N-DETECT TEST PATTERN GENERATION AND RELAXATION USING ZDD

[1]B.DINESH KUMAR REDDY, [2]P.SAI KRISHNA, [3]M.PRABHU, [4]K. SIVA PRASAD REDDY, [5]NAVYA MOHAN, [6]G. VAMSI KRISHNA

[1,2,3,4,5]Department of Electronics and Communication Engineering, Amrita University, Amrita Vishwa Vidyapeetham, Coimbatore, India.
E-mail: bdineshsmile@gmail.com, saikrishna.041@gmail.com, john.prabhu22@gmail.com

Abstract- 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. This paper mainly deals with a method to generate N-detect test sets that provide high defect coverage making judicious use of new pattern quality metrics which is basically depended on the concept of ZDD. Simulation results for benchmark circuits show that, this method provides higher fault coverage and coverage ramp-up compared to other methods using BDD as ZDD provides an efficient way of solving problems expressed in terms of set theory and cube vectors.

I. INTRODUCTION

The basic process of testing is done by asking questions followed by analyzing the results by matching them to the correct or expected results. In VLSI testing one should have thorough knowledge about the object which is tested and then devise tests such that if it produces the expected results then it can be guaranteed to meet the requirements. Certain typical errors that are likely to occur are assumed and tests are devised to uncover these errors and certain algorithms are used to correct these them from occurring.

Electronic testing also uses this method which is formerly known as fault modelling and tests are guaranteed for assumed fault models. With the presence of single incorrect test response will fail a VLSI chip which increases the cost of manufacturing. So, it is the most important step to test a devise before manufacturing it and correcting those errors which would otherwise make it a faulty one. Suppose a devise is designed, fabricated, tested and it fails the test, then there may be many causes for the failure, something like fabrication errors, design errors etc.

It is highly impossible to check each and every node of a circuit for faults as it increases the test power and also requires a lot of time. The main idea of testing is to detect the fault and the idea of diagnosis is to determine the cause of the fault.

Testing helps in finding the minimal test vectors for which any error can be detected which would save both power and time which proves that quality andeconomy are the major benefits of testing.

Many methods are available to determine the faults for a particular circuit namely Fault equivalence, Fault dominance, Checkpoint theorem etc.

II. PROPOSED METHODOLOGY

In the proposed method, every test in $\mathcal{T}$ is systematically replaced by another test which has more number of unspecified bits. The algorithm focuses on a single fault at a time to determine different tests $\{t_j, j = 1, 2, \ldots, n\} \in \mathcal{T}$ that detect the fault in order to maximize the number of bits that can be relaxed. The algorithm determines *n* tests to target the fault and maximizes the unspecified bits that are required to detect the fault. Firstly, we provide the theoretical framework and present all the steps of the proposed methodology.

A. Theoretical Framework

Consider a fault f_i detected by $\mathcal{T}$. Let $\mathcal{T}_i \subseteq \mathcal{T}$ denote the set of tests in $\mathcal{T}$ that detect fault f_i. The algorithm finds the *n* tests in $\mathcal{T}_i$, given in $\mathcal{T}_i^n \subseteq \mathcal{T}_i$ that should detect fault f_i. Consider a test $t_k \in \mathcal{T}_i$. Let the number of specified bits in t_k that can be unspecified if t_k no longer detects f_i be denoted by c_{ik}. In other words, c_{ik} is the contribution of fault f_i in test t_k. Then, the total number of specified bits in $\mathcal{T}$ that can become unspecified if fault f_i is only detected by test $t_j \in \mathcal{T}_i$ is given by

$$G_{ij} = \sum C_{ik}, \qquad t_k \in \{T_i - t_j\} \qquad (1)$$

Thus, G_{ij} denotes the gain in unspecified bits if fault f_i is only targeted during the test generation by test t_j. Of course, coincidental detection of f_i by other tests may occur but this is done with no extra specified bits.

