Path Delay Test Compaction with Process Variation Tolerance

Seiji Kajihara Masayasu Fukunaga
Xiaoqing Wen
Kyushu Institute of Technology
680-4 Kawazu, Iizuka, 820-8502 Japan
e-mail:{kajihara, fukunaga, wen}
@aries30.cse.kyutech.ac.jp

Toshiyuki Maeda Shuji Hamada
Yasuo Sato
Semiconductor Technology Academic Research Center
3-17-2 Shinyokohama, Kita-ku,
Yokohama, 222-0033 Japan
e-mail:{maeda, hamada.shuji, satoh.y}@starc.or.jp

ABSTRACT

In this paper we propose a test compaction method for path delay faults in a logic circuit. The method generates a compact set of two-pattern tests for faults on long paths selected with a criterion. While the proposed method generates each two-pattern test for more than one fault in the target fault list as well as ordinary test compaction methods, secondary target faults are selected from the fault list such that many other faults, which may not be included in the fault list, are detected by the test pattern. Even if faults on long paths in a manufactured circuit are not included in the fault list due to a process variation or noise, the compact test set would detect the longer untargeted faults, i.e., the test set has a noise or variation tolerant nature. Experimental results show that the proposed method can generate a compact test set and it detects longer untargeted path delay faults efficiently.

Categories and Subject Descriptors

M.1.6 [**Testing, test generation and debugging**]

Keywords

delay testing, test compaction, path delay fault, process variation

1. Introduction

For recent DSM circuits, defects affecting timing behavior are becoming dominant, and thus testing for delay faults is becoming more and more important. Among delay fault models for test generation and fault diagnosis [1-3], the path delay fault model [2] has many advantages since it models localized as well as distributed excessive delays. Test patterns generated for a path delay fault can detect most of other types of delay fault such as gate delay faults [3] on the path.

On the other hand, the number of paths in a circuit is sometimes too large to allow efficient test pattern generation for all path delay faults. For example, the ISCAS-85 benchmark circuit of c6288, which is a 16-bit multiplier, has more than 10^{19} paths. Hence in test generation for path delay faults, we need to select a subset of paths to be targeted directly. Since it is necessary to select paths that are likely to be faulty, longer paths are usually selected according to a certain criterion. A simple approach of path selection is to select *N* longest paths in order of the path length. The length of any selected path is longer than the length of any unselected path. However, the selected paths may not be distributed all over the circuit and may be locally concentrated in a part of the circuit. In the approach of [4-8], a set of paths is selected that contains at least one of the longest paths through each line. These approaches are based on structural information of the circuit. However, in the DSM era, structurally longest paths may not be actual longest paths in a manufactured circuit due to process variation and/or noise [9-10]. Statistical or dynamic analysis based approaches for path selection have been proposed [11-12] too. However, it is difficult to know exact delay distribution of manufactured circuits. In addition, the longest paths may be different for each manufactured circuit. Hence, statistical approaches are still insufficient.

In order to make up for the incompleteness of path selection, [14,15] proposed a test generation method that selects two subsets of paths. For paths in the primary set consisting of longest paths, test patterns are guaranteed to be generated. For paths in the secondary set consisting of next-longest paths, fault detection is not guaranteed, but it is considered so as to maximize accidental detection by the test patterns for paths in the primary set.

After a subset of paths is obtained for test generation, test patterns for the selected paths are generated. Since a two-pattern test is required to detect a delay fault and the constraints of test patterns for path delay faults are more than those for stuck-at faults, the number of test patterns for delay faults is usually large. In order to reduce test application time, test compaction is required to achieve maximum fault coverage with a smallest possible number of test patterns [16,17].

