository and offers a trade-off between pattern count and test quality by appropriately selecting the value of *k*.

In the second experiment we study the effect of parameter *N* used for generating the repository, on the defect coverage of the proposed method. To this end we generated various repositories for *N* = 2, 4, 6, 8 and 10 and we applied the proposed method *Prop_SD* for *k* = 30% and *m* = 10 for the representative benchmark circuit *systemcaes*. The results are shown in Fig. 7. The lest y-axis presents the transition fault coverage, the right y-axis presents the size of the generated test sets and the x-axis presents the values of *N* used for the experiments. As in the previous experiments for comparison reasons we also append the transition fault coverage and the test set size of the original *Reg_SD* case (leftmost case in the chart). It is obvious that as *N* increases test sets of higher quality are generated with no adverse impact on the size of the test set (note that in all cases the same volume of test cubes are selected from the repository). The reason is that as *N* increases the quality of the test patterns comprising the repository improves and these patterns are effectively identified and selected by the proposed output deviation-based metric. Thus, we conclude that the value of *N* for the repository should be as large as it is computationally feasible in order to provide test sets of high quality.

In the next experiment, we study the effect of the value of *m* on the defect coverage of the generated test sets (recall that *m* is the volume of random fillings applied at each test cube during Step 2 for evaluating each test cube as well as during Step 5 for filling the remaining Xs of the selected test cubes). We again apply the proposed method for the benchmark circuit *systemcaes* and we set *N* = 10 and *k* = 30%. The results are shown in Fig. 8. The left y-axis presents the transition fault coverage, the right y-axis presents the size of the generated test sets and the x-axis presents the values of *m* used for the experiments. As in the previous cases, the leftmost reported results correspond to the *Reg_SD* case. It is obvious that as *m* increases, the transition fault coverage increases while at the same time the size of the test set is not affected. Note that the slightly non-monotonic nature of the fault coverage in Fig. 8 is expected due to the random selection involved in some of the steps.

From the above three experiments we conclude 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%. Additionally, we assume the value *N* = 10 for the *Emb_ND* method proposed in [7] in order to ensure a fair comparison with this method.

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 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*). As shown in Table I, 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 as we mentioned earlier, 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 the vast majority of the cases, *Emb _ND*, *Prop_SD*, and *Prop_ND* provide significantly higher transition-fault

