On Test Generation for Transition Faults with Minimized Peak Power Dissipation

Wei Li [1] Sudhakar M. Reddy [1]
Dept. of ECE
Univ. of Iowa
Iowa City, IA 52242
USA
{li100, reddy}@engineering.uiowa.edu

Irith Pomeranz [2]
School of ECE
Purdue University
West Lafayette, IN 47907
USA
pomeranz@ecn.purdue.edu

ABSTRACT
This paper presents a method of generating tests for transition faults using tests for stuck-at faults such that the peak power is the minimum possible using a given set of tests for stuck-at faults. The proposed method is suitable for use in testing scan designs that employ enhanced scan. The method reduces the peak power consumption in benchmark circuits by 19% on the average with essentially the same test set size and the same fault coverage compared to an earlier method.

Categories and subject descriptors
B.8.1 [**Performance and Reliability**]: Reliability, Testing and Fault-Tolerance

General terms
Algorithm, Design, Reliability

Keywords
Power Dissipation, Test Generation, Transition Faults

1. INTRODUCTION
Physical defects that cause the signal propagation delays in circuits to increase can be modeled as delay faults. Two general types of delay fault models, the gate delay fault model [1,2] and the path delay fault model [3,4], have been used to model delay defects. The path delay fault model is more comprehensive in modeling delay defects, but often difficult to use in practice due to the large number of paths in large circuits. The gate delay fault model is more practical for large circuits. The most commonly used gate delay fault model is the transition fault model [1]. Under the transition fault model, there are two types of transition faults, slow-to-rise faults and slow-to-fall faults. A test pattern pair <V1, V2> is required to detect the transition fault *f* on line *l*. Two conditions have to be satisfied by <V1, V2> to detect the transition fault. (1) The initial vector V1 must set the target node to an initial value 0 for the slow-to-rise fault and to an initial value 1 for the slow-to-fall fault. (2) The test vector V2 has to launch the corresponding transition at the target node and also propagate the fault effect to the primary output. Thus, V2 is a test for a s-a-0 (s-a-1) fault if the transition fault is the slow-to-rise (slow-to-fall) fault. Many works have been reported on generating and compacting tests for transition faults [3-6].

It is known that power dissipation may be considerably higher in test mode than in the normal system mode [7,8]. During normal operation, typically a relatively small portion of the flip-flops change value in each clock cycle. However while scanning in test vectors as well as during capture cycles, many more flip-flops may change value in each clock cycle, thus causing much higher average and peak power dissipation than in the normal operation mode. This can cause problems both with heat dissipation and with current spikes. Another problem arises when wafer probe testing is done through spring loaded contacts which may not be able to deliver large enough current to meet the peak current requirement of some scan tests. This may cause a chip to fail a test even when it is defect-free, thus reducing the yield. Recently, yield loss due to voltage drops caused by higher peak current during test of an ASIC chip has been reported in [9]. Using tests with reduced peak current requirement will help reduce the possibility of such occurrences.

1. Research supported in part by NSF Grant No. CCR-0097905 and in part by SRC Grant No. 2001-TJ-949
2. Research supported in part by NSF Grant No. CCR-0098091 and in part by SRC Grant No. 2001-TJ-950

DAC 2004, June 7–11, 2004, San Diego, California, USA

Various techniques have been proposed to reduce power dissipation during test. Two examples are [10] and [11] that consider standard scan designs.

In this paper, we present a power conscious transition fault ATPG scheme based on the work in [6]. Our scheme consists of three phases. First, we use a MinMax Spanning Tree (MMST) [18] algorithm to construct tests for the easy-to-detect faults. Second, we compute the minimum peak power dissipation value without compromising transition fault coverage when tests generated from a given stuck-at fault test set are used. Then, we use this minimum value as a constraint to select test pairs for the remaining faults. Finally, in the third phase we use forward-looking fault simulation [13] to compact the test set. The proposed scheme keeps all the advantages noted in [6] and in addition achieves minimum peak power dissipation. Compared to the method in [6], the peak power reduction is 19% on the average and up to 40%.

