Multiplets, Models, and the Search for Meaning: Improving Per-Test Fault Diagnosis

David B. Lavo
Agilent Technologies
Semiconductor Products Group
Santa Clara, CA

Ismed Hartanto
Agilent Technologies
Semiconductor Products Group
Santa Clara, CA

Tracy Larrabee
Computer Engineering
University of California
Santa Cruz, CA

Abstract

The advantage to "one test at a time" fault diagnosis is its ability to implicate the components of complicated defect behaviors. The disadvantage is the large size and opacity of the diagnostic answer. In this paper, we address the problems of per-test fault diagnosis by improving the candidate matching, introducing scoring and ranking techniques, and by developing a method to translate the results into common defect scenarios. Our experimental results on simulated and introduced defects indicate that not only are the results improved on complex behaviors, but by considering passing test results we improve a common case where per-test algorithms can perform significantly worse than traditional diagnosis algorithms. Finally, our method of candidate analysis provides a way to bridge the per-test approach with traditional model-based fault diagnosis.

1. Introduction

Fault diagnosis, especially in its initial stage, can be a daunting task. Not only does the failure analysis engineer not know what kind of defect he is dealing with, but there may in fact be multiple separate defects, any number of which may interfere with each other to modify expected fault behaviors. The defect behavior may be intermittent or difficult to reproduce. Also, the size of the circuit may make application of all but the simplest diagnosis algorithms impractical.

Given these facts, the *single fault assumption* – that there is one defect in the circuit under diagnosis that can be modeled by a single instance of a particular fault model – apparently does not apply for modern fault diagnosis. While it has simplified many diagnostic approaches, some of which have worked quite well despite real-world violations of the premise, the single fault assumption has led to problems with two common defect types: multiple faults, and complex faults. As defined here, *complex faults* are faults in which the fault behavior involves several circuit nodes, involves multiple erroneous logic values, is pattern-dependent, or is otherwise intermittent or unpredictable.

Traditionally, the single fault assumption has led to the expectation of a certain internal consistency, or some dependence between the test results, with regard to defective circuit behavior. In cause-effect diagnosis, a fault model is selected beforehand, and the observed faulty behavior is compared, as a single collection of failing patterns and outputs, to fault signatures obtained by simulation. In effect-cause diagnosis, many algorithms look for test results that prove that certain nodes in the circuit are able to toggle, and are therefore fault-free throughout the rest of the test set. In either case, the assumption is that individual test results are not independent, but are rather wholly determined by the presence of the single unknown defect.

From the beginning, however, a few diagnosis techniques eschewed the single fault assumption, especially those that directly addressed multiple faults. These approaches, either implicitly or explicitly, forsake inter-test dependence and instead consider each test independently. The advantage to such approaches is that pattern-dependent and intermittent faults can still be identified, as can the component faults of complex defects. The drawback is that a conclusion drawn about the defect from one test cannot be applied to any other test, and the net result is (in effect) a diagnosis for each test pattern. This can lead to large candidate sets that are difficult to understand and use, especially as guidance for physical failure analysis. Also, since these algorithms no longer implicate a single instance of a fault model, there is now the problem of constructing a plausible defect scenario to explain the observed behavior.

This paper will attempt to address these drawbacks by improving both the process and the product of per-test fault diagnosis. First, the process will be improved by including more information to score candidates, and paring down the candidate list to a manageable number. Second, the product will be improved by suggesting a way of interpreting the candidates to infer the most likely defect type. The result is a general-purpose approach to identifying likely sources of defective behavior in a circuit despite the complexity or unpredictability of the actual defects.

