A Test Pattern Ordering Algorithm for Diagnosis with Truncated Fail Data

Gang Chen
Mentor Graphics Corp.
8005 SW Boeckman Road
Wilsonville, OR 97070

Sudhakar M. Reddy*
Dept. of ECE
University of Iowa
Iowa City, IA 52242

Irith Pomeranz*
School of ECE
Purdue University
W. Lafayette, IN 47907

Janusz Rajski
Mentor Graphics Corp.
8005 SW Boeckman Road
Wilsonville, OR 97070

ABSTRACT

In this paper, we propose a test pattern ordering algorithm for fault diagnosis. Test pattern ordering is effective in situations where the fail log is truncated and contains a limited number of fail data. In such cases, higher diagnostic resolution can be achieved with the test set appropriately ordered. Test pattern ordering is independent of the diagnosis algorithm used. The higher resolution achieved by test pattern ordering is obtained at no additional cost once the test patterns have been appropriately ordered. Experimental results on two industrial designs are presented to demonstrate the effectiveness of the proposed method.

Categories and Subject Descriptors

B.8.1 [**Integrated Circuits**]: Reliability, Testing, and Fault-Tolerance

General Terms

Algorithms, Reliability

Keywords

Test pattern ordering, diagnosis, truncated fail data.

1. INTRODUCTION

In addition to the continuous effort to derive test suites with high defect coverage, one major concern of the semiconductor industry is to correlate the collected fail data from the testers to the underlying defect in devices that fail the test. Defect diagnosis plays an important role in silicon debug and field return analysis. As feature size of integrated circuits continues to shrink, the effects of feature-related defects are becoming increasingly important [1]. As a result, initial yield is more limited by systematic failures due to feature-related defects than failures caused by random particles present during fabrication. To determine cause for yield limitation and to achieve fast and reliable yield ramp-up, there is an increasing demand to diagnose defects on large failing samples in production environment.

The purpose of this work is to order a given test set such that higher diagnostic resolution can be achieved. In this work we assume that the fail data available to the diagnostic tool is truncated and contains only a limited amount of fault detection information. This is justified for the following reasons. (1) Automatic test equipments (ATE) have a buffer of limited size for fail data storage. The fail data exceeding the buffer capacity will be truncated during capture. (2) It takes considerable time for testers to log a large amount of fail data. Hence in a high volume diagnosis monitoring flow, testers are configured to store only a limited number of fail data so that test throughput can be maintained at an acceptable level. Test pattern ordering for diagnosis purpose is effective when fail data presented to diagnostic tools is truncated. Besides the two situations described above, test pattern ordering can also be helpful in diagnosis applications using fault dictionaries [2,3,4]. Methods based on fault dictionaries create simulation response database for the modeled faults and compare this database with observed failure responses to identify the probable causes of the failure. In these methods, the dictionary size can grow out of control for large designs if all the simulated failure responses of faults are recorded. Thus, in practice only a limited number of fault detections are stored in fault dictionaries, in which case tests applied in different orders can cause different diagnosis results.

It is important to point out that test patterns may be ordered using a posteriori probabilities of detection of defects by a given pattern after testing of several failing devices [5]. The method proposed in this work does not explicitly consider the a posteriori probabilities of defect detection by tests. However, it can be adapted to use a posteriori probabilities.

In this paper, we propose a method to order test patterns for the purpose of improving diagnosis resolution when fail logs contain a limited amount of fail data. With any given test set, the proposed method arranges the test pattern order by considering both the number of fault pairs each pattern can distinguish and the fault detection time when a fault is distinguished. With appropriately ordered test patterns, truncated fail logs can contain more information useful for fault separation, such that higher diagnostic resolution can be achieved. Moreover, the benefit offered by test pattern ordering is independent of the diagnosis method being used. Thus it can be used with any diagnosis algorithm. Lastly, test pattern ordering for diagnosis is appealing as it has no cost except for the run-time required by the one-time execution of the algorithm. It is important to note that we use the resolution of pairs of single stuck-at faults to guide test pattern ordering. However the ordered tests can be used to diagnose any type of defect for which a diagnosis tool is normally used.

The rest of the paper is organized as follows. Section 2 briefly presents prior works. Section 3 describes, with an example, how

* Research reported was supported in part by SRC Grants TJ-1243 (SMR) and TJ-1244(IP).

test patterns in different orders affect the diagnosis outcome. Section 4 presents the proposed test pattern ordering procedure. In section 5 we present experimental results. Section 6 concludes the paper.