The remainder of the paper is organized as follows. Section 2 gives definitions of terms used in this paper and a brief review of the previous work in [6]. In Section 3 we formulate the problem and give a sketch of the proposed method. Section 4 introduces the MMST based algorithm. In Section 5, we give the method to select tests for the remaining transition faults under the peak power constraint. In Section 6, the test compaction method is discussed. In Section 7 experimental results are presented. Section 8 concludes the paper.

2. PRELIMINARIES AND REVIEW

2.1 Terminology and background

Below we give the definitions for some terms used in this paper.

Definition 1: The *weighted switching activity (WSA)* of a node is the number of state changes at the node multiplied by (1+node fan-out). The "1" in the formula is to include the output capacitance of the driving gate. The WSA of the entire circuit is obtained by summing the WSA of all the nodes in the circuit.

WSA is used to represent the power dissipated in a circuit when a two-pattern test is applied. WSA was also used to represent instantaneous power in earlier works [11]. Instead one can use an actual power calculation, but it requires a large computation time.

Definition 2: The *peak power* is the highest value of power dissipation at any given instant of switching activity caused by test application. In this paper, the highest value of WSA is used as the peak power.

2.2 Test application

Transition fault tests can be applied in three different ways: broadside [14], skewed-load [15] and enhanced-scan [16]. In this paper, we only consider enhanced-scan transition fault tests.

When enhanced scan is used, scan cells are made up of three latches. During scan-in of a new vector the previously applied vector is stored in the third latches of the scan flip-flops, which drive the combinational logic of the circuit under test. For this reason during scan-in no power is dissipated in the combinational logic of the circuit under test. Power is dissipated in the combinational logic only during capture cycles.

2.3 Review of previous work

For designs with enhanced scan one can generate tests for transition faults by composing two-pattern tests from tests for stuck-at faults [6,12]. Compared to the conventional test generation and test application methods for transition faults in standard scan designs, this has the following advantages [6].

- Higher fault coverage compared to that achievable in standard scan designs. Enhanced scan allows the detection of all the irredundant transition faults whereas standard scan based tests may not detect all the irredundant faults.
- Low average power dissipation due to the use of enhanced scan cells, since the inputs to the combinational logic are held at a constant value while the new state is scanned in.
- Reduced storage space. Normally the size of a transition fault test set is four to five times larger than the size of a stuck-at fault test set. Since the transition fault tests are chosen from a stuck-at fault test set, only the stuck-at fault tests need to be stored and the test data volume is significantly reduced.
- Simplified validation process. Since the stuck-at fault tests have gone through a validation process, the validation process for the transition fault tests is unnecessary.

Despite all the advantages above, the peak power during test may exceed that in normal operation since there is no power saving action addressing the capture cycle.

3. PROBLEM FORMULATION

A transition fault test set for an enhanced scan design consists of a sequence of primary input vectors and state vectors that are scanned in. For the sake of simplicity of explanation, we assume as in [6] that all the inputs (i.e., primary inputs and state variables) are scanned. For this case, let $T_s=<t_1, t_2, \ldots, t_N>$ be a sequence of test vectors scanned in and let T_c be a test set for stuck-at faults. We assume that all the tests in T_c have to be applied in some order. A pair of consecutive vectors of T_s constitutes a two-pattern test for transition faults. For example $<t_1, t_2>$, $<t_2, t_3>$, …,$<t_i, t_{i+1}>$ form two-pattern tests. If the transition fault tests are formed from tests for stuck-at faults, then each t_i is a member of the test set T_c for stuck-at faults. In the test

sequence T_s for transition faults, some of the tests in T_c may appear more than once. One obtains a two-pattern test <t_1, t_2> for say a slow-to-rise fault on circuit line *r* by using a test, say t_1, to detect the s-a-1 fault on line *r* (note that t_1 sets line r to 0) and a test t_2 that detects the s-a-0 fault on *r*. Actually, any test in T_c that sets *r* to 0 can be used for t_1 (even if it does not detect a s-a-1 fault on r) and any test in T_c that detects a s-a-0 fault on *r* can be used for t_2.

