On Improving Defect Coverage of Stuck-at Fault Tests

Kohei Miyase[1], Kenta Terashima[2], Seiji Kajihara[2], Xiaoqing Wen[2] and Sudhakar M. Reddy[3]

1: Innovation Plaza, Fukuoka, Japan Science and Technology Agency

2: Department of Computer Science and Electronics, Kyushu Institute of Technology

3: Department of Electrical and Computer Engineering, University of Iowa

Abstract

Recently design for manufacturability (DFM) has been required to achieve higher process yield. Information obtained from silicon by testing and/or fault analysis is sometimes fed back for redesign of VLSI circuits. In this paper we propose a method to maximize defect coverage of a test set initially generated for stuck-at faults in a full scan sequential circuit by using feed back information from fault analysis. If a test set for more complex faults than stuck-at faults is generated, higher defect coverage would be obtained. Such a test set, however, would have a large number of test vectors, and hence the test costs would go up. The proposed method improves defect coverage of the test set by not adding new test vectors but modifying test vectors with the information obtained from fault analysis. Therefore there are no negative impacts on test data volume and test application time. The initial fault coverage for stuck-at faults of the test set is guaranteed with modified test vectors. In this paper we focus on detecting as many as possible non-feedback AND/OR-type bridging faults. Experimental results show that the proposed method significantly decreases the number of non-feedback AND/OR-type bridging faults undetected by a test set generated for stuck-at faults.

1. Introduction

Development of manufacturing process for VLSI circuits has been shrinking transistor feature size. A serious problem resulting from shrinking feature size is that it has become difficult to improve process yield. In the past, improvement of yield is considered at manufacturing process stage. After the start of nano-technology it is necessary to consider improvement of yield at design stage prior to manufacturing process stage. Such concept is referred to as design for manufacturability (DFM). Nowadays, information obtained from testing and/or fault analysis has been used in order to enhance efficiency of DFM. This is because information on defects that occur often and location where they often occur, can indicate how to redesign VLSI circuits for improvement of yield. It is considered that such information would also be useful for testing, i.e., if we have data on type and location of defects which are more likely to occur, tests could be generated to cover the likely defects.

In the past, it is considered that test sets for stuck-at faults could detect enough defects. But now, while the test set for stuck-at faults is still necessary to detect logical faults, this test set is not sufficient to cover non-stuck-at faults. This is because behavior of defects due to the effects of shrinking feature size is becoming much more complicated than before. Consequently the goal of test pattern generation is changing from high stuck-at fault coverage to cover more complex fault models in addition to stuck-at faults [1][2][3][4].

An easy way to detect unmodeled defects is to generate tests that detect each stuck-at fault n times [22]. The method is referred to as n-detection. Since a stuck-at fault is activated and propagated by n different test vectors in an n-detection test set, defects on the site of the stuck-at fault are likely detected. Thus a test set for n-detection has higher defect coverage. Besides, the method is easy to implement since all we have to do is to generate test vectors for single stuck-at faults. However, the number of test vectors increases in proportion to n, which increases test application time and hence test cost.

When we generate a test set assuming a more complex fault model it may detect a larger number of defects than a test set for stuck-at faults. The problem is that the test set size for the complex fault model may become larger than the one for stuck-at faults. It is possible that test data volume would exceed the limit of tester memory as well as increase test application time if we use or add a test set targeting a complex fault model.

Obviously detection of many defects with a small size of a test set is really important to achieve higher test quality and to reduce the test cost of test application. However, it is impossible to detect all conceivable defects

1081-7735/05 $20.00 © 2005 IEEE
IEEE COMPUTER SOCIETY

with a small test set. One solution to achieve high test quality with a small test set is to concentrate detection of faults covering defects that often occur. Especially with VLSI circuits redesigned with DFM concept, we can select effective faults with information obtained from testing and/or fault analysis.

