Generating Compact Robust and Non-Robust Tests for Complete Coverage of Path Delay Faults Based on Stuck-at Tests *

Dong Xiang
School of Software,
Tsinghua University,
Beijing 100084, China
dxiang@tsinghua.edu.cn

Kaiwei Li
School of Software,
Tsinghua University,
Beijing 100084
likaiwei@mails.tsinghua.edu.cn

Hideo Fujiwara
Grad. Sch. of Inform. Sci.,
Nara Institute of Sci. and Techn.
Ikoma, Nara 630-0101, Japan
fujiwara@is.naist.jp

Jiaguang Sun
School of Software,
Tsinghua University,
Beijing 100084
sunjg@tsinghua.edu.cn

Abstract — A new test generation method of fully scanned or combinational circuits is proposed for complete coverage of path delay faults based on single stuck-at tests. The proposed method adds the target path into the original circuit, where all off inputs of the path are connected with corresponding nodes in the original circuit. Test generation of the path delay fault is reduced to that of the single stuck-at fault at the fanout branch, where the additional path connects with its source node in the original circuit. A disjoint dynamic test compaction scheme is proposed to reduce the size of the test set in the process of test generation. A conjoint test compaction scheme is proposed based on fanout counts of the paths. The proposed method presents a very compact test set for complete coverage of robustly and non-robustly testable path delay faults.

Keywords —Dynamic test compaction, path delay faults, single stuck-at faults, test generation.

I. Introduction

Test generation finds a compact test set to cover all modeled faults. The most popular fault models are the single stuck-at fault and the path delay fault. Test generation and testing techniques for the single stuck-at fault have been well studied. Test generation for fully scanned circuits or combinational circuits has been resolved well in [4, 6, 9, 17]. The problem is how to reduce test application time, test data volume, and test power consumption generated by test application.

One of the most important problems for test generation of path delay faults is the huge number of paths in the circuit, which can increase exponentially with the size of the circuit. It is observed that most of the path delay faults in a circuit are usually redundant. It is unnecessary to spend much effort on test generation and fault simulation of the redundant path delay faults [2, 13, 8].

Cheng *et al.* [2] pointed out that it is unnecessary to detect functional redundant faults. Path delay faults are classified into nonrobust testable, functional redundant, and functional sensitizable faults. Recently, a very good structure called zero-suppressed binary decision diagram is used to select testable or critical paths in reasonable time [13]. The test compaction procedures in Bose *et al.* [1] used the concept of primary and secondary target faults. Once a test is found for a primary fault, it is expanded so that it also detects one or more secondary faults.

One of the most important differences between test generation of path delay faults and single stuck-at faults is that the fault effect of a single stuck-at fault can be propagated along multiple paths while the transition of a path delay fault can only be propagated along the corresponding path. Enough methods have been proposed to handle test generation of path delay faults by the single stuck-at fault test generation techniques [5, 12, 15]. Saldanha, *et al.* [15] proposed the first test generation procedure by transforming the circuit. It is shown that the robust test vector pairs for path delay faults in a circuit can be generated by the test vectors for a single stuck-at fault in the transformed circuit. Sufficient theoretical analyses on equivalence between path fault testing and single stuck-at fault testability were presented in [5, 15]. However, the size of the transformed circuit can be very large for circuits with a huge number of paths. Recently, Ohtake, *et al.* [12] proposed a new test generation method for path delay faults based on single stuck-at tests using partial circuit transformation. Partial circuit transformation modifies a circuit based on a subset of paths. Frequent transformation of the circuit partially can still be very time-consuming.

Test compaction has also been studied in a number of previous methods for stuck-at faults [6, 10] and path delay faults [1, 16, 7, 11, 14]. Saxena, *et al.* [16] presented a concept called compatible faults to generate a compact test set for path delay faults. The PODEM algorithm [6] is extended to compact test generation of path delay faults based on an effective path ordering heuristic in [1]. Pomeranz and Reddy in [14] proposed length-based, valued-based and count-based dynamic test compaction schemes to reduce the test size of path delay faults. The length-based scheme tries to cover critical paths as many as possible while the count-based scheme tries to cover as many as possible path delay faults. Kajihara *et al.* in [7] proposed a length-based static compaction scheme for non-robustly testable path delay faults most recently. However, static test compaction can be time-consuming based on the original test set with one test per fault. Michael and Tragoudas [11] recently proposed a non-enumerative compact test generation method based on binary decision diagrams. However, the method in [11] does not guarantee complete coverage for a number of circuits.