In this paper we propose a method of test compaction for a given set of path delay faults. The proposed method is an effective solution for two major problems in test generation for path delay faults, namely reducing the number of test patterns and achieving high fault coverage against process variation and noise. In test compaction, each two-pattern test is generated for more than one fault in the targeted fault list as well as ordinary test compaction methods. The proposed method selects secondary target faults from the target fault list such that many faults on other long paths, which may not be included in the target fault list, can be accidentally detected. Even if longer paths in a manufactured circuit are not included in the target fault list, the compact test set generated by the proposed method would detect the longer untargeted faults. Hence the proposed method potentially improves the quality of test patterns while reducing the number of test patterns. Experimental results show that the proposed method can generate a compact two-pattern test set and it detects longer untargeted path delay faults efficiently.

This paper is organized as follows. In Section 2, we explain path selection approaches and test compaction techniques. In Section 3, we describe the proposed test compaction method. In Section 4, an example of the procedure that realizes the proposed test compaction method is given and experimental results are given. Finally, we conclude this paper in Section 5.

2. Related works

2.1 Path selection

When path delay testing is considered, a subset of paths in a circuit needs to be selected because it is generally impractical to test all paths in the circuit. Selection of paths which are targeted in test generation is an important step for testing path delay faults. If a faulty path of a manufactured circuit is not included in the target fault list, generated test patterns will not be able to detect the existence of the fault. Therefore long paths that are likely to be faulty are usually selected.

Some path selection criteria have been developed [4-8,11-13]. A simple approach is to select N longest paths in order of the path length. The length of any selected path is longer than that of any unselected path. In the approach of [4-8], a set of paths is selected that contains at least one of the longest paths through each line in the circuit. In the DSM era, structurally longest paths may not be actual longest paths in a manufactured circuit due to process variation and/or noise. In order to make up for the variation or noise problems, statistical approaches for path selection have been proposed [12].

In this work we assume that a list of target path delay faults which are selected using a criterion is given. Depending on the criterion used for path selection, different paths might be selected. However the proposed test compaction method in this paper does not depend on the path selection criterion.

During path selection, we need to be aware of the existence of untestable paths because it is known that there are many untestable paths in a circuit [18-20]. If untestable faults are included in the target faults, the fault coverage would be so low that additional paths need to be selected until a sufficient number of selected paths are testable. This is a time-consuming process because of the time wasted on test generation efforts for untestable paths. Therefore it is desirable that untestable paths are excluded from a fault list as much as possible.

2.2 Test compaction

As classic test compaction procedures, static compaction and dynamic compaction are well-known [21]. *Static compaction* reduces the number of test patterns by merging multiple individually generated test patterns into one. Since a test pattern generated for a fault contains unspecified values in general, compatible test patterns can be merged. For example, suppose that test vectors 0x10 and x1x0 are generated for two faults, respectively. In this case, these two can be merged into one test vector 0110.

Dynamic compaction [21] is a method of generating a test pattern that detects undetected faults as many as possible. By using unspecified values in a test pattern generated for a fault, test generation tries detecting another undetected fault. Dynamic compaction has a higher ability of test compaction than static compaction, but test generation time of dynamic compaction may be larger. The test compaction method proposed in this paper focuses on combination of faults detected by same test pattern. It is independent of compaction techniques used for test generation, i.e., the proposed method can be introduced into either static compaction or dynamic compaction.

Although a delay fault need two patterns to be detected, test compaction techniques such as static and dynamic compaction are still applicable. Note that, in the rest of the paper, a test pattern means a test-pattern-pair since we treat test patterns for delay faults. A test compaction method for transition faults in [22] is based on techniques developed test generation for stuck-at faults [23,24]. For path delay faults, dynamic compaction methods have been reported in [16,17]. These works aimed at minimizing the number of test patterns without losing fault coverage.

3. Proposed test compaction

3.1 Basic concept

The proposed method aims at not only minimizing the number of generated test patterns but also enhancing the test quality of generated test patterns, as shown in Fig. 1. Test quality enhancement is achieved by detecting more faults not included in the fault list. In general, a test pattern generated for a fault detects faults other than the target fault accidentally. If the accidentally detected faults are included in the target fault list, the number of test patterns would be reduced. Even if the accidentally detected faults are not included in the target fault list, it would contribute to the enhancement of test quality. In our method, while test generation targets path delay faults in a given fault list, test compaction is performed such that untargeted path delay faults are detected utilizing parts of the paths targeted in test generation.