In order to find out which *n* tests of $\mathcal{T} = \{t_1, t_2, \ldots, t_m\}$ must explicitly target fault f_i, we calculate

$$\widehat{G}_i = max_j \{G_{ij}\}, \quad t_j \in T_i \qquad (2)$$

n different times by excluding t_j from $\mathcal{T}_i$ each time this calculation is made. All the tests that are selected form the set of tests $\mathcal{T}_i^n$ that target fault f_i.

It can be said that, as we have to find the number of test set bits that can be made unspecified after keeping only n detections, (1) and (2) should consider all the available combinations of detections. In other words, it has to be checked whether keeping the detections that give more number of specified bits relaxation is as effective as keeping the combination of n detections that give the larger relaxation. Next, we prove that the two decision criteria are identical. First we slightly modify equations (1) and (2) in order to evaluate the gain and maximum gain in specified bits considering all combinations of n detections for the same fault

$$G_{ij}^n = \sum C_{ik}\,,\ \mathrm{t_k} \in \{T_i - \tau_j\} \qquad (3)$$

With t_j we denote a subset of $\mathcal{T}_i$ of size n. Thus, $\hat{G}_i^n$ is calculated for all combinations of n tests out of all tests that detect fault f_i. Similarly, (2) becomes

$$\widehat{G_i^n} = max_j\{G_{ij}^n\}\,,\ \ \tau_j \subset T_i \qquad (4)$$

When an n-detect test set $\mathcal{T}$ is fault simulated against a fault list F, there exist a set of faults $F_m \subseteq F$ that are detected more than n times. For a fault $f_i \in F_m$ we keep those detections (n tests) that when using (2) give the n higher values, in set $S(f_i)$. Moreover, for the same fault $f_i \in F$ we keep those detections (tests in T_j) that when using (4) give the higher value, in set $C(f_i)$

Theorem 1: For some fault $f_i \in F_m$, the set of tests $S(f_i)$ is identical to the set of tests $C(f_i)$.

Proof: Set $S(f_i)$ contains the n tests that give the maximum values when calculating (2), for fault f_i. By substituting (1) in (2) we have

$$G_i = max_j\left\{\sum_{t_k\in\{T_i - t_j\}}(C_{ik})\right\},\ t_j \in T_i \ \Leftrightarrow$$
$$G_i = max_j\left\{\sum_{t_k\in T_i}(C_{ik}) - C_{ik}\right\}, t_j \in T_i \qquad (5)$$

However, the term $\sum_{t_k\in\mathcal{T}_i}(c_{ik})$ is a constant since it does not depend on j and thus

$$G_i = \sum_{t_k\in T_i}(C_{ik}) + max_j\{-C_{ij}\}, \qquad t_j \in T_i.$$

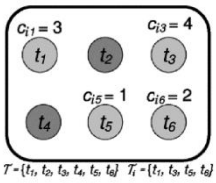
The tests to be included in test set $S(f_i)$ are selected by obtaining n times the $\hat{G}_i$, i.e., n times

$$t = \arg max_j\{-C_{ij}\}, \quad t_j \in T_i.$$

Since all c_{ij} are non-negative integers the same test is obtained by

$$t = \arg min_j\{C_{ij}\}, \quad t_j \in T_i. \qquad (6)$$

Set $C(F_i)$ contains the n tests which are elements of T_j that give the maximum value in (4), for fault f_i by Substituting (3) in (4) we have



Equation 5

t_j	$\sum_{t_k\in T_i}(c_{ik}) - c_{ij}$	G_{ij}
t_1	10 – 3	7
t_2	n/d	
t_3	10 – 4	6
t_4	n/d	
t_5	**10 – 1**	9
t_6	**10 –2**	8

Equation 7

τ_j	$\sum_{t_k\in T_i} c_{ik} - \sum_{t_h\in\tau_j} c_{ih}$	G_{ij}^2	τ_j	$\sum_{t_k\in T_i} c_{ik} - \sum_{t_h\in\tau_j} c_{ih}$	G_{ij}^2
t_1, t_2	n/d		t_2, t_5	n/d	
t_1, t_3	10 - (3+4)	3	t_2, t_6	n/d	
t_1, t_4	n/d		t_3, t_4	n/d	
t_1, t_5	10 - (3+1)	6	t_3, t_5	10 - (4+1)	5
t_1, t_6	10 - (3+2)	5	t_3, t_6	10 - (4+2)	4
t_2, t_3	n/d		t_4, t_5	n/d	
t_2, t_4	n/d		t_4, t_5	n/d	
			t_5, t_6	**10 - (2+1)**	7