Main contributions of this paper includes: A new test generation method for complete coverage of path delay faults in combinational or fully scanned circuits is proposed based on single stuck-at tests without circuit transformation. Two dynamic test compaction scheme is proposed to generate a compact test set for path delay faults based on the influence cone and the output cone of a target path.

*This work was partially supported by the National Science Foundation of China under grants 60373009 and 60425203.

II. Definitions and Notation

Let a path p be g_1-g_2- ... -g_n, and $g_i(v)$ be the value of gate g_i when applying the test vector v to the circuit, where g_1 and g_n are primary input and primary output. Here, g_1 and g_n are called the source node and the sink node of the path, respectively. The off-inputs $off(g_i, p)$ are the inputs of g_i that are not g_{i-1}. The path can have a rising or falling transition at g_1. A path p has a path delay fault if propagation time of the rising or falling transition through the path exceeds a limit. Falling transition and rising transition at the start signal of a path p are p_f and p_r, respectively.

A test of a path delay fault with respect to a path $p = g_1$-g_2- ... -g_n which guarantees to detect the path delay fault when no other path delay fault is present is called *nonrobust test*. A path delay fault for which a nonrobust test exists is called a singly testable path delay fault. A test of a path delay fault with respect to a path $p = g_1$-g_2- ... -g_n which guarantees to produce an incorrect value at the sink node if the delay of the path under test exceeds a specified time interval (or clock period), irrespective of the delay distribution in the circuit, is called a *robust test* of the path delay fault.

The proposed method gets the testable path delay faults by using the recent ZBDD-based redundant identification scheme [13]. A series of selected path circuits (SPC) are constructed based on the testable paths, where the SPC circuit is fanout-free. Usually, the time to construct the SPC circuit and the testable path identification is trivial [13] compared with the test generation time and fault simulation time.

III. Test Generation for Path Delay Faults with Stuck-at Fault Tests

A new scheme is proposed to generate tests for path delay faults, which does not need any circuit transformation. For any target path, a new circuit (called equivalent test generation circuit, ETGC) can be constructed as follows: The new circuit includes the original circuit in addition to another copy of the target path, where all off-input lines of the target path are connected with the corresponding lines in the original circuit. Test generation of the target path delay fault is equivalent to test generation of the single stuck-at fault at the source of the additional path.

Theorem 1 *Generating a nonrobust test of the target path is equivalent to generating a test for the single stuck-at fault at the source of the additional path in the ETGC.*

Proof: Let us consider the falling transition at the source of the path without loss of generalization. Generating a test for the falling transition at the source of the path is reduced to generate a test for the stuck-at-1 fault at a' ($a'/1$). In order to detect the path delay fault, a falling transition must be generated at a, which corresponds to activating the $a'/1$. There exist a single path from a' to its end e' as shown in Fig. 1, therefore, fault effect of the fault $a'/1$ must be propagated along the single path. The nodes b', c', ..., d' must be assigned non-controlling values, respectively in order to propagate the fault effect of the single stuck-at fault $a'/1$ to e'. The nodes b, c, ..., d are assigned sensitization values simultaneously. ■

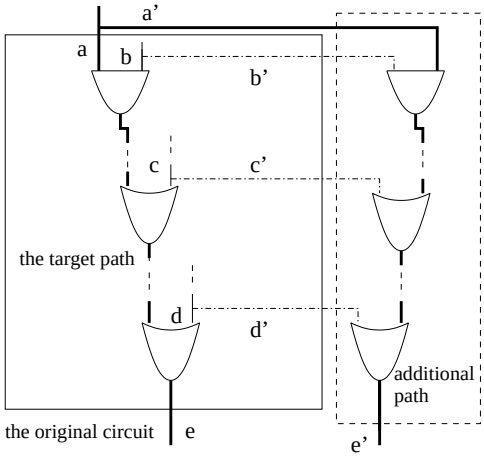

Figure 1: Generating non-robust tests for a path delay fault by a single stuck-at fault test without circuit transformation.

