Multiple-Fault Diagnosis Based on Single-Fault Activation and Single-Output Observation

Yung-Chieh Lin Kwang-Ting Cheng
Dept. of ECE, University of California, Santa Barbara
Santa Barbara, CA 93106, USA
{younglin, timcheng}@ece.ucsb.edu

Abstract

In this paper, we propose a new circuit transformation technique in conjunction with the use of a special diagnostic test pattern, named SO-SLAT pattern, to achieve higher multiple-fault diagnosis resolutions. For a given list of candidate faults, which could be stuck-at, transition, bridging, or other faults, we generate a set of SO-SLAT patterns, each of which attempts to activate only one fault in the list and propagate its effects to only one observation point. Observing the responses to SO-SLAT patterns helps more precisely identify fault candidates. The method can also tolerate most of the timing hazards for more accurate diagnosis of failures caused by timing faults. The experimental results demonstrate the effectiveness of the proposed method for diagnosing multiple faults.

1. INTRODUCTION

Defects can be modeled at the logic level by faults that affect single or multiple circuit locations and produce erroneous output responses for one or more input test vectors. Fault diagnosis analyzes the observed failing responses and the structure of the circuit under diagnosis (CUD) to search for locations that are potentially faulty. This information is then used in defect analysis, where the CUD is physically examined to determine the failure mechanism. Because physical examination is inevitably slow due to the immense amount of resources needed, the efficiency of defect analysis depends to a great extent on the resolution of fault diagnosis.

Single-fault diagnosis is a well-studied problem with various linear-time techniques proposed to solve it [1]. However, the single-fault model may not be adequate for diagnosing defects in modern devices, which tend to cluster and affect multiple lines in failing chips [2]. Recent experiments [3] confirm that more than 41% of defects found in failing chips cannot be diagnosed using the single stuck-at fault model. Moreover, due to aggressive clocking strategies in modern designs, failures caused by timing defects become more common, and thus accurately identifying timing defects becomes more critical for rapid production volume ramp-up. Diagnosis of circuits with multiple delay faults is a very difficult task because of at least two factors: (1) The solution space grows exponentially with the number of faults, and the interactions between different fault effects further complicate the diagnosis algorithms. (2) The timing defects behave unpredictably due to timing uncertainty and increasing parametric variations.

Many fault-diagnosis approaches with promising results use the idea of SLAT (Single Location At a Time) [3]-[7]. A SLAT pattern is a failing pattern which can be explained by a single location fault. SLAT patterns are used to determine the locations of faults and to build up a composite picture of the multiple faults using the fewest faulty locations. These approaches attempt to find simple fault activation patterns (only one fault is activated by a pattern), so that the diagnosis algorithms, such as response-matching and candidate-scoring, could work better. However, most of existing fault diagnosis methods deal with only static faults. When extending them to diagnose delay faults, the assumption that the fault simulation results match delay defect behavior in real silicon becomes unrealistic. Therefore, their matching mechanisms are likely to produce misleading results.

On the other hand, algorithms using critical-path tracing [8] [9] can alleviate the problem caused by the approaches relying on fault simulation. Based on a single-fault assumption, these algorithms back-trace the sensitized paths from each failing observation point (primary output or scan-cell) and locate possible candidates through intersection of fainin cones of failing observation points. This method is conservative in terms of pruning false candidates and thus will report a larger number of fault candidates. In addition, the concept of "faults must exist in the intersection of fanin cones of failing observation points" generally does not apply in the presence of multiple faults.

We propose a new diagnosis method that combines the benefits of SLAT and path tracing techniques while avoiding their drawbacks. The proposed method is compatible with other state-of-the-art diagnosis methods and thus can be used in conjunction with them. It starts with a fault candidate location list, produced by any existing diagnosis method, which includes all true fault locations of the multiple-fault to be diagnosed. The multiple-fault has an unknown multiplicity with static and/or dynamic (e.g. delay) fault components. We develop a diagnostic test generation procedure based on a transformed circuit model. Special kinds of tests, called the Single-Observation SLAT (SO-SLAT) tests, are generated and applied to identify true fault

3-9810801-0-6/DATE06 © 2006 EDAA

locations and prune false candidates. The SO-SLAT test is a test pattern that, given a list of faults containing a target fault, detects the target fault at a single observation point while guaranteeing that the presence of other faults in the given list will not mask the fault at this observation point. This is achieved by ensuring that the test for the target fault does not activate other faults which have sensitizable paths to the specific observation point.