2. SLAT, STAT, and All That

While increasing in recent popularity, the idea of conducting fault diagnosis one test pattern at a time is a venerable one. Waicukauski and Lindbloom (WL) [WaiLin89, EicLin91], and, more recently, the POIROT [VenDru00] and SLAT [BarHea01] diagnostic systems all suggest or rely on per-test fault diagnosis to address multiple or complex faults. We can, without too much license, state the primary axiom of the one-test-at-a-time approach as follows:

> *For any single test, an exact match between the observed failures (at circuit outputs or flip-flops) with those predicted by a simulated fault is strong evidence that the fault is present in the circuit, if only during that test.*

The underlying concept is uncontroversial, as it underpins both traditional fault diagnosis as well as scientific modeling and prediction: A match between model and observation supports the assumptions of the model or implicates the modeled cause. The difference here is that the traditional comparison of model to observed behavior is decomposed into comparisons on individual test vectors, with a stricter threshold of exact matching to produce stronger implications.

The statement that "the fault is present" should not be taken too broadly. It does not mean that the fault (or modeled defect) is physically present, or that any conclusions can be drawn about the defect in any other circumstance other than the specific failing test vector. Applied most commonly to stuck-at faults, all that can be inferred from a match is that a particular node has the wrong logic value for a particular test. However, that node is not implicated as the source of any other failures, nor is the node actually "stuck-at" any value at all, since there is no evidence that it doesn't toggle during other test vectors.

Note also that the axiom cannot claim that an exact match constitutes *proof* that a particular fault is present. A per-test diagnosis approach can be fooled by aliasing, when the fault effects from multiple or complex faults mimic the response from a simple stuck-at fault. This can happen, for instance, if the propagation from a fault site is altered by the presence of other simultaneous faults, or due to defect-induced behaviors such as complex bridging faults. The probability of such aliasing is, in practice, impossible to determine, given the variety of ways in which it could occur. Per-test diagnosis approaches rely on the assumption that this probability is small, and on the hope that, should aliasing incorrectly implicate the wrong fault, that this fault is not wholly unrelated to the actual defect and is therefore not completely misleading.

A secondary axiom, implicit in the WL paper but stated in different terms in the SLAT paper, is the following:

> *There will be some tests during which the defect(s) to be diagnosed will behave as a single, simple fault, which will, by application of the primary axiom, implicate something about the defect(s).*

What this axiom states is that, for any defective chip, there will be some tests for which the failing outputs will exactly match the predicted failing outputs of one or more simple (generally stuck-at) faults. This assertion relies on the observation that many complex defects will, for some applied tests, behave like stuck-at faults that are in some way related to the actual defect. For example, a bridging fault will occasionally behave, on some tests, just like a stuck-at fault on one of the bridged nodes.

The way that a per-test fault diagnosis algorithm proceeds is to find these *simple failing tests* (referred to in the SLAT paper as *SLAT patterns*), and identify and collect the faults that match them. The candidate faults are arranged into sets of faults that cover all the matched tests. The SLAT authors call these collections of faults *multiplets*. As a simple example, consider the following three tests, with the associated matching fault candidates:

Test Number	Exactly-Matching Faults
1	A
2	B
3	C, D, E

Figure 1. Simple per-test diagnosis example.

In this example, fault A is a match for test #1, which means that the predicted failing outputs for fault A on test #1 match exactly with the observed failing outputs for that test. Similarly, fault B matches on test #2, while for test #3 three faults match: C, D, and E. The SLAT algorithm will build the following multiplets as a diagnosis: (A, B, C), (A, B, D), and (A, B, E). Each multiplet explains, or covers, all of the simple failing test patterns. SLAT uses a simple recursive covering algorithm to traverse all covering sets smaller than a pre-set maximum size, and then only reports minimal-sized coverings (multiplets) in its final diagnosis.

For comparison, the WL algorithm will report one set of faults – (A, B, C, D, E) – in its diagnosis on the above example, with a note that fault C, D, and E are equivalent explanations for test #3. The POIROT algorithm will produce the same results, with a score based on how many tests are explained by each fault (in this case, all faults would get the same score).