In order to construct a sequence T_s for transition faults from a test set T_c such that the maximum WSA for any two-pattern test is the minimum possible using the tests in T_c, we create a graph $G_s(V,E)$, which is a complete graph called the transition graph. In G_s there is a node corresponding to each test in T_c and the edge (t_i, t_j) has weight w_{ij} which is equal to the WSA during the capture cycle when the two-pattern test <t_i, t_j> is applied. Since the WSA for (t_i, t_j) is equal to the WSA for (t_j, t_i), G_s is an undirected graph with a weight $w_{ij}=w_{ji}$ on the edge between t_i and t_j. An example given below illustrates the construction of G_s and how transition fault tests with minimum WSA can be obtained from G_s.

Example: Let T_c contain four tests {t_1, t_2, t_3, t_4}. The corresponding G_s is given in Figure 1. In G_s traversing an edge corresponds to a two-pattern test, which also depends on the direction of traversal. For example <t_1, t_2> and <t_2, t_1> are two different two-pattern tests that detect different transition faults. Thus there are twelve different two-pattern tests corresponding to the six edges in G_s of Figure 1. From these tests we need to select a minimal number of tests to detect all the transition faults that can be detected if all twelve tests are applied. In addition, the maximum of the WSAs in the circuit for all the two-pattern tests used should be minimum. A test sequence T_s corresponds to traversing the graph G_s starting from a node corresponding to the first test in T_s. For example, the test sequence T_s = <t_1, t_2, t_3, t_2, t_4> corresponds to traversing G_s starting from t_1. The two-pattern tests obtained from T_s are <t_1, t_2>, <t_2, t_3>, <t_3, t_2> and <t_2, t_4>.

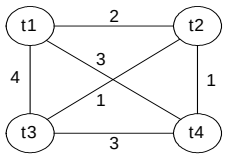

Figure 1. An example transition graph

In the method we propose the test sequence T_s is constructed in three phases. In the first phase we construct a sequence T_{s1} that includes all the tests in T_c at least once and has the property that the maximum value of WSA for any two consecutive tests in T_{s1} is minimum. To define T_{s1}, we determine a path in G_s such that the path includes every node in G_s at least once and the largest weight of an edge included in the path is the minimum among all the paths that can be constructed to include each node of G_s at least once. It is well known that an algorithm based on Minimum Spanning Tree traversal provides a two-approximate solution for a special case of TSP (the Traveling Salesman Problem) called ΔTSP, which requires visiting each node exactly once with minimum cost [17]. The cost in this case is the sum of the weights of the edges in the path used to visit the nodes. In the problem we consider the goal is to visit each node at least once such that the maximum weight of an edge in the selected path is minimum. To obtain such a path we use a MinMax Spanning Tree [18] (defined later in this section) for G_s. In the second phase we extend T_{s1} as follows. First we simulate the two-pattern tests obtained from T_{s1} to drop all the transition faults detected by them. T_{s1} is extended into a sequence T_s that includes tests to detect yet-undetected transition faults such that the maximum WSA of the tests included in the extended test sequence is again the minimum among all the possible extensions. The extended sequence T_s is compacted without increasing the maximum WSA in the third phase to obtain a minimal length test sequence.

Definition 3: Given a graph G(V,E), a *spanning tree* G_T is a connected, cycle-free subgraph that includes all the nodes in G.

Definition 4: Given a graph G(V,E) with weighted edges, a spanning tree is called a *MinMax Spanning Tree* (MMST) if the maximum of the weights of the edges in G_T is the minimum among all the spanning trees of G.

For example, a MMST for the transition graph of Figure 1 is given in Figure 2. In the next three sections we give the details of the proposed method to generate transition fault tests from a given stuck-at fault test set.

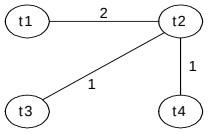

Figure 2. A MMST for the transition graph of Figure 1

4. TEST SEQUENCE BASED ON A MMST

The first part of the test sequence for transition faults is derived by constructing a MMST for the transition graph of a given stuck-at test set and traversing the graph using the following procedure.