The rest of the paper is organized as follows. In Section 2, we explain the background of diagnostic test generation and the motivation of using observation point information. Section 3 gives the definition of the SO-SLAT tests and the flow of the proposed diagnosis method. Section 4 shows the experimental results, and Section 5 concludes the paper.

2. BACKGROUND AND MOTIVATION

Manufacturing tests generated by standard ATPG tools have low diagnosability because they often detect multiple faults with a single test [2] and propagate the fault effects only to one or a few observation points, from which the tools can easily generate a test pattern [10]. Therefore, to improve the diagnosability, special patterns with higher diagnosability are generated in addition to the detection test set [11]-[15]. The goal of diagnostic test generation is to find a test such that the circuit produces different responses for different faults. Diagnostic test generation methods in [11] [12] [13] are based on various circuit modification techniques that allow a standard test-generation algorithm for fault-detection to be directly used for diagnostic test generation. The diagnostic test generation procedure proposed in [14] starts with a complete fault-detection test set. For any pair of faults that are not distinguished by the test set, the procedure eliminates some patterns from the original tests, so that they detect only one of the two undistinguished faults. In [15], a special type of fault called Fault Distinguishing Pattern Fault (FDPF) is modeled for an existing ATPG program to effectively generate fault-distinguishing patterns.

Existing diagnostic test-generation methods focus on generating patterns to distinguish a pair of faults. However, in the presence of multiple faults, it is not sufficient merely to differentiate faults in pairs because the activation of other faults beyond the two being distinguished might result in masking or unexpected circuit behavior. Due to the huge set of possible multiple-fault candidates, it is also not feasible to explicitly try all the possible combinations of multiple faults by traditional approaches. A SAT-based diagnosis approach [16] leverages the advances in SAT solving engines and cleverly transforms the multiple-fault diagnosis problem into a SAT problem. However, this approach suffers from huge memory requirements. Besides, when the cardinality of multiple faults increases, this approach would not have high diagnosis resolution.

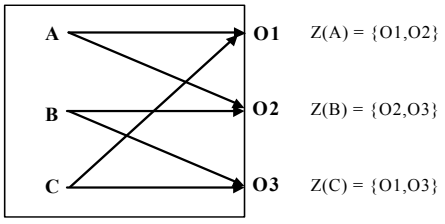

Fig. 1. Example illustrating the use of observation point information

Another interesting fault-diagnosis approach uses the SLAT concept [3]. All the observed failures for a SLAT pattern can be explained fully by a single fault location. The SLAT-based approaches make the assumption that if the observed failures match with the simulation result of a fault, then the fault is present in the CUD. Thus, to determine whether a fault is present or not, an intuitive way is to apply a SLAT pattern for a fault and observe the response of the CUD. However, for existing ATPG tools, it is hard to generate a diagnostic test pattern which activates only one fault and guarantees the fault effect is observable. Existing ATPG algorithms are developed to detect a fault without considering whether other faults will or will not be detected. Thus, we need to impose proper constraints upon ATPG tools such that a SLAT pattern can be generated for every fault in the fault candidate list. These constraints would ensure that only the target fault is activated and that its fault effect can be uniquely observed.

In [17], a concept of Z-Set is presented. A Z-Set of fault f_i, $Z(f_i)$, is a collection of observation points (O_1, O_2,…O_N). A directed path in the circuit leads from the location of f_i to each of the N observation points. Fault pairs with different Z-Sets are always distinguishable because their fault effects can be propagated to different observation points. The authors also reported that public benchmark circuits as well as industrial circuits that they have used for experiments contain a large percentage of faults having unique Z-Sets. This inspires us to consider generating a SLAT pattern which propagates the fault effect of the target fault only to selected observation points. These chosen observation points are in the Z-Set of the target fault, but not in other faults' Z-Sets. In particular, choosing only one observation point seems to be an easy and reasonable heuristic.

