# Generation of Compact Test Sets with High Defect Coverage

Xrysovalantis Kavousianos
Dept. of Computer Science, University of Ioannina
45110 Ioannina, Greece
kabousia@cs.uoi.gr

Krishnendu Chakrabarty*
Dept. of Electrical & Computer Engineering, Duke University
27708 Durham, NC, USA
krish@ee.duke.edu

***Abstract*-Multi-detect (*N*-detect) testing suffers from the drawback that its test length grows linearly with *N*. We present a new method to generate compact test sets that provide high defect coverage. The proposed technique makes judicious use of a new pattern-quality metric based on the concept of output deviations. We select the most effective patterns from a large *N*-detect pattern repository, and guarantee a small test set as well as complete stuck-at coverage. Simulation results for benchmark circuits show that with a compact, 1-detect stuck-at test set, the proposed method provides considerably higher transition-fault coverage and coverage ramp-up compared to another recently-published method. Moreover, in all cases, the proposed method either outperforms or is as effective as the competing approach in terms of bridging-fault coverage and the surrogate BCE+ metric. In many cases, higher transition-fault coverage is obtained 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.**

## I. INTRODUCTION

Manufacturing test is a major challenge for very-deep submicron (VDSM) integrated circuits. Mandated product-quality levels must be ensured by screening all defective devices before they are shipped. However, defect screening remains a formidable problem, especially for VDSM process technologies, since it is impossible to explicitly target every possible defect. 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. This model 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 [12].

The inadequacy of the stuck-at model for achieving high defect coverage led to the development of new fault models, e.g. transition-delay faults, bridging faults, etc., which reflect the behavior of realistic defects more accurately than the stuck-at fault model. However, even these fault models are inadequate for many new defect types and they are accompanied by practical problems that hinder 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.

An alternative approach to increase defect coverage is to rely on multi-detect testing, also referred to as *N*-detect testing. This method was proposed in [11], and since then it has been identified as a very effective test strategy [1]-[6], [8]-[10], [13]-[17]. 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. However, *N*-detect testing has one significant disadvantage—the size of the test set increases linearly as the value of $N$ increases in order to provide test patterns of higher quality. An alternative method proposed recently in [7] addresses this problem by embedding multi-detection of faults within a compact 1-detect test set, thereby increasing test quality without significantly increasing test-set size.

In this paper, we propose a new method to generate high-quality compact test sets with test lengths similar to that of 1-detect stuck-at test sets. In contrast to [7], we do not embed multi-detection of stuck-at faults within 1-detect stuck-at patterns. Instead, we first select a small number of the most-efficient test patterns from an *N*-detect pattern repository. Next we embed single-detection of stuck-at faults within the selected high-quality test patterns. Despite its compact size, the test set generated by the proposed method provides significantly higher coverage of transition-delay faults and comparable coverage for bridging faults, when compared to the baseline 1-detect test set and the test set obtained using [7]. It also provides higher coverage of transition-delay faults than larger *N*-detect test sets for several values of *N*.

The proposed method relies on a new quality metric for evaluating the effectiveness of each test pattern. We also present an efficient algorithm for selecting the most efficient test patterns. The quality metric is based on the concept of output

*The work of K. Chakrabarty was supported in part by the Semiconductor Research Corporation under contract no. 1588.

