Multiple-Fault Diagnosis Based On Adaptive Diagnostic Test Pattern Generation

Yung-Chieh Lin, Feng Lu, *Member, IEEE*, and Kwang-Ting Cheng, *Fellow, IEEE*

***Abstract*—In this paper, we propose two fault-diagnosis methods for improving multiple-fault diagnosis resolution. The first method, based on the principle of single-fault activation and single-output observation, employs a new circuit transformation technique in conjunction with the use of a special type of diagnostic test pattern, named single-observation single-location-at-a-time (SO-SLAT) pattern. Given a list of candidate suspects (which could be stuck-at, transition, bridging, or other faults obtained by any existing diagnosis method), we generate a set of SO-SLAT patterns, each of which attempts to activate only one fault in the list and propagate its effects only to a specific observation point. Observing the responses of the circuit under diagnosis to the SO-SLAT patterns helps more precisely determine whether each fault suspect is a true or false candidate. The method can tolerate most of the timing hazards for a more accurate diagnosis of failures caused by timing faults. The second method generates and applies limited-cycle sequential tests, based on a Boolean satisfiability solver, to identify multiple defective signals which can jointly explain the circuit's faulty behavior. These two methods can be applied independently and/or jointly after any existing state-of-the-art diagnosis process to further improve the diagnosis resolution. The experimental results demonstrate the effectiveness of the proposed methods for diagnosing multiple faults, including timing faults.**

***Index Terms*—Boolean satisfiability, diagnosis, testing, very large scale integration (VLSI).**

I. 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 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 solutions [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 a failing chip [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 both microprocessor and system-on-chip designs, failures caused by timing defects become more common. Thus, accurately identifying timing defects becomes more critical for rapid production volume ramp-up. Diagnosis of circuits with multiple delay faults is a very challenging 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 problem. 2) The timing defects behave unpredictably due to various sources of timing uncertainty, including increasing parametric variations.

Many fault-diagnosis approaches with promising results use the idea of single location at a time (SLAT) [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 build up a composite picture of multiple faults involving the fewest faulty locations. These approaches attempt to find simple fault-activation patterns, each of which activates one fault only, so that the diagnosis algorithms, such as response matching and candidate scoring, could work more effectively. However, most of the existing fault-diagnosis methods deal with static faults only. When extending them to diagnose delay faults, the assumption that the fault simulation results will match the delay defect behavior in real silicon becomes unrealistic. Therefore, their matching mechanisms are likely to produce misleading results.

On the other hand, algorithms based on critical-path tracing [8], [9] can alleviate the problem caused by the approaches relying on fault simulation. Based on the single-fault assumption, these algorithms back-trace the sensitized paths from each failing observation point (a primary output or output of a scan cell) and locate possible candidates through the intersection of the fanin 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 assumption that "faults must exist in the intersection of the fanin cones of failing observation points" is no longer valid in the presence of multiple faults.

We propose a diagnosis method which combines the advantages of the SLAT and the path-tracing techniques while avoiding their drawbacks. The proposed method is compatible with other state-of-the-art diagnosis methods and can be used after the application of other diagnosis methods with lower resolution. As shown in Fig. 1, the proposed approach starts with a list of fault candidate locations 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

Manuscript received January 7, 2006; revised April 11, 2006. This paper was recommended by Associate Editor A. Ivanov.

The authors are with the University of California, Santa Barbara, CA 93106 USA.

Digital Object Identifier 10.1109/TCAD.2006.884486

0278-0070/$25.00 © 2007 IEEE

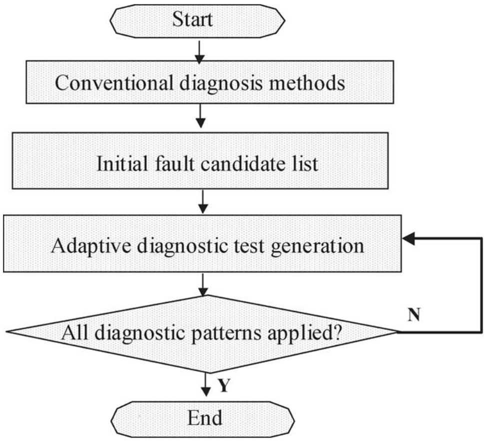

Start
Conventional diagnosis methods
Initial fault candidate list
Adaptive diagnostic test generation
All diagnostic patterns applied?
N
Y
End

Fig. 1. Proposed two-stage diagnosis procedure.

components. We further develop a diagnostic test-generation procedure based on a transformed circuit model. A special kind of test, called single-observation SLAT (SO-SLAT) test, is generated and applied to identify true fault locations and prune false candidates. For a given list of faults which contain the target fault, a SO-SLAT test detects the target fault at a single observation point and guarantees that the presence of other faults in the given fault 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.

Due to the huge set of possible multiple-fault candidates, it is not feasible to explicitly try all the possible combinations of multiple faults by traditional approaches. A SAT-based diagnosis approach [10], [11] leverages the advances in SAT solving engines and cleverly transforms the multiple-fault diagnosis problem into a SAT problem. This approach is capable of identifying fault multiplets. A multiplet is a collection of faults which can jointly explain all the failing patterns. However, this approach suffers from huge memory requirements. These requirements arise from the need for duplicate copies of the circuit model for each of the applied test vectors. Thus, to reduce the memory requirement while maintaining the diagnostic capability of identifying fault multiplets, in the second part of this paper, we further propose an intelligent diagnostic test-generation method which can: 1) reduce the number of test vectors and 2) avoid the circuit duplication to improve the performance of SAT-based diagnosis method. We propose the use of a special kind of test called the multiple-capture antidetecting (MC–AD) test. Given a set of faults containing a target fault, the MC–AD tests are a set of limited-cycle sequential test vectors that, while detecting other faults in the fault list, do not detect the target fault. The MC–AD tests are particularly useful for SAT-based diagnosis which can identify multiplets. We employ an efficient sequential SAT solver [12] which utilizes the MC–AD tests to improve the performance of SAT-based diagnosis.

The rest of the paper is organized as follows. In Section II, we explain the background of diagnostic test generation and fault diagnosis, especially the SAT-based diagnosis approach. In addition, the motivation for using observation point information and MC–AD tests is described. Section III gives the definition of the SO-SLAT tests and the flow of the proposed diagnosis method using SO-SLAT tests. Section IV presents the definitions the MC–AD tests, the advantages of using these tests for diagnosis, and the corresponding diagnostic flow. Section V shows the experimental results, and Section VI presents the conclusion.