The additional path can be replaced by any fanout free subcircuit of the original circuit. Test generation of any path of the fanout free subcircuit corresponds to generating tests of the corresponding single stuck-at fault in the additional subcircuit. In this paper, test generation of path delay faults can be completed by constructing a series of fanout free selected path circuits (SPC), where each selected path circuit contains only the testable paths or critical paths. The additional path in the above ETGC is replaced by each of the selected path circuits.

The proposed method gets the testable path delay faults by using the recent ZBDD-based redundant identification scheme [13]. A series of selected path circuits (SPC) are constructed for fault simulation based on the testable paths, where the SPC circuit is fanout-free. Fault simulation for a test can be completed by a single pass if SPC circuit can be can be constructed within available memory limits in the computer.

The proposed test generation method can be extended to robust test genration based on a 10-valued logic system $s0$ (0,0), $\overline{s}0$ (0,1), $\overline{s}1$ (1,0), $s1$ (1,1), x0 (x,0), U, x1(x,1), U0, U1, and xx. Fig. 2 presents the Hasse diagram of the 10-valued logic system. As shown in Fig. 2, (1,0) and (0,1) are used to replace $\overline{s}1$ and $\overline{s}0$, respectively. Here, $\overline{s}1$ and $\overline{s}0$ are not only represent the falling and rising transitions, where the details can be found from [3].

The above scheme can be extended to robust test generation easily. We still use the circuit as shown in Fig. 1 to generate robust tests. Similarly, an extra target path is inserted the circuit, where the original target path and the extra target path share the same source node. All off-path inputs of the extra path are connected to the original off-path inputs in the original circuit.

Let us consider the falling transition at the source of the path without loss of generalization. Generating a robust test for the falling transition at the source of the target path is reduced to generation of a test for the stuck-at-1 fault on the source node a' of the extra path. In order to detect the path delay fault, a falling transition must be generated at a, which corresponds to the falling transition at a'. There exists a unique path from a' to its sink node e' as shown in Fig. 1, therefore, robust test generation of the target path delay fault can be replaced by test generation

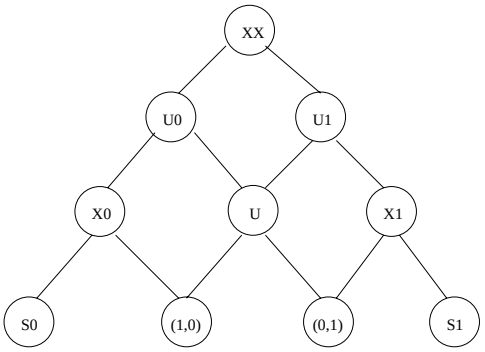

Figure 2: The Hasse diagram of the 10 values of the logic system.

of the stuck-at-1 fault on a'.

The nodes b', c', ..., d' must be assigned sensitization values in order to propagate the corresponding transitions robustly. If the gate fed by the on-path input is an AND or a NAND gate, its off-path inputs must be assigned $s1$ for a falling transition on the on-path input, and its off-path inputs must be assigned x1 for a rising transition. If the gate fed by the on-path input is an OR or a NOR gate, its off-path inputs must be assigned x0 for a falling transition on the on-path input, and its off-path inputs must be assigned s0 for a rising transition. Just like non-robust test generation for path delay faults, a single stuck-at test generator can be used to generate robust tests for path delay faults based on the 10-valued logic system and the constraints presented earlier in this paragraph.

IV. Dynamic Test Compaction Based on Output Cones

The output cone *out*(po) of a primary output (or pseudo primary output) *po* is defined as the subcircuit, where all nodes in the subcircuit reach *po*.

Lemma 1 *Test generation of two path delay faults with the same sink node can assign specified values to the same set of primary (or pseudo primary) inputs.*

Lemma 2 *Test vectors of two path delay faults can be compacted if the subsets of primary inputs (or pseudo primary inputs) that reach the sink nodes of the paths are disjoint.*

As shown in Fig. 3(a), influence cones of two path delay faults are presented. And influences of two path delay faults are clearly disjoint. Tests of two faults can be compacted certainly as shown in Fig. 3(b). We call the scheme output-cone-based test compaction scheme.