2. PRIOR WORK

Test pattern ordering has been considered for different purposes, including power dissipation reduction [16, 17, 18], power profile reshaping [19], and fault coverage steepening [5, 6]. However, to the best of our knowledge, no published work exists on test pattern ordering for the purpose of diagnosis with truncated fail data.

In [5, 6], test pattern ordering was considered to steepen fault coverage curves. These methods propose to order test patterns in a test set such that the fault/defect coverage would increase by a maximal amount with the inclusion of each additional test pattern. One goal of ordering test patterns according to fault/defect coverage is to detect defects earlier so as to save test application time when using the abort-on-fail technique. However, as multi-site testing strategy [11, 12, 13, 14] (which is a method to test multiple copies of the same circuit in parallel on one test equipment) is increasingly adopted, the savings in time offered by the early defect detection diminishes because the entire test set needs to be applied to the end as long as there is one device under test that is defect free. The experimental results in [15] show that the abort-on-fail technique has rather limited benefits when used in combination with multi-site testing.

3. PRELIMINARIES

We describe in this section how test pattern ordering can affect the diagnosis capability of a test set when only a limited number of fault detections are allowed for fault diagnosis. Let M^f denote the faulty circuit with fault *f*. We first give definitions essential to understanding the effects of test pattern ordering on diagnosis results.

Definition 1 [7]: Two faults *f* and *g* are *indistinguished with respect to a test set T*, if for every test vector the values on every primary output for M^f and M^g are equal or one or both output values is unknown; otherwise, they are distinguished.

Definition 2 [7]: An *indistinguished fault set with respect to T* (denoted by *IFS(T)*) is a set of faults such that every pair of faults in the set is indistinguished.

Table 1. Output responses with unknowns

tests	*t1*	*t2*
good ckt	0x0	10x
M^{f1}	1x0	0x1
M^{f2}	110	0xx
M^{f2}	0x0	11x

Consider the output responses with unknowns for an example circuit with three outputs as shown in Table 1. We give in Table 1 the output responses of a good and three faulty circuits under the test set *T* = {*t1*, *t2*}. It can be seen that *f1* and *f2* are indistinguished with respect to *T* according to Definition 1. {*f1*, *f2*} is an indistinguished fault set with respect to *T*. Moreover, *f3* is distinguished from *f1* and *f2* under test set *T*.

Given a test set *T*, all the faults that are undetected by *T* belong to one indistinguished fault set. Table 2 gives the output responses of an example circuit with two outputs on the application of the test set *T* comprising four patterns. We list in the table the output responses of the good and seven faulty circuits. This will serve as an explanatory example throughout the paper. For clarity, we also show the fault detection status in the table. Each fault under a particular test is marked as detected (D) or undetected (U). For example, *f1* is detected by tests *t1*, *t2* and *t4*, but is not detected by *t3*. In this example, we note that all seven faults are detectable by *T*. From Table 1 we can determine the list of *IFSs(T)*. For example, *f1* and *f2* are indistinguished with respect to *t1* since they both cause error output response "01" under *t1*. We show in Table 3 the indistinguished fault sets obtained by simulating each test in *T* separately.

Table 2. Output responses of an example circuit

tests	*t1*	*t2*	*t3*	*t4*
good ckt	00	00	11	11
M^{f1}	01 (D)	01 (D)	11 (U)	01 (D)
M^{f2}	01 (D)	00 (U)	11 (U)	11 (U)
M^{f3}	10 (D)	10 (D)	10 (D)	10 (D)
M^{f4}	10 (D)	10 (D)	10 (D)	11 (U)
M^{f5}	00 (U)	10 (D)	10 (D)	11 (U)
M^{f6}	10 (D)	10 (D)	01 (D)	10 (D)
M^{f7}	10 (D)	10 (D)	01 (D)	11 (U)

Table 3. Indistinguished fault sets generated by each test pattern

tests	Indistinguished fault sets
t1	(1,2); (3,4,6,7); (5)
t2	(1); (2); (3,4,5,6,7)
t3	(1,2); (3,4,5); (6,7)
t4	(1); (3,6); (2,4,5,7)

In order to evaluate the distinguishability of faults with respect to an ordered test set when a limited number of fault detections are allowed, we define the following.