Fig 1. Example illustrating theorem 1

$$\widehat{G_i^n} = max_j\left\{\sum_{t_k\in\{T_i - t_j\}} C_{ik}\right\},\ \tau_j \subset T_i \ \Leftrightarrow$$
$$\widehat{G_i^n} = max_j\left\{\sum_{t_k\in T_i} C_{ik} - \sum_{t_h\in T_i} C_{ih}\right\}, \tau_j \ \ T_i \quad (7)$$

Again, the term $\sum_{t_k\in\mathcal{T}_i} c_{ik}$ does not depend on j, so we get

$$G_i^n = \sum_{t_k\in T_i} C_{ik} + max_j\left\{-\sum_{t_h\in T_j} C_{ih}\right\}, \quad T_j \subset T_i.$$

The set of tests that give the maximum reduction in specified bits can be obtained by

$$C(f_i) = \arg max_j\left\{\sum_{t_h\in T_j} C_{ih}\right\}, \qquad T_j \subset T_i.$$

Since all c_{ih} are non-negative integers we get

$$C(f_i) = \arg min_j\left\{\sum_{t_h\in T_j} C_{ih}\right\}, \qquad T_j \subset T_i. \qquad (8)$$

Equation (6) implies that the set $S(f_i)$ contains the n tests from $T_i(T_i^n)$ that have the minimum c_{ij}, i.e., the tests that are elements of the subset of T_i of size n that has the minimum sum of cij. The subset of Ti of size n that has the minimum sum of cij is given in (8) which, however, gives the set C (fi). Hence, the sets S ( ) and C (fi) are identical. Theorem 1 suggests that using (2) for selecting the best tests to detect each fault gives exactly the same result as using (4). Thus, there is no need to find the contribution in specified bits for all combinations of n tests in Ti for same fault order to keep those tests that give the highest reduction in terms of specified bits.

Fig. 1 presents an indicative example that illustrates the rationale of Theorem 1 and lists all the necessary calculations. The diagram on the left shows a test set $\mathcal{T}$ and a sub $\mathcal{T}_i$ that includes all the four tests that cover fault f_i i.e., t_1, t_3, t_5, and t_3. The number above each test indicates the number of specified bits that can become don't cares if f_i (and only f_i) is no longer detected by the test (i.e., this is c_{ij}). For instance, 4 bits of t_3 can become don't

cares and this relaxation does not affect any other fault detection of t_3 except that of f_i. Now we assume that $n = 2$ and according to the problem examined we have to select that subset of $\mathcal{T}_i$ with cardinality 2 that will give me the largest gain in specified bits, defined above as $C(f_i)$ This re- quires the calculation of G_{ij}[2] for all six pairs of tests in $\mathcal{T}_i$ and select the maximum (4). These calculations are done using (7) and are listed on the right table of Fig. 1. For instance, if tests t_3 and t_5 are selected to detect f_i, 5 bits can be converted to don't cares. Obviously, these calculations will lead to selecting the set of tests $T_j = \{t_5, t_6\}$ as $C(f_i)$, since this gives the largest relaxing benefit n/d here means that the corresponding combination will reduce the *n*-detect fault coverage of the test and it should not be considered as a possible solution.

According to Theorem 1 the same sets of test will be selected if (2) instead of (4) is used for *n* consecutive times. Observe that (2) computes the benefit in specified bits when the detection of f_i is achieved by only one test at a time. These calculations are done using (5) and are listed on the left table of Fig. 1. For example if f_i is enforced to be detected only by t_5 and no other test in $\mathcal{T}_i$ 9 bits can become don't cares. Obviously, the desired test in this case is t_5, since it gives the maximum gain in specified bits. The second best test is t_3, giving eight specified bits reduction. These two tests, i.e., t_3 and t_5 are by definition the elements of $S(f_i).S(f_i)$ coincides with $C(f_i)$ even though two different equations have been used for their construction. Observe that, the values of G_{i5} and G_{i6} (9 and 8) do not reflect the actual number of specified bits that become unspecified (i.e., 7), yet the same set is formed with smaller effort (4) calculation in- stead of 6). The effort reduction is even larger in real cases where a lot of faults have a large number of tests detecting them and *n* is typically between 5 and 10. The actual number of specified bits that become don't cares can be obtained by subtracting the cost corresponding to the selected tests from the total number of specified bit that can be relaxed when f_i is no more detected by any test, i.e.,