978-3-9810801-5-5/DATE09 © 2009 EDAA

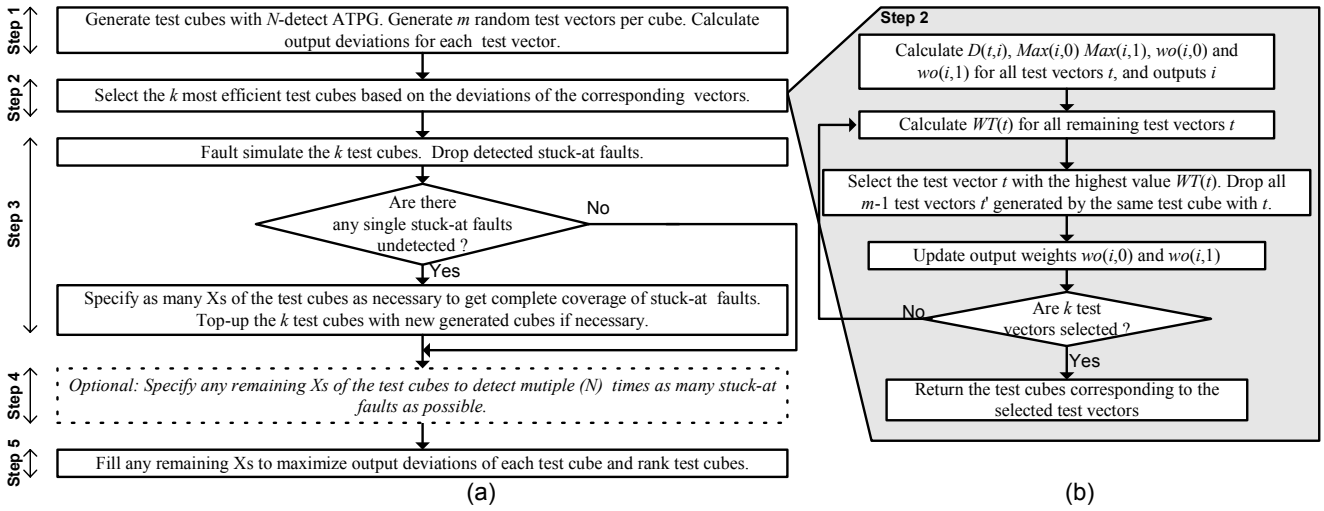


Fig. 1. Proposed test-generation flow.

deviations [19], which offer an effective probabilistic means to evaluate test vectors based on their potential for detecting arbitrary defects and, most importantly, without being biased towards any particular fault model. As shown in [18], unbiased testing provides higher test quality than a test method that is biased by a particular fault model. Simulations results for benchmark circuits demonstrate the effectiveness of the proposed method for defect screening. These results also 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 (described in detail in [19]) are probability measures at primary outputs, as well as pseudo-outputs for full-scan designs (all denoted as circuit outputs hereafter), that reflect the likelihood of error detection at these outputs. As it is shown in [19], 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: $R_i = ( r_i^{0...00}\ r_i^{0...01} ... r_i^{1...11} )$, 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:

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

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

Likewise, the signal probabilities can be easily computed for other gate types. For any logic gate (or output) $G_i$ 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. 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. In the next section, we define a new output-deviations measure that can be used for test generation.

## III. Proposed Test-Generation Method

The flow of the proposed test generation method is shown in Fig. 1(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, using $N$-detect ATPG with as high a value of $N$ as is computationally feasible. During ATPG, the Xs of the test cubes are left unspecified but the dynamic-compaction option is turned on in order to limit the size of the repository. Next, the generated test cubes are evaluated according to the proposed deviation-based quality metric. Since output deviations are not defined in [19] 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 list $L$ and their output deviations are calculated as proposed in [19]. Note that the $m$ random test vectors per cube are used only for evaluating the cube, and they are discarded after the selection of the most efficient cubes.

**Step 2**: The second step uses a quality metric to evaluate all the test vectors generated in the previous step, and selects $k$ out of them. These $k$ vectors are deemed to be of the highest quality and they correspond to $k$ different test cubes. The se-

lection process provides the set of $k$ test cubes from which the $k$ selected test vectors were generated. The selection is based on the following two rules:

(i) *Rule* 1: A test vector is selected only if it provides a high deviation value at one or more outputs.

(ii) *Rule* 2: The final set of selected test vectors provides high deviation value at least once at every output. However, the number of selected test vectors that provide high deviation value at output $i$ may be higher than the number of selected test vectors that provide high deviation at output $j$, if the potential of output $i$ to observe defects is higher than that of output $j$, based on the structure of the circuit.