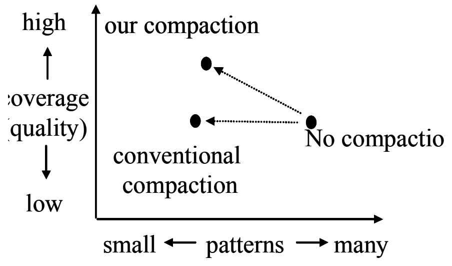
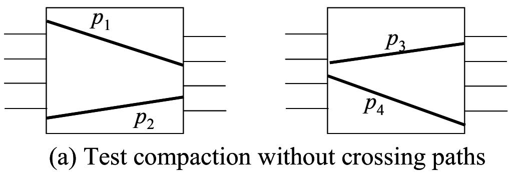


Fig. 1: Enhancement of test quality through test compaction

We use an example to explain the idea of test compaction used in the proposed method. Suppose that four paths p_1, p_2, p_3 and p_4 are tested. Through test compaction, some paths are tested by a test pattern simultaneously. As shown in Fig 2(a), if p_1 and p_2 are tested by a test pattern, and if p_3 and p_4 are tested by another test pattern, no other paths would be tested necessarily. On the other hand, if p_1 and p_3 are tested simultaneously as shown in Fig. 2(b) where p_1 and p_3 cross at a gate, paths other than p_1 and p_3 can be tested. Fig. 3 illustrates that there are two paths p_5 and p_6 consisting of partial paths of p_1 and p_3. Since paths p_1 and p_3 have a common gate, a test pattern for p_1 and p_3 can test paths p_5 and p_6 in addition to p_1 and p_3. Note that p_1 and p_3 are included in the target fault list, but p_5 and p_6 are not included necessarily.



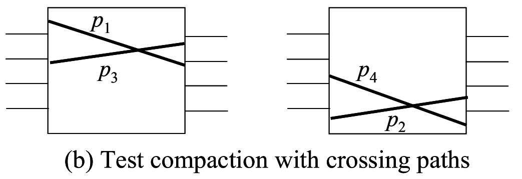


Fig. 2: Combinations of tested paths by same test

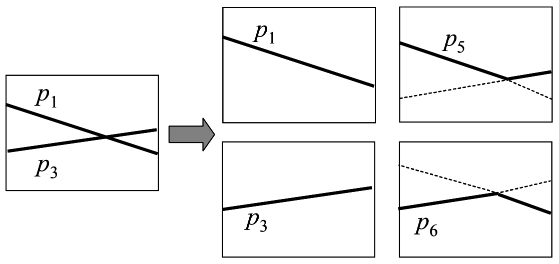


Fig. 3: Tested paths by accidental detection

3.2 Conditions of crossing paths

In general, when crossing paths on which there is a common gate are tested simultaneously, non-target paths consisting of partial paths of the target paths can be tested simultaneously too. We generate a test pattern such that two path delay faults with a common gate in the fault list are detected. Two path delay faults with a common gate can be tested when the following conditions are satisfied:

1) Two paths have same transition at the common gate each other.
2) The transition at the common gate is from the controlling value [25] of the gate to the non-controlling value.

Fig. 4(a) shows an example of a test pattern that can test two paths through a common gate. At the OR gate in the circuit, two paths p_1 and p_2 meet each other with a transition from the controlling value to the non-controlling value. If the arrival of one of input transitions is delayed as shown in Fig. 5(a), then the output transition is also delayed. Therefore if either path is delayed, it would be detected. However, when two inputs of a gate have transitions from the non-controlling value to the controlling value as shown in Fig. 5(b), the transition arrived at the input earlier determines the output transition. Hence a delay fault through the path would be masked. Thus a common gate must have a transition from the controlling value to the non-controlling value.