Definition 3: An *indistinguished fault set with respect to an ordered test set T and a limit of K on the number of detections* (denoted by *IFS(T, K)*) is a set of faults such that every pair of faults in the set remains indistinguished up to the *K*-th failing pattern of each fault when the ordered test set *T* is simulated.

One can determine the indistinguished fault sets with respect to a test set by building a diagnosis tree based on output responses. A diagnosis tree is a directed acyclic graph. Each level of the diagnosis tree corresponds to a simulated test pattern. Before any test is applied, all faults belong to one class, which is represented by the root. Each tree node (or leaf) corresponds to a set of indistinguished faults that produce exactly the same output responses under the simulated test sequence. No faults in different leaves of the diagnosis tree have the same output responses for all tests.

By applying a test set in different orders, the diagnosis tree for a circuit can have different formations. In Figure 1, we show two diagnosis trees for the example circuit when *T* is applied in the order (*t1, t2, t3, t4*) and in the reversed order (*t4, t3, t2, t1*). Besides each node we give the number of times the faults in the node are detected by the tests applied up to the time corresponding to the level of the node. It can be seen from Figure 1 that, if all the failing patterns are recorded and used for diagnosis, all seven faults in the example circuit can be fully distinguished from each other regardless in which order the test is applied. However, in case that only a limited number of fault detections are allowed to be recorded, it is advantageous for fault diagnosis to use the ordered tests (*t4, t3, t2, t1*) rather than (*t1, t2, t3, t4*). Because at least four fault detections need to be recorded to fully distinguish every fault from other faults when using the ordered tests (*t1, t2, t3, t4*), while three fault detections are sufficient to fully distinguish every fault if the ordered tests (*t4, t3, t2, t1*) are used.

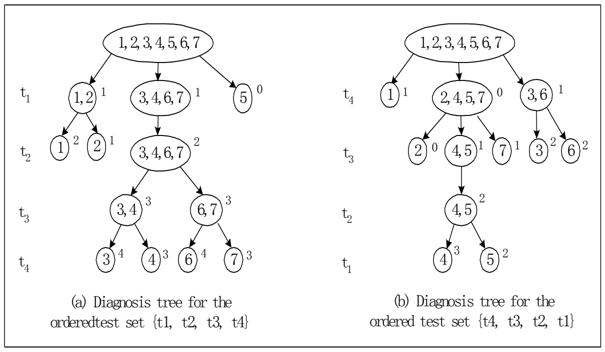

(a) Diagnosis tree for the orderedtest set {t1, t2, t3, t4}
(b) Diagnosis tree for the ordered test set {t4, t3, t2, t1}

Figure 1: Diagnosis trees for tests applied in different orders

4. THE PROPOSED TEST PATTERN ORDERING PROCEDURE

Several measures have been proposed in the literature to determine the diagnosis capability of a given test set [7-10]. One diagnosis measure proposed in [8] is *Diagnostic Resolution* (*DR*), which is the fraction of fault pairs distinguished by a test set. With an ordered test set *T*, let *N(k)* be the number of fault pairs that remain indistinguished when up to k fault detections are considered for fault differentiation. *N(k)* is a decreasing function of *k*. In our work we used *N(k)* as the diagnosis measure when ordering test sets. Thus the test pattern ordering algorithm we propose in this section targets improving the steepness of the *N(k)* vs. *k* curve achieved by the reordered tests.

4.1 A Greedy Algorithm for Test Pattern Ordering

Since finding an optimal test order is a NP-hard problem, the proposed test ordering algorithm follows a greedy approach. It iteratively selects, from the original test set *T*, patterns that are most beneficial according to prescribed criteria, and includes each additional pattern into the reordered test set *T'* until all the patterns are selected. The test ordering process can be viewed as a process of building a diagnosis tree. Every tree node is tagged by the current number of detections of the corresponding indistinguished fault set (*IFS*). Every time a pattern is newly selected and simulated, the diagnosis tree grows one more level and some tree nodes (i.e. *IFS*) split.

In the proposed test pattern ordering algorithm, we consider two criteria when selecting the next pattern to be added.

1) Criterion 1: Starting from the current level of the diagnosis tree, it is considered advantageous to select a pattern before others if it causes the maximum reduction in the number of indistinguished fault pairs. This strategy allows more faults to be distinguished as early as possible, hence the faults tend to be detected fewer number of times when they are distinguished from others.