The first rule ensures that a test vector will not be selected if it does not provide high deviation value, that is high observability at, at least one output. The deviation value $D(t,i)$ for test vector $t$ at output $i$ is high if it is close to the maximum deviation value provided by any test vector at this output. However, since we cannot consider all possible test vectors due to computation constraints, we limit the calculation of the maximum deviation values to the test vectors in list $L$. In addition, the maximum deviation at output $i$ when the fault-free response is 0 can be different than the maximum deviation at the same output when the fault-free response is 1. Therefore, in contrast to [19], for every output $i$, $1 \le i \le NO$, where $NO$ is the number of circuit outputs, we calculate two values, namely $Max(i,0)$ and $Max(i,1)$. These are the maximum deviation values achieved by any of the test vectors producing the fault-free response 0 (referred to as test vector subset $T_{i,0}$) and 1 (referred to as $T_{i,1}$) at this output ($T_{i,0} \cup T_{i,1}=L$). Thus, $Max(i,v)= max\{D(t,i)\}$, where $t \in T_{i,v}$, $v$=0 or 1. We say that $D(t,i)$ is high if

$$D(t,i) \ge F_1 \cdot Max(i,v), \text{ where } F_1 \le 1,\ v = 0 \text{ or } 1, \qquad (1)$$

where $v$ is the fault-free response *of* $t$ at output $i$. Note that $F_1$ is a real-valued quantity that should be close to 1 for selecting vectors with maximum output deviation values. However, in practice a value of $F_1 = 1$ is very strict and sometimes it results in a failure to select vectors, especially after the first few most-efficient vectors are selected. We experimentally verified that a value of $F_1$ in the range [0.99, 0.995] provides high-quality vectors, and thus for the experiments reported here, we set $F_1 = 0.995$.

The second rule ensures that the test vectors will be selected to eventually provide high deviation value at every output at least once, in order to provide high observability for as many defects as possible. It also considers the fact that the number of defects that can be observed at the output of a large logic cone is likely to be higher than the corresponding number at the output of a small logic cone. Therefore, every output $i$ is initially assigned a pair of weights $wo(i,0)$ and $wo(i,1)$, one for each set $T_{i,0}$ and $T_{i,1}$, respectively. Both these weights are initially set equal to the number of lines in the fan-in logic cone of output $i$, a measure of the volume of defects that can possibly affect this output. Then, a weight $WT(t)$ is calculated for every test vector $t$ as follows: let $MS(t)$ be the set consisting of all outputs $i$, $1 \le i \le NO$, with high deviation value $D(t,i)$ when vector $t$ is applied at the circuit (i.e. outputs $i$ satisfy inequality (1) for vector $t$). $MS(t)$ is partitioned into two subsets $MS_0(t)$, $MS_1(t)$; all outputs with fault-free response 0 are placed in $MS_0(t)$ and the rest in $MS_1(t)$. Then we have:

$$WT(t) = \sum_{i \in MS_0[t]} wo(i,0) + \sum_{i \in MS_1[t]} wo(i,1) \qquad (2)$$

$WT(t)$ is equal to the sum of the weights of all outputs with high observability during the application of $t$. The vector $t$ with the highest weight $WT(t)$ is selected every time.

If the selection is based only on Equation (2) then it will be biased towards test vectors that increase the deviations at a subset of outputs (with high weights), thus adversely affecting observability for other nodes in the circuit. In order to avoid this problem, every time a test vector $t$ that provides high deviation value at output $i$ is selected, the weight $wo(i,v)$, where $i \in MS_v(t)$, $v$=0, 1 is divided by a constant factor $F_2$. Therefore, outputs that exhibit increased observability by the application of $t$ are considered as less effective in detecting defects during the selection of the test vectors following. In this way the succeeding selected test vectors will provide high deviation value at other outputs too. Note that, by using two weights $wo(i,0)$ and $wo(i,1)$ for each output $i$ we enforce the selection of test vectors which provide high deviation values for both fault free responses 0 and 1 at this output since we control the number of vectors which provide high deviation value for each response separately. We verified experimentally that a value of $F_2$ 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 $F_2 = 8$ in the experiments reported in Section 4. The overall flow for this selection process is highlighted in Fig. 1(b).