Figure 1 illustrates the above idea. Assume fault A has Z-Set (O1, O2), fault B has Z-Set (O2, O3), and fault C has Z-Set (O1, O3). We generate and apply a pattern that detects fault C at O3 while fault B is not activated. Because fault B is not activated and fault A has no sensitizable path to O3, fault C is the only fault that can affect the response at O3 under the application of this pattern. If the CUD's response at O3 is fault-free, then fault C is not present in CUD; otherwise, it is present.

Here, the path-tracing technique can be used to reduce the number of faults to be explicitly considered for inactivation, and in turn, the number of ATPG constraints during the diagnostic test generation. Assume we restrict the response observation at a specific observation point, says O_x.

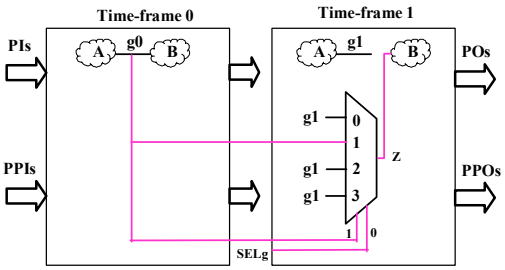

Fig. 2. Circuit model for slow-to-rise transition fault

By tracing back the fanin cone of O_x, we can easily identify faults whose Z-Sets do not contain O_x. We do not need to consider those faults because their fault effects can never be observed at O_x.

Restricting the number of observation points has another advantage. This approach provides a workaround for timing failure diagnosis in the absence of timing information. Timing defects and timing uncertainty resulting from parametric variations, hazards, and pattern-dependency, are too complicated to be modeled in logic/timing simulation. Thus, the mismatches between the responses to at-speed tests and the simulation results have presented a major obstacle for delay fault diagnosis. In our approach, instead of checking all observation points (which very likely have some responses mismatching with the simulation results), we intentionally mask all observation points except the chosen one. We generate a SLAT test and observe the target fault response at only one observation point. Since responses at other observation points are ignored, we would be less likely to be misled by the mismatches.

3. SINGLE OBSERVATION SLAT (SO-SLAT) PATTERNS

Since our diagnosis method starts from a fault candidate list provided by *any* existing diagnosis technique, we do not assume that we have the information of SLAT patterns. In addition, even if the candidate list is provided by a SLAT-based diagnosis program, those SLAT patterns might not be useful for our purpose because they cannot guarantee that only one fault in the fanin core of a failing observation point is activated. Therefore, we first develop a technique to generate a SLAT pattern, SO-SLAT, which detects the fault at a specific observation point.

There are functional equivalence and dominance relationships between the fault candidates. Therefore, before performing the SO-SLAT test generation, we pre-process the fault candidate list to identify both equivalent fault classes and fault dominance relationships among the faults in the candidate list. After this checking process, only one representative fault for each equivalent class is considered as a candidate. The fault dominance relationships are used for pruning false candidates at the end of the diagnosis process.

3.1. Circuit Model for SO-SLAT Pattern Generation

To detect a transition fault, it is necessary to apply two-pattern tests. The first pattern initializes the circuit, and the second pattern activates the fault and propagates the fault effect to observation points. We first transform a sequential circuit into a two time-frame combinational model with a 4-to-1 multiplexer inserted at the location of each fault candidate. Figure 2 shows the circuit model for a slow-to-rise transition fault. The model for a slow-to-fall fault can be constructed in a similar fashion. This model can detect the transition fault activation condition and inject the fault when the condition occurs.