Consider the example shown in Table 2. Before any test is applied, all the seven faults belong to one set, the number of indistinguished fault pairs is 21. From Table 3, it can be seen that the number of fault pairs left indistinguished is seven if *t1* is applied first. The indistinguished fault pair numbers are 10, 5, and 7, respectively, for the cases where test *t2, t3*, and *t4* are applied first, respectively. Since *t3* causes the maximum reduction in the number of indistinguished fault pairs (from 21 to 5), *t3* is first placed into the reordered test set. In Figure 2(a), we show the diagnosis tree obtained after simulating *t3* first.

2) Criterion 2: The pattern to be selected next is the one which splits the sets of faults associated with nodes at the current level of the diagnosis tree which are tagged with higher number of fault detections. This criterion places more focus on those faults that are detected many times without being distinguished by the preceding test patterns, so that they can be distinguished early by the reordered tests.

Consider the partially built diagnosis tree after simulating *t3*, which is shown in Figure 2(a). Among the three nodes in Figure 2(a), one ((1,2)) is tagged with 0, two ((3,4,5) and (6,7)) are tagged with 1. Based on criterion 2, it is beneficial to select a pattern to split node (3, 4, 5) or node (6, 7). Thus, it is advantageous to select *t1* or *t4* rather than *t2* as the second pattern in the reordered test set.

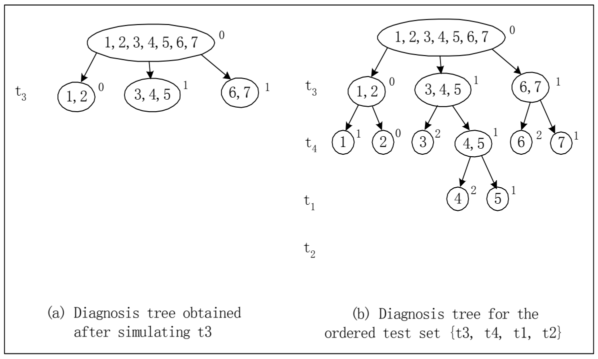

(a) Diagnosis tree obtained after simulating t3
(b) Diagnosis tree for the ordered test set {t3, t4, t1, t2}

Figure 2: Diagnosis trees for the test set starting with t_3

In order to take both criteria into consideration, we use a weight function to balance the contribution of each criterion. Let IFS_i be a indistinguished fault set that can be split after simulating test t_k. Let IFS_{ij} be the indistinguished fault set that is a descendent from IFS_i after simulating test t_k. Let d_i denote the current detection time of the faults in IFS_i. Let n_i (n_{ij}) be the cardinality of the fault set IFS_i (IFS_{ij}). We compute weight $w(t_k)$ for every test pattern not included in the ordered sets of patterns by using the following weight function:

$$w(t_k) = \sum_i (d_i + 1)^{\alpha} \cdot \Delta_i$$

$$= \sum_i (d_i + 1)^{\alpha} \cdot \left(n_i (n_i - 1) - \sum_j n_{ij} (n_{ij} - 1) \right) \cdots \quad (1)$$

where α (α>0) is a parameter. In our experiment, α was chosen to be 1.

Using the weight function described for pattern selection, one can arrange the test patterns in *T* as follows. First, determine *F*, the set of faults that are detected by *T*. Next the following steps are repeated until no test pattern is left in *T*. (1) simulate every pattern *t* in *T* on all the faults in *F*, compute *w(t)* for *t*; (2) select a test pattern t_i from *T* such that t_i has the maximum weight; (3) remove t_i from *T* and append t_i to the end of the ordered test set *T'* (*T'* is initially empty); (4) simulate t_i on *F* and update the *IFS* accordingly if it is split by t_i, drop fault f_i from *F* if f_i is found to be fully distinguished from all other faults in *F*. It is necessary to re-simulate the test patterns in *T* in Step 1 in order to determine their current weights, considering the change in the indistinguished fault sets due to the simulation of the selected test in Step 4. We show in Figure 2(b) the optimal ordered test set {*t3, t4, t1, t2*} obtained by the ordering procedure and the corresponding diagnosis tree for the example circuit in Table 2. It can be seen from Figure 2(b) that only two fault detections need to be recorded in order to fully distinguish every fault.