$$\sum_{t_k \in T_i} C_{ik} - \sum_{t_h \in T_j, T_j \subset T_i} C_{ih} = 10 - (C_{i5} + C_{i6}) = 7$$

B. Proposed algorithm

Fig. 2 shows the proposed algorithm. The input parameters are the circuit-under-test the test set to be relaxed $\mathcal{T}$, the n-detect parameter *n*, and the considered fault model $\mathcal{M}$ based on which the targeted fault list $\mathcal{F}$ is derived (lines 1–2 of Fig. 2).

First, fault simulation is performed to derive the complete fault list $\mathcal{F}$ as well as the fault lists F_j for each test $t_j \in \mathcal{T}$. To achieve this no fault dropping is allowed. Then, the algorithm iterates over each fault f_i, following a predefined ordering order to determine the "best" n tests to detect f_i. This is done by examining only tests in $\mathcal{T}$ that detect f_i, that is $\mathcal{T}_i$ (lines 6–19). For every test $t_j \in \mathcal{T}_i$ the

n_detect_relax

Inputs: circuit $\mathcal{C}$, test set $\mathcal{T}$, n, fault model $\mathcal{M}$
Outputs: relaxed test set $\mathcal{T}'$
01: fault simulate $\mathcal{T}$ based on fault model $\mathcal{M}$
02: F = list of faults detected by $\mathcal{T}$
03: **for each** test $t_j \in \mathcal{T}$
04: F_j = list of faults detected by t_j
05: **for each** fault $f_i \in F$
06: $\mathcal{T}_i$ = list of tests detecting f_i
07: **for each** test $t_j \in \mathcal{T}_i$
08: use F_j to calculate c_{ij}
09: **for each** test $t_j \in \mathcal{T}_i$
10: calculate $G_{ij} = \sum c_{ik}, \quad k \in \{\mathcal{T}_i - t_j\}$
11: $\mathcal{T}_i^n = \emptyset$
12: **for** $d = 1$ to n
13: $\hat{G}_{im_d} = \max\{G_{ij}\}, \quad t_j \in \{\mathcal{T}_i - \mathcal{T}_i^n\}$
14: $\mathcal{T}_i^n = \mathcal{T}_i^n + t_{m_d}$
15: *% tests in $\mathcal{T}_i^n$ keep f_i, tests in $\{\mathcal{T}_i - \mathcal{T}_i^n\}$ drop f_i*
16: **for each** $t_j \in \{\mathcal{T}_i - \mathcal{T}_i^n\}$
17: $F_j = F_j - f_i$
18: **if** $F_j = \emptyset$
19: $\mathcal{T} = \mathcal{T} - t_j$ *% drop test t_j*
20: $\mathcal{T}' = \emptyset$
21: **for each** test $t_j \in \mathcal{T}$
22: generate test t_j' that detects all faults in F_j
23: add t_j' to $\mathcal{T}'$
24: **return** $\mathcal{T}'$

Fig 2. Proposed *n*-detect algorithm

contribution of f_i in t_j (c_{ij}) is first calculated (line 8). This is a crucial step which invokes a test generation routine. Specifically, to find for a fault f_i and a test t_j detecting the faults in F_j, we generate a test cube t' targeting faults in f_i. If the number of specified bits in a test t_j is denoted by c_{ij}, then $s(t_j) - s(t')$. This is the number of specified bits savings if test t_j no longer detects fault f_i. Once c_{ij} is calculated for every test $t_j \in \mathcal{T}_i$, the total gain G_{ij} in unspecified bits (meaning f_i is detected by t_j but not by $\{\mathcal{T}_i - t_j\}$) for every test t_j is easily computed (line 10). Consequently, the *n* tests giving the maximum gain are determined (lines 11–14). This is achieved by calculating *n* times the maximum gain, each time removing all the previously found tests with maximum gain. Tests $t_{m_d} = \mathcal{T}_i$, d= $1, 2, \ldots, n$ form the set $\mathcal{T}_i^n$ containing all tests that detect f_i.