This paper proposes a new per-test algorithm, similar in style to the SLAT diagnosis technique, but able to use more information and produce a better, more quantified, diagnostic result. The SLAT technique is focused on determining fault *locations*, hence the name: "Single Location At a Time". The new approach will instead focus on the faults themselves, but will, like SLAT, diagnose test patterns one at a time. Borrowing the nomenclature, however, we will refer to the process of per-test diagnosis as "STAT" – "Single Test At a Time".[1] The new algorithm is called "iSTAT", for "improved STAT". Like SLAT, the iSTAT algorithm uses stuck-at faults to build multiplets, but differs from SLAT in two important ways. First, it uses a scoring mechanism to rank (or order) multiplets to narrow the resulting candidate set. Second, it can use the results from both passing and complex failing tests to improve the scoring of candidate fault sets.

3. Multiplet Scoring

The biggest problem with a STAT-based diagnosis is that, since each test is essentially an individual diagnosis, the number of candidates can become quite large. It can take many multiplets to explain the entire set of failing patterns, and each multiplet will be composed of multiple faults. This section presents iSTAT's approach to this problem, a method to score and rank the multiplets to indicate a preference between them.

The basic motivation of STAT-based approaches, as expressed in the first axiom, is that an exact match between failing and predicted outputs on a single test is strong evidence for the fault. While this much seems reasonable, it seems just as obvious that the evidence provided by a failing test is diluted if there are many fault candidates that match. For instance, in the simple example given above, the evidence for fault A is much stronger than that for any of faults C, D, or E, simply because fault A is the only candidate (according to the axiom) that can explain the failures of test #1. The evidence provided by test #3 is just as significant as the evidence from test #1, it is just shared among three possible explanations.

This division of evidence can also be illustrated by imagining failures on outputs with a lot of fan-in, or a defect in an area with many equivalent faults. While there will be a number of faults that match the failure exactly, test results will not provide much compelling evidence to point to any particular fault instance.

[1] We will hereafter refer to the class of diagnosis algorithms that includes WL, POIROT, SLAT, and the new iSTAT algorithm as "STAT", or "per-test", diagnosis algorithms.

The first way that iSTAT improves per-test diagnosis is to consider the weight of evidence pointing to individual faults, and to quantify and collect that evidence into multiplet scores. The mechanism that iSTAT uses to quantify diagnostic evidence is the Dempster-Shafer method of evidentiary reasoning.

3.1 "A Mathematical Theory of Evidence"

A means of quantitatively manipulating evidence was developed by Arthur Dempster in the 1960's, and refined by his student Glen Shafer in 1976 [Sha76]. The Dempster-Shafer method is a generalization of the familiar *Bayes Rule of Conditioning*, in which prior and conditional probabilities are multiplied to produce a final posterior probability for purposes of estimation and prediction.

The Dempster-Shafer method was developed to address certain difficulties with Bayes Rule when it is applied to the conditions of *epistemic* probability, in which probability assignments are based on belief or personal judgement, rather than its usual application to *aleatory* probability, where probabilities are determined by chance.

The conditions of epistemic probability are familiar to most people: a person will assign a *degree of belief* to a proposition relative to the strength of evidence presented in its favor. There is an explicit and unavoidable role of judgement in such a process. It is possible or likely that no prior information or belief about the problem exists before the evidence is considered. Finally, there is a possibility that a judgement cannot be made, or belief will be reserved, in the case of ignorance or lack of evidence.

The Dempster-Shafer method is designed with these considerations in mind. It is best illustrated geometrically: the basic element of the Dempster-Shafer method is a *belief function*, which can be thought of as a division of a unit line segment into various probability assignments. Each assignment represents the belief accorded to an individual element based on some evidence; in addition, an explicit degree of doubt or ignorance about the evidence can be assigned. The total of all probability assignments equals one:

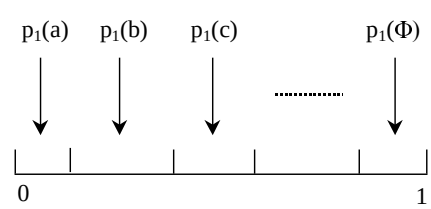

Figure 2. An example belief function.