II. Background and Motivation

A. Diagnostic Test Generation

Manufacturing tests generated by standard automatic test pattern generation (ATPG) tools have low diagnosability because each of the tests often detects multiple faults [2] and propagates the fault effects to only one or few observation points, for which the ATPG tools can easily generate a test pattern [13]. Therefore, to improve the diagnosability, special patterns with higher diagnosability are needed in addition to the detection test set [14]–[18]. 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 [14]–[16] 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 [17] starts with a complete fault-detection test set. For any pair of faults that cannot be distinguished by the test set, the procedure eliminates some patterns from the original tests, so that the remaining tests detect only one of the two undistinguished faults. In [18], a special type of fault, called the fault distinguishing pattern fault, 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. In the presence of multiple faults, it is not sufficient merely to differentiate faults in pairs because the activation of other faults beyond the pair being distinguished might result in masking or unexpected circuit behavior.

B. Deterministic SLAT Patterns for SLAT-Based Diagnosis

SLAT-based diagnosis 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. These approaches attempt to find SLAT patterns from the manufacturing detection test set [3]. However, SLAT patterns might not be available in the set for some faults in the fault candidate list. Thus, it would be necessary to generate and apply additional deterministic SLAT patterns for those faults and observe the responses of the CUD. Existing ATPG algorithms are developed to detect a fault without considering whether other faults could be detected. Thus, proper constraints need to be imposed upon ATPG tools for SLAT pattern generation. These constraints would ensure that only the target fault is activated and that its fault effect can be uniquely observed.

In [19], a concept of Z-set is presented. A Z-set of fault f_i, $Z(f_i)$, is a collection of observation points $(O_1, O_2, \ldots, O_N)$. A directed path in the circuit leads from the location of f_i to each of the N observation points. Fault pairs with different

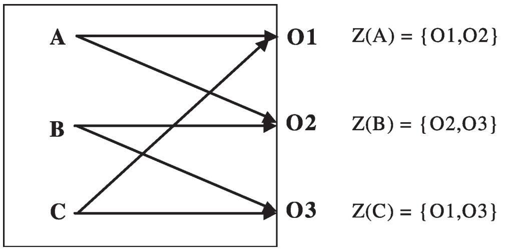


Fig. 2. Example illustrating the use of observation point information.

Z-sets are always distinguishable because their fault effects can be propagated to different observation points. This research 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 inspired 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.

Fig. 2 illustrates the above idea of using observation point information. 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 for this pattern. Fault C is present if and only if the CUD's response at O3 is faulty.

The path-tracing technique can be used to reduce the number of faults to be explicitly considered for inactivation, and, in turn, to reduce the number of ATPG constraints during the diagnostic test generation. Suppose we restrict the response observation at a specific observation point O_x. By tracing back the fanin cone of O_x, we can easily identify faults whose Z-sets do not contain O_x. During ATPG, we do not need to explicitly avoid activating those faults because their fault effects can never be observed at O_x.

Restricting the number of observation points has another advantage—it is more tolerant to timing uncertainties for diagnosing delay faults. Timing defects and timing uncertainty resulting from parametric variations, hazards, and pattern-dependence 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. By applying a SLAT test, observing the target fault response at only one observation point, and ignoring responses at all other observation points, the probability of being misled by the mismatches would be significantly reduced.

C. Non-SLAT Patterns for SAT-Based Diagnosis

A Boolean-satisfiability-based (SAT-based) fault diagnosis approach for multiple faults was proposed in [10] and [11]. This approach formulates the fault diagnosis task as a SAT problem and utilizes a SAT solver to find fault multiplets which could correct the given diagnostic vectors. The approach introduces a multiplexer (MUX) at each potential faulty line. The zero input of the MUX is connected to the original signal, and the one input is an extra input. Therefore, at the beginning when all N signals in the given candidate list are candidate faulty signals, this model introduces N MUXs and $2N$ extra primary inputs (N inputs for the one inputs and N inputs for the select lines of the MUXs). By properly assigning values at the select lines and the one-inputs of the MUXs, the augmented circuit can emulate a circuit behavior matching the observed test responses. A SAT solver is used to find such proper value assignments for the modified circuit which is typically represented in a conjunctive normal form (CNF). This approach is able to identify solutions consisting of fault multiplets.

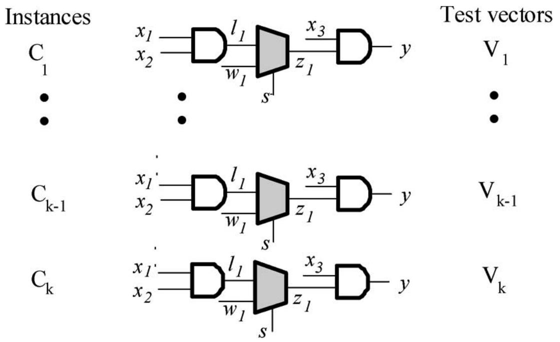


Fig. 3. Circuit construction for SAT-based diagnosis.

Given a circuit netlist and a set of k test vectors V, the SAT-based diagnosis algorithm builds a CNF formula

$$\Phi_s = \prod_{j=1}^{k} \prod_{m=1}^{m_j} C^{j,m}(L^{j,m}, W^{j,m}, X^{j,m}, Q_I, Y^{j,m}, S). \quad (1)$$

The Φ_s is the conjunction of the number of $k \times m_j$ CNF formulas $C^{j,m}(L^{j,m}, W^{j,m}, X^{j,m}, Q_I, Y^{j,m}, S)$, where $1 \leqq j \leqq k$, $1 \leqq m \leqq m_j$, k is the number of vectors, and m_j is the length of test sequence V_j. As shown in Fig. 3, each $C^{j,m}$ encodes constraints from test sequence V_j on the logic netlist C_j consisting of $L^{j,m}$, $W^{j,m}$, $X^{j,m}$, $Y^{j,m}$, and S, where capital letters L, W, X, and Y represent a set of signals. As in the case where $k = 2$ and $m_j = 3$, $X^{2,3}$ represents $\{x_1^{1,1}, x_1^{1,2}, x_1^{1,3}, x_1^{2,1}, x_1^{2,2}, x_1^{2,3}, x_2^{1,1}, x_2^{1,2}, x_2^{1,3}, x_2^{2,1}, x_2^{2,2}, x_2^{2,3}\}$. S represents the set of select lines of the added MUXs in the logic netlist. Note that, for each select line s in S, the value must be the same for all vectors. Q_I is the initial state variables for all vectors. It is shown that Φ_s is satisfied if and only if there is a set of faulty values that can be injected into the circuit, so that the circuit responses match the observed failing responses at the primary outputs Y for all vectors in V. In other words, the conjunction requires that every candidate set of faults satisfy all constraints imposed from all vectors, similar to the intersection of solutions in traditional effect–cause diagnosis.