4.2 Speeding-up the Test Pattern Ordering Procedure

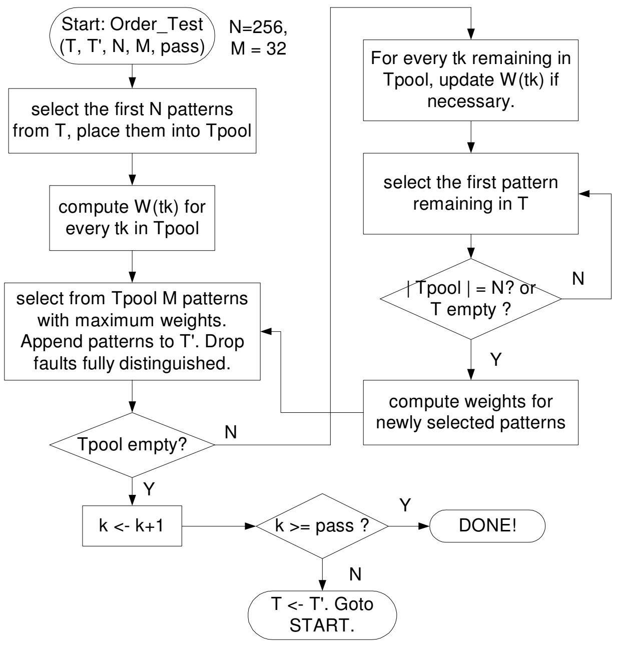
The test pattern ordering algorithm described in Section 4.1 is very time-consuming, since it requires re-simulation of all the remaining test patterns in order to update their weights every time a test pattern is included into the reordered test set. To speed up the test pattern ordering process, we propose an efficient procedure to order test patterns in *T*. The flowchart of the proposed test set ordering algorithm is presented in Figure 3. It imitates the near-optimal ordering process presented in Section 4.1. However, instead of selecting patterns to be ordered from the entire remaining test set *T*, the proposed algorithm makes its selection from a smaller set of *N* test patterns, which is a subset of *T*. Thus, every time a test pattern is placed in the ordered test set, at most *N* test patterns need to be simulated to update their weights. Moreover, instead of updating test pattern weights after each additional test pattern is selected and simulated, the proposed algorithm waits to update the test pattern weights until *M* (*M* < *N*) test patterns are selected, which allows to take advantage of parallel pattern simulation. The test pattern ordering procedure repeatedly selects *M* test patterns from the test pool consisting of *N* test patterns (*N* was set to 256 and *M* was set to 32 in our experiment). The proposed algorithm can be applied iteratively to further optimize the order of the test set. The variable *pass* is set to control the number of iterations. When the ordering procedure stops, it returns the ordered test set *T'*.

5. EXPERIMENTAL RESULTS

We applied the proposed test pattern ordering procedure to order the test sets derived for single stuck-at fault detection. We conducted experiments on two industrial designs, D1 and D2. In the experiments, we arranged test patterns by considering single stuck-at faults detected by the given tests. The results presented in this section were obtained using a 2.4GHz Linux machine with 2GB RAM.

Table 4. Run times of the proposed procedure

circuit	#TP	#gate	#det. flts	$RT(sec)
D1	2248	543K	850458	2079
D2	5000	313K	473745	2307



Start: Order_Test (T, T', N, M, pass)
N=256, M = 32
select the first N patterns from T, place them into Tpool
compute W(tk) for every tk in Tpool
select from Tpool M patterns with maximum weights. Append patterns to T'. Drop faults fully distinguished.
Tpool empty?
N
Y
k <- k+1
k >= pass ?
Y
DONE!
N
T <- T'. Goto START.
For every tk remaining in Tpool, update W(tk) if necessary.
select the first pattern remaining in T
| Tpool | = N? or T empty ?
N
Y
compute weights for newly selected patterns

Figure 3: Flowchart of the test set ordering algorithm

We first report in Table 4 the run time of the proposed procedure. For each design, after the circuit name, we give the pattern count of the test set being ordered, then report the number of gates in the design and the number of single stuck-at faults detected by the test set. In the last column we give the run time spent for one pass of the test pattern ordering procedure.