Procedure_MMST(G_s(V,E))

Given: A transition graph $G_s(V,E)$ corresponding to a stuck-at fault test set T_c

Step 1: Sort the edges in E by weight into a list E'

Step 2: Remove from E' all the edges whose weight is equal to the maximum edge weight

Step 3: Check to see if a spanning tree made up of edges from E' exists for the graph. If such a spanning tree exists, go back to Step 2. Otherwise go to Step 4.

Step 4: Restore the edges removed in the last execution of Step 2 into E'

Step 5: Construct a spanning tree for G_s. This is a MMST for the given G_s. The method we use to construct a spanning tree is discussed below.

Step 6: Find a path based on the spanning tree constructed in Step 5. The method we use to construct the path is discussed below. This path defines the first part T_{s1} of the transition fault test sequence.

To generate a spanning tree in Step 5 we apply the Depth-First Search (DFS) algorithm to the graph G_s. Since G_s is connected, the order by which the nodes are visited for the first time defines a spanning tree called the DFS tree [17]. For example, suppose that the graph shown in Figure 3(a) is obtained after Step 4. Let us start the DFS algorithm from t_1. Suppose that t_2 is visited first. The algorithm then backtracks to t_1. Suppose that t_3 and t_4 are visited next. Then the spanning tree is the one shown in Figure 3(b).

To construct the path in Step 6 we traverse the spanning tree constructed in Step 5 in depth first manner such that a subtree with a lower depth is traversed earlier. For example, in the spanning tree of Figure 3(b) we first traverse the subtree rooted at t_2 as it has fewer levels than the subtree rooted at t_3 which will be traversed later. The traversal of the subtree rooted at t_2 results in the subsequence t1→t2. Next, we return to t1 and traverse the subtree rooted at t3. This results in the sequence $t_1 \rightarrow t_2 \rightarrow t_1 \rightarrow t_3 \rightarrow t_4$.

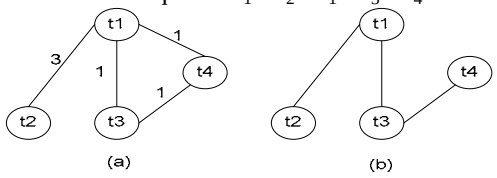

Figure 3. Method to construct depth-first MMST

When we traverse the MMST as discussed above, some nodes may be visited more than once. We shortcut a node that has been visited before by using an edge not in the MMST if the weight of the edge does not exceed the maximum weight of the edges in the MMST; otherwise the shortcut is not taken. The resulting test sequence is called T_{s1}.

The test sequence T_{s1} is fault simulated and all the detected transition faults are dropped. To detect the yet-undetected faults the procedure described in the next section is used for extending T_{s1}.

5. FAULT-ORIENTED TEST SELECTION

We extend the sequence T_{s1} into a sequence T_s by including tests for the transition faults not detected by T_{s1}. Let F_r be the set of remaining transition faults. The extension has two parts. In the first part we determine the minimum value of WSA needed to detect the transition faults in F_r that are detectable by using tests in T_c. We also order these detectable transition faults. In the second part of this phase we extend T_{s1} to detect all the detectable transition faults in F_r.

5.1 Determination of detectable transition faults and minimum WSA

We know that for every transition fault detected there is a corresponding stuck-at fault that is detected. We determine the set of stuck-at faults called F_s corresponding to the transition faults in F_r. We fault simulate without fault dropping the stuck-at faults in F_s using the tests in T_c. During the fault simulation process, for each stuck-at-α fault *f* in F_s we determine the set of tests in T_c that detect the fault *f* as well as the set of tests that set the faulty line to α. These sets of tests in T_c correspond to the tests that can detect the corresponding transition fault and initialize the fault, respectively. If the set of initializing tests or the set of detection tests for some transition fault in F_r is empty, then the fault is not detectable using the tests in the given T_c. Undetectable transition faults in F_r are dropped.