For SAT-based diagnosis, each input test vector (corresponds to a set of constraints) results in a unique copy of the CUD. Thus, this approach needs to limit the number of applied test vectors to keep memory requirement within a reasonable range. In the implementation of [11], a total of 20 vectors are divided into four sections, which then are applied sequentially. In addition, because it is impractical to consider the whole diagnosis space $(\#\text{ckt line})^{(\#\text{ of errors})}$, the authors employ a second component $E_N(S)$ to (1) to encode constraints on the cardinality of injected faults as a user-specified parameter. Thus, the approach can report multiplets with a limit on the number of faulty signals specified by the user.

To reduce the memory requirement while maintaining the diagnostic capability of identifying fault multiplets, we propose an enhanced method with the following goals: 1) reducing the number of test vectors without comprising resolution and 2) avoiding circuit duplication.

Two observations lead to our proposed SAT-based diagnosis procedure.

1) Observation 1: While SLAT patterns are good for traditional effect–cause diagnosis, non-SLAT patterns are better for SAT-based diagnosis. The reason is that, in SAT-based approaches, a SLAT pattern, which only activates and sensitizes one fault location, will result in only one MUX select line with value assignment and leave all other MUX select lines as free (i.e., unassigned) variables. Consequently, there will be far too many solutions reported by a SAT solver for such patterns. Therefore, the SAT solver would need a large number of SLAT patterns in order to restrict its search space. On the other hand, non-SLAT patterns which could activate and sensitize multiple faults would likely result in significantly fewer solutions.

2) Observation 2: Fault candidates identified by existing diagnosis methods are often indistinguishable by the manufacturing tests used for 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 for diagnosis.

Based on the first observation, we would prefer to select non-SLAT patterns for SAT-based diagnosis. It has been observed, however, that the majority of the manufacturing failing test patterns of scan designs can be attributed to a single faulty location [3], [6], i.e., the majority of the manufacturing scan patterns that detect actual defects are SLAT patterns. However, as reported by the study in [3], sequential tests usually do not have the SLAT property. That is, the failing responses produced by the sequential tests designed for a set of target faults are unlikely to be explained by a single faulty location. In other words, the failing sequential tests are more likely to be non-SLAT patterns. Based on this observation, we propose to generate a special type of non-SLAT limited-cycle sequential patterns, named MC tests, which maximize the number of faults activated and increase the sequential reconvergence (interaction between fault effects). The experimental results reported in [20] indicate that activating multiple faults will increase the probability of fault convergence and fault masking. A similar experiment, reported in [21], for correcting design errors also confirms that the presence of error effect interaction grows with the number of activated errors.

The second observation indicates the need for generating additional distinguishing patterns for diagnosis. However, generating distinguishing patterns is computationally expensive if multiple faults are present. The procedure must exhaust all pairs of faults in the fault-candidate list, each of which requires a distinguishing pattern. To reduce the computational complexity and take multiple faults into account, we propose to generate a special type of distinguishing patterns called AD tests. We further combine the characteristics of both MC and AD tests and generate diagnostic test patterns called MC–AD tests, which are particularly suitable for SAT-based diagnosis.

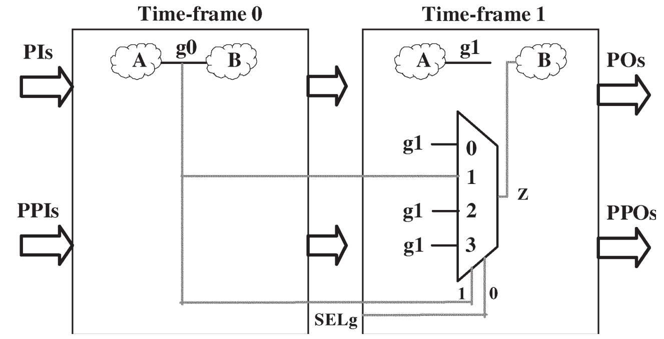


Fig. 4. Circuit model for slow-to-rise transition fault.

III. SLAT-Based Diagnosis Using SO-SLAT Patterns

We assume that the proposed diagnosis method starts from a fault candidate list, which can be provided by any existing diagnosis technique. In this section, we describe the procedure for generating a special type of SLAT pattern, called SO-SLAT pattern, which detects a fault at a specific observation point. We use transition faults as an example for illustrating the test-generation process.

There are functional equivalence and dominance relationships between the fault candidates in the given fault list. Therefore, before performing the SO-SLAT test generation, we preprocess 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.

A. Circuit Model for SO-SLAT Pattern Generation for Transition Faults

To detect a transition fault, it is necessary to apply a two-pattern test. 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 four-to-one MUX inserted at the location of each fault candidate. Fig. 4 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 a signal g is a slow-to-rise transition fault candidate. Then, port-0, port-2, and port-3 of the MUX's data inputs are connected to the faulty signal at time-frame one $(g1)$, and port-1 is connected to the faulty signal at time-frame zero $(g0)$. The select line port-0 is connected to an extra primary input

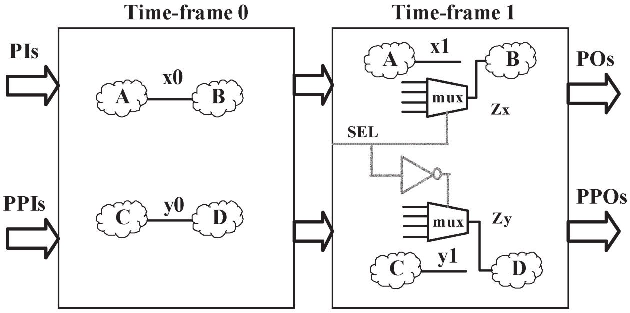


Fig. 5. Circuit model for checking fault equivalence.

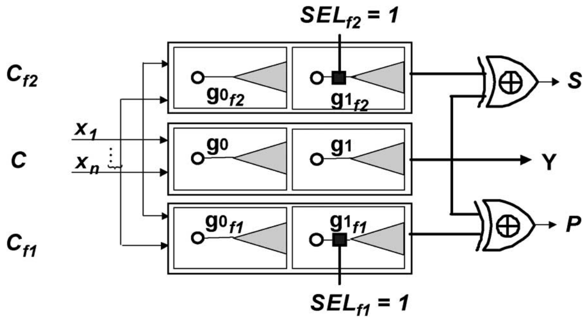


Fig. 6. Circuit model for checking fault dominance.