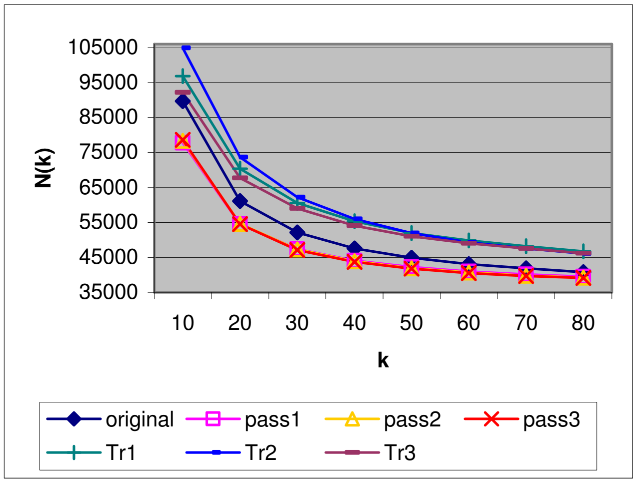


Figure 4: *N(k)* vs. *k* for D1

In the next experiment we performed, we compared the diagnosis measure, the number of indistinguished fault pairs, *N(k),* for the single detection stuck-at test sets before and after ordering when up to *k* fault detection times were considered for fault differentiation. We show the *N(k)* vs. *k* curve for design D1 in Figure 4. The curve for the original test set and those for each of the three passes of the proposed ordering procedure are plotted in Figure 4. For comparison purpose, we also show the curves for the tests T_{r1}, T_{r2} and T_{r3}, which are obtained by randomly ordering the original test set. It can be seen that the proposed procedure improves the steepness of the *N(k)* vs. *k* curve even after one pass of the ordering procedure. And the curves corresponding to subsequent passes of the proposed procedure are very close to the one corresponding to the first pass. Moreover, as expected, the gain offered by the reordered test sets gradually diminishes as k is increased.

The proposed test pattern ordering procedure is based on stuck-at fault model. The next set of experiments we performed was to validate the use of the proposed procedure to achieve test set orders with higher defect diagnosis capability. For each circuit, a number of test cases were created by injecting defects. Each test case of a circuit is an instance of that circuit with a two-line non-feedback bridge defect. And each test case of a circuit is simulated with the original single detection stuck-at test set T and the test set T' obtained after the first pass of the proposed test pattern ordering procedure. We created different sets of fail logs for each test case by using T and T'. Then a commercial diagnostic tool was used to diagnose the truncated fail logs that contain only up to a limited number of failing patterns. In order to evaluate the effectiveness of the proposed test pattern ordering procedure, a set of parameters are defined as follows.

Contain: the number of test cases where the diagnosis result contains one or more than one candidate pairs and the injected bridged node pair is included in the reported candidate list.

Partial: the number of test cases where the diagnosis result contains only one node of the injected bridge nodes in the reported list.

None: the number of test cases where the diagnosis result does not include any of the bridged nodes.

Ave: the average number of candidates in the candidate list reported by the diagnosis tool.

We conducted experiments on two industrial designs. We created for each design 5000 test cases all of which are detectable by the given single detection stuck-at test set. For design D1 where the layout is not available, we injected a randomly selected non-feedback bridge in each test case. The 5000 injected bridges are set to AND or OR-type with equal probability. For design D2 where the layout is available, in each test case we injected a layout extracted non-feedback bridge with the resistance ranging from 0Ω to 500Ω. The failure responses for each injected bridge are obtained by a two-stage bridge simulator, which uses SPICE to simulate the bridge defect site. In addition to the original stuck-at test set T and its ordered version T', for comparison purpose we also create fail logs for each test case by using tests T_{r1}, T_{r2} and T_{r3}, which are obtained by randomly ordering the test set T.

We report the diagnosis results in Table 5 and Figures 5, 6, 7 and 8. In Table 5, for different values of fault detection limit (from 10 up to 40) set for fail log truncation, we give the diagnosis results for different test sets in corresponding rows. In Figure 5(6), we show for design D1(D2) ***Ave*** for different ordered test sets when fault detection limit is set to different values. From the results in Table 5 and Figure 5, 6, we can see that the test set obtained by the proposed test pattern ordering procedure improves the resolution of defect diagnosis when the fail log is truncated. For example, when the detection limit is set to 10 for design D1, ***Ave*** for the original test set *T* is 6.84 whereas ***Ave*** for the ordered test set *T'* is reduced to 6.55. Another phenomenon we can observe is that, compared to the test set arranged in random order, the original test set generated by the ATPG tool has relatively higher diagnosis quality when diagnosing truncated fail logs. In Figure 7(8), we show for design D1(D2) the distribution of the size of the candidate list reported by the diagnosis tool when the detection limit is set to 10. For each size range, we show the number of test cases for which the sizes of the reported candidate lists fall into that range. We consider seven size ranges, ([1, 2)), ([2, 5]), ((5, 10]), ((10, 20]), ((20, 50]), ((50, 100]) and (>100). In Figure 7(8), for each size range, the bars represent from left to right, the number of test cases for the test sets *T, T'*, T_{r1}, T_{r2} and T_{r3}. It can be seen that, by using the ordered tests *T'*, more number of test cases resulted in short candidate lists (with size of ([1, 2)) or ([2, 5])) after diagnosis.