The assignment p(Φ) represents the degree of doubt regarding the evidence or the assignments. The introduction of a second piece of evidence results in the creation of a second belief function, with a new assignment of probabilities to a possibly-different set of elements:

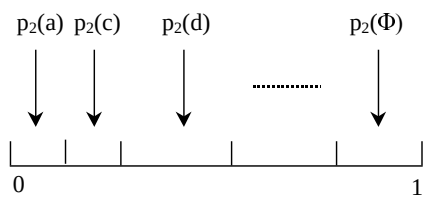

Figure 3. Another belief function.

Dempster's rule of combination performs an orthogonal combination of these two belief functions. Geometrically, the two line segments are combined to produce a square, which represents the new total probability mass of the combination:

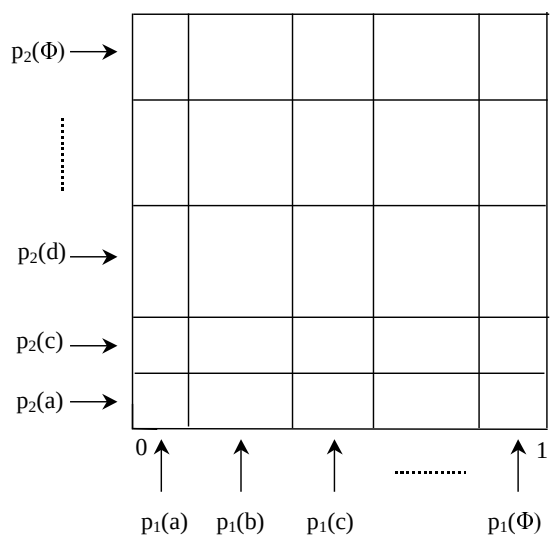

Figure 4. The combination of two belief functions.

The total combined probability of an element is the sum of all non-contradictory assignments to that element; therefore, $p_3(a) = p_2(a)p_1(a) + p_2(a)p_1(\Phi) + p_2(\Phi)p_1(a)$. The final probability assigned to each element is re-normalized by dividing by total probability mass assigned to contradictory combinations:

$$p(C) = \frac{\sum_{\substack{i,j \\ A_i \cap B_j = C}} p_1(A_i)\, p_2(B_j)}{1 - \sum_{\substack{i,j \\ A_i \cap B_j = 0}} p(A_i)\, p(B_j)}$$

3.2 From Evidence to Scored Multiplets

Applying the Dempster-Shafer method to per-test diagnosis scoring is a relatively straightforward process. Each failing test that is matched exactly by one or more fault candidates results in a belief function; each candidate is assigned an equal portion of the belief assigned by the test result. Also, some probability mass is reserved to account for the possibility of aliasing, discussed earlier. Since an exact match on a test result is the strongest evidence implicating fault candidates, this reserved belief is small.

A short example will illustrate the scoring process. Figure 5 presents some test-matching results.

Test Number	Matching Faults
1	A
2	A, D
3	B
4	C, D

Figure 5. Example test results with matching faults.

The result of test #1 results in a belief function in which all evidence supports fault A. The amount of ignorance regarding this test result (whether fault A is really the cause of the behavior) is arbitrary; the iSTAT algorithm uses the value p(Φ) = 0.01, so the support awarded to fault A for test #1 is $p_1(A) = 0.99$.

For test #2, the evidence supports both faults A and D, so the total belief is split between these faults: p2(A) = p2(D) = 0.495. A geometric representation of the combination of these belief functions is shown below. The proportion of area allotted to p(Φ) is exaggerated in the figure for readability.