The next steps (lines 16–19) convey the dynamic nature of the algorithm. Once set $\mathcal{T}_i^n$ is determined, it is no longer necessary for tests $\{\mathcal{T}_i - \mathcal{T}_i^n\}$ to detect f_i. Therefore, the fault list F_j for each of the remaining tests $t_j \in \mathcal{T}_i$ is updated. In this manner, fault f_i will never be targeted in any subsequent test generation step (line 8). Observe that if a test's fault list becomes empty at any point, the test can be fully relaxed which means it can be dropped since all of the faults it used to detect are now detected by some other test(s). The fault coverage of $\mathcal{T}$ is maintained since every fault f_i is guaranteed to be detected by n different tests $t_{m_d} \in \mathcal{T}_i^n$, d=1,2,3,….n with $F_{m_d} \neq \emptyset$.

Once all faults are examined, the relaxed test set $\mathcal{T}'$ is generated based on the updated fault list F_j for each test t_j that has remained in (lines 20–23). Each new test $t'_j \in \mathcal{T}'$ is guaranteed to detect a subset of the faults detected by the corresponding test $t_j \in \mathcal{T}$, since the size of the updated fault list per test is reduced or, in the worst case, remains the same.

The algorithm of Fig. 2 can be slightly modified to enforce a different number of detections (instead of n) for each fault. The algorithm can take as input an array containing the desired number of detections for each fault and traverse that array instead of the loop at line 12 of the pseudo code. Alternatively, the number of detections per fault can be obtained dynamically based on the characteristics of the given test set. For example, the number of detections for a fault f_i can be a percentage p of the number of detections for that fault in the given test set $\mathcal{T}$, i.e., a subset of $\mathcal{T}_i$ with size p.

The effectiveness of the proposed method depends greatly on the ability of the test generation process (line 8 and line 22 of Fig. 2) to derive tests with a large number of unspecified bits. Several existing methods can be used to solve this problem effectively. Both of the structural methods of [1], [2] propose specific structural-based ATPG-like routines (using implications, justifications, and testability measure concepts) to find a large test cube (test with a large number of unspecified bits) that detects a number of faults. Alternatively the function-based framework of [3], can derive a large cube by extracting the shortest path in a BDD-based implementation. We discuss this framework in Section VI. Any of these techniques can be used by the proposed method whose main contribution is not on this specific single test generation problem but on finding a systematic method to replace an entire test set such that the total number of specified bits is minimized.

The proposed algorithm takes $|\mathcal{T}|$ fault simulations plus, in the worst case, $|\mathcal{T}| \cdot F| + |\mathcal{T}|$ test generations. In practice, how- ever, the factor $|\mathcal{T}| \cdot |F|$ is much smaller since each fault $f_i \in \mathcal{F}$ is examined only against the small number of tests in $\mathcal{T}_i \subseteq \mathcal{T}$ that detect the fault, and not for the entire test set $\mathcal{T}$.

III. EXPERIMENTAL RESULTS

In the method described in Section II the test generation procedure is of great importance. Specifically, this procedure should be able to efficiently generate a single test pattern that detects a specific set of faults and contains a small number of specified bits. This procedure can be implemented using either a structural-based or a function-based test pattern generation/manipulation framework without affecting the effectiveness of the proposed methodology.

In a structural-based framework, existing routines such as those in [1] and [2] can be integrated in the proposed method- ology outlined in Fig. 2. For example, step 8 of Fig. 2 can be a call to the specific routine of [2] or [1] which returns a partially specified test pattern that detects one or more given faults. This test can be used to calculate the specified bits contribution for each fault/test combination (i.e., cij for fault fi and test (t_j) (t_i) which is a critical criterion for the selection of the tests that will detect each fault in the final test set. Both these two approaches for test generation are of linear, to the size of the circuit, complexity giving a very good alternative to the test generation process that is called a large number of times in the pro- posed methodology.