SELg, and select line port-1 is connected to $g0$. It can be verified that, when the extra primary input SELg is set to zero, the model represents the fault-free circuit. On the other hand, when SELg is set to one, if an input pattern activates g slow-to-rise fault ($g0 = 0$ and $g1 = 1$), the output of the MUX (Z) is forced to be the value of $g0$ (i.e., from port-1 of the MUX's data inputs). If the 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 one, 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 the g slow-to-rise fault is equivalent to that of generating a test for the SELg stuck-at-0 (or s-a-1) fault. This is because the model with SELg $= 0$ represents the fault-free circuit, and SELg $= 1$ represents the faulty circuit with the g slow-to-rise transition 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 Fig. 5, by complementing the extra input (SEL) which is connected to port-1 of the select lines of an inserted MUX 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 transition faults. For Fig. 5, if a test cannot be generated for the select line SEL s-a-1 fault, then those two faults (x and y) are equivalent.

B. Fault Equivalence and Dominance Identification

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 the contra-positive law, if under the condition of not detecting f_i no tests exist for f_j, then f_i dominates f_j.

By employing the methods proposed in [22] and [24], we use a standard ATPG tool to preprocess the fault candidate list to identify fault-equivalence classes and fault-dominance relationships. Fig. 6 illustrates the circuit model for deriving the fault-dominance relationship. It is a combinational circuit consisting of two miter circuits [25]. One is the miter circuit connecting (through an XOR gate) the fault-free circuit C and the faulty circuit C_{f1}, where fault f_1 is injected by setting the corresponding MUX select line (SEL_{f1}) to one. The other is a miter circuit of the fault-free circuit and the 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 zero. 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, say, fault f_2, a SAT-based technique is invoked to further verify the dominance relationship between f_1 and f_2. As illustrated in Fig. 6, under the conditions that $\mathrm{SEL}_{f1} = 1$ and $\mathrm{SEL}_{f2} = 1$, we check the satisfiability of objectives $P = 0$ (i.e., fault f_1 is not detected) and $S = 1$ (i.e., fault f_2 is detected). If the objectives cannot be satisfied simultaneously, then f_1 dominates f_2. For each fault candidate, the 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.

C. 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 Fig. 4.

Step 1) Identify fault-equivalence classes and fault-dominance relationships, as described in Section III-B. Select only one representative fault for each equivalence class to form a unique fault candidate list F.

Step 2) 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 1s in the jth column of the M matrix.

Step 3) 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 the smallest $|F(O_j)|$

in z_i. Impose ATPG constraints using the method mentioned in Section III-A to inactivate all faults in $F(O_j)$, except f_i. Run ATPG targeting SEL_i s-a-1 fault for detection only 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 $\mathrm{SST}(f_i)$. 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 a SO-SLAT pattern with respect to O_k. This process iterates until an $\mathrm{SST}(f_i)$ is generated, or until all observation points have been exhausted. If no SO-SLAT pattern can be generated for any of the unidentified faults in the fault list, go to Step 4) skipping Step 3).

For each fault candidate, Step 2) searches for an observation point to which fewer faults can reach (i.e., having a smaller $|F(O_j)|$); 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 4) Apply generated SO-SLAT patterns to the CUD. Based on the test response of each SO-SLAT pattern, classify each of the faults, which have SO-SLAT patterns, as either a true fault location or a false candidate. Go to Step 2) and repeat.

For instance, if $\mathrm{SST}(f_i)$ 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 possible 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 access to the tester multiple times. Step 3) collects data to classify the fault candidates (as true or false) and impose corresponding constraints on the MUX select lines for the future runs. The port-0 of the corresponding MUXs select lines of the false faults will be set to zero and the true faults will be set to one. In addition, the ATPG constraints with respect to the activation conditions of the 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. Based on the analysis in [26], pattern-dependent effects can cause faulty values to disappear but cannot create new faulty values. Thus, any failing pattern among the applied n SO-SLAT patterns would indicate a true fault. Applying an n-detection SO-SLAT test set can avoid misclassifying fault candidates due to the pattern-dependent timing behavior.

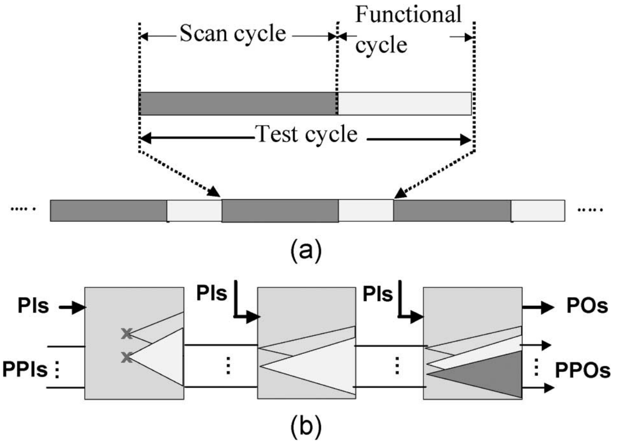


Fig. 7. MC tests.

Step 5) Collapse fault candidates, using identified fault-dominance relationships.

IV. SAT-Based Diagnosis Using MC–AD Patterns

In the following, we give the definitions and the test-generation procedure for the MC–AD tests. The test application and implementation details of the proposed diagnosis procedure will be explained later in this section.

A. MC–AD Test

Fig. 7 illustrates the test application scheme of test patterns having MC property and how these patterns benefit SAT-based diagnosis. We use the following definitions.

Scan cycle — A scan cycle is the period during which a test pattern is shifted into (or the response is shifted out of) the scan chains. If the length of the longest scan chain is N, then one scan cycle corresponds to N clock cycles.

Functional cycle — A functional cycle is the period during which the circuit is in the functional (i.e., capture) mode. A functional cycle may consist of one to several capture cycles. A functional cycle is between two adjacent scan cycles.

Test cycle — A test cycle consists of one scan cycle immediately followed by a functional cycle.

As shown in Fig. 7(a), a test pattern with MC property (MC test) for a scan-based design is a sequence of input vectors which have specified values at primary inputs in every cycle. Test vectors for the scan cells are shifted into the scan chains only during scan cycles. The values in the scan cells during the functional cycle are derived by the system functional logic. Most commercial ATPG tools have the capability of generating MC tests for a scan-based design.

A fault detected by an MC test must be activated and propagated through multiple timeframes that represent the operation of the circuit in MC cycles. Performing test generation on this multiple-timeframes circuit model creates complex fault activation and propagation conditions, as shown in Fig. 7(b).