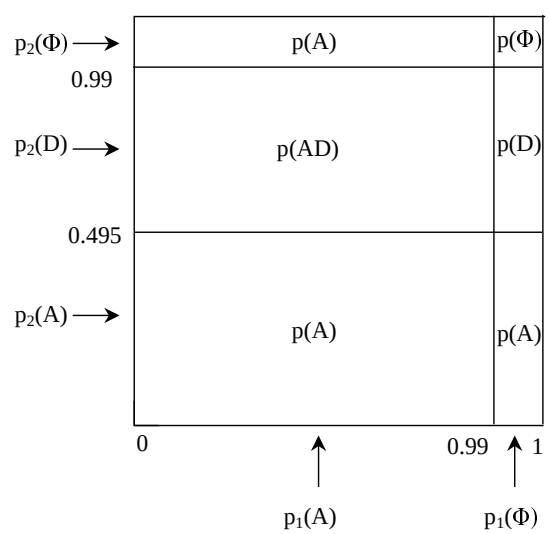

Figure 6. Combination of evidence from the first two tests.

The calculation of combined probabilities is as follows:

$$\begin{aligned} p(A) &= p_2(A)p_1(A) + p_2(A)p_1(\Phi) + p_2(\Phi)p_1(A) \\ &= 0.5049 \\ p(D) &= p_2(D)p_1(D) + p_2(D)p_1(\Phi) + p_2(\Phi)p_1(D) \\ &= 0.00495 \\ p(AD) &= p_2(D)p_1(A) \\ &= 0.49005 \\ p(\Phi) &= p_2(\Phi)p_1(\Phi) \\ &= 0.0001 \end{aligned}$$

As you can see, slightly more belief is distributed to p(A) than p(AD), and a small residual belief is accorded to p(D). After the application of the third test, the results of which match with fault B, the revised probabilities are:

$$\begin{aligned} p(AB) &= (0.99)(0.5049) = 0.499851 \\ p(A) &= (0.01)(0.5049) = 0.005049 \\ p(ABD) &= (0.99)(0.49005) = 0.4851495 \\ p(AD) &= (0.01)(0.49005) = 0.0049005 \\ p(BD) &= (0.99)(0.00495) = 0.0049005 \\ p(D) &= (0.01)(0.00495) = 0.0000495 \\ p(B) &= (0.99)(0.0001) = 0.000099 \\ p(\Phi) &= (0.0001)(0.01) = 0.000001 \end{aligned}$$

The highest probability is assigned to (AB), which makes intuitive sense given the test results so far. Finally, test #4 matches faults C and D, and the top two combinations are (after rounding):

$$\begin{aligned} p(ABD) &= 0.492 \\ p(ABC) &= 0.247 \end{aligned}$$

As a comparison, the SLAT algorithm reports the same final multiplet candidates, (ABC) and (ABD). Intuitively, we prefer multiplet (ABD) to multiplet (ABC), based on the notion that there exists more evidential support for fault D than fault C. The calculations above support this intuition, showing that the Dempster-Shafer method assigns almost twice the support to the multiplet containing fault D.

The application of this scoring alone makes the iSTAT algorithm preferable to other per-test diagnosis algorithms; all such algorithms produce essentially the same candidate faults, but by assigning a probability score to each candidate set it provides much more guidance in selecting candidates out of what can be large diagnoses. But, there is more information that per-test approaches usually fail to consider and that can be applied to produce even better final diagnoses.

3.3 Matching Passing Tests

Most STAT-based algorithms completely ignore passing tests, probably because passing tests don't fit well with the basic axioms expressed earlier: it is difficult to infer a failure when no failure has occurred. But, STAT algorithms will suffer a loss of resolution, especially when compared with traditional non-STAT algorithms, when dealing with some defects.

For example, consider an observed behavior that mimics a classic stuck-at fault. In such a case (which is surprisingly common, for power or ground shorts, signal-to-signal shorts, and for opens), a traditional diagnosis algorithm that matches both failing and passing tests will produce either a single fault candidate, or a list of faults that are behaviorally equivalent under the applied test set. But, a per-test algorithm that ignores passing tests will produce the same equivalence list, *plus* all fault candidates whose fault signatures are supersets of the observed behavior. A simple example is a non-controlling stuck-at fault on a gate input. Most STAT algorithms will implicate a stuck-at fault on the output of the gate as strongly as the (preferred) input fault, simply because the output fault explains all the failing test patterns.