Functional-based frameworks can utilize various techniques for representing and manipulating the Boolean functions, such as binary

TABLE I
Test set relaxation for 8-detect test sets

Circuit	PIs	Initial Test Set[5]				After Proposed Method			
		No of Faults	$\|\mathcal{T}\|$	SP. Bits	N(T)%	$\|\mathcal{T}'\|$	SP. Bits	N(T')%	K(T')
S208	18	210	271	4878	73.190	167	1124	73.424	0.23
S298	17	332	234	3978	100	234	2352	100	0.59
S344	24	334	138	3312	100	136	2049	100	0.62
S382	24	418	253	6072	100	252	3019	100	0.50
S386	13	430	201	2613	60.442	201	1924	60.490	0.74
S420	34	446	433	14722	58.991	275	2983	59.236	0.20
S510	25	572	543	13575	100	543	4291	100	0.32
S526	24	625	492	11808	100	491	6844	100	0.58
S641	54	518	227	12258	100	227	6497	100	0.53
S820	23	1018	949	21827	100	944	10512	100	0.48
S953	45	1078	766	34470	100	764	11341	100	0.33
S1196	32	1294	1233	39456	97.295	1157	16452	97.284	0.42
S1423	91	1408	269	24479	100	266	13249	100	0.54
S1488	14	1642	209	2926	55.164	209	2629	55.196	0.90
S9234	247	6960	1132	279604	100	1132	102471	100	0.37
S13207	700	9788	2341	1638700	100	2341	128489	100	0.08
S15850	611	11182	983	600613	100	983	144621	100	0.24
S38417	1664	31183	784	1304576	100	784	849206	100	0.65
				Average	91.394		Average	91.42	0.46

Decision diagrams (BDDs) [4], Boolean expression diagrams (BEDs), Boolean satisfiability or even combinations of these. Without any loss of generality, the proposed algorithm was implemented using a BDD- based framework (Expanding an in-house ATPG tool) similar to that utilized in [3] The major advantages of the utilized function-based framework are: 1) the complete set of tests for a fault or group of faults is implicitly considered and 2) obtaining a test with many unspecified bits can be done efficiently. Specifically, obtaining the test that has the most don't care bits from a test function accounts in obtaining the largest cube

from that function. The latter, while can be reduced to the NP-hard problem of Boolean satisfiability, is done efficiently (not necessarily optimally) when using BDDs. As long as the diagram representing the function can be constructed, obtaining the largest cube amounts to identifying the BDD's shortest path, for the root node to the terminal node one, which is a linear, to the size of the diagram, operation. This process gives the largest cube of the corresponding function, under a certain BDD variable ordering. The reader is referred to [3] for further details on the underlying BDD-based test generation framework.

Specified bits reduction First we present the test set characteristics before and after applying the proposed method on the compact 10-detect test sets of [7]. Table I shows the number of Primary Inputs in Column 2, next to the circuit name.

Column 3 reports the number of faults considered for each circuit. The number of faults considered for each circuit was obtained after applying Function-based fault equivalence rules similar to those used in [6] on top of the Checkpoint Theorem [5].