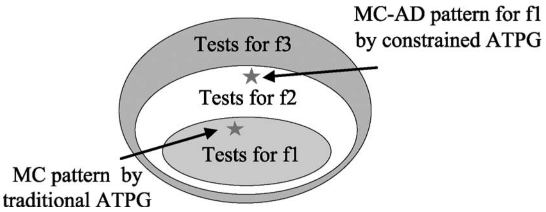


Fig. 8. MC test versus MC–AD tests.

The shaded areas depict the fault effect propagation zone across three timeframes of two faults. The MC tests are generated, in some sense, to detect faults multiple times. That is, if we could observe the values at both primary outputs and the scan cells in every cycle, we could observe erroneous responses in every cycle. Therefore, we can view an MC test as a collection of multiple-detection tests. Multiple-detection tests have been shown to be effective for fault diagnosis [27]. In addition to the benefits to multiple-fault diagnosis, MC tests are particularly useful for SAT-based diagnosis. The sequential reconvergence and multiple-detection properties of the MC tests impose stricter constraints on the SAT solver, limit its search space, and thus reduce the number of possible candidate locations reported by the SAT solver.

Test patterns with AD property are intended to differentiate one fault from the others. Let a set of faults F contain N faults. AD tests for fault f_i in F is a collection of patterns that do not detect f_i but do detect the rest $N-1$ faults f_j, where $j \neq i$. The circuit model shown in Fig. 6 can be used in generating AD tests for fault f_i by setting the corresponding MUX select line (SEL_{f1}) to one, i.e., injecting fault f_i. A standard ATPG tool can be used to generate tests for the rest $N-1$ faults in C with an additional constraint that the output P of the miter circuit has to be zero. Because of the miter circuit structure and the constraint, the ATPG tool will not be able to generate any test detecting f_i, whereas it can generate tests for all other faults. AD tests may consist of several patterns. Also, it might not always be possible to detect all other faults. If there are undetectable faults under the AD constraint, there must exist some equivalence or dominance relationships among the fault candidates.

The fault candidates reported by conventional diagnosis methods often have fault-dominance relationships among them, which causes challenges for further improvement to diagnosis resolution. However, the AD tests can best be used for such situations. Fig. 8 illustrates an example which consists of three faults, f1, f2, and f3. As shown in the figure, fault f1 is dominated by fault f2, and faults f1 and f2 are dominated by fault f3. If we generate an MC test by a traditional ATPG tool, after test compression, the generated test would likely detect all three faults (as the objective of traditional ATPG and test compression is to detect as many faults with as few tests as possible). This pattern would not be effective for SAT-based diagnosis because, although it activates three faults, their fault effects cannot be distinguished. On the other hand, if we generate an MC test with the additional AD requirement for f1, the patterns would be in the middle set, as shown in the figure. This pattern will then be useful for distinguishing f1 from the others, and thus, the SAT solver would report fewer solutions.

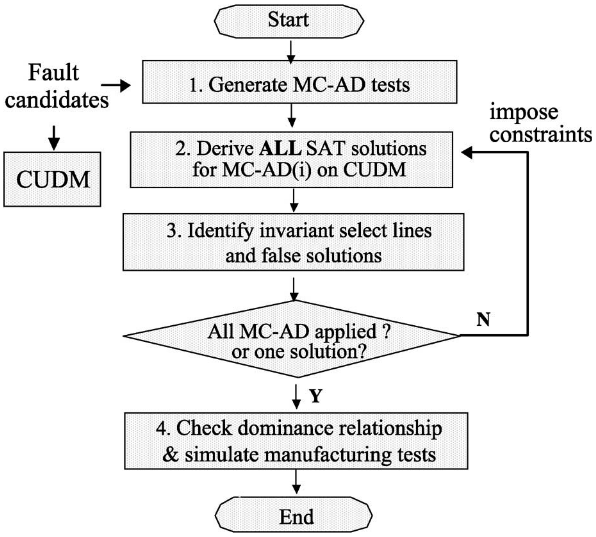

Start
Fault candidates
1. Generate MC-AD tests
CUDM
2. Derive ALL SAT solutions for MC-AD(i) on CUDM
impose constraints
3. Identify invariant select lines and false solutions
All MC-AD applied ? or one solution?
N
Y
4. Check dominance relationship & simulate manufacturing tests
End

Fig. 9. SAT-based diagnosis procedure.

B. Procedure of Generation and Application of MC–AD Patterns

The second diagnosis procedure described in the following combines the benefits of MC tests (which maximize non-SLAT property with a small number of patterns) and AD tests (which distinguish one fault from the others). The procedure first generates MC–AD tests for fault candidates in the suspect list and then applies an efficient sequential SAT solver [12] with performance-enhancement heuristics specific to this application. The proposed SAT-based diagnosis flow is summarized in Fig. 9.

Step 1) Identify fault-equivalence classes and fault-dominance relationships, as described in Section III-B.

Step 2) Construct a modified CUD with additional MUXs inserted at locations of all fault candidates, denoted as CUDM (the insertion was discussed in Section II-C). For every fault f_i in the fault candidate list F containing N faults, generate a set of AD tests, each of which is a k-cycle MC test as well. The set is denoted as MC-AD(i), where $1 \leqq i \leqq N$. In our experiment, we set k to three.

Step 3) For each fault candidate, simulate its MC-AD(i) on the CUD and record the circuit output responses. The MC-AD(i) and its corresponding circuit output responses are then converted into a set of constraints, denoted as SAT_CON(i). Start from the first constraint; impose it upon CUDM and apply a SAT solver to find a solution which satisfies the constraint. A solution is a set of value assignments at the select lines of the inserted MUXs that makes the CUDM's behaviors match the observed faulty responses at the CUD outputs. Derive all SAT solutions, denoted as SAT_SOL(i).

We use an efficient circuit-based sequential SAT solver [12]. It does not explicitly expand the circuit into multiple timeframes. Instead, it works on a single copy of the CUDM while imposing different sets of constraints (from the pattern in the corresponding timeframe) in different cycles. Therefore, it is highly memory-efficient and more scalable to larger circuits. In addition, it is capable of deriving multiple solutions per run and can accumulate previous solutions as conflict clauses to prevent producing the same solutions; thus, it can derive all solutions in a relatively short time.

Step 4) Perform the intersection of all derived solutions. Based on the intersection results, identify invariants at MUXs' select lines and false solutions.