It is unnecessary to get the output cone for each path delay fault, which spends much CPU time. Path delay faults are classified based on output cones. It is interesting that we can only get the information whether a primary output (or pseudo primary output) is reachable from a primary input. This can be completed by calculating a separate list $reach(i)$ for output i before test generation in order to avoid repeated calculation, which includes all inputs that reach the output i. All path delay faults with

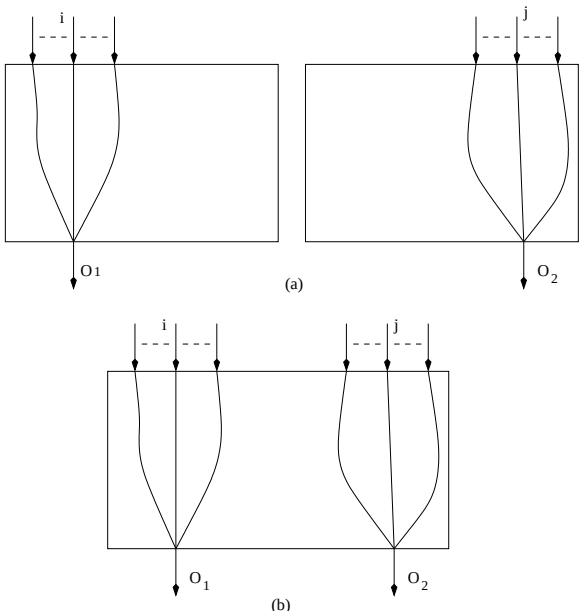

Figure 3: Dynamic compaction for tests of two path delay faults with disjoint output cones, (a) output cones of two path delay faults, and (b) compacting tests of two path delay faults with disjoint output cones.

test−compaction−with−output−cone( )

1. While the fault list is not empty, do 2, 3, 4
2. Set PI as the empty set; select a path p whose sink node is po; generate a test t for the path delay fault; add all primary inputs that have been assigned specified values to PI; dtc=0,*f(po)=1*;
3. while (dtc=0), do /* further compaction is possible. */
   (1) dtc =1; for each primary output po1 in POr , if*f(po1)=1*, (2)(3)(4);
   (2) for each pi in PI, if pi is not in*reach(po1)* ,*f(po1)=1*;
   (3) dtc =0, select a path from P(po 1);
   (4) generate a new test t' for p under the constraints given in the test t; t = t'; delete p from P(po 1). If P(po 1) has been empty, delete po 1 from PO r.
4. fault simulation with the test t on the selected testable path circuit for the remaining faults, delete all covered paths from the lists based on separate output cones. Delete po from PO r if the path list P(po) has been empty.

Figure 4: Dynamic test compaction for path delay faults based on output cones.

the same primary output (or pseudo primary output) have the same influence cone. Calculation of the reachability lists is trivial compared with the test generation and fault simulation time. The lists $reach()$ corresponding to each input of outputs can be completed as follows: Traverse the circuit from each primary input j or pseudo primary input level by level until reaching a primary output or a pseudo primary output. Put input j into the reachability lists of outputs that are reachable from input j.

A test is generated for a target path delay fault, which can assign 0, 1 values to all inputs in the output cone of the sink node of the path. A new target fault with a sink node i is selected, where i is unreachable of each of the inputs assigned a specified value. That is, test of a path with sink node whose output cone is disjoint with the inputs that have been assigned specified values can be compacted into the current test. A separate list $reach(i)$ is calculated for each output before test generation, which records the subset of inputs that reach the output i. A path with a

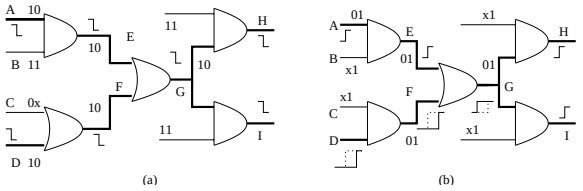

Figure 5: Test compaction for path delay faults with conjoint output cones: (a) Reasonable compaction, (b) unreasonable compaction.

sink node i can be compacted into the current test if all inputs in $reach(i)$ have not been assigned specified values. The above process continue until all outputs have been checked.