Recently methods that allow us to modify test vectors without increasing the size of a test set and without changing stuck-at fault coverage have been proposed [7][8][9]. The methods identify don't-care bits in fully specified test vectors. Then they assign logic values to don't-cares so as to satisfy some purposes such as test compression [10][11] or test power reduction [12]. In this paper, we propose a method to generate a test set that not only guarantees to detect stuck-at faults but also maximizes the detection of faults from another fault model. We can select the fault model from information on defects that often occur in the manufacturing process. According to such information the proposed method modifies a test set generated for stuck-at faults so as to additionally detect other faults.

In this work, we assume that defects modeled by non-feedback AND/OR-type bridging faults are the likely to occur faults. The number of bridging faults in a circuit with L lines is of the order of $O(L^2)$ if we don't use layout information. But if we can access layout information it may be reduced to $O(L)$. We also assume that a test set for single stuck-at faults where test vectors are fully specified is given. In the first step of the proposed method, we identify as many don't-cares as possible in the test vectors. Even though arbitrary logic values are assigned to the don't-cares, stuck-at fault coverage of the initial test set is still guaranteed. In the next step we assign logic values to them such that detection of non-feedback AND/OR-type bridging faults is maximized. The conditions to detect a bridging fault are to detect the stuck-at fault on one of the lines in the pair of bridged lines and to set the opposite value on the other line of the bridged lines. Since the conditions are comprised of conditions to detect a stuck-at fault, we employ a dynamic compaction technique of ATPG [13] to assign logic values to bridged lines. As a result we obtain a test set for stuck-at faults which detects many non-feedback AND/OR-type bridging faults without increasing the number of test vectors. Experimental results for ISCAS benchmark circuits show that the test sets obtained by the proposed method detect more bridging faults than the test sets initially generated for stuck-at faults.

The paper is organized as follows. In Section 2, we review a method of test vector modification and dynamic compaction. We propose a method to improve defect coverage in Section 3. In Section 4 we describe how to improve coverage of non-feedback AND/OR-type bridging faults with the proposed method. Next, we give experimental results for benchmark circuits in Section 5. Finally, we conclude this paper in Section 6.

2. Preliminaries

2.1. Don't cares in test vectors

Don't-cares (Xs) in test vectors play an important role for testing logic circuits today. Depending on logic values assigned to the Xs, different features can be imparted to the test vectors. For example the existence of Xs generally facilitates test compression [5][6][9][10][11]. There are two methods to obtain test vectors that include Xs. One is not to specify logic values to Xs (unspecified bits) just after test vectors are generated. The Xs are left unspecified until all test vectors are generated. The drawback of this method is that more test vectors are generated [14][15] because this method misses accidental detection of faults with random fill or static/dynamic compaction methods[13]. Another method to obtain tests with Xs is identification of entries in a fully-specified test set that can be Xs [7][8][9]. The methods of [7][8] can identify the input values in test vectors that can be set to Xs with reasonable computing time. Furthermore even in compacted test sets these methods show that up to 50% of inputs in the test sets can be set to Xs. Therefore we can effectively modify test vectors without increase of test vectors to achieve higher defect coverage by filling the Xs appropriately.

2.2. Dynamic compaction

Dynamic compaction is known as a classic technique to reduce the number of test vectors during ATPG [13]. The concept is to detect as many yet undetected faults as possible by each newly generated test vector. Usually just after test generation for a fault, there are many unspecified values left in the test vector. Dynamic compaction assigns logic values to the unspecified values using ATPG so as to detect other undetected faults.

3. Improving Defect Coverage

3.1. Overview of the proposed method

The proposed method improves defect coverage with information on defects. Therefore, we assume that such information can be obtained from fault analysis for manufactured chips. Fig.1 shows the flow of the proposed method. Given a test set T for stuck-at faults, we first identify as many positions as possible that can be set X to obtain test set T' with Xs. Before logic value assignment to Xs, we select a fault model based on information on