**Step 3**: Next, we 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 detected stuck-at fault. Then, we specify their Xs in order to detect as many undetected stuck-at faults as possible, and if necessary, we top-up the $k$ test cubes with additional dynamically-compacted test cubes. The additional test cubes are not selected from the repository in order to extend the set of the $k$ test cubes with the smallest possible number of top-up test cubes.

**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].

**Step 5**: At this step all remaining Xs, are specified in such a way as to maximize the output deviations. Therefore, for each test cube, $m$ random test vectors are generated and the best test vector is selected, according to the $WT$ measure as in Step 2. Finally, the selected test vectors are ranked in descending order of their $WT$ measures.

## IV. Simulation Results

In this section, we evaluate the defect coverage of the proposed test-generation method for the largest ISCAS'89 and a subset of IWLS'05 benchmark circuits. The test-generation flow (Fig. 1), excluding ATPG and fault simulation, was implemented in C. Commercial tools were used for all ATPG-related and fault simulation steps.

The quality of the proposed method, with respect to defect

TABLE I
BENCHMARKS AND TEST-SET SIZES.

| | Circuit | # Inp. | # Scan Cells | # Gates | Reg SD/ND | Prop SD/ND 2-*p* | Prop SD/ND 1-*p* | Reposit. size |
|---|---|---|---|---|---|---|---|---|
| ISCAS'89 | **s5378** | 35 | 179 | 3114 | **142** | 147 | 146 | 498 |
| | **s9234** | 36 | 211 | 4636 | 190 | **188** | 206 | 1358 |
| | **s13207** | 62 | 638 | 6837 | **287** | 301 | 322 | 949 |
| | **s15850** | 77 | 534 | 7949 | 178 | **167** | 171 | 728 |
| | **s38417** | 28 | 1636 | 21104 | **155** | 160 | 159 | 692 |
| | **s38584** | 38 | 1426 | 22780 | 313 | **246** | 283 | 1326 |
| IWLS'05 | **sytemcaes** | 258 | 670 | 16574 | 343 | 325 | **309** | 1428 |
| | **tv80** | 13 | 359 | 13460 | **747** | 752 | 753 | 4198 |
| | **usb_funct** | 112 | 1746 | 23379 | 181 | **169** | 175 | 1049 |
| | **ac97_ctrl** | 54 | 2199 | 24055 | 217 | 179 | **176** | 1711 |
| | **mem_ctrl** | 116 | 1078 | 22402 | **607** | 643 | 647 | 2839 |
| | **pci_bridge32** | 159 | 3358 | 38015 | 1007 | 739 | **727** | 7157 |

TABLE II
BCE+ VALUES AND BRIDGING FAULT COVERAGE (1-*P* TEST SETS)

| | BCE+ | | | | Bridging fault coverage | | | |
|---|---|---|---|---|---|---|---|---|
| Circuit | Reg SD | Reg ND | Prop SD | Prop ND | Reg SD | Reg ND | Prop SD | Prop ND |
| **s5378** | 94.65 | 96.50 | 96.06 | **96.86** | 93.91 | 95.30 | 95.07 | **95.63** |
| **s9234** | 88.24 | 89.68 | 88.85 | **89.81** | 87.32 | 88.68 | 88.03 | **88.92** |
| **s13207** | 93.81 | 95.75 | 95.43 | **96.41** | 92.99 | 94.54 | 94.32 | **94.98** |
| **s15850** | 94.72 | 95.73 | 95.35 | **95.76** | 94.18 | 95.00 | 94.65 | **95.07** |
| **s38417** | 98.38 | 99.07 | 98.69 | **99.14** | 97.59 | 98.34 | 97.79 | **98.42** |
| **s38584** | 91.68 | **93.09** | 91.54 | 92.36 | 91.13 | **92.04** | 90.99 | 91.48 |
| **sytemcaes** | 99.08 | **99.23** | 99.07 | 99.08 | 96.73 | **96.76** | 96.54 | 96.52 |
| **tv80** | 91.91 | 92.71 | 92.31 | **92.72** | 89.61 | **90.02** | 89.86 | 89.97 |
| **usb_funct** | 94.87 | **96.55** | 95.23 | 96.45 | 96.13 | **97.19** | 96.36 | 97.16 |
| **ac97_ctrl** | 98.99 | **99.31** | 98.79 | 98.95 | 99.27 | **99.42** | 99.15 | 99.23 |
| **mem_ctrl** | 62.34 | 63.31 | 63.06 | **63.77** | 74.64 | 75.30 | 75.18 | **75.68** |
| **pci_bridge32** | 98.70 | **99.13** | 98.63 | 98.85 | 98.48 | **98.76** | 98.46 | 98.61 |