Assume signal g is a slow-to-rise transition fault candidate. Then the port-0, port-2, and port-3 of the multiplexer's data inputs are connected to the faulty signal at time-frame 1 (g1), and the port-1 is connected to the faulty signal at time-frame 0 (g0). The select line port-0 is connected to an extra primary input SELg and select line port-1 is connected to g0. It can be verified that when the extra primary input SELg is set to 0, the model represents the fault-free circuit. On the other hand, when SELg is set to 1, if an input pattern activates g slow-to-rise fault (g0 = 0 and g1 = 1), the output of the multiplexer (Z) is forced to be the value of g0 (i.e. from port-1 of the multiplexer's data inputs). If g slow-to-rise fault is not activated (either g0=g1=0 or g0=1), the Z remains the value of g1. That is, while SELg is set to 1, the g slow-to-rise fault will be injected when it is activated, and this model represents the faulty circuit.

This model can be used for various purposes. (1) Under this model, the task of generating a test for g slow-to-rise fault is equivalent to that of generating a test for SELg stuck-at-0 (or s-a-1) fault. (2) Setting constraints on the signals at timeframe 0 and timeframe 1 can constrain a transition fault to be activated or not. For example, constraining (g0, g1)=(0, 1) will activate the g slow-to-rise transition fault, whereas other value combinations of (g0, g1) will not. (3) As shown in Figure 3, by complementing the extra input (SEL). which is connected to port-1 of the select lines of an inserted multiplexer for a fault at x and connecting it to that for another fault at y, this model can be used for checking fault equivalence between these two faults. For Figure 3, if a test cannot be generated for the select line (SEL) stuck-at-1 fault, then those two faults (x and y) are equivalent.

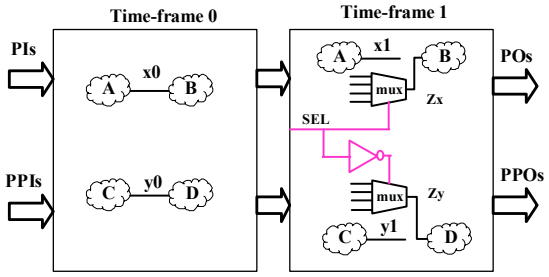

Fig. 3. Circuit model for checking transition fault equivalence

3.2. Fault Equivalence and Dominance Identification

Fault candidates identified by existing diagnosis methods are often indistinguishable by the detection test set used during the diagnosis process. These fault candidates tend to cluster, and among them, there are functional equivalence and dominance relationships with respect to the patterns used. A fault f_i is said to dominate fault f_j if the set of tests that detect fault f_j is a subset of all tests that detect fault f_i. In other words, by contra-positive law, if under the condition of not detecting f_i, no tests exist for f_j, then f_i functionally dominates f_j.

Employing the methods proposed in [18] [19], we use a standard ATPG tool to pre-process the fault candidate list to identify fault-equivalence classes and fault-dominance relationships. Figure 4 illustrates the circuit model for deriving the fault dominance relationship. It is a combinational circuit consists of two *miter* circuits. One is the miter circuit of fault-free circuit C and faulty circuit C_{f1} where fault f_1 is injected by setting the corresponding MUX select line (SEL_{f1}) to 1. The other is a miter circuit of fault-free circuit and faulty circuit C_{f2} with fault f_2 injected. We include fault candidates in C, except f_1, into the target fault list and run ATPG to generate tests with an additional constraint imposed: setting P, the output of the bottom miter circuit, to 0. Because of the miter circuit structure and the imposed constraints, f_1 is untestable. Any fault dominated by f_1 will be untestable as well. By targeting other faults for ATPG using this model, we can identify faults dominated by f_1.

For some faults, the ATPG complexity under this model might be too high so the search could be aborted. If the ATPG process is aborted for fault f_2, a Boolean Satisfiability (SAT)-based technique is invoked to further verify the dominance relationship between f_1 and f_2. As illustrated in Figure 4, under the conditions that $SEL_{f1} = 1$ and $SEL_{f2} = 1$, we check the satisfiability of objectives $P = 0$ (fault f_1 is not detected) and $S = 1$ (fault f_2 is detected). If the objectives cannot be satisfied simultaneously, then f_1 functionally dominates f_2. For each fault candidate, the functional dominance relationship can be derived and expressed in a dominance matrix D, in which an entry $D(i, j) = 1$ if f_i dominates f_j. If two faults dominate each other, then they are functionally equivalent.