IEEE COMPUTER SOCIETY

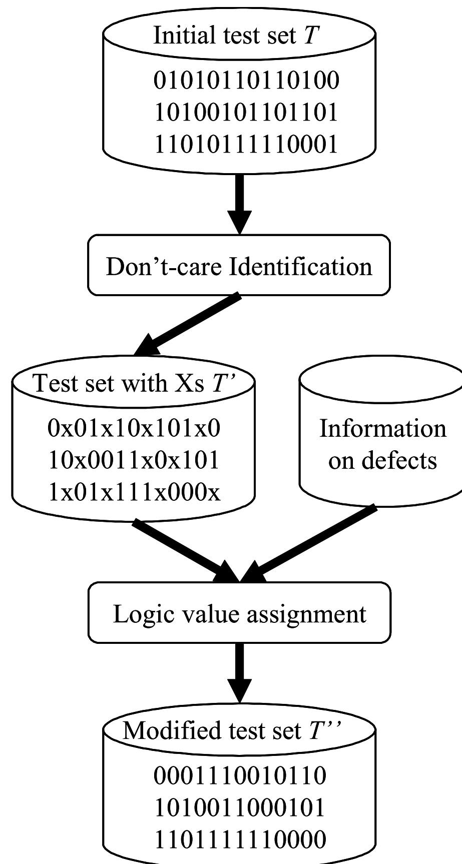

Fig. 1: Flow of test vector modification

defects. Then we assign logic values to the Xs so as to detect another type of faults. Finally we obtain the modified test set *T''*.

3.2. Target faults

When we generate a test set for complex non-stuck-at faults, the number of test vectors becomes larger than that for stuck-at faults. Therefore we focus on enhancing test quality of a generated test set. Since the method proposed in this paper does not add new test vectors but modifies test vectors, there are no negative impacts on test data volume and test application time.

3.2.1. Selection of faults based on characteristic of defects

When information obtained from fault analysis indicates that defects often occur with a particular behavior, we select fault model representing the defects such as 4-way bridging faults [15][16], or X-faults[2]. The conditions to detect a stuck-at fault usually are also necessary conditions for detection of other logical faults. Therefore a test set for stuck-at faults already has potential to detect them. In Section 4, we show how to improve the defect coverage for non-feedback AND/OR-type bridging faults.

3.2.2. Selection of faults based on location

Recently sophisticated works of fault location have been proposed [20][21]. One can use statistics of fault locations for many failed chips to determine sites that are likely to have defects. When information obtained from fault analysis does not indicate the characteristic of defects but only sites where defects likely occur, we modify the initial test set such that stuck-at faults on the suspicious sites are detected multiple times. The concept is similar to *n*-detection [22], but not all faults are required to be detected *n*-times. Therefore we can avoid making test set size larger that what is necessary for the detection of single stuck-at faults.

4. Improvement of Non-Feedback AND/OR Bridging Fault Coverage

In this paper, we assume that defects which occurred in many circuits under test are bridges and that the behavior of the defects corresponds to an AND-type bridging fault or an OR-type bridging fault. Obviously we can't detect all bridging faults with test vector modification only, since the number of bridging faults in a circuit with L lines is $O(L^2)$. We attempt to detect as many bridging faults as possible without increasing the number of tests generated for single line stuck-at faults.

4.1. Logic value assignment for detection of non-feedback AND/OR bridging faults

After we obtain a test set including Xs, we assign logic values to Xs so as to detect non-feedback AND/OR-type bridging faults. To abbreviate the explanation we only discuss non-feedback AND-type bridging faults. The condition of detection for a non-feedback AND-type bridging fault is to detect the stuck-at 0 fault on a bridged line and to set logic value 0 on the other line of the pair of bridged lines. Suppose that lines a and b are bridging. This bridging fault can be detected if a stuck-at 0 fault on line $a(b)$ can be detected and logic value of line $b(a)$ is 0. Since each condition is covered by the condition to detect a stuck-at fault, we can detect a non-feedback AND-type bridging fault using the technique of dynamic compaction [13] using the modified test vectors. When we assign logic values to Xs in the test set for the detection of bridges, there are three cases. In the following we explain how to assign logic values for each case.