6. CONCLUSIONS

There are situations when the fail data used for fault diagnosis is truncated. A test pattern ordering algorithm is proposed such that higher diagnostic resolution can be achieved with appropriately ordered test set when diagnosing based on truncated fail logs. Experiments conducted on benchmark circuits and industrial designs validated the effectiveness of the proposed method.

7. REFERENCES

[1] R. Radojcic and M. Rencher, “Old Rules No Longer Apply”, EETimes, April 2003.

[2] I. Pomeranz, S. M. Reddy, “On Dictionary-based Fault Location in Digital Logic Circuits”, IEEE Trans. on Computers, vol. 46, iss. 1, pp. 48-59, 1997.

[3] V. Boppana, I. Hartanto, W. K. Fuchs, “Full Fault Dictionary Storage Based on Labeled Tree Encoding”, VTS, pp. 174-179, 1996.

[4] B. Chess, T. Larrabee, “Creating Small Fault Dictionaries”, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 18, iss. 3, pp. 346-356, 1999.

[5] W. Jiang and B. Vinnakota, “Defect-oriented Test Scheduling”, VTS, pp. 433-438, 1999.

[6] X. Lin, Janusz Rajski, I. Pomeranz, S. M. Reddy, “On Static Test Compaction and Test Pattern Ordering for Scan Designs”, ITC, pp. 1088-1097, 2001.

[7] E. M. Rudnick, W. K. Fuchs, and J. H. Patel, “Diagnostic Fault Simulation of Sequential Circuits”, ITC, pp. 178-186, 1992.

[8] P. Camurati, et. al., “A Diagnostic Test Pattern Generation Algorithm”, ITC, pp. 52-58, 1990.

[9] K. Kubiak, S. Parkes, W. K. Fuchs, R. Saleh, “Exact Evaluation of Diagnostic Test Resolution”, DAC, pp. 347-352, 1992.

[10] P. G. Ryan, W. K. Fuchs, I. Pomeranz, “Fault Dictionary Compression and Equivalence Class Computation for Sequential Circuits”, ICCAD, pp. 508-511, 1993.

[11] A. C. Evans, “Applications of Semiconductor Test Economics, and Multisite Testing to Lower Cost of Test”, ITC, pp. 113-123, 1999.

[12] A. Khoche, R. Kapur, D. Armstrong, T. W. Williams, M. Tegethoff, and J. Rivoir, “A New Methodology for Improved Tester Utilization”, ITC, pp. 916-923, 2001.

[13] E. Volkerink, A. Khoche, J. Rivior, and K. D. Hilliges, "Test Economics for Multi-site Test with Modern Cost Reduction Techniques", VTS, pp. 411-416, 2002.

[14] V. Iyengar, S. K. Goel, K. Chakrabarty, and E. J. Marinissen, "Test Resource Optimization for Multi-site Testing of SOCs under ATE Memory Depth Constraints", ITC, pp. 1159-1168, 2002.

[15] S. K. Goel and E. J. Marinissen, "Optimisation of On-chip Design-for-test Infrastructure for Maximal Multi-site Test Throughput", IEE Proc. On Computer and Digital Techniques, pp. 442-456, vol. 152, no. 3, 2005.

[16] V. Dabholkar, S. Chakravarty, I. Pomeranz, S. Reddy, "Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application", IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, pp. 1325-1333, vol. 17, no. 12, 1998.

[17] P. Flores, J. Monteiro, and J. Marques-Silva, "Assignment and Reordering of Incompletely Specified Pattern Sequences Targeting Minimum Power Dissipation", VLSI Design, pp. 37-41, 1999.

[18] P. Girard, C. Landrault, S. Pravossoudovitch, D. Severac, "Reduction of Power Consumption During Test Application by Test Vector Ordering", IEE Electronic Letters, pp. 1752-1754, vol. 33, no. 21, 1997.

[19] P. M. Rosinger, B. M. Al-Hashimi, and N. Nicolici, "Power Profile Manipulation: a New Approach for Reducing Test Application Time under Power Constraints:, IEEE Trans. On Computer-Aided Design of Integrated Circuits and Systems, pp. 1217-1225, vol. 21, no. 10, 2002.