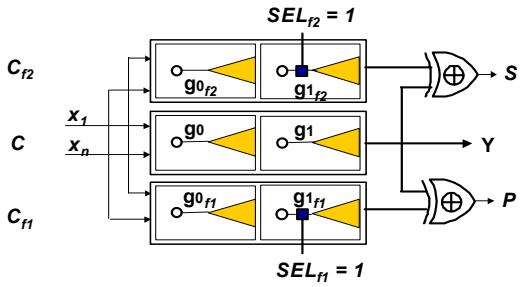

Fig. 4. Circuit model for checking fault dominance

3.3. Procedure of Generation and Application of SO-SLAT Patterns

The following describes the procedure of generating SO-SLAT patterns using a standard ATPG tool based on the circuit model depicted in Figure 2.

Step 0. Identify fault-equivalence classes and fault-dominance relationships, as described in Section 3.2.

Step 1. For a set of unique fault candidates $F = \{f_1, f_2, \ldots, f_n\}$, collect all of their reachable observation points $Z = \cup \{z_1, z_2, \ldots, z_n\} = \{O_1, O_2, \ldots, O_m\}$, where z_i is the Z-Set of fault f_i and O_i is an observation point. Build a fault-observation matrix M, in which an entry $M(i, j) = 1$ if the fault effect of f_i can be propagated to observation point O_j. The set of faults whose fault effects can be propagated to O_j is denoted as $F(O_j)$ and $|F(O_j)|$ is the number of 1's in the j-th column of the M matrix.

Step 2. For each fault f_i in F, which has not been identified as a true fault location or a false candidate, find an observation point O_j with smallest $|F(O_j)|$ in z_i. Impose ATPG constraints to inactivate all faults in $F(O_j)$ except f_i. Run ATPG targeting SEL_i s-a-1 fault for detection at observation point O_j (i.e. the appearance of fault effects at observation points other than O_j is not considered detection). If a test is successfully generated, it is a SO-SLAT pattern for fault f_i, denoted by $SST(f_i)$. If the ATPG tool fails to generate a test, then find the next observation O_k, where $k \neq j$, and $|F(O_j)| \leqq |F(O_k)|$, and try to generate SO-SLAT patterns until a $SST(f_i)$ is generated, or until all observation points have been exhausted and still no $SST(f_i)$ has been successfully generated. If no SO-SLAT pattern can be generated for any of the unidentified faults in the fault list, go to Step 4.

For each fault candidate, Step 2 searches for an observation point to which fewer faults (smaller $|F(O_j)|$) can reach, thus fewer ATPG constraints need to be imposed to inactivate some of the faults. Consequently, the chance of successfully generating a SO-SLAT pattern is higher.

Step 3. Apply generated SO-SLAT patterns to CUD. Based on the test responses of each SO-SLAT pattern, classify faults which have SO-SLAT patterns as true fault locations or false candidates. Go to Step 2.

For instance, if $SST(f_i)$ is a failing SO-SLAT pattern, then f_i is a true fault location; otherwise, fault f_i does not exist in the CUD. However, due to the imposed ATPG constraints (not to activate other faults which have not yet been processed for classification as true or false candidates), it might not be able to generate a SO-SLAT pattern for every fault in a single pattern generation pass. Therefore, this SO-SLAT-based diagnosis procedure is an iterative process that requires several accesses to the tester. Step 3 collects identified fault information (true or false) and impose corresponding constraints on the MUX select lines for the later runs. The port-0 of corresponding MUX's select lines of false faults will be set to 0 and the true faults' will be set

be set to 1. In addition, ATPG constraints for the activation conditions of identified false faults can be removed.