coverage, was evaluated using two surrogate fault models—the transition-delay fault model (using the launch-on-capture technique) and the bridging fault model. In the former case we calculated the proposed quality metric using the response to the second vector for each vector pair, (consisting of each stuck-at test and the response to the stuck-at test) while in the latter case, we use the response for every single vector (i.e., every stuck-at test).

In order to highlight the benefits of the proposed test-generation method, we compare it with traditional 1-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 1-detect test set, with the Xs specified randomly.

***Reg_ND***: traditional dynamically compacted 1-detect test set, with the Xs specified in such a way as to detect a stuck-at fault as many times (up to *N*) as possible. The approach of [7] for $N=10$ was implemented for this purpose.

***Prop_SD***: compact 1-detect test set generated by the proposed test-generation flow, with the Xs specified exclusively to maximize output deviation (Step 4 is omitted).

***Prop_ND***: compact 1-detect test set generated by the proposed test-generation flow, with the Xs specified in order to detect first multiple (up to $N=10$) 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 deviations.

For *Prop_SD* and *Prop_ND* test sets, 10-detect test sets were used as repositories. The number *k* of selected cubes from the repositories was set equal to 1/3 of the number of cubes in the *Reg_SD* test sets for circuits mem_ctrl, tv80, s13207, and to 2/3 for the rest of the circuits. Finally, *m* was set equal to 10.

The basic characteristics of the benchmarks are shown in Columns 1-4 of Table I. Column 5 presents the number of test vectors in *Reg_SD* and *Reg_ND* (the pattern counts are the same). Column 6 presents the (identical) number of test vectors in *Prop_SD* and *Prop_ND*, generated by applying the quality metric to vector pairs (denoted as 2-*p* in the Table I). Column 7 presents the same data generated by applying the quality metric to single vectors (denoted as 1-*p*). Finally, column 8 lists the number of the test cubes in the 10-detect pattern repositories. For each circuit, the size of the smallest test set generated is bold. In a majority of cases, the size of the test set generated by the proposed method is smaller than the test sets generated by the other methods and in all other cases, the test-set sizes are comparable. The test-set sizes for the proposed method can be reduced further by selecting a smaller value of *k*. A large *k* leads to higher test quality, therefore pattern count can be traded off with test quality by appropriately selecting *k*.

Next, we compare the four test sets with respect to the coverage achieved for transition-delay faults (Fig. 2). As expected, in the vast majority of the cases, *Reg_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 *Reg_ND* generated using the method proposed in [7]. In most cases, the highest coverage is provided by the *Prop_ND* test sets. Moreover, the proposed method provides higher ramp-up than the other methods, and it offers reduced test application time in an abort-at-first-fail environment.

In the second experiment, we study the multiple-detection profile of each test set. Due to limited space, we present results for three representative cases. The other benchmarks exhibit similar behavior. Each curve in Fig. 3 presents the percentage of stuck-at faults detected *n* times or more for $n = 1, 2, \ldots, 11$. Our results show that a high degree of multi-detection is not always necessary for high defect coverage. The test sets of the ac97_ctrl and system_caes for the proposed method provide less multi-detection than the two baseline methods, yet they provide higher transition-fault coverage. We therefore note 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 [16] was used as an estimate of bridging fault coverage. In addition, 400K bridging faults were 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 Table II. Columns 2-5 and 6-9 show the BCE+ measures and the bridging fault coverage