Table 5. Comparison of diagnosis results for industrial designs

circuit	tests	10			20			30			40		
		contain	partial	none	contain	partial	none	contain	partial	none	contain	partial	none
D1	T	3452	1497	51	3527	1425	48	3635	1319	46	3694	1260	46
	T'	3474	1475	51	3546	1406	48	3660	1293	47	3728	1226	46
	T_{r1}	3359	1588	53	3448	1503	49	3555	1396	49	3615	1337	48
	T_{r2}	3345	1603	52	3440	1511	49	3545	1407	48	3601	1351	48
	T_{r3}	3384	1564	52	3454	1498	48	3579	1374	47	3634	1320	46
D2	T	2695	2247	58	2713	2229	58	2726	2217	57	2740	2204	56
	T'	2703	2239	58	2719	2223	58	2727	2215	58	2740	2202	58
	T_{r1}	2649	2293	58	2671	2271	58	2683	2259	58	2692	2250	58
	T_{r2}	2630	2311	59	2633	2308	59	2640	2303	57	2656	2288	56
	T_{r3}	2623	2319	58	2646	2296	58	2657	2285	58	2678	2266	56

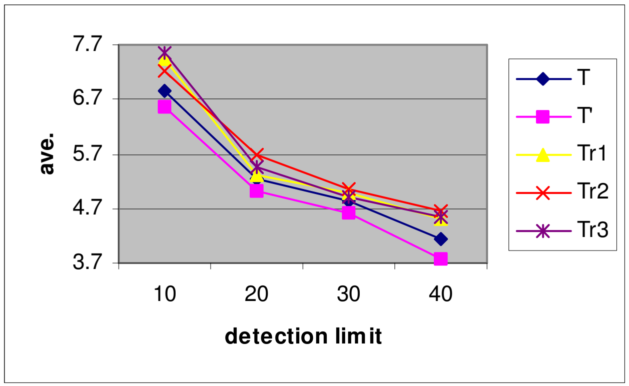


Figure 5: Comparison of Ave. for D1

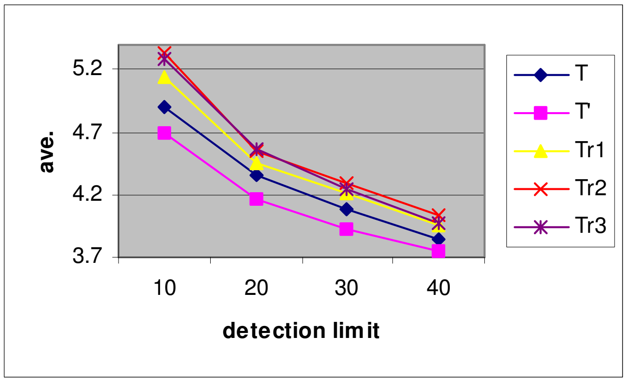


Figure 6: Comparison of Ave. for D2

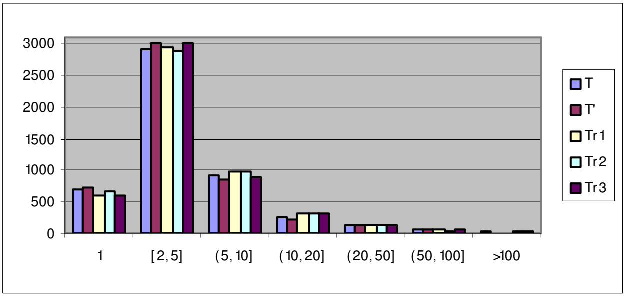


Figure 7: The number of test cases by candidate list size at detection limit of 10 for D1

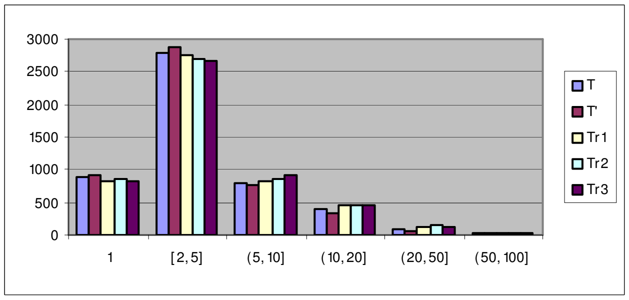


Figure 8: The number of test cases by candidate list size at detection limit of 10 for D2