As shown in Fig. 4, inputs of the algorithm *compaction-with-output-cone*() is the selected testable paths, the fanout-free circuit formed by the selected testable paths. Path delay faults are classified based on outputs, where $P(po)$ store the subset of paths with the sink node po. The primary input set PI keeps the set of inputs that have been assigned specified values for the current compacted test. PO_r stores the set of primary outputs, for each $p \in PO_r$, P(p) contains at least one path that still have not been covered by a test. Initially, PO_r is the set of primary outputs or pseudo primary outputs that are the sink node of at least one testable path delay faults. We have $j \in reach(i)$ if output i is reachable from input j. Test of a path p can be compacted into the current test if no input that has been assigned a specified value reaches the sink node of the path p. Let po be the sink node of a path p. We have $f(po) = 1$ if a path with the sink node po has been compacted into the current test, or any path with the sink node po cannot be compacted into the current test. The calculation of the algorithm is very simple, where each output in PO_r is checked only once. For each output po, it is necessary to check whether each $pi \in reach(po)$ has been assigned a specified value. The computing complexity of procedure in Fig. 4 is $O(N \cdot \#PIs)$, where N and $\#PIs$ are the circuit size and the number of primary and pseudo primary inputs. However, the CPU time to traverse a circuit from a primary input or a pseudo primary input to the primary outputs or pseudo primary outputs that are reachable from the input needs CPU time much less than N.

A selected path circuit (SPC) is selected path based on the testable or critical path sets. Fault simulation is done in the SPC circuit. Fault simulation can be replaced by logic simulation, where fault dropping is still used to delete the SPC circuits. The SPC circuit is reduced in the process of test generation and fault simulation until it becomes empty. Selective tracing is also used in the process of logic simulation by only tracing the active part of the SPC circuit.

V. Dynamic Test Compaction for Paths with Conjoint Output Cones

It is possible for two path delay faults with the same influence cones or output cones to have the same test pattern. We may generate a test for several path delay faults although they have the same influence cones or output

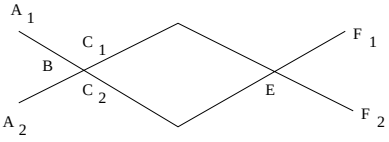

Figure 6: Uniform inversion parity feature of two conjoint paths.

cones. Multiple extra paths are inserted into the circuit as presented in Fig. 1. Test generation reduces to multiple goal test generation with all robust off-path contraints of the path delay faults. Usually, more compact test set can be obtained in this case, however, much more CPU time is necessary. A new dynamic test compaction scheme for path delay faults with conjoint influence cones or output cones is presented in this section.

The output cone and influence cone based dynamic test compaction schemes are pessimistic. Tests of two path delay faults can still be compacted although their influece cones or output cones are conjoint. The following conditions must be satisfied in order to compact the tests of two path delay faults into one,

- Two paths have the same transition at the common gate;
- the transition at the common gate is from the controlling value to the non-controlling value.

As shown in Fig. 5, two paths A-E-G-H and D-F-G-I have the same output cones. Fig. 5(a) presents a reasonable compaction. There exists a falling transition at the inputs of the gate G (from the controlling value to the non-controlling value). The delay of the transition on either path can be propagated to the output of gate G. The compaction presented in Fig. 5(a) can cover 4 path delay faults A-E-G-I(f), A-E-G-H(f), D-F-G-I(f), and D-F-G-H(f) simultaneously. However, Fig. 5(b) presents an unreasonable test compaction. Both transitions at E and F are from the non-controlling value to the controlling value although E and F have the same transition. The fault effect is masked at G consider the rising transition reaches F a little later. It is unable to demonstrate the fault effect at the output of gate G. Fig. 6 presents two paths A_1-B-C_2-E-F_1 and A_2-B-C_1-E-F_2 with two common nodes B and E, where C_1 and C_2 are fanout branches of node B. It is required that all inputs of each common node have the same transition from the controlling value to the non-controlling value.