Set the initial intersection of solutions Int_SOL to SAT_SOL(1) which is the SAT solution for the first set of MC–AD tests. After finishing applying the MC-AD(j), intersect the solutions SAT_SOL(j) with the previous solutions SAT_SOL(i), where $1 \leqq i < j$ and store the intersected solutions [which are the same solutions appearing in both SAT_SOL(i) and SAT_SOL(j)] as Int_SOL(j). Because the true solution must be in each of the SAT_SOLs, it would be in the intersection of SAT_SOLs as well. Therefore, if the select line of a specific inserted MUX has the same value for all solutions in Int_SOL(j), then the fault corresponding to the MUX can be determined as present or absent (a present fault if the select line is always one and absent if it is always zero). Moreover, during the solving process, a solution not in the Int_SOLs must be a false solution. The identified invariants at select lines and the false solutions can then be imposed upon the circuit model as constraints in the later runs of SAT solving, such that the overall SAT solving process becomes more and more efficient toward the later runs.

Step 5) Check the dominance relationship for undetermined fault candidates and simulate the original manufacturing detection tests (which identify the defective part under diagnosis) to validate each fault multiplet.

After all MC–AD tests are applied, there might be more than one solution. This is because the SAT-based diagnosis might not be able to distinguish fault candidates with a limited number of MC–AD tests (each of which is limited to a certain number of capture cycles) due to the functional equivalence and dominance relationships among candidates of fault multiplets. At this moment, the fault-dominance matrix D can be used to determine whether the remaining multiplets are equivalent or have dominance relationships. After we collapse all possible equivalent and dominated solutions, the original manufacturing tests, including both passing and failing patterns, are simulated to validate each of the remaining possible fault multiplets. A multiplet is considered a false candidate and thus eliminated from the candidate list if the simulation results for the corresponding faulty circuit model (which contains the fault multiplet) do not produce the same response as that of the faulty chip.

In the above discussion, we use the transition fault model to illustrate the concept, model, and procedures. However, the proposed method can be used for other fault models with minor circuit model modification [23]. For example, for stuck-at fault diagnosis, we can construct a one-time-frame combinational model. A two-to-one MUX 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. In practice, we can first operate the tester at lower speed and identify static faults using the 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 (low bridging resistance) or dynamic faults (high bridging resistance), the fault candidate list contains most nets involved in bridge faults.

V. Experimental Results

Our experiments first employ the two proposed techniques separately and then evaluate their combined effectiveness on several circuits from ISCAS-89 benchmark set. 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 the transition faults. The subcolumns under the column labeled "circuit" are the statistics of fault diagnosis results reported by an existing diagnosis method [6]. They include the number of reported fault candidates, which are around 2–3.5 times the number of the injected defects, and the number of initial multiplets candidates which are $2^{(\#\text{ of initial fault candidates})}$. We use those candidates as the starting point of our method.

The column labeled "SO-SLAT (a)" shows the results after applying the proposed SO-SLAT-based diagnosis procedure. The first subcolumn shows the number of final fault candidates and the second subcolumn shows the number of required iterations, each of which requires tester access for applying and observing a set of SO-SLAT patterns until no new fault can be identified. The average numbers of all cases for different circuits are also listed. Note that, for each fault candidate, we attempt to generate a corresponding SO-SLAT pattern. Thus, the number of SO-SLAT patterns is the same as or smaller than the number of fault candidates. Numbers in the column labeled "MC–AD (b)" are the results of applying the MC–AD-based diagnosis method. The first subcolumn under "MC–AD (b)" shows the number of multiplets after applying the MC–AD-based diagnosis procedure. The second subcolumn shows the number of MC–AD tests generated. The third subcolumn lists the CPU time consumed. In all cases, the proposed SO-SLAT-based method is able to either accurately identify the exact set of injected faults or reduce the original fault candidate list to a much smaller set of faults that include the true faults. In some cases, the number of final fault candidates was even smaller than that of the injected faults. This is because there are

TABLE I
MULTIPLE STUCK-AT FAULTS DIAGNOSIS

	Circuit			SO-SLAT (a)		MC-AD (b)			(a) + (b)				
		Ci	MPi	Ca	T	MPb	# of pttn	CPU	Cab	MPab	T	# of pttn	CPU
2 Faults	S5378	6.2	74	2.1	2.2	1.0	6.6	8.1	2.0	1.0	3.2	2.5	0.9
	S13207	4.8	28	2.0	1.5	1.0	4.1	2.6	2.0	1.0	2.5	2.1	0.8
	S15850	5.3	39	2.2	1.8	1.0	4.8	3.2	2.0	1.0	2.8	2.9	2.0
	S35932	7.1	137	2.3	1.5	1.1	7.8	56.6	2.0	1.0	2.5	7.1	23.9
	S38417	6.0	64	2.5	1.7	1.1	5.6	31.3	2.1	1.1	2.7	3.5	16.7
	S38584	5.4	42	2.4	1.8	1.1	5.1	83.8	2.1	1.1	2.8	2.8	10.3
	Avg	5.80	64	2.25	1.75	1.05	5.67	30.9	2.03	1.03	2.75	3.48	9.1
3 Faults	S5378	10.8	1783	4.2	2.2	1.0	14.9	67.9	3.0	1.0	3.2	12.5	49.4
	S13207	7.6	194	3.5	1.5	1.3	7.9	14.1	3.0	1.0	2.5	4.8	59.1
	S15850	8.0	256	3.1	2.1	1.0	9.2	8.3	3.0	1.0	3.1	4.7	13.7
	S35932	10.7	1663	3.6	1.9	1.3	16.3	158.3	3.2	1.2	2.9	11.8	60.4
	S38417	9.0	512	4.1	1.1	1.0	10.8	154.8	3.0	1.0	2.1	8.8	68.4
	S38584	9.6	776	3.7	1.9	1.5	12.6	558.9	3.3	1.3	2.9	11.2	298.1
	Avg	9.28	864	3.7	1.78	1.18	11.95	160.4	3.08	1.08	2.78	8.97	91.5
4 Faults	S5378	13.5	11585	4.3	2.7	1.0	21.4	165.9	4.0	1.0	3.7	11.3	11.4
	S13207	11.4	2702	4.4	2.0	2.8	13.7	73.1	4.4	2.8	3.0	9.2	25.0
	S15850	10.9	1911	4.5	2.3	1.0	15.8	63.5	4.0	1.0	3.3	8.0	43.8
	S35932	13.1	8780	4.4	2.0	1.3	21.0	190.5	4.1	1.1	3.0	17.4	115.1
	S38417	11.9	3822	6.6	1.9	1.1	17.8	260.8	4.1	1.1	2.9	11.0	158.1
	S38584	13.0	8192	4.8	2.4	1.2	18.7	1530.2	4.2	1.3	3.4	11.4	682.5
	Avg	12.3	6165	4.83	2.22	1.4	18.07	380.7	4.13	1.38	3.22	11.38	172.6