Case 1: In this case a stuck-at 0 fault can be detected on one of the bridging lines, and the logic value is an X on the other bridging line for some test. In order to detect the

IEEE COMPUTER SOCIETY

bridging fault, we assign logic values to primary inputs that satisfy the conditions to detect the bridging fault. In this case the detection of a stuck-at 0 on a bridged line is already satisfied. Therefore, we attempt to set 0 on the other line of the bridge. We show an example in Fig. 2. Suppose that there is a non-feedback AND-type bridging fault between lines *d* and *e*, and that we have obtained a partially specified test vector *v* = <*a*, *b*, *c*, *f*, *g*> = <0, 1, x, 0, x> by X-identification. We perform fault simulation with the test vector, and we find out that the stuck-at 0 fault on line *d* can be detected. In this case we set 0 to line *e*. This operation corresponds to activation of the fault site in ATPG using dynamic compaction. In order to set 0 on line *e*, we assign 1 to line *c*. Finally we obtain test vector *v* = <*a*, *b*, *c*, *f*, *g*> = <0, 1, 1, 0, x> which detects the non-feedback AND-type bridging fault between lines *d* and *e*.

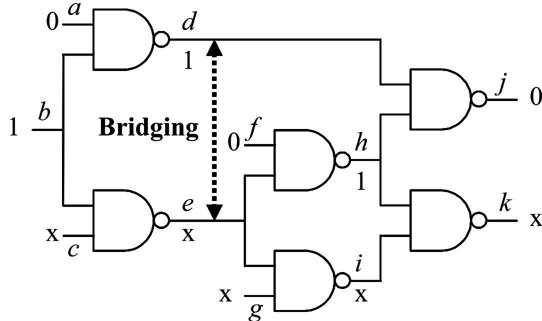

Fig. 2: Logic value assignment for Case 1

Case 2: Suppose that a logic value on one of the bridged lines is 0, and the logic value on the other bridging line is an X. This case satisfies the condition that a logic value is 0 on one of the bridged lines. So, we assign logic values to primary inputs so as to detect the stuck-at 0 fault on the other line. An example is shown in Fig. 3. Suppose that there is a non-feedback AND-type bridging fault between lines *d* and *e*, and that we have obtained partially specified test vector *v* = <*a*, *b*, *c*, *f*, *g*> = <1, 1, x, 0, x> by X-identification. We try to detect stuck-at 0 fault on line *e*. In order to set 1 to line *e* for the activation of the stuck-at 0 fault, we assign 0 to line *c* and we also assign 1 to line *g*

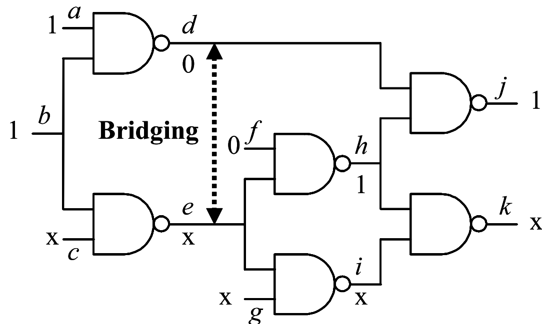

Fig. 3: Logic value assignment for Case 2

for the propagation of the fault. As a result we obtain test vector *v* = <*a*, *b*, *c*, *f*, *g*> = <1, 1, 0, 0, 1> which detects the non-feedback AND-type bridging fault between lines *d* and *e*.

Case 3: Suppose that the logic values on both the bridged lines are Xs. In this case, we assign logic values to primary inputs so as to detect a stuck-at 0 fault on one of the bridging lines and to set 0 on the other bridging line. An example is given in Fig. 4. Assume that there is a non-feedback AND-type bridging fault between lines *d* and *e*, and that we have obtained a partially specified test vector *v* = <*a*, *b*, *c*, *f*, *g*> = <x, 1, x, 0, 0> by X-identification. We assign 0 to line *a*, and 1 to line *c*, and we can detect stuck-at 0 fault on line *d* and set 0 on line *e*. As a result we obtain test vector *v* = <*a*, *b*, *c*, *f*, *g*> = <0, 1, 1, 0, 0> which detects the non-feedback AND-type bridging fault between lines *d* and *e*.