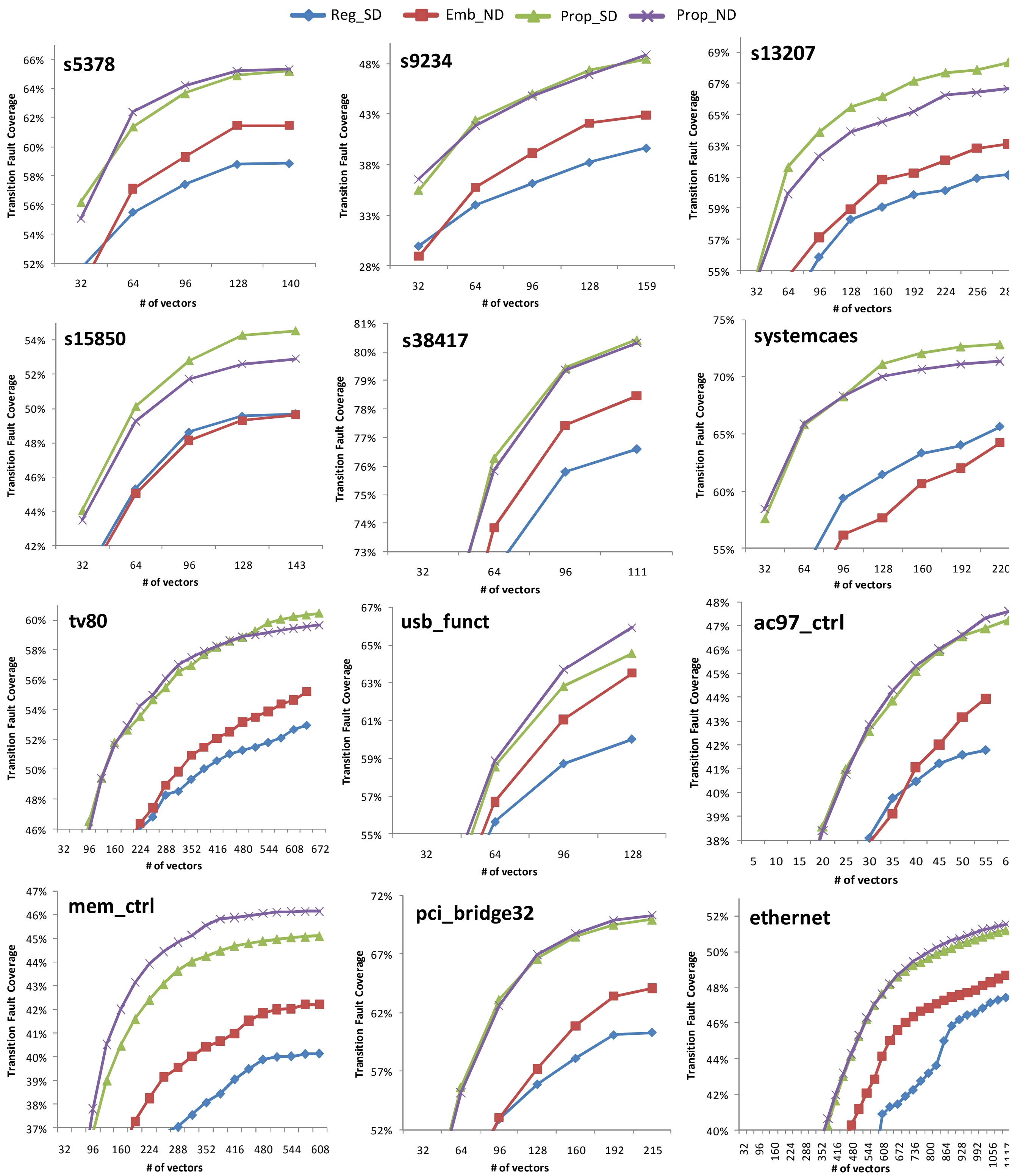


Fig. 9. Transition-fault coverage ramp-up.

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* generated using the method proposed in [7]. In more than half of the cases, the highest coverage is provided by the *Prop_ND* test sets.

In Fig. 9 we present the transition fault coverage ramp-up for these methods. In each of the charts the x-axis presents the volume of test vectors applied and the y-axis the transition

TABLE II
TRANSITION FAULT COVERAGE, BCE+ VALUES AND BRIDGING FAULT COVERAGE (%)

	Transition Fault Coverage				Bridging fault coverage							
					BCE+				Random Bridging Faults			
	Reg_SD	Emb_ND	Prop_SD	Prop_ND	Reg_SD	Emb_ND	Prop_SD	Prop_ND	Reg_SD	Emb_ND	Prop_SD	Prop_ND
s5378	58.53	61.58	65.20	**65.33**	94.32	96.32	95.62	**96.60**	93.57	95.05	94.61	**95.41**
s9234	39.65	42.87	48.46	**48.85**	87.00	88.42	87.36	**88.68**	86.07	87.41	86.50	**87.86**
s13207	61.14	63.10	**68.36**	67.02	92.87	95.16	94.60	**95.91**	92.14	93.99	93.68	**94.62**
s15850	50.90	49.65	**54.55**	52.91	94.04	95.22	94.42	**95.35**	93.11	94.14	93.55	**94.40**
s38417	77.07	78.45	**80.42**	80.31	97.91	98.68	98.13	**98.77**	96.69	97.56	97.00	**97.77**
s38584	60.86	61.69	**62.38**	62.14	94.14	**96.33**	95.15	96.30	89.42	90.53	89.75	**90.65**
sytemcaes	65.62	64.24	**72.83**	71.35	98.05	98.52	98.28	**98.52**	95.37	95.80	95.69	**95.86**
tv80	52.95	55.20	**60.45**	59.65	90.95	91.89	91.60	**92.23**	88.85	89.45	89.41	**89.60**
usb_funct	60.01	63.64	64.57	**65.92**	93.17	**95.38**	93.46	95.13	94.73	**96.25**	95.08	96.09
ac97_ctrl	42.81	43.93	47.24	**47.60**	93.72	95.40	94.34	**96.15**	96.44	97.16	96.76	**97.66**
mem_ctrl	40.13	42.22	45.10	**46.15**	61.91	63.02	62.89	**63.54**	74.26	75.04	75.08	**75.48**
pci_bridge32	59.67	64.09	69.96	**70.31**	94.44	96.36	95.56	**96.76**	95.71	96.82	96.34	**97.00**
ethernet	47.44	48.69	51.21	**51.55**	88.79	89.69	89.20	**89.69**	90.53	**91.49**	90.81	91.33

fault coverage. It is clear that both the proposed methods provide higher ramp-up than the other methods, and thus they offer reduced test application time in an abort-at-first-fail environment.