Pomeranz and Reddy in [14] proposed length-based, valued-based and count-based dynamic test compaction schemes to reduce test sizes of path delay faults. The length-based scheme take priority to selecting critical paths to compact. That is, paths with greater length have greater priority to be compacted into the current test. The count-based scheme tries to compact a path into the current test, which includes as many as possible testable path delay faults. Kajihara *et al.* in [7] proposed a length-based static compaction scheme. Our method is based on the number of the fanouts in paths. That is, paths with more fanouts have greater priority to be considered. It is quite possible for a test vector to cover more testable paths when two paths with more fanouts can be detected by the same test vector. The fanout-based scheme can be easier to calculate than the count-based scheme, which presents more accurate information to generate compact tests. The fanout-based

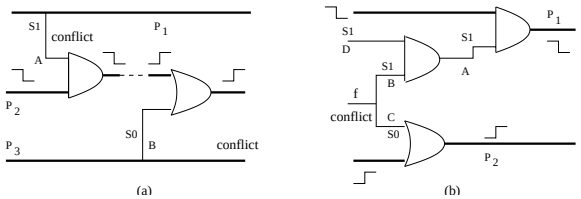

Figure 7: Off-path checking for incompatible paths: (a) Stable values must be assigned on target paths, (b) conflicts generated by off-path assignments and unique implication.

scheme can also compact a test of more testable paths into the same test than the length-based scheme. Benefits of this scheme can be shown from comparison of the proposed test compaction scheme with that in [14] in Table 2.

Path delay faults with more fanouts have greater priority to be considered after a test vector has been generated. Paths with enough number of fanouts are checked first in order to generate more compact tests. Our method uses on-path checking and off-path checking to exclude some paths. As for the cases as stated in Fig. 6, transitions of both paths can be propagated from the source nodes to the common fanouts, where both conditions can be checked easily. This on-path checking scheme can exclude many paths before test compaction. Off-path checking scheme tries to find incompatible paths as early as possible based on the unique implications of the necessary assignments on the off-path inputs. The bold-faced lines in Fig. 7 present the paths under consideration. As shown in Fig. 7(a), nodes A and B must be assigned $s1$ and $s0$, respectively in order to generate a robust test for the path delay fault P_2. Both values are in conflict with the on-path values of P_1 and P_3 because the on-path values of both paths must be transitions. Therefore, both paths are incompatible with P_2, and P_2 cannot have a common test with either path. As shown in Fig. 7(b), nodes C and A must be assigned values $s0$ and $s1$, respectively in order to generate robust tests for both paths. Nodes D and B must be assigned $s1$ simultaneously in order to set A to value $s1$. A conflict occurs at the fanout f as shown in Fig. 7(b).

Initially, the testable paths are stored in S, and the K longest paths are kept in S', where K is determined based on a trade-off between the cpu time to generate compact tests and the compactness. The proposed method selects a path delay fault p from S'. A test t is generated for p. Each path p' in S' is checked whether it is compatible with the current test t based on on-path checking and off-path checking as presented in Fig. 7. Paths with more fanouts have greater priority being selected for test compaction. A new test t' is generated under the constraints of t for fault p' if p' is compatible with the test t. Update t as t'. Continue the above process until all paths in S' has been checked. Fault simulation is completed for the test t on the paths in S. All paths P covered by the test t are deleted from S, and all paths P' covered by the test t are deleted from S'. $|P'|+1$ longest new paths are selected from S, which are added to S'. One of the paths in the original set S' is selected as the next target path delay fault. Continue the above process until S has been empty. The above procedure tries to find a trade-off between the cpu time and the compactness of the tests unlike many of the previous methods tries all faults for each test. The technique to make the number of path delay faults in S' unchanging is to guarantee compactness.

VI. Experimental Results

The proposed test generation method is implemented using C language and runs on a Blade 2000 workstation with two 900MHz CPU. Table 1 presents performance of the proposed path delay fault test generation method. All circuits used are the combinational parts of the ISCAS89 circuits. In Column 1 of Table 1, *paths* represents the number of nonrobust and robust testable path using the algorithm presented in [13]. The parameter *comp.* represents the average number of detected paths by a single test. The base test generation algorithm is the ATALANTA [9], which used most important techniques of the FAN algorithm [4]. The parameter *vec.* stands for the number of test vectors generated. The cpu time includes 5 different parts: *dyn.*, *atpg*, *init.*, *fsim* and *static* represent the time used by the dynamic test compaction scheme, cpu time used by the ATALANTA algorithm, the time used to construct the fault simulation circuit, the time used for fault simulation, and the cpu time required by the ATALANTA to do some simple static compaction after test generation, respectively. All circuits obtain *complete fault coverage* for the complete nonrobustly and robustly testable path sets in reasonable time. The dynamic test compaction schemes gets better compactness for all circuits. The last five rows in Table 1 present results of the proposed method on the selected critical paths.