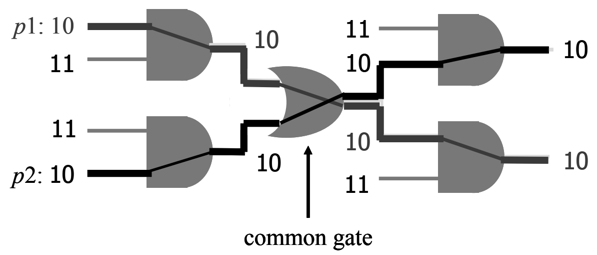


Fig. 4: Test pattern with accidental detection

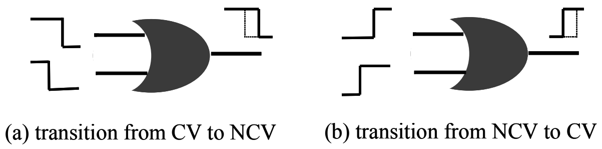


Fig. 5: Transition at a common gate

In order to detect faults additionally, two paths have to branch off on the way to outputs from the common gate. If two paths have same routes from the common gate to the output, there is no other path tested simultaneously. Such a case is illustrated in Fig. 6(a). On the other hand, in case two paths have two common gates with fan-outs on the way to the outputs as shown in Fig. 6(b), 6 paths can be tested in addition to paths p_1 and p_2 because 8 paths can be constructed using p_1 and p_2. Thus the number of paths tested simultaneously increases exponentially to the number of common gates with branches.

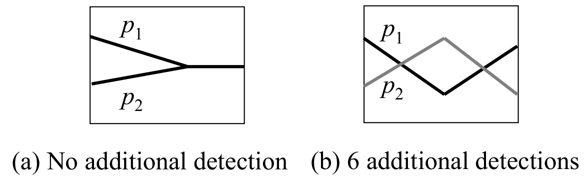


Fig. 6: Tested paths by accidental detection

3.3 Variation-tolerant nature of test patterns

In this section we state advantages of test patterns generated by using the proposed compaction method. Since paths that are likely to be faulty should be tested, longer paths are selected according to a criterion. Test patterns generated would detect path delay faults on the selected paths certainly if they are testable. However, it is difficult to predict the delay size of a path in manufactured circuits because of process variation or noise. As a result, there remain paths that are more likely to be faulty than the selected ones and the generated test patterns might miss a fault on the paths.

Test patterns generated by our method, however, would detect not only faults on the selected paths but also some faults on unselected paths. If the unselected paths whose faults are accidentally detected consist of parts of the selected paths, the length of the unselected paths is relatively long because the selected paths are long. Therefore the test patterns potentially compensate the detection of untargeted faults.

Another advantage is that the test patterns potentially cover alternative faults to untestable target faults. Although most of untestable paths can be identified in path selection, it is difficult to exclude all untestable paths from the fault list because ATPG is required to check if a path is testable or not. If a selected path is untestable, another path which is the next-longest should be selected and a test pattern for testing the path should be generated. But, since test patterns generated by our method potentially detect faults on unselected long paths, we would keep high fault coverage without retrying path selection and test generation.

4. Experimental results

4.1 Procedure

We implemented the proposed test compaction method in a simple static compaction procedure, which is shown below. This procedure generates test patterns for given faults with heuristics to increase common gates on paths tested by each test pattern.

Step 1: Set $T_{fin} = \phi$, and generate an initial test pattern set T_{init} for each path delay fault in a given fault list one by one. Note that T_{fin} is a final test pattern set.
Step 2: Remove all overlapped test patterns in T_{init}.
Step 3: Pick up one test pattern, t_{sel}, from T_{init} which sensitizes longer path than others, and remove t_{sel} from T_{init}.
Step 4: If there are any test patterns in T_{init} which are compatible with t_{sel}, go to Step 5. Otherwise, go to Step8.
Step 5: Pick up a test pattern, t_{merge}, from T_{init} such that the following conditions are satisfied:
1. t_{sel} and t_{merge} are compatible.
2. Paths sensitized by t_{sel} and t_{merge} have the largest number of common gates.
Step 6: Remove t_{merge} form T_{init}, and merge t_{merge} into t_{sel}.
Step 7: If there is any compatible test pattern with t_{sel} in T_{init}, return to Step 5. Otherwise, go to Step 8.
Step 8: Add t_{sel} to T_{fin}.
Step 9: If T_{init} is not empty, return to Step 3. Otherwise, this algorithm finishes.