Knowing the set of tests in T_c that can initialize a transition fault *f* and the set of tests in T_c that can detect *f*, we also know the set of two-pattern tests formed out of tests in T_c that detect *f*. Thus for each transition fault *f* in F_r, we can determine from G_s the minimum WSA, denoted by $mWSA_f$, needed to detect the fault. Then the minimum WSA needed to detect all the detectable transition faults, denoted by WSA_{min}, is:

$$WSA_{min} = \max(W1, \max_{f} (mWSA_f)),$$

where W1 is the maximum WSA of an edge in the MMST found in Phase 1.

For each detectable transition fault *f*, we determine a level of difficulty which is equal to the minimum of the number of tests in T_c that initialize the fault and the number of tests in T_c that detect the fault. We then order the faults in F_r in ascending order of their level of difficulty. Let the ordered set of detectable transition faults be F_r'. In the next step described below we extend the test sequence T_{s1} obtained after Phase 1 to detect the faults in F_r'. We consider the faults in F_r' in order.

5.2 Extending T_{s1}

We extend T_{s1} in three ways to detect faults in F_r. These are illustrated in Figure 4. In Figure 4(a) a sequence T_{s1} with tests from T_c is given. This sequence yields three two-pattern tests $\langle t_1, t_2\rangle$, $\langle t_2, t_3\rangle$, $\langle t_3, t_4\rangle$. This sequence can be extended in three different ways depicted in Figure 4(b) through 4(d). The extended sequence of Figure 4(b) gives one additional two-pattern test $\langle t_4, t_5\rangle$, whereas the extended sequences of Figure 4(c) and Figure 4(d) give two additional two-pattern tests. The sequence of Figure 4(c) gives additional tests $\langle t_4, t_5\rangle$ and $\langle t_5, t_6\rangle$, and the sequence of Figure 4(d) gives additional tests $\langle t_2, t_5\rangle$ and $\langle t_5, t_2\rangle$.

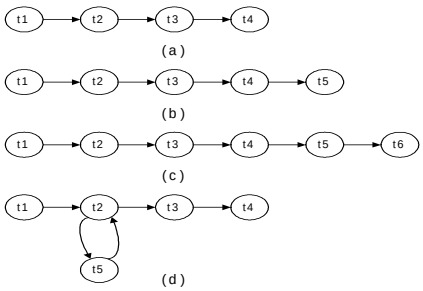

Figure 4. Methods to extend T_{s1}

In extending T_{s1} into a test sequence T_s we pick a yet-undetected transition fault, say *f*, at the top of the ordered set of faults F_r' and then extend the current test sequence by one or two nodes as illustrated in Figure 4 such that the resulting two-pattern tests detect *f* and none of the two-pattern tests added by this extension require WSA greater than WSA_{min} determined in the first part of this phase. We fault simulate the two-pattern tests added by the extensions and remove all the detected faults from F_r'. We repeat the extension step until all the faults in F_r' are detected. The final test sequence obtained is called T_s.

An extension of T_{s1} to detect all the faults in F_r' always exists for the following reason. Consider an arbitrary fault in F_r' that can be detected by a two-pattern test $<t_i, t_j>$. Since t_i exists in T_{s1}, and since WSA is the same for $<t_i, t_j>$ and $<t_j, t_i>$, the extension of Figure 4(d) can be used to add $<t_i, t_j>$ to T_{s1} and detect the fault.

6. COMPACTION

After the fault-oriented test selection described above, the test sequence obtained guarantees the detection of all the detectable faults with a minimum peak WSA. As is well known, test sequences can be compacted further in a post-processing step. We use forward-looking fault simulation [13] to compact the test sequence. Compared to reverse-order fault simulation, forward-looking simulation can often achieve a larger reduction of test set size with a reduced run time.