The proposed test generation scheme called SPC with effective dynamic test compaction schemes is compared with the most recent test generation methods [11, 14] in Table 2. We present comparison between two methods for the smaller ISCAS89 circuits on compactness and cpu time because [11, 14] did not present results of the larger circuits. The proposed method gets better compactness for almost all circuits for the non-robust test sets and robust test sets.

VII. Conclusions

A new test generation scheme for complete coverage of robustly and non-robustly testable path delay faults in combinational or fully scanned circuits was proposed based on single stuck-at fault tests without circuit transformation. A disjoint dynamic test compaction scheme called the output-cone-based method was proposed. The output-cone-based method does not select a path, where the output cone of its sink node contains one of the inputs that have been assigned specified values. Another conjoint dynamic test compaction scheme based on the fanout count was proposed. Experimental results were presented to compare with two recent test generation methods on compactness.

References

[1] S. Bose, P. Agrawal, and V. D. Agrawal, "Generation of compact delay tests by multiple path activation," *Proc. of IEEE Int. Test Conference*, pp. 714-723, 1993.

Table 1: Performance of the Proposed Path Delay Fault Test Generation Method

circuits	paths	detected paths	robust							non−robust								
			vec	comp	cpu(s)					paths	detected paths	vec	comp	cpu(s)				
					init	dyn	atpg	fsim	static					init	dyn	atpg	fsim	static
s1423	28696	28696	4508	6.37	2.3	49.2	4489	137.4	71.3	45198	45198	2215	20.41	4.72	8.28	487	108	108
s5378	18656	18656	929	20.08	1.07	10.7	3308	13.4	0	21928	21928	467	46.96	1.12	1.53	209	8.53	5.85
s9234	21389	21363	1728	12.36	2.1	7.0	5525	42	0	59854	59854	1107	54.07	7.42	6.67	1044	91	68
s13207	27603	27603	2727	10.12	4.17	41.9	19635	78.7	51.7	476143	476143	2439	195.2	162	54	4009	1238	1646
s15850	182673	182673	7557	24.17	478	82.8	26960	5813	7874	121525	121525	2417	50.28	447	22.6	2782	3583	3391
s35932	21783	21783	278	78.36	6.4	4.2	16423	15.7	14.3	58657	58657	69	850.1	18.2	1.35	1092	9.97	8.43
s38417	598062	598062	32348	18.49	585	1188	183616	20625	19320	1138194	1138194	15658	72.69	1114	501	1017	15471	0
s38584	92239	92239	3484	26.48	18.3	31.5	22533	282	250	334922	334922	3842	87.17	188	35	3384	1281	1464

Table 2: Comparison with Two of the Recent Test Generation Methods

circuits	robust				non−robust			
	paths	SPC (conj.)	enrich [14]	Neat [11]	paths	SPC (disconj.)	SPC (conj.)	Neat [11]
		vec/comp	vec/comp	vec/comp		vec/comp	vec/comp	vec/comp
s298	343	62/5.53	64/5.36	**61 / 5.62**	364	40 / 9.10	**30/12.13**	64/5.68
s344	611	**95/6.43**	98/6.23	96 / 6.36	654	65 / 10.06	**44/14.86**	102/6.19
s349	611	**95/6.43**	—	108/ 5.65	656	72 / 9.11	**44/14.91**	97/6.53
s382	667	**103/6.48**	106/6.29	110/ 6.06	734	86 / 8.53	**60/12.23**	118/6.22
s386	413	120/3.44	**118/3.50**	**118/3.50**	414	76 / 6.46	**68/6.09**	101/4.10
s400	663	102/6.50	102/6.50	101/ 6.50	753	84/ 8.96	**60/12.55**	107/7.05
s420	738	**244/3.02**	282/2.62	—	738	202 / 3.65	**184/4.01**	310/2.38
s444	586	105/5.58	**97/6.03**	—	813	85 / 9.56	**60/13.55**	83/7.05
s510	729	**219/3.33**	227/3.21	—	738	80 / 9.23	**68/10.85**	—
s526	694	**127/5.46**	131/5.30	133/ 5.77	720	90 / 8.0	**81/8.89**	116/6.21
s641	1979	**183/10.8**	187/10.58	186/10.59	2270	246/ 9.23	**117/19.40**	181/12.54
s713	1184	**92/12.87**	—	205/5.77	4922	246/20.01	**118/41.71**	259/19.0
s820	980	**249/3.94**	250/3.92	250/3.92	984	120 / 8.20	**100/9.84**	209/4.71
s832	984	**255/3.86**	—	265/3.71	996	124/ 8.03	**102/9.76**	210/4.74
s838	2018	**681/2.96**	—	—	2018	615/ 3.28	**595/3.39**	—
s953	2302	**399/5.77**	411/5.60	411/5.60	2312	251/ 9.21	**201/11.5**	361/6.4
s1196	3581	**552/6.49**	556/6.44	555/6.45	3759	413/ 9.10	**314/11.97**	477/7.88
s1238	3589	**522/6.88**	—	595/6.03	3684	415/ 8.88	**305/12.08**	416/8.85
s1488	1875	395/4.75	**390/4.81**	—	1916	160/11.97	**127/15.09**	—
s1494	1882	**387/4.74**	—	—	1927	160/12.04	**129/14.94**	—