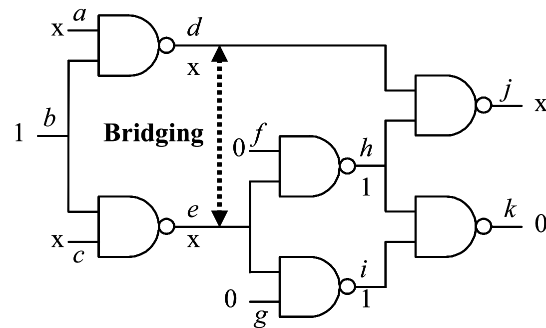

Fig. 4: Logic value assignment for Case 3

4.2. Procedure of logic value assignment

In Fig.5, we show the procedure to detect a bridging fault with test vector modification. After we obtain a test set *T'* with Xs, we first perform fault simulation for the targeted non-feedback bridging faults and collect undetected faults. After that, for each test vector *t'*, we perform fault simulation for stuck-at faults. Then we apply three modification cases proposed in the previous section in sequence using the results of fault simulation for stuck-at faults. After each modification, we remove the bridging faults *bf* which can be detected by the modification. Finally we obtain the modified test set *T"* which detects non-bridging faults in addition to stuck-at faults.

5. Experimental Results

We implemented the proposed method using C programming language on a PC (OS: FreeBSD 4.11 Release, CPU: Pentium4 2.8GHz, memory: 1.0GB), and applied to ISCAS'85 benchmark circuits and full-scan versions of ISCAS'89 benchmark circuits. For a test set to be modified by the proposed method we used a compacted

1081-7735/05 $20.00 © 2005 IEEE
IEEE COMPUTER SOCIETY

```
--------------------------------------------------
Procedure to detect bridge faults (C, T')
Circuit C; Test set with Xs T';
{
    Fault_simulation_for_BF(T');
    BF = collect_undetected_BF();

    For each test vector t' in T'{
        Fault_simulation_for_SF(t');
        Modification_Case1(BF);
        BF = BF-detected_bf;
        Modification_Case2(BF);
        BF = BF-detected_bf;
        Modification_Case3(BF);
        BF = BF-detected_bf;
        Fault_simulation_for_BF(t');
    }
    return modified T''
}
--------------------------------------------------
```

Fig. 5: Procedure of logic assignment

test set [16] for single stuck-at faults. The number of non-feedback AND/OR-type bridging faults is still large compared to the number of stuck-at faults. The large number of faults takes unrealistic time for fault simulation. Besides the small number of bridging faults selected randomly is enough for estimation of the fault coverage [17]. Therefore, we pick non-feedback AND-type bridging faults and non-feedback OR-type bridging faults ten times as the number of signal lines for smaller circuits and five times for larger circuits, respectively.

Table 1 shows the results of test vector modification in order to detect non-feedback AND-type bridging faults. Table 2 shows the results for non-feedback OR-type bridging faults. In both of the tables, the first three columns show circuit name, the number of primary inputs and the number of test vectors, respectively. Next column "%X" shows the percentage of Xs in a test vector on the average. Next column "#brfaults" shows the number of bridging faults in the fault list treated in this work. Results under the column headed "before" shows the number of bridging faults undetected and the percentage of the undetected faults before test vector modification. Results under the column headed "after" is the number and the percentage after test vector modification. Last column shows the percentage of undetected bridging faults reduced by the proposed method.

Tables 1 and 2 show that the proposed method could reduce the percentage of undetected bridging faults especially for larger circuits such as s15850, s35932, s38417, and s38584. For s35932 circuit, the proposed method could reduce over 1000 undetected non-feedback AND or OR bridging faults. For some circuits, the proposed method increase the number of undetected bridging faults. This is because the don't-care identification guarantees not to decrease stuck-at fault coverage, but doesn't consider bridging fault coverage. For these circuits the original test set should be used instead of the test set by the proposed method. In this experiment we tried to detect as many bridging faults as possible. If bridging faults are scored according to significance of detection, the proposed method can detect faults in the order of decreasing significance.