Column 4 reports the size of the initial, fully specified test set and Column 5 gives the number of specified bits in. Column 6 reports the n-detect detect fault coverage calculated using Definition 2. Columns 7–9 list the same information for the derived relaxed test set. Moreover, the specified to total bits ratio K(T') (Definition 3), after the test relaxation is reported, in Column 10. The initial test sets are fully-specified, thus K(T)=1, in all cases. For all circuits reported the n-detect fault coverage has been preserved. For circuits s386, s420, s1196, and s1488 the n -detect fault coverage is marginally improved due to coincidental detections of faults that have less than n-detections in the initial test set. The latter occurs since, as we mentioned before, our method performs test generation for the list of faults remained for each test and, thus, extra fault detections may arise. Clearly, the proposed method helps significantly in identifying bits that can get don't care values. The reduction in specified bits is, significant in all cases, ranging between 90% and 8% and with an average of 62% since the average K(T') is 0.46. This reduction is significant, despite the fact that the 10-detect test sets used are very compact (close to the optimal size for 10 detect). Typically, less compact test sets allow for higher specified bits reduction. It is also important to note that in many cases the proposed method produced smaller test sets. This compaction effect is significant for some cases (circuit's s208, s420, s1196, and s1423). Since, to our knowledge, there is no prior work on test set relaxation for traditional n-detect test sets we have implemented a simple test relaxation technique in order to demonstrate the effectiveness of the proposed method. Specifically, this method is a brute-force method in which each test is fault simulated and only the detected faults that have not been covered n covered times are used to generate a new test, with fewer

Table II
Comparing with Brute-Force technique

Circuit	$\|T\|$	Brute-Force $K(T')$	Proposed $K(T')$
S208	271	0.48	0.23
S228	234	0.82	0.59
S344	138	0.72	0.62
S382	253	0.89	0.50
S386	201	0.91	0.74
S422	433	0.35	0.20
S510	543	0.67	0.32
S526	492	0.74	0.58
S641	227	0.78	0.53
S820	949	0.66	0.48
S953	766	0.45	0.33
S1196	1233	0.54	0.42
S1423	269	0.69	0.54
S1488	209	0.91	0.90
S9234	1132	0.57	0.37
S13207	2341	0.69	0.08
S15850	983	0.38	0.24
S38417	784	0.72	0.65

specified bits, to replace the one from the original test set. Consequently fault dropping is performed after each test replacement. In this manner, each considered test to be relaxed will no longer have to target a fault if it has already been detected n times . Table II lists the obtained results. The initial test sets are the same as those used for the experiment in Table I. Columns 3 and 4 list the specified to total bits ratio K(T') for the brute-force and the proposed approach, respectively. In all cases the proposed methodology is more effective in decreasing the number of specified bits. This demonstrates that the optimization goal targeted in the proposed approach helps in finding better sets of n (10 for this experiment) tests to target a fault such that the number of specified bits is reduced, than a straightforward approach that selects these test sets in a brute-force manner.

CONCLUSION

In this work, we have investigated the impact of test set relaxation in n detect test sets. We presented a systematic methodology for decreasing the number of specified bits in a given n-detect test set or a multiple detect test set. The experimental results reported demonstrate the effectiveness of the proposed method in achieving high specified bit reduction rates n-detect test sets, while maintaining the n- detect fault coverage. Provided discussion and experimentation data also explain how the defect coverage and non-targeted fault coverage of the relaxed detect fault coverage. Provided discussion and experimentation data also explain how the defect coverage and non-targeted fault coverage of the relaxed.

REFERENCES

[1] El-Maleh and A. Al-Suwaiyan, "An efficient test relaxation technique for combinational & amp; full-scan sequential circuits," in Proc. VTS, 2002, pp. 53–59.

[2] K. Miyase and S. Kajihara, "XID: Don't care identification of test pat- terns for combinational circuits," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 23, no. 2, pp. 321–326, 2004.

[3] S. Neophytou and M. Michael, "Test set generation with a large number of unspecified bits using static and dynamic techniques," IEEE Trans. Comput., vol. 59, no. 3, pp. 301–316, Mar. 2010.

[4] R. Bryant, "Graph-Based algorithms for boolean function manipula- tion," IEEE Trans. Comput., vol. C-35, no. 8, pp. 677–691, Aug. 1986.

[5] M. Bushnell and V. Agrawal, Essentials of Electronic Testing. Nor- well, MA: Kluwer, 2000.

[6] R. Adapa, S. Tragoudas, and M. Michael, "Evaluation of col lapsing methods for fault diagnosis," in Proc. ISQED, 2006, pp. 439–444.

[7] I. pomeranz and S. M. Reddy, "Forming n-detectioin test sets without test generation" ACM Trans. Design Autom. Electron Syst., vol 12, no.2, Apr. 1, 2007. Article 18.

★★★