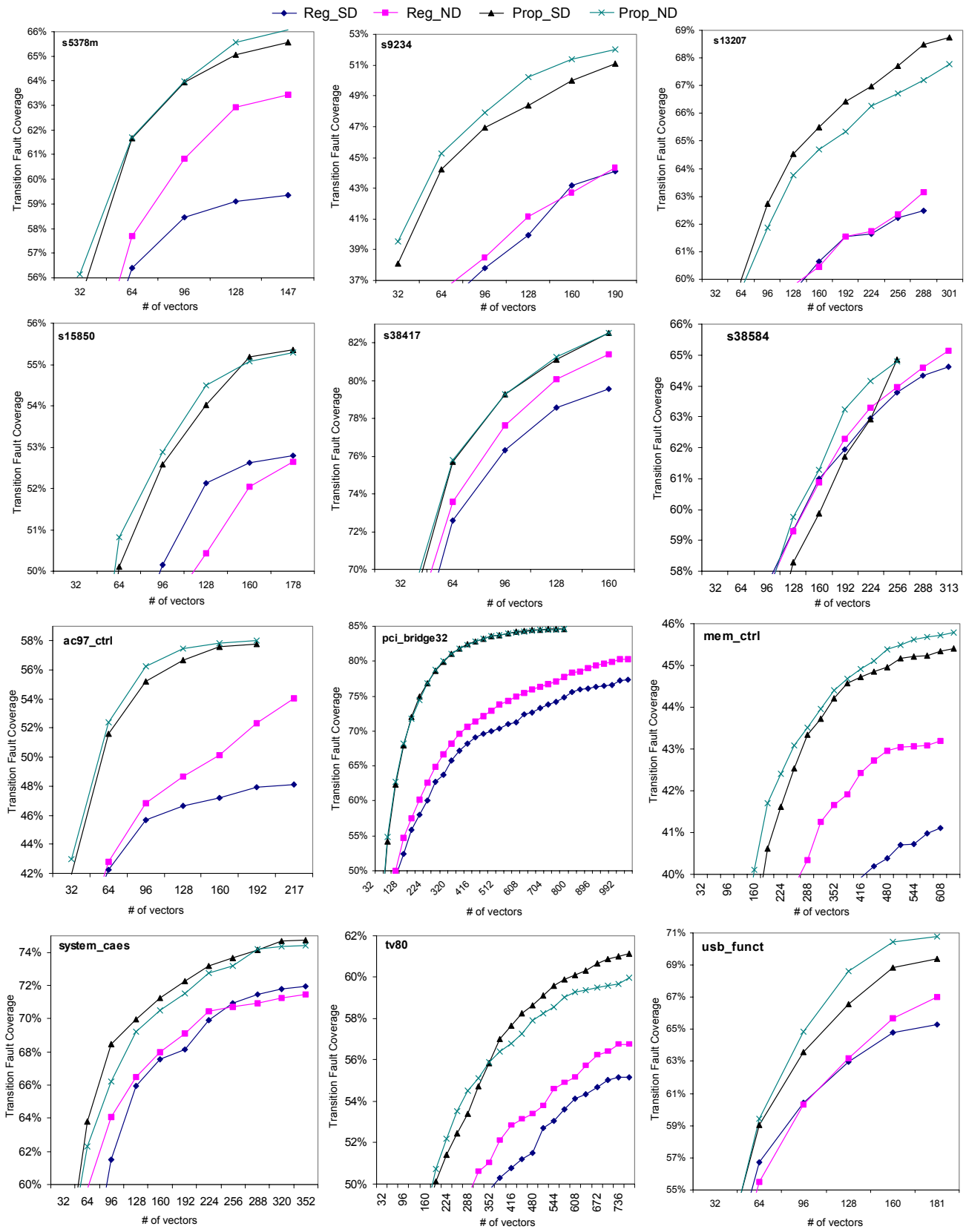


Fig. 2. Transition-fault coverage for the different test sets and the various benchmarks (2-*p* test sets).