[2] K. T. Cheng and H. C. Chen, "Classification and identification of nonrobust untestable path delay faults," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 8, Aug. 1996.

[3] K. Fuchs, F. Fink, and M. H. Schultz, "DYNAMITE: An efficient automatic test pattern generation system for path delay faults," *IEEE Trans. on Computer-Aided Design*, vol. 10, no. 10, pp. 1323-1335, 1991.

[4] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. on Computers*, vol. 32, no. 12, pp. 1137-1144, Dec. 1983.

[5] M. A. Gharaybeh, M. L. Bushnell, and V. D. Agrawal, "Classification and test generation for path delay faults using single stuck-fault tests," Proc. of *IEEE Int. Test Conference*, pp. 139-148, 1995.

[6] P. Goel, "An implicit enumeration algorithm to generate tests for combinational circuits," *IEEE Trans. on Computers*, vol. 30, no.3, pp. 215-222, Mar. 1981.

[7] S. Kajihara, M. Fukunaga, X. Wen, T. Maeda, S. Hamada, and Y. Sato, "Path delay test compaction with process variation tolerance," in *Proc. of 42th ACM/IEEE Design Automation Conference*, pp. 845-850, 2005.

[8] A. Krstic and K. T. Cheng, *Delay Fault Testing for VLSI Circuits,* Kluwer Academic Publishers, 1998.

[9] H. K. Lee and D. S. Ha, "On the generation of test patterns for combinational circuits," Technical report12-93, Dept. of Electrical Eng., Virginia Polytechnic Institute and State University, 1993.

[10] X. Lin, I. Pomeranz, and S. M. Reddy, "On static test compaction and test pattern ordering for scan designs," Proc. of *IEEE Int. Test Conference*, pp. 1088-1097, 2001.

[11] M. K. Michael and S. Tragoudas, "Function-based compact test pattern generation for path delay faults," *IEEE Trans. on VLSI Systems*, vol. 13, no. 8, pp. 996-1001, 2005.

[12] S. Ohtake, K. Ohtani, and H. Fujiwara, "A method of test generation for path delay faults using stuck-at fault test generation algorithms," Proc. of *IEEE Int. Conf. on Design, Automation and Test in Europe*, pp. 310-315, 2003.

[13] S. Padmanaban and S. Tragoudas, "Efficient identification of (critical) testable path delay faults using decision diagrams," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 77-87, 2005.

[14] I. Pomeranz and S.M. Reddy, "Test enrichment for path delay faults using multiple sets of target faults," *IEEE Trans. on Computer-Aided Design*, vol. 22, no. 1, pp. 82-90, 2003.

[15] A. Saldanha, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Equivalence of robust delay-fault and single stuck-at fault test generation," Proc. of 29th *ACM/IEEE Design Automation Conference*, pp. 173-176, 1992.

[16] J. Saxena and D. K. Pradhan, "A method to derive compact test sets for path delay faults in combinational circuits," in *Proc. of IEEE Int. Test Conference*, pp. 724-733, 1993.

[17] M. Schulz, E. Frischler, and T. M. Sarfert, "SOCRATES: A highly efficient test pattern generation system," *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 1, pp. 126-137, 1988.