There are many other examples of faults whose faulty behaviors are supersets of those of other harder-to-detect faults. In any case, it is especially disappointing for STAT algorithms not to be able to perform as well as traditional algorithms at distinguishing such simple and classic behaviors as single stuck-at faults.

To remedy this, the iSTAT algorithm must deal with passing tests. The process of matching passing patterns is very similar to matching simple failing patterns: candidates that predict a passing test will share in belief assigned based on that test. These belief values are combined according to Dempster's rule of combination, as with the failing tests.

An important difference in dealing with passing tests is that only multiplets (candidate fault sets that explain all simple failing tests) are considered, not individual faults. The reason is that passing tests don't, according to the per-test axioms stated earlier, provide any evidence for individual faults. Rather, they only imply the lack of fault sensitization or unmodeled fault behavior.

It is difficult to infer much about the conditional probability of a set of faults given a passing test result. Obviously, if all of the component faults are predicted to pass on a particular passing test, that result provides some evidence in support of that multiplet. If, however, some of the component faults of a multiplet predict failures for a passing test, it is possible that none of these faults were activated, or if any such fault was sensitized then none of its failures propagated to

observable outputs. Either condition could occur due to interactions between multiple faults. The likelihood of interference with sensitization or propagation is difficult to calculate, especially for larger multiplets.[2]

It seems reasonable to assume that the likelihood of no sensitization and propagation is proportional to the number of components in a multiplet that predict a pass for any test. For each passing test, then, a multiplet will be assigned an initial belief value, from a maximum of 1.0 (all faults predict a pass) to a minimum of 0.0 (all faults predict some failure). This initial score is divided by the total score over all multiplets, so that the total belief accorded over all multiplets is equal to 1.0.

Since the evidence provided by any passing test is relatively weak, any inference made from one is not strong, and so the degree of doubt or ignorance assigned to a passing test should be high. The iSTAT algorithm uses a value of $p(\Phi) = 0.5$. The belief invested in each multiplet is therefore adjusted one last time, by multiplying by 0.5, to re-normalize the total belief to 1.0.

3.4 Matching Complex Failures

The SLAT algorithm ignores any failing test pattern that doesn't match exactly with one or more candidate faults. If we refer to the easily-matched patterns as *simple* failing tests, then the question becomes what to do with the *complex* failing tests, or tests that don't match exactly with any stuck-at fault.

The POIROT algorithm uses a greedy covering algorithm on such failing output sets, using individual faults to explain subsets of the failing outputs. The iSTAT algorithm takes a different approach. First, as with passing patterns only multiplets are considered when trying to match the failing outputs, and not individual stuck-at faults. Second, instead of trying to match subsets of the failing outputs, we attempt a much simpler and more conservative matching process, as explained below.

Determining which outputs are predicted to fail by a multiplet is not easy, because we have no way of knowing how the fault effects of the individual fault components will interact for any test vector. The fault effects of one activated fault could prevent the propagation to some outputs of a second activated fault. Or, one fault could prevent the sensitization of another fault completely, or cause another fault to become sensitized that normally would not.

[2] If per-test I_{DDQ} pass-fail information were available, it would indicate whether a logical pass actually indicates the absence of a defect or not. On a test that passes scan tests but fails I_{DDQ}, then, a multiplet that predicts failure would not be subject to a scoring penalty.

It is not practical to investigate all of the various permutations of these fault interactions, especially if electrical effects such as drive fights or variable logic thresholds are concerned. So, iSTAT ignores these complications and instead chooses a conservative path of matching by combining all the failing outputs and then ignoring misprediction (or overprediction) of the observed failing outputs. For example, if the following faults are contained in the multiplet (A, B, C):

Fault	Predicted Failing Outputs
A	1, 5, 8
B	2, 5
C	2, 10

Figure 7. Example of constructing a set of possibly-failing outputs for a multiplet.