6. Conclusions

We proposed a method to improve defect coverage without increasing the size of a test set and without changing stuck-at fault coverage. We assumed that we could obtain information on defects that often occur from fault analysis. In this work we selected non-feedback AND/OR-type bridging faults representing the defects and we modified a test set so as to detect the bridging faults in addition to stuck-at faults. In the test vector modification, we identified Xs in the test set and assigned logic values to the Xs so as to detect the bridging faults. Experimental results showed that the proposed method decreased the number of non-feedback AND/OR-type bridging faults undetected by the test sets initially generated for stuck-at faults.

Acknowledgment

The work of S. Kajihara and X. Wen were supported in part by JSPS Grants-in-Aid in Scientific Research 16500036 and 17500039, respectively.

References

[1] S. Chakravarty, A. Jain, N. Radhakrisnan, E. W. Savage, S. T. Zachariah, "Experimental Evaluation of Scan Tests for Bridges," *Int'l Test Conf.*, pp. 509-518, 2002.

[2] X. Wen, H. Tamamoto, K. K. Saluja, and K. Kinoshita, "Fault Diagnosis for Physical Defects of Unknown Behaviours," *Asian Test Symp.*, pp. 236-241, 2003.

[3] B. Kruseman, A. Majhi, C. Hora, S, Eichenberger, J. Meirlevede, "Systematic Defects in Deep Sub-Micron Technologies," *Int'l Test Conf.*, pp. 290-299, 2004.

[4] S. Irajpour, S. K. Gupta, M. A. Breuer, "TIMING-INDEPENDENT TESTTING OF CROSSTALK IN THE PRESENCE OF DELAY PRODUCING DETECTS USING SURROGATE FAULT MODELS," *Int'l Test Conf.*, pp. 1024-1033, 2004.

[5] A. Chandra, K. Chakrabarty, "Test Data Compression and Test Resource Partitioning for System-on-a-Chip Using Frequency-Directed Run-Length (FDR) Codes," *IEEE Trans. on Comput.*, Vol. 52, No. 8, Aug. 2003.