Note that some delay faults may show pattern-dependent behavior because of the transitions on other lines. To cope with pattern-dependent transition faults, it might be beneficial to generate n different SO-SLAT patterns for a target fault candidate, where $1 < n < m$, and m is the number of maximum obtainable different patterns. Based on the analysis in [20], pattern-dependent effects can cause faulty values to disappear but cannot cause new faulty values to be created. Thus, any failing pattern among the applied n SO-SLAT patterns indicates a true fault. Applying an n-detection SO-SLAT test set can avoid misclassifying fault candidates due to the pattern-dependent timing behavior.

Step 4. Collapse fault candidates using identified fault equivalence and dominance relationships.

3.4. Considering Other Fault Models

In addition to transition fault diagnosis, the proposed method can be used for other fault models with little circuit model modification. The circuit model shown in Figure 5(a) is for stuck-at fault diagnosis. It is a one time-frame combinational model. A 2-to-1 multiplexer is inserted at the location of each stuck-at fault candidate. The port-0 of the multiplier is connected to the original signal, and the port-1 is tied to the stuck-at-value (s-a-v) of the fault candidate. For bridge fault diagnosis, we use the 4-way bridge fault model in [21]. If there is a 4-way bridge fault between lines s_1 and s_2, s_1 assumes a faulty value of s_2 when s_1 and s_2 are at opposite values. Figure 5(b) shows the circuit model for a bridge fault. For each of possible bridging nets, a 2-to-1 multiplexer is inserted and the port-0 of the multiplexer is connected to the victim net. The port-1 is connected to another net (the aggressor) which possibly dominates this victim net. Note that for bridge fault model, the SO-SLAT patterns are generated slightly differently. We consider each fault in the candidate list as a possible dominated net and generate SO-SLAT patterns detecting the target net, without masking other possible bridge faults (because there are too many combinations). Because of this relaxation of ATPG constraints, we can generate SO-SLAT patterns for all fault candidates in a single pass. But the responses from the application of SO-SLAT patterns for bridge faults have to be interpreted differently. A failing SO-SLAT pattern

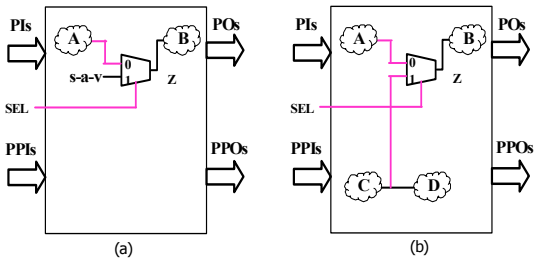

Fig. 5. Circuit model for stuck-at fault and bridge fault models

Table I. Multiple stuck-at faults diagnosis

Circuit	2 faults			3 faults			4 faults		
	I	F	T	I	F	T	I	F	T
S953	6.0	2.1	2.3	9.3	3.4	2.9	12.2	4.2	3.0
S5378	6.0	2.1	2.2	10.8	4.1	2.2	13.5	4.3	2.7
S13207	4.8	2.0	1.5	7.6	3.3	1.5	11.4	4.3	2.0
S15850	5.3	2.2	1.8	8.0	3.1	2.1	10.9	4.5	3.2
S35932	7.1	2.3	1.5	10.7	3.7	1.9	13.1	4.4	2.0
S38417	6.0	2.4	1.7	8.8	3.6	1.4	11.9	5.2	2.3
S38584	5.2	2.4	1.8	10	3.7	2.0	13.0	4.8	2.4

I: # of initial fault candidates F: # of final fault candidates
T: # of tester operations

can no longer be used to identify a true fault because we do not know whether the failing responses are caused by other faults or not. On the other hand, if a fault candidate has no failing responses for all its corresponding SO-SLAT patterns, it would very likely be a false candidate.

In practice, we can first operate the tester at lower speed and identify static faults using a stuck-at fault model. Then, the identified static fault information can be carried to the next phase, which runs the tester at-speed to perform transition fault diagnosis. Because bridge faults often behave either like static fault (low bridging resistance) or dynamic fault (high bridging resistance), thus the fault candidate list contains most nets involved in bridge faults. If the diagnosis resolutions from stuck-at and transition faults are not satisfiable, we can start looking for bridge faults.