The total list of failing outputs for this multiplet is (1, 2, 5, 8, 10). A successful match, then, is a match with any subset of these outputs, such as (1), (2, 5, 10), (1, 10), and so on. A match with any subset is considered an "explanation" of the failures, but any non-subset, such as (1, 2, 6) is not. It is possible that fault interaction could cause such an unexpected propagation and therefore a mismatch, but iSTAT will tolerate this (assumed small) probability of error if it generally aids in ranking candidate multiplets.

This matching on complex failing tests results in either a success or a failure for each multiplet on each test. The degree of belief assigned to each matching multiplet is 1.0 divided by the number of matching multiplets. As with passing tests, the evidence provided by a complex failing test is not perfect, and so iSTAT assigned a degree of doubt $p(\Phi) = 0.1$ and the belief assigned to individual matching multiplets is normalized by multiplying by 0.9.

3.5 Size is an Issue

In addition to matching all the simple failing tests, the SLAT paper implicitly introduces another criterion for judging multiplets, namely multiplet size: only minimal-sized multiplets are considered in the final diagnosis. Consider an example:

Test Number	Matching Faults
1	A
2	A
3	B
4	B, C, D

Figure 8. Multiplets (A,B), (A,B,C), and (A,B,D) explain all test results, but (A,B) is smallest and so preferred.

A minimally-sized multiplet that covers all of the failing vectors is (A, B). But, it is also possible to cover the failing vectors with the multiplet (A, B, C) by

choosing fault C to explain the failures on test #4. The same is true for multiplet (A, B, D). Intuitively, (A, B) seems to be the best, and more likely, candidate due to the evidence for fault B from test #3. There is also the principle of Occam's Razor [Tor38], which states "Causes shall not be multiplied beyond necessity", or more commonly, "The simplest answer is best"[3]. The application of Occam's Razor therefore argues for choosing multiplets of minimal size.

Test Number	Matching Faults
1	A, B
2	A, C
3	A, C
4	A, B

Figure 9. The choice of best multiplet is difficult if (A) predicts additional failures but (B, C) does not.

But consider a slightly less simple scenario, demonstrated in Figure 9. While iSTAT will build and score the multiplets (A) and (B,C), SLAT will only consider the multiplet (A). At first glance it would appear that multiplet (A) is a simpler and therefore better choice than (B, C). But suppose that fault A is also predicted to fail other tests that don't fail on the tester, while faults B and C are only predicted to fail on tests #1 through #4. We would then be faced with the choice of explaining the behavior with either an intermittent stuck-at fault (A), or a well-behaved pair of stuck-at faults (B, C). In such a case, Occam's Razor may not be the best tool to slice out the best or simplest answer.

For the example above, the iSTAT algorithm will assign the following probabilities to the two multiplets after processing the simple failing tests:

$$p(A) = 0.5002$$
$$p(BC) = 0.4998$$

If, however, test #5 is a passing test and faults B and C are both predicted to pass while fault A is predicted to fail, the multiplet probabilities are adjusted to the following:

$$p(A) = 0.3335$$
$$p(BC) = 0.6665$$

The actual values calculated will depend upon the value of $p(\Phi)$ assigned for passing tests, which in turn is determined by the judgement of the algorithm designer or user.

The iSTAT algorithm follows the SLAT convention of rejecting multiplets with redundant or superfluous faults, such as (A, B, C) in the example of Figure 8. But, by allowing such non-minimal multiplets as (B, C) in the second example, the iSTAT algorithm can consider a wider range of defect scenarios than can SLAT and many other per-test algorithms.

4. Experimental Results – Multiplet Ranking

This section presents results on some simulated defects in an industrial circuit. These defects were created by modifying the circuit netlist and simulating the test vectors to obtain faulty behaviors. Only logical fault simulation was done; in none of the cases was any electrical-level (or SPICE-level) simulation performed. The idea was to create defects of varying complexity, and of the types that per-test diagnosis algorithms usually target: multiple and intermittent stuck-at faults, wired-logic bridging faults, and faults clustered on nets and gates.