[6] A. Wurtenberger, C. S. Tautermann, S. Hellebrand, "DATA

IEEE COMPUTER SOCIETY

COMPRESSION FOR MULTIPLE SCAN CHAINS USING DICTIONARIES WITH CORRECTIONS," *Int'l Test Conf.*, pp. 926-935, 2004.
[7] S. Kajihara, K. Miyase, "On Identifying Don't Care Inputs of Test Patterns for Combinational Circuits," *Int'l Conf. on Computer-Aided Design*, pp. 364-369, 2001.
[8] K. Miyase, S. Kajihara, "XID: Don't Care Identification of Test Patterns for Combinational Circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 23, No. 2, pp. 321-326, Feb. 2004.
[9] A. El-Maleh and A. Al-Suwaiyan, "An Efficient Test Relaxation Technique for Combinational & Full-Scan Sequential Circuits," *VLSI Test Symp.*, pp. 53-59, 2002.
[10] S. M. Reddy, K. Miyase, S. Kajihara, I. Pomeranz, "On Test Data Volume Reduction for Multiple Scan Chain Designs," *ACM Trans. on Design Automation of Electronic Systems,* Vol. 8, No.4, pp. 460-469, Oct. 2003.
[11] S. Kajihara, Y. Doi, L. Li, K. Chakrabarty, "On Combining Pinpoint Test Set Relaxation and Run-Length Codes for Reducing Test Data Volume," *International Conf. on Computer Design*, pp. 387-392, Oct. 2003.
[12] X. Wen, Y. Yamashita, S. Kajihara, L.-T. Wang, K. K. Saluja, K. Kinoshita, "On Low-Capture-Power Test Generation for Scan Testing," *VLSI Test Symposium*, pp. 265-270, May 2005.
[13] P. Goel, and B. C. Rosales, "Test Generation and Dynamic Compaction of Tests," *Digest of Papers 1979 Test Conf.*, pp. 189-192, Oct. 1979.
[14] B. Koenemann, et. al., "A Smart BIST Variant Guaranteed Encoding," *Asian Test Symp.*, pp. 325-330, 2001.
[15] R. Sankaralingam, R. R. Oruganti, N. A. Touba, "Static Compaction Techniques to Control Scan Vector Power Dissipation," *VLSI Test Symp.*, pp. 35-40, 2000.
[16] 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.
[17] I. Pomeranz, S. M. Reddy, S. Kundu, "On the Characterization of Hard-to-Detect Bridging Faults," *Design Automation and Test in Europe Conf. and Exhibition*, pp. 11012-11017, 2003.
[18] S. Sengupta et. Al., "Defect-Based Tests: A Key Enabler for Successful Migration to Structural Test," *Intel Technology Journal*, Q.1, 1999.
[19] V. Krishnaswamy, A. B. Ma, P. Vishakantaiah, "A Study of Bridging Defect Probabilities on a Pentium (tm) 4 CPU," *Int'l Test Conf.*, 2001, pp.688-695.
[20] T. Bartenstein, et. al. "Diagnosing Combinational Logic Designs Using the Single Location At-a-Time (SLAT) Paradigm," *Int'l Test Conf.*, pp. 287-296, 2001.
[21] D. Lavo, I. Hartanto and T. Larrabee, "Multiplets, Models, and the Search for Meaning: Improving Per-Test Fault Diagnosis," *Int'l Test Conf.*, pp. 250-259, 2002.
[22] S. M. Reddy, I. Pomeranz, and S. Kajihara, "Compact Test Sets for High Defect Coverage," *IEEE Trans. on CAD.*, Vol. 16, No. 8, Aug. 1997.

Table 1: Experimenatal results for non-feedback AND-type bridging faults

circuits	#PIs	#tests	%X	#brfaults	before		after		%imp
					#undet	%	#undet	%	
c3540	50	93	53.33	35620	1110	3.12	1023	2.87	7.84
c5315	178	46	61.36	54380	999	1.84	608	1.12	39.14
c7552	207	75	54.53	76600	580	0.76	498	0.65	14.14
s5378	214	100	73.29	27615	160	0.58	284	1.03	-77.50
s9234	247	111	69.15	47420	1740	3.67	1458	3.07	16.21
s13207	700	235	92.04	69845	722	1.03	924	1.32	-27.98
s15850	611	97	77.30	82655	1602	1.94	1224	1.48	23.60
s35932	1763	12	36.20	188300	25005	13.28	23705	12.59	5.20
s38417	1664	87	74.80	200405	1073	0.54	784	0.39	26.93
s38584	1464	114	81.15	200810	6297	3.14	6021	3.00	4.38

Table 2: Experimenatal results for non-feedback OR-type bridging faults

circuits	#PIs	#tests	%X	#brfaults	before		after		%imp
					#undet	%	#undet	%	
c3540	50	93	53.33	35620	1784	5.01	1556	4.37	12.78
c5315	178	46	61.36	54380	856	1.57	673	1.24	21.38
c7552	207	75	54.53	76600	511	0.67	544	0.71	-6.46
s5378	214	100	73.29	27615	187	0.68	235	0.85	-25.67
s9234	247	111	69.15	47420	1566	3.30	1386	2.92	11.49
s13207	700	235	92.04	69845	1296	1.86	1162	1.66	10.34
s15850	611	97	77.30	82655	1831	2.22	1597	1.93	12.78
s35932	1763	12	36.20	188300	15318	8.13	13763	7.31	10.15
s38417	1664	87	74.80	200405	908	0.45	563	0.28	38.00
s38584	1464	114	81.15	200810	8720	4.34	7230	3.60	17.09