4. EXPERIMENTAL RESULTS

Our experiment employs the proposed techniques on several circuits from ISCAS-89 benchmark circuits. Ten faulty instances were randomly generated for each case. Table I shows the diagnosis results of multiple stuck-at faults averaged from ten faulty instances and Table II shows those for transition faults. The first sub-columns under each column labeled "X faults" are the numbers of initial single fault candidates reported by an existing diagnosis method [6], which are around 2 to 3.5 times the number of the injected defects. We use those candidates as the starting point of our method. Second sub-columns show the numbers of fault candidates after applying the proposed diagnosis procedure. Numbers in third sub-columns are the numbers of tester operations for applying SO-SLAT pat-

Table II. Multiple transition faults diagnosis

Circuit	2 faults			3 faults			4 faults		
	I	F	T	I	F	T	I	F	T
S953	5.6	2.0	1.7	8.4	3.0	2.7	12.0	4.0	2.7
S5378	6.0	2.1	1.5	9.3	3.4	1.4	12.6	4.4	1.8
S13207	6.0	2.0	1.9	8.1	3.3	1.7	9.6	4.0	1.6
S15850	6.3	2.1	1.7	10.2	3.3	2.3	12.8	4.0	2.3
S35932	5.9	2.3	1.9	9.4	3.4	2.0	13.5	4.6	2.1
S38417	5.9	2.3	1.8	9.6	3.9	1.9	12.3	4.5	1.9
S38584	5.9	2.1	1.5	9.0	3.0	2.0	13.7	4.3	2.8

I: # of initial fault candidates F: # of final fault candidates
T: # of tester operations

Table III. Bridge faults diagnosis

Circuit	1 fault			2 faults		
	I	F	T	I	F	T
S953	4.6	2.1	1	9.2	4.5	1
S5378	6.4	2.5	1	11.5	5.8	1
S13207	5.1	2.3	1	8.4	4.7	1
S15850	5.7	2.1	1	10.9	5.1	1
S35932	5.9	2.4	1	13.5	6.5	1
S38417	5.9	3.0	1	13.2	5.9	1
S38584	5.8	2.1	1	9.8	4.3	1

terns until no new fault can be identified. In all cases, the proposed method is able to either accurately identify the injected faults or reduce the original fault candidate list to a smaller set of faults which include the true faults. In some cases, the numbers of final fault candidates are even smaller than those of the injected faults. This is because equivalence/dominance relationships exist between the original injected faults. In those cases, we record them just as we do the numbers of injected faults. Note that numbers shown are averaged numbers. In some cases, a poor performance can be attributed to a few faulty instances which are difficult to diagnose using the proposed diagnosis approach. The counts of tester operations are fewer than 3 in all cases. Note that the numbers of SO-SLAT patterns always are less than the numbers of initial fault candidates.

Table III shows the results of bridge fault diagnosis. We conducted experiments only for cases with one or two faults injected because the diagnosability of such faults is relatively low for the diagnosis method [6] we used to produce the initial candidate list. Numbers shown in second sub-columns (F) are in units of nets; thus a number "2" corresponds to one bridge fault. Because of the relaxation of ATPG constraints (faults in the fanin cone of the observation point are not made inactivated), the diagnosis resolution would be less accurate. Some faults might be misclassified as false candidates if their fault effects are always masked by other true faults. However, we did not observe any event of this sort, probably because of the probability is extremely low that every SO-SLAT pattern will activate the true fault and mask the fault effect of the target fault.

5. CONCLUSIONS

We have proposed a new diagnosis method which offers better diagnosis resolution and can be used to enhance any existing state-of-the-art diagnosis processes. The method first identifies fault equivalence and dominance relationships among the faults in the initial candidate fault list obtained by the existing methods. Then, special diagnostic tests are generated, each of which activates only one fault, propagates its fault effect to particular observation points, and checks the response at only one observation point. The experimental results indicate that the method offers very high diagnosis resolution for multiple faults.

6. REFERENCES

[1] N. Jha and S. Gupta, Testing of Digital Systems, Cambridge, U.K., Cambridge Univ. Press, 2003.