In the next experiment, we study the multi-detection profile of each test set. We present results for three representative cases (the other benchmarks exhibit similar behavior). Each curve in Fig. 10 presents the percentage of stuck-at faults detected n times or more for n = 1, 2, …, 11. Our results show that a high degree of multi-detection is not always necessary for high defect coverage. For example the test set of the systemcaes benchmark circuit for the proposed method provides less multi-detection than the two baseline methods, yet it provides higher transition-fault coverage. In all three cases reported in Fig. 10, 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 (*Prop_ND*) offers the most promising solution.

Next, we compare the four test sets using the bridging fault model. For each generated test set, the BCE+ metric proposed in [17] was used as an estimate of bridging fault coverage:

$$BCE^{+} = \sum_{i=1}^{n} \frac{f_i^{sa-v}}{|F|} \cdot \left[\sum_{j=1}^{|S|} \frac{1}{|S|} \left(1 - \left(1 - p_{j,v} \right)^{i} \right) \right],$$

where v = 0, 1. The parameter f_i^{sa-v} refers to the number of stuck-at-0 faults (for v = 0) and stuck-at-1 faults (for v = 1) that are detected i times by the test vectors (n is the maximum number of detections for any stuck-at fault). $|S|$ is the number of circuit lines, $|F|$ is the total number of stuck-at faults and $p_{j,v}$ is the probability of signal j to receive the logic value 0 (for v = 0) and 1 (for v = 1). As noted in [17], BCE^{+} is not very accurate for estimating the real bridging fault coverage of a method, but it is very useful for comparing two different methods (the method with the highest value of BCE^{+} is deemed to be more effective for defect screening). Therefore, in addition to the calculation of the BCE^{+} values, 400K bridging faults was simulated as follows: 100K pairs of lines were selected randomly for each circuit, and four bridging faults were simulated for each pair by considering both lines as aggressors and victims, as well as by considering both AND and OR bridging faults. The results for the various test sets are shown in the last 8 columns of Table II. Columns 6-9 and 10-13 show the BCE+ measures and the random bridging fault coverage for all test sets, respectively. The best result is boldfaced in each case. We note that in most of the cases, the *Prop_ND* test set provides the best results, both in terms of the BCE+ measure and bridging fault coverage. Only in very few cases *Emb_ND* provides marginally higher bridging fault coverage

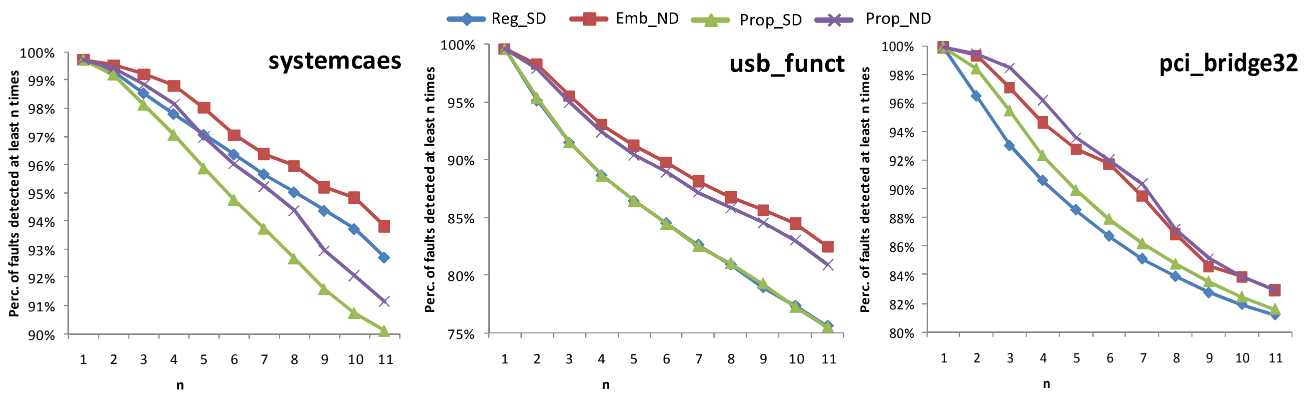


Fig. 10. Multi-detection results for the different test sets.

TABLE III

SMALLER TEST SETS ARE MORE EFFECTIVE THAN *N*-DETECT TEST SETS.