For transition fault tests which are given as a sequence of tests, we have to make sure the resulting sequence would not violate the peak power constraint of WSA_{min} before we drop any test from the sequence during test compaction. Next we discuss the procedure we use to compact T_s. Let T_s be $\{t_1, t_2, \ldots, t_n\}$ where every t_i, $1 \le i \le n$, is an element from the test set T_c.

- Mark all the test vectors from t_1 to t_n as UNUSED.
- Conduct forward-looking transition fault simulation on pairs $\{<t_1, t_2>, <t_2, t_3>, \ldots, <t_{n-1}, t_n>\}$. If any pair does not detect any new fault or should be skipped without simulation (a feature of forward-looking fault simulation [13]), continue with the next test. Otherwise, mark both vectors in the pair as USED.
- Scan the test sequence $\{t_1, t_2, \ldots, t_n\}$ from the start vector t_1. For every UNUSED segment (a segment such that all its vectors are UNUSED): (a) If the UNUSED segment is at the start or the end of T_s, remove it. (b) If the UNUSED segment is between two USED vectors t_i and t_j and if $WSA(t_i, t_j)$ is no greater than WSA_{min}, remove the entire segment. Otherwise try to replace the UNUSED segment with an element t_k from T_c such that both $WSA(t_i, t_k)$ and $WSA(t_k, t_j)$ are less than or equal to WSA_{min}. If the entire UNUSED segment cannot be replaced by a test t_k, shorten the UNUSED segment by marking its last element as USED and try to remove the shortened UNUSED segment or replace it with a single element of T_c. Repeat the last step as necessary to determine if a subsequence of the UNUSED segment can be deleted.

7. EXPERIMENTAL RESULTS

We implemented the scheme described above in C and experimented with ISCAS89 benchmark circuits. The experiments were run on a Linux machine with a 1400 MHz Intel Pentium IV processor and 512MB memory.

The stuck-at fault test set we used can detect all the irredundant stuck-at faults [19]. We reproduced the procedure in [6] for comparison. For a fair comparison, we made the following two changes in this reproduced procedure. i) We use forward-looking fault simulation in place of reverse order fault simulation used in [6] since forward-looking fault simulation gives higher compaction. ii) We apply forward-looking fault simulation using all the test vectors instead of just the vectors from the fault-oriented phase (i.e., Phase 2) as done in [6]. This gives shorter test sequences.

Experimental results are given in Table 1. After the circuit name, we give the number of tests in T_c and the number of transition faults (the set of transition faults is denoted by F_{tr}). Next, we give the results obtained by the method in [6] including the number of transition fault tests in T_s and the number of transition faults detected. The scheme in [6] uses an arbitrary order to visit tests in T_c in the first phase and thus may result in different peak WSA values depending on the order. For this reason, we conducted experiments using ten different random orders and recorded both the average peak WSA and maximum peak WSA from these ten experiments. These are reported in the next two columns of Table 1. The run time of the procedure from [6] is given next. The experimental data obtained by using the proposed method are reported in Table 1 next. The first two columns under the proposed method are similar to those for the method of [6]. Next the peak WSA and percentage reduction in WSA compared to the average peak WSA and the maximum peak WSA for the method of [6] is given. Finally, the run time is given.

From Table 1, it can be seen that the proposed method achieves the same transition fault coverage as [6] with reduced peak power dissipation. Compared to the peak WSA in [6], the proposed method reduces the peak WSA by 19% on the average. The run time for the proposed method is higher. The number of tests for the proposed method is similar to the number of tests for the method of [6].

8. CONCLUSION

In this work, we proposed a new low power transition fault ATPG scheme for designs with enhanced scan based on reusing stuck-at fault tests. The peak power requirement of the proposed method is minimum under the constraint of i) a given stuck-at fault test set and ii) uncompromised transition fault coverage. Experimental results showed that the proposed scheme achieves the same transition fault coverage while reducing the peak power by 19% on the average compared to an earlier method.

9. REFERENCES

[1] J. A. Waicukauski, E. Lindbloom, B. K. Rosen and V. S. Iyengar, "Transition Fault Simulation", IEEE Design & Test of Computers, Vol. 4, No. 2, April 1987.