[2] R. C. Aitken, Modeling the Unmodelable: Algorithmic Fault Diagnosis, IEEE Design & Test of Computers, Vol. 14, No. 3, Jul./Sep. 1997, pp. 98-103.

[3] L. M. Huisman, Diagnosing Arbitrary Defects in Logic Designs Using Single Location at a Time (SLAT), IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 23, No. 1, Jan. 2004, pp. 91-101.

[4] A. Waicukauski and E. Lindbloom, Failure Diagnosis of Structured VLSI, IEEE Design & Test of Computer, Vol. 6, Aug. 1989, pp. 49-60.

[5] D. B. Lavo, I. Hartanto, and T. Larrabee, Multiplets, Model, and the Search for Meaning: Improving Per-Test Fault Diagnosis, Proc. International Test Conference, 2002, pp. 250-259.

[6] Z. Wang, K.-H. Tsai, M. Marek-Sadowska, and J. Rajski, An Efficient and Effective Methodology on the Multiple Fault Diagnosis, Proc. International Test Conference, 2003, pp. 329-338.

[7] S. Venkataraman and S. B. Drummonds, Poirot: Applications of a Logic Fault Diagnosis Tool, IEEE Design & Test of Computers, Jan. 2001, pp. 19-29.

[8] M. Abramovici, P. R. Memon, and D. T. Miller, Critical Path Tracing – An Alternative to fault simulation, Proc. Design Automation Conference, 1983, pp. 214-220.

[9] P. Girard, C. Landrault, and S. Pravossoudovitch, Delay Fault Diagnosis by Critical-Path Tracing, IEEE Design & Test of Computers, 1992, pp. 27-32.

[10] I. Park, A. AL-Yamani, and E. McCluskey, Effective TARO Pattern Generation, Proc. VTS, 2005, pp. 161-166.

[11] I. Hartanto, V. B oppana, J. H. Patel, and W. K. Fuchs, Diagnostic Test Generation for Sequential Circuits, Proc. VLSI Test Symposium, 1997, pp. 196-202.

[12] A. Veneris, R. Chang, M. S. Abadir, M. Amir, Fault Equivalence and Diagnostic Test Generation using ATPG, Proc. International Symposium on Circuits and Systems, 2004, pp. 221-224.

[13] V. D. Agrawal, D. H. Baik, Y. C. Kim, and K. K. Saluja, Exclusive Test and it's Application to Fault Diagnosis, Proc. International Conference on VLSI Design, 2003, pp. 143-148.

[14] I. Pomeranz and S. M. Reddy, A Diagnostic Test Generation Procedure for Synchronous Sequential Circuits Based on Test Elimination, Proc. In'l Test Conference, 1998, pp. 1074-1083.

[15] T. Bartenstein, Fault Distinguishing Pattern Generation, Proc. International Test Conference, 2000, pp. 820-828.

[16] A. Smith, A. Veneris, and A. Viglas, Design Diagnosis Using Boolean Satisfiability, Proc. Asia and South Pacific Design Automation Conference, 2004, pp. 218-223.

[17] I. Pomeranz, S. Venkataraman, S. M. Reddy, and B. Seshadri, Z-Sets and Z-Detections: Circuit Characteristics that Simplify Fault Diagnosis, DATE, 2004, pp. 68-73.

[18] Y.-C. Lin, F. Lu, and K.-T. Cheng, Accurate Diagnosis of Multiple Faults, in Proc. ICCD, 2005, pp. .

[19] R. K. K. R. Sandireddy and V. D. Agrawal, Diagnostic and Detection Fault Collapsing for Multiple Output Circuits, Proc. Design, Automation and Test in Europe, 2005, pp. 1014-1019.

[20] I. Pomeranz and S. M. Reddy, On Diagnosis of Pattern-Dependent Delay Faults, Proc. DAC, 2000, pp. 59-62.

[21] S. Sengupta et al., Defect-Based Test: A Key Enabler for Successfully Migration to Structural Test, Intel Technology Journal, Q1, 1999.