Ci: # of initial fault candidates MPi: # of initial multiplet candidates T: # of tester accesses CPU: CPU time (sec)

Cx: # of final fault candidates after applying method x MPx: # of multiplets after applying method x # of pttn: # of MC-AD patterns applied

TABLE II
MULTIPLE TRANSITION FAULTS DIAGNOSIS

	Circuit			SO-SLAT (a)		MC-AD (b)			(a) + (b)				
		Ci	MPI	Ca	T	MPb	# of pttn	CPU	Cab	MPab	T	# of pttn	CPU
2 Faults	S5378	7.0	128	2.6	1.6	1.6	12.6	9.8	2.1	1.3	2.6	9.7	2.9
	S13207	6.0	64	2.0	1.9	2.5	8.2	35.5	2.0	1.0	2.9	7.0	4.3
	S15850	6.3	79	2.1	1.6	4.4	10.1	104.6	2.0	1.0	2.6	8.1	7.9
	S35932	5.9	60	2.2	2.0	1.0	7.8	139.3	2.0	1.0	3.0	7.8	62.0
	S38417	5.9	60	2.3	1.9	1.1	10.3	141.9	2.0	1.0	2.9	7.7	49.2
	S38584	5.9	60	2.1	1.5	1.0	11.3	99.4	2.0	1.0	2.5	7.8	19.3
	Avg	6.17	75.2	2.22	1.75	1.93	10.05	88.42	2.02	1.05	2.75	8.02	24.26
3 Faults	S5378	9.3	630	3.3	1.4	1.4	16.6	19.2	3.0	1.0	2.4	12.7	3.1
	S13207	8.1	274	3.1	1.7	1.6	14.2	458.4	3.1	1.2	2.7	11.3	8.2
	S15850	10.2	1176	3.3	2.2	1.1	19.8	181.6	3.0	1.0	3.2	17.4	22.1
	S35932	9.4	676	3.4	2.0	1.1	17.8	396.1	3.1	1.1	3.0	17.8	136.8
	S38417	9.6	776	3.8	2.0	1.2	19.9	248.7	3.0	1.0	3.0	16.3	80.33
	S38584	9.0	512	3.0	2.0	1.3	19.6	308.3	3.0	1.0	3.0	12.0	42.3
	Avg	9.27	674.0	3.32	1.88	1.28	17.98	268.72	3.03	1.05	2.88	14.58	48.81
4 Faults	S5378	12.6	6208	4.2	1.8	1.7	30.4	46.2	4.2	1.6	2.8	23.1	6.8
	S13207	9.6	776	4.0	1.6	6.3	21.1	910.6	4.0	1.0	2.6	14.5	8.6
	S15850	12.8	7131	4.0	2.3	1.0	28.9	124.3	4.0	1.0	3.3	25.5	28.4
	S35932	13.1	8780	4.6	2.1	7.5	33.2	1647.0	4.2	1.3	3.1	26.7	132.0
	S38417	12.2	4705	4.5	1.9	1.2	28.0	600.7	4.2	1.2	2.9	28.0	600.6
	S38584	13.7	13308	4.1	3.0	1.4	30.3	1946.8	4.0	1.0	4.0	25.9	145.8
	Avg	12.33	6818	4.23	2.12	3.18	28.65	879.27	4.10	1.18	3.12	23.95	153.70

Ci: # of initial fault candidates MPi: # of initial multiplet candidates T: # of tester accesses CPU: CPU time (sec)

Cx: # of final fault candidates after applying method x MPx: # of multiplets after applying method x # of pttn: # of MC-AD patterns applied equivalence and/or dominance relationships among the original set of injected faults. For those cases, we record the number just as we did with the injected faults. Note that the listed number is the average number. In a few cases, a poor performance can be attributed to a few faulty instances which are difficult to diagnose using the SO-SLAT-based approach. The worst case among our experiments improves the original diagnosis resolution from 12 to 8 fault candidates. The count of tester accesses is smaller than three in all cases. Similarly, for all cases diagnosed by the MC–AD approach, the proposed technique is able to either accurately identify the true multiplet or report a very small set of multiplets which includes the true one.

Applying the MC–AD-based approach following the SO-SLAT-based technique can leverage the smaller list of fault candidates derived by the SO-SLAT-based technique. Thus, it can improve the performance of SAT-based technique—resulting in fewer MC–AD patterns, less CPU time, and most importantly, higher resolution for identifying true multiplets as indicated in the (a) + (b) column.

VI. Conclusion

We have proposed two new diagnosis methods which offer better diagnosis resolution and can be used to enhance any existing state-of-the-art diagnosis processes. Through novel circuit modeling techniques, our method first uses a standard ATPG tool to efficiently identify nontrivial fault equivalence and dominance relationships among the faults in the initial candidate fault list obtained by the existing methods. Then, special diagnostic tests are adaptively generated and applied in incrementally filtering out false candidates and in identifying true fault locations. The first method generates deterministic SLAT patterns to isolate fault candidates by observing responses at a selected output for each fault candidate. The second method relies on a special type of test, namely, limited-cycle sequential MC–AD test for accurate multifault diagnosis. The approach analyzes the device under diagnosis responses to the MC–AD tests using a sequential SAT solver to prune false fault multiplets and identify the true one. Both approaches work for both static (such as stuck-at) and dynamic (such as transition) faults. The experimental results indicate that the combined method offers very high diagnosis resolution for multiple faults.

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 Des. Test. Comput.*, vol. 14, no. 3, pp. 98–103, Jul.–Sep. 1997.

[3] L. M. Huisman, “Diagnosing arbitrary defects in logic designs using single location at a time (SLAT),” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 1, pp. 91–101, Jan. 2004.

[4] A. Waicukauski and E. Lindbloom, “Failure diagnosis of structured VLSI,” *IEEE Des. Test. Comput.*, vol. 6, no. 4, pp. 49–60, Aug. 1989.

[5] D. B. Lavo, I. Hartanto, and T. Larrabee, “Multiplets, model, and the search for meaning: Improving per-test fault diagnosis,” in *Proc. Int. Test Conf.*, 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,” in *Proc. Int. Test Conf.*, 2003, pp. 329–338.

[7] S. Venkataraman and S. B. Drummonds, “Poirot: Applications of a logic fault diagnosis tool,” *IEEE Des. Test. Comput.*, vol. 18, no. 1, pp. 19–30, Jan./Feb. 2001.