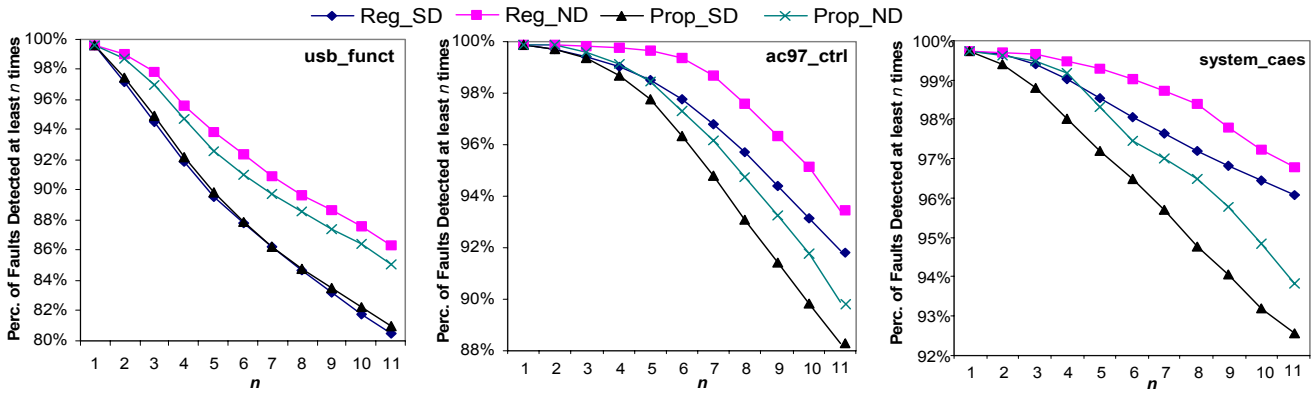


Fig. 3. Multi-detection results for the different test sets.

TABLE III
SMALLER TEST SETS ARE MORE EFFECTIVE FOR DEFECT COVERAGE THAN *N*-DETECT TEST SETS.

| Circuit | $N^*$: Threshold on $N$ | Test-Set Size | | Size Reduction (%) |
|---|---|---|---|---|
| | | $N^*$-detect | Proposed | |
| **s5378** | 7 | 367 | 147 | 59.9% |
| **s9234** | 6 | 876 | 188 | 78.5% |
| **s13207** | 6 | 660 | 301 | 54.4% |
| **s15850** | 5 | 448 | 167 | 62.7% |
| **s38417** | 2 | 226 | 160 | 29.2% |
| **s38584** | 2 | 482 | 246 | 49.0% |
| **sytemcaes** | 2 | 488 | 325 | 33.4% |
| **tv80** | 3 | 1632 | 752 | 53.9% |
| **usb_funct** | 3 | 397 | 169 | 57.4% |
| **ac97_ctrl** | 8 | 1396 | 179 | 87.2% |
| **mem_ctrl** | 5 | 1718 | 643 | 62.6% |
| **pci_bridge32** | 3 | 2739 | 739 | 73.0% |

for all test sets, respectively. We note that in most cases, the *Prop_ND* test set provides the best results, both in terms of the BCE+ measure and bridging fault coverage. In some cases, *Reg_ND* provides slightly higher bridging fault coverage and BCE+ values. However, note that in all these cases, the proposed test sets are smaller than the test-set sizes for *Reg_ND*. We expect that a larger value of $k$, and thus a larger test set (enriched with higher-quality patterns), will lead to higher BCE+ and bridging fault coverage using *Prop_ND*.

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 test set (all test sets provide complete coverage of detectable stuck-at faults). The results in Table III demonstrate that, for most benchmarks, the proposed method leads to smaller but more effective test sets than several *N*-detect test sets.

## V. CONCLUSIONS

We have presented a new method, based on the judicious use of output deviations, for generating compact, high-quality test sets. Simulation results show that compact test sets can be generated for complete single stuck-at coverage and higher coverage of unmodeled transition-delay faults and bridging faults 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] 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.

[10] 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.

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

[12] 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.

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

[14] 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.

[15] 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.

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

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

[18] 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.

[19] 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.