When an initial test pattern set, T_{init}, is generated for each path delay fault, we do not assign any logic value to unspecified inputs of each test pattern, i.e., unspecified bits remain for static compaction. Next, redundant test patterns in T_{init} are removed if exist, and a test pattern, t_{sel}, is selected from T_{init}, which sensitizes longer path than others. Next, compatible test patterns, t_{merge}, are searched for in order to be merged with t_{sel}. Note that the current procedure does not take branches to outputs into consideration. Hence no additional path may be sensitized even if two paths sensitized by the compatible test patterns have a common gate. In Step5, if there is more than one test pattern that has the same number of common gates, the test pattern is selected that sensitizes longer paths than others. In Step6, after t_{merge} is removed from T_{init}, a new test pattern is generated by merging t_{sel} and t_{merge}, and the resulting test pattern is denoted by t_{sel}. If there is at least one compatible test pattern with the new t_{sel} in T_{init}, the process returns to Step5. Otherwise, after logic values are filled randomly to all unspecified bits of t_{sel}, t_{sel} is added to the final test set T_{fin}. In Step9, if there remains any test pattern in T_{init}, the process returns to Step3. Otherwise, the process finishes.

4.2 Results for benchmark circuits

We implemented the procedure of static compaction using C programming language on a PC (Pentium III Xeon 2GHz, 4GB memory) and applied it to full scan version of ISCAS'89 benchmark circuits. We constructed a given fault list such that all the longest potentially testable paths through each line of the circuit are included. Note that the length of a path is determined by the number of logic gates on the path.

Table 1 shows statistics of each circuit in terms of testable paths and selected paths. The columns of Table 1 give the circuit name, the total number of logical paths i.e. path delay faults, the number of testable paths which can be calculated by ATPG for all paths, and the number of selected paths and the number of testable paths out of the selected paths. In the selected paths some untestable paths existed except for s35932 because of the incompleteness of untestable path analysis in path selection.

Table 1: Selected paths and testable paths

circuit	#total paths	#testable paths	#selected paths	#testable paths in selected paths
s5378	27,084	21,928	9,644	9,524
s9234	489,708	59,854	15,458	15,377
s13207	2,690,738	476,145	27,111	26,054
s15850	329,476,092	10,782,994	89,298	85,938
s35932	394,282	58,657	39,124	39,124
s38417	2,783,158	1,138,194	224,101	209,161
s38584	2,161,446	334,927	59,519	58,221

Table 2 gives test generation results. The four columns followed by circuit name show results of test generation without test compaction where each test pattern is generated for an undetected fault in the fault list and fault simulation is performed for the generated test pattern after random-filling for unspecified bits. The last four columns show results of test compaction according to the procedure described above. The columns "#tests" gives the number of two-pattern tests. Fault efficiency is defined as the percentage of paths tested by generated test patterns for all the testable paths. The columns "#tested paths per test" gives the average number of newly tested paths for each two-pattern test. The sizes of generated test patterns were compacted approximately to 25 % of the uncompacted test sets by test compaction while the uncompacted test sets have higher fault efficiency than the compacted test sets, i.e., the uncompacted test sets could detect more faults which are not included in the given fault list. However, the difference of fault efficiency is not large compared with the difference of test set sizes. The number of newly tested paths by each test pattern in the compacted test sets was four times of the uncompacted test sets on average.