Circuit	N^*: Threshold on N	Test-Set Size		Size Reduction (%)
		N^*-detect	Proposed	
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%

and BCE+ values 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. The first column shows the name of each benchmark circuit. Columns 2, 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. CONCLUSIONS

We have presented a new method, which exploits a new output deviation-based metric to identify the most effective (in terms of un-modeled defect coverage) test patterns from an *N*-detect repository, and uses these test patterns to build a compact single-detect stuck-at test set which offers high defect coverage. Simulation results show that compact test sets can be generated for complete single stuck-at coverage which offer higher coverage of un-modeled defects compared to other methods. The effectiveness of the proposed method can be attributed to the combination of multi-detect ATPG and pattern selection based on 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. ITC, pp. 669-678, 2004.

[2] B. Benware, et. all., “Impact of Multiple-Detect Test Patterns on Product Quality”, in Proc. ITC, pp. 1031-1040, 2003.

[3] R. D. Blanton, K. Dwarakanath and A. Shah, “Analyzing the Effectiveness of Multiple-Detect Test Sets”, in Proc. ITC,pp. 876-885, 2003.

[4] E. J. McCluskey and C.-W. Tseng, “Stuck-Fault Tests vs. Actual Defects”, in Proc. ITC, pp. 336-343, 2000.

[5] J. Dworak, et. all., “Defect-Oriented Testing and Defect-Part-Level Prediction”, IEEE Design & Test of Computers, pp. 31-41, 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. ITC, pp. 653-662, 1995.

[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. ITC, 2007.

[8] M. R. Grimaila, et. al, “REDO – Random Excitation and Deterministic Observation – First Commercial Experiment”, in Proc. VTS, pp. 268-274, 1999.

[9] X. Kavousianos and K. Chakrabarty, "Generation of Compat Test Sets with High Defect Coverage", in Proc. DATE, pp. 1130-1135, 2009

[10] S. Lee, B. Cobb, J. Dworak, M.R. Grimaila and M.R. Mercer, “A New ATPG Algorithm to Limit Test Set Size and Achieve Multiple Detections of all Faults”, in Proc. DATE Conf., pp. 92-99, 2002.

[11] Y.–T. Lin, O. Poku, N. Bhatti and R. Blanton, “Physically-Aware N-Detect Test Pattern Selection”, in Proc. DATE Conf., pp. 634-639, 2008.

[12] S. Ma, P. Franco and E.J. McCluskey, “An Experimental Chip to Evaluate Test Techniques Experiment Results”, in Proc. ITC, pp. 663-672, 1995.

[13] P. Maxwell, R. Aitken, V. Johansen and I. Chiang, “The Effect of Different Test Sets on Quality Level Prediction: When is 80% better than 90%”, in Proc. ITC., pp. 358-364, 1991.

[14] J. Nelson, J. Brown, R. Desineni and R. Blanton, “Multiple-Detect ATPG Based on Physical Neighborhoods”, in Proc. DAC, pp. 1099-1102, 2006.

[15] I. Pomeranz and S. M. Reddy, “A Measure of Quality for n-Detection Test Sets”, IEEE Trans. Computers, vol. 53, No 11, pp. 1497-1503, 2004.

[16] I. Pomeranz and S. M. Reddy, “Worst-Case and Average Case Analysis of n-Detection Test Sets”, in Proc. DATE Conf., pp. 444-449, 2005.

[17] H. Tang et. al., “Defect Aware Test Patterns”, in Proc. DATE Conf., pp. 450-455, 2005

[18] S. Venkataraman et. al, “An Experimental Study of N-Detect Scan ATPG Patterns on a Processor”, in Proc. VTS, pp. 23-28, 2004.

[19] B. Vermeulen et al., “Trends in testing integrated circuits,” in Proc. ITC, 2004, pp. 688-697.

[20] L.-C. Wang, P.R. Mercer, S.W. Kao and T.W. Williams, “On the Decline of Testing Efficiency as Fault Coverage Approaches 100%”, in Proc. VTS, pp.74-83, 1995.

[21] Z. Wang and K. Chakrabarty, "Test-Quality/Cost Optimization Using Output-Deviation-Based Reordering of Test Patterns", IEEE Trans. CAD, vol. 27, No 2, pp. 352-365, 2008.

[22] Li-C. Wang, T. W. Williams, M. R. Mercer, "Using Target Faults to Detect Non-Target Defects", in Proc. ITC, pp. 629-638, 1996.

[23] IWLS’05 benchmark circuits [online] http://www.iwls.org/iwls2005/benchmarks.html