[2] A. K. Pramanick and S. M. Reddy, "On the Detection of Delay Faults", Proc. ITC, pp. 845-856, Sept. 1988

[3] G. L. Smith, "Model for Delay Faults Based Upon Paths", Proc. ITC, pp. 342-349, Sept. 1985

[4] C. J. Lin and S. M. Reddy, "On Delay Fault Testing in Logic Circuits", IEEE TCAD, pp. 694-703, Sept. 1985

[5] I. Pomeranz and S. M. Reddy, "Static Compaction for Two-Pattern Test Sets", Proc. ATS, pp. 222-228, 1995

[6] X. Liu, M. S. Hsiao, S. Chakravarty and P. J. Thadikaran, "Novel ATPG Algorithms for Transition Faults", Proc. ETW, pp. 47-52, May 2002

[7] H. J. Wunderlich and Y. Zorian, "Built-In Self Test (BIST) Synthesis of Self-Testable Systems", Kluwer Academic Publisher, 1997.

[8] Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices", Proc. VTS, pp. 4-9, 1993

[9] J. Saxena, et. al., "A Case Study of IR-Drop in Structured At-Speed Testing", Proc. ITC, 2003. pp.1098-1104

[10] V. Dabholkar, S. Chakravarty, I. Pomeranz and S. M. Reddy, "Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application", IEEE TCAD, pp. 1325-1333, 1998

[11] S. Gerstendorfer and H. J. Wunderlich, "Minimized Power Consumption for Scan-based BIST", Proc. ITC, pp. 77-84, 1999.

[12] I. Pomeranz and S.M. Reddy, "On n-Detection Test Sets and Variable n-Detection Test Sets for Transition Faults", IEEE TCAD, March 2000, pp. 372-383.

[13] I. Pomeranz and S. M. Reddy, "Forward-Looking Fault Simulation for Improved Static Compaction", IEEE TCAD, pp. 1262-1265, 2001

[14] J. Savir and S. Patil, "On Broad-Side Delay Test", Proc. VTS, pp. 284-290, Sept. 1994

[15] J. Savir and S. Patil, "Scan-Based Transition Test", IEEE TCAD, pp. 1232-1241, August 1993

[16] B. Dervisoglu and G. Stong, "Design for Testability: Using Scanpath Techniques for Path-Delay Test and Measurement", Proc. ITC, pp.365-374, 1991

[17] T. H. Cormen, C. E. Leiserson and R. L. Rivest, "Introduction to Algorithms", MIT Press and McGraw-Hill Book Company, 6th edition, 1992

[18] W. Cook, W. H. Cunningham, W. Pulleyblank and A. Schrijver, "Combinatorial Optimization", John Wiley & Sons, 1998

[19] S. Kajihara, I. Pomeranz, K. Kinoshita and S. M. Reddy, "Cost-Effective Generation of Minimum Test Sets for Stuck-at Faults in Combinational Logic Circuits", IEEE TCAD, pp. 1496-1504, Dec. 1995.

Table 1 Experimental Results

	Size of T_c&F_{tr}		The method in [6]					The proposed method					
	$\|T_c\|$	F_{tr}	$\|T_s\|$	tr_det	avg peak	max peak	time (s)	$\|T_s\|$	tr_det	peak	avg%	max%	time (s)
s1196	113	2110	343	2110	810	888	0.63	379	2110	486	40.00	45.27	2.40
s9234	111	10700	595	10694	7226	7393	13.17	578	10694	5605	22.43	24.19	41.03
s13207	235	15387	895	15379	10558	10669	28.45	890	15379	8267	21.70	22.51	186.97
s15850	97	18427	610	18401	12901	13140	25.22	614	18401	11300	12.41	14.00	61.17
s38417	87	49558	671	49544	31628	31811	68.82	653	49544	29628	6.32	6.86	166.75
s38584	114	59130	1057	58978	31095	31592	99.90	1006	58978	27848	10.44	11.85	240.68
Avg.	-	-	-	-	-	-	-	-	-	-	18.88	20.78	-