Table2: Test generation results

circuit	uncompacted tests				compacted tests			
	#tests	fault efficiency	#tested paths per test	coverage for 10,000 paths	#tests	fault efficiency	#tested paths per test	coverage for 10,000 paths
s5378	1,681	88.52%	5.77	92.26%	400	84.81%	23.25	87.85%
s9234	2,106	62.66%	8.90	62.39%	640	58.25%	27.24	61.66%
s13207	1,861	43.28%	55.37	78.91%	733	40.72%	132.26	77.98%
s15850	3,118	18.66%	322.72	96.00%	1,615	18.16%	606.18	<u>97.84%</u>
s35932	275	98.79%	105.36	99.66%	33	82.83%	736.18	90.43%
s38417	29,714	63.17%	12.10	99.97%	3,497	55.91%	90.99	99.88%
s38584	4,581	65.84%	24.07	81.80%	1,172	65.28%	93.27	<u>82.86%</u>

When we watch only 10,000 longest testable paths in each circuit, we can observe that the compacted test sets could test longer paths efficiently. For circuits s15850 and s38584, the compacted tests could test more paths than the uncompacted test sets in spite of much less number of test patterns. These results imply that the proposed method is useful for testing longer paths which are not targeted in test generation.

Table 3 gives data on crossing paths which the test compaction procedure results in. Path delay faults on the crossing paths can be detected by the generated test patterns certainly. The column "#crossing paths" of Table 3 gives the number of crossing paths created through the compaction process. The columns "%crossing paths" gives the percentages of crossing paths for the tested paths. The columns "#crossing paths per test" gives the average number of crossing paths for each test pattern. The number of crossing paths for each test pattern was not so large. This means that the implemented procedure, which is simple static compaction, does not have high compaction ability. There is still enough room for optimization of the compaction algorithm. For example, applying dynamic compaction would improve the results. And better heuristics to find more crossing paths would be able to be developed.

5. Conclusion

In this paper we showed a solution for problems of test generation for path delay faults that are reduction of test patterns and achieving high fault coverage against process variation and noise. In test compaction, we proposed to test paths with cross points simultaneously so as to accidentally detect many faults which may not be included in the target fault list. Experimental results showed that the proposed method could generate a compact two-pattern test set and it could detect longer untargeted path delay faults efficiently. However, the compaction algorithm implemented in this work is still insufficient for test quality enhancement. As a future work, we will develop more efficient compaction algorithm to derive the effects of the proposed idea.

Table3: Created crossing paths

circuit	#crossing paths	%crossing paths	#crossing paths per test
s5378	1,556	8.37%	1.95
s9234	2,548	7.31%	1.99
s13207	2,721	1.40%	1.86
s15850	142,942	7.30%	44.25
s35932	1,349	2.78%	20.44
s38417	14,284	2.24%	2.04
s38584	7,306	3.34%	3.12

ACKNOWLEDGMENTS

This work was supported by the New Energy and Industrial Technology Development Organization (NEDO).

REFERENCES

[1] M. L. Bushnell, and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits*, Kluwer Academic Publishers, 2000.

[2] G. L. Smith, "Model for delay faults based upon paths," *Int'l Test Conf.*, pp.342-349, 1985.

[3] Z.Barzilai and B.K.Rosen, "Comparison of AC Self-testing Procedures," *Int'l Test Conf.*, pp.89-01, 1983.

[4] W.-N.Li, S.M.Reddy, S.K.Sahni, "On Path Selection in Combinational Logic Circuits," *IEEE Trans. on CAD.*, vol.8, pp.56-63, 1989

[5] A. Murakami, S. Kajihara, T. Sasao, I. Pomeranz, and S. M. Reddy, "Selection of Potentially Testable Path Delay Faults for Test Generation," *Int'l Test Conf.*, pp. 376-384, 2000.

[6] M. Sharma and J. H. Patel, "Finding a Small Set of Longest Testable Paths that Cover Every Gate," *Int'l Test Conf.*, pp. 974-982, Oct. 2002.