[8] M. Abramovici, P. R. Memon, and D. T. Miller, “Critical path tracing—An alternative to fault simulation,” in *Proc. Des. Autom. Conf.*, 1983, pp. 214–220.

[9] P. Girard, C. Landrault, and S. Pravossoudovitch, “Delay fault diagnosis by critical-path tracing,” *IEEE Des. Test. Comput.*, vol. 9, no. 4, pp. 27–32, Dec. 1992.

[10] A. Smith, A. Veneris, and A. Viglas, “Design diagnosis using Boolean satisfiability,” in *Proc. Asia and South Pacific Des. Autom. Conf.*, 2004, pp. 218–223.

[11] M. Fahim Ali, A. Veneris, A. Smith, S. Safarpour, R. Drechsler, and M. Abadir, “Debugging sequential circuits using Boolean satisfiability,” in *Proc. Int. Conf. Comput.-Aided Des.*, 2004, pp. 204–209.

[12] F. Lu, G. Parthasarathy, M. K. Iyer, L.-C. Wang, K.-T. Cheng, and K. C. Chen, “An efficient sequential SAT solver with improved search strategies,” in *Proc. Des. Autom. and Test Eur.*, 2005, pp. 1102–1107.

[13] I. Park, A. AL-Yamani, and E. McCluskey, “Effective TARO pattern generation,” in *Proc. VLSI Test Symp.*, 2005, pp. 161–166.

[14] I. Hartanto, V. Boppana, J. H. Patel, and W. K. Fuchs, “Diagnostic test generation for sequential circuits,” in *Proc. VLSI Test Symp.*, 1997, pp. 196–202.

[15] A. Veneris, R. Chang, M. S. Abadir, and M. Amir, “Fault equivalence and diagnostic test generation using ATPG,” in *Proc. Int. Symp. Circuits and Syst.*, 2004, pp. 221–224.

[16] V. D. Agrawal, D. H. Baik, Y. C. Kim, and K. K. Saluja, “Exclusive test and it’s application to fault diagnosis,” in *Proc. Int. Conf. VLSI Des.*, 2003, pp. 143–148.

[17] I. Pomeranz and S. M. Reddy, “A diagnostic test generation procedure for synchronous sequential circuits based on test elimination,” in *Proc. Int. Test Conf.*, 1998, pp. 1074–1083.

[18] T. Bartenstein, “Fault distinguishing pattern generation,” in *Proc. Int. Test Conf.*, 2000, pp. 820–828.

[19] I. Pomeranz, S. Venkataraman, S. M. Reddy, and B. Seshadri, “Z-sets and Z-detections: Circuit characteristics that simplify fault diagnosis,” in *Proc. Des. Autom. and Test Eur.*, 2004, pp. 68–73.

[20] J. B. Liu and A. Veneris, “Incremental fault diagnosis,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 2, pp. 240–251, Feb. 2005.

[21] I. Pomeranz and S. M. Reddy, “On correction of multiple design errors,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 14, no. 2, pp. 255–264, Feb. 1995.

[22] Y.-C. Lin, F. Lu, and K.-T. Cheng, “Accurate diagnosis of multiple faults,” in *Proc. Int. Conf. Comput. Des.*, 2005, pp. 153–156.

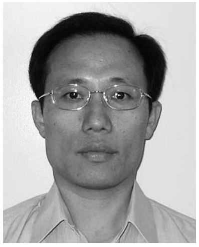
[23] Y.-C. Lin and K.-T. Cheng, “Multiple-fault diagnosis based on single-fault activation and single-output observation,” in *Proc. Des. Autom. and Test Eur.*, 2006, pp. 424–429.

[24] R. K. K. R. Sandireddy and V. D. Agrawal, “Diagnostic and detection fault collapsing for multiple output circuits,” in *Proc. Des. Autom. and Test Eur.*, 2005, pp. 1014–1019.

[25] D. Brand, “Verification of large synthesized designs,” in *Proc. Int. Conf. Comput.-Aided Des.*, 1993, pp. 534–537.

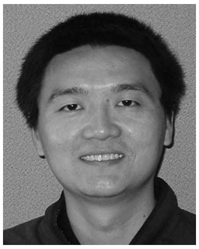
[26] I. Pomeranz and S. M. Reddy, “On diagnosis of pattern-dependent delay faults,” in *Proc. Des. Autom. Conf.*, 2000, pp. 59–62.

[27] Z. Wang, M. Marek-Sadowska, K.-H. Tsai, and J. Rajski, “Multiple fault diagnosis using n-detection tests,” in *Proc. Int. Conf. Comput. Des.*, 2003, pp. 198–201.



Yung-Chieh Lin received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1991, and the M.S. and Ph.D. degrees in electrical and computer engineering from University of California, Santa Barbara, in 1994 and 2006, respectively.

He is currently with the Hon-Hai Precision Industry Company, Ltd., Taiwan.



Feng Lu (M'02) received the B.S. degree in computer science from Civil Aviation Institute, Tenjing, China, in 1993, and the M.S. degree in computer science from Tsinghua University, Beijing, China, in 1996. He is currently working toward the Ph.D. degree at the University of California, Santa Barbara.

His research interests include SAT algorithm, formal verification, and testing.

Kwang-Ting Cheng (S'88–M'88–SM'98–F'00) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1983, and the Ph.D. degree in electrical engineering and computer science from University of California, Berkeley, in 1988.

He worked with the Bell Laboratories, Murray Hill, NJ, from 1988 to 1993 and joined the faculty at the University of California, Santa Barbara, in 1993, where he is currently a Professor and Chair of electrical and computer engineering. His current research interests include very large scale integration testing, design verification, and multimedia computing. He has published over 250 technical papers, coauthored three books, and holds ten U.S. patents in these areas. He has also been working closely with the U.S. industry for projects in these areas.

Dr. Cheng received the Best Paper Awards at the 1994 and 1999 Design Automation Conferences, 2001 Annual Best Paper Award in *Journal of Information Science and Engineering*, Best Paper Award in 2003 Conference of Design Automation and Test in Europe (DATE 2003), and the Best Paper award at 1987 AT&T Conference on Electronic Testing. He currently serves as the Associate Editor-in-Chief for IEEE Design and Test of Computers, Associate Editor for ACM Transactions on Design Automation of Electronic Systems, Editor for *Journal of Electronic Testing: Theory and Applications*, and Editor for Foundations and Trends in Electronic Design Automation. He has also served on the Editorial Boards of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN. He has been the General Chair and Program Chair of IEEE International Test Synthesis Workshop and Program Co-Chair of International Mixed-Signal Test Workshop and served on the technical program committees for a number of international conferences on design, design automation, and test.