[7] Y. Shao, S. M. Reddy, I. Pomeranz, S. Kajihara, "On Selecting Paths to Test in Scan Designs," *Journal of Electronic Testing Theory and Applications*, volume 19, pp. 447-456, August 2003.

[8] W. Qiu and D. M. H. Walker, "An Efficient Algorithm for Finding the K Longest Testable Paths Through Each Gate in a Combinational Circuit," *Int'l Test Conf.*, pp. 592-601, Sept. 2003.

[9] L.-C. Chen, S. K. Gupta and M. A. Breuer, "High Quality Robust Tests for Path Delay Faults," *VLSI Test Symp.*, pp. 88-93, April 1997.

[10] K-T Cheng, S. Dey, M. Rodgers, K. Roy. "Test Challenges for Deep Sub-Micron Technologies," *Design Automation Conf.*, pp.142-149, June 2000.

[11] J.-J. Liou, A. Krstic, Y.-M. Jiang and K.-T. Cheng, "Path Selection and Pattern Generation for Dynamic Timing Analysis Considering Power Supply Noise Effects," *Intl. Conf. on Computer-Aided Design*, pp. 493-496, Nov. 2000.

[12] J.-J. Liou, A. Krstic, L.-C. Wang, K.-T. Cheng. "False-Path-Aware Statistical Timing Analysis and Efficient Path Selection for Delay Testing and Timing Validation," *Design Automation Conf.*, pp.566-569, 2002.

[13] S. Tragoudas, S. Padmanaban, "A Critical Path Selection Method for Delay Testing," *Int'l Test Conf.*, pp. 232-241, Oct. 2004.

[14] I. Pomeranz and S. M. Reddy, "Test Enrichment for Path Delay Faults Using Multiple Sets of Target Faults," *Conf. on Design Automation and Test in Europe*, pp. 722-729, March 2002.

[15] I. Pomeranz and S. M. Reddy, "A Postprocessing Procedure of Test Enrichment for Path Delay Faults," *Asian Test Symposium*, pp. 448-453, Nov. 2004.

[16] S. Bose, P. Agrawal, V. Agrawal, "Generation of compact delay tests by multiple path activation," *Int'l Test Conf.*, pp. 714-723, Oct. 1993.

[17] J. Saxena; D.K.Pradhan, "A method to derive compact test sets for path delay faults in combinational circuits," *Int'l Test Conf.*, pp. 724-733, Oct. 1993.

[18] S.Kajihara, K.Kinoshita, I.Pomeranz, S.M.Reddy, "A Method for Identifying Robust Dependent and Functionally Unsensitizable Paths," *Int'l Conf. on VLSI Design*, pp.82-87, 1997.

[19] Z.Li, Y.Min, R.K.Brayton, "Efficient Identification of Non-Robustly Untestable Path Delay Faults," *Int'l Test Conf.*, pp.992-997, 1997.

[20] K.Heragu, J.H.Patel, V.D.Agrawal, "Fast Identification of Untestable Delay Faults Using Implications," *Intl. Conf. on Computer-Aided Design*, pp.642-647, 1997.

[21] P. Goel and B. C. Rosales, "Test Generation & Dynamic Compaction of Tests," in Digest of Papers *1979 Test Conf.* , pp. 189-192, Oct. 1979.

[22] I. Hamzaoglu, J.H. Patel, "Compact two-pattern test set generation for combinational and full scan circuits," *Int'l Test Conf.*, pp. 944-953, Oct. 1998.

[23] S. Kajihara, I. Pomeranz, K. Kinoshita and S. M. Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 12, pp.1496-1504, Dec. 1995.

[24] I. Hamzaoglu and J. H. Patel, "Test Set Compaction Algorithms for Combinational Circuits," *Intl. Conf. on Computer-Aided Design*, pp. 283-289, Oct. 1998.

[25] M. Abramovici, M. A. Breuer, A. D. Friedman, *Digital Systems Testing and Testable Design,* Piscataway, New Jersey: IEEE Press, 1990.