The iSTAT algorithm was performed on each simulated defect, including all of the matching and scoring methods described earlier. For each trial, a diagnosis consists of a set of multiplets. For each diagnosis, Table 1 below reports the type of defect we simulated and the size of the multiplets, where the size indicates the number of component faults in each multiplet. All SLAT multiplets contain the same number of faults (by construction); for these experiments, so did all top-ranked iSTAT multiplets.

The next column reports the number of SLAT multiplets, built according to the SLAT algorithm. There is one difference, however, between the SLAT multiplets described here and those described in the SLAT paper: These multiplets contain stuck-at faults (describing a circuit node and fault polarity), while original SLAT multiplets consist of only faulty circuit nodes (faults of opposite polarity on the same node are collapsed into one "location").

For each diagnosis, we then report the number of top-ranked iSTAT multiplets. This value gives the number of multiplets that all receive the same top score. A higher number indicates lower resolution, as the algorithm expresses no preference among these candidates. The comparison of this number with the number of SLAT multiplets indicates the improvement in resolution over the SLAT algorithm. The next column reports whether the diagnosis was a success or not, defined as the correct multiplet receiving the highest score.

[3] Or, less commonly, "Nunquam ponenda est pluralitas sine necesitate".

Defect No.	Simulated Defect	Faults in SLAT and Top-Ranked iSTAT Multiplets	SLAT Multiplets	Top-Ranked iSTAT Multiplets	Success?
1	Single stuck-at fault	1	7	4	Y
2	2 independent stuck-at faults	2	21	8	Y
3	2 independent stuck-at faults	2	1	1	Y
4	2 interfering stuck-at faults	2	9	4	Y
5	3 interfering stuck-at faults	3	2	1	Y
6	4 stuck-at faults, 3 interfering	4	2	1	Y
7	Two-line wired-OR bridge	2	2	1	Y
8	Two-line wired-AND bridge	2	2	1	Y
9	Two-line wired-AND bridge	2	1	1	Y
10	Two-line wired-XNOR bridge	3	13	7	Y
11	Two-line dominance bridge	1	3	1	P
12	Two-line dominance bridge	1	2	1	P
13	Net fault (3 branch stuck-at faults)	4	90	1	Y
14	Net fault (3 branch stuck-at faults)	3	4	1	Y
15	Gate replacement (OR to AND)	1	1	1	Y
16	Gate replacement (OR to NOR)	2	11	7	Y
17	Gate replacement (MUX to NAND)	2	3	2	Y
18	Gate output inversion	1	3	1	Y
19	Multiple logic errors on one gate	1	1	1	Y
20	Multiple logic errors on one gate	2	27	10	Y

Table 1. Results from scoring and ranking multiplets on some simulated defects.

For the two-line bridging defects, the result can be a partial ("P") success if only one node of the bridge is identified by the faults in a multiplet. A complete success ("Y") requires that both nodes be represented in the multiplet. Therefore, it is very unlikely that the diagnosis of a dominance bridge can be anything but a partial success, because no faults ever originate from the dominating node. Also, the implication of both nodes of a non-dominance bridging fault is highly dependent upon the test set. In order for both nodes to appear in a multiplet, the test set will have to propagate failures from both nodes and put opposite logic values on those nodes during the detecting tests.

On other defects, a successful diagnosis is expected to identify exactly the faults inserted. So, if two stuck-at faults were inserted, the correct multiplet should have two faults of the correct polarity. One exception was defect #13, where the diagnosis was judged a success even though three faults were inserted, since the fourth implicated fault was the stem of the net fault.

Overall, both the SLAT algorithm and the iSTAT algorithm produce a correct diagnosis on all trials. This is a remarkable success rate even for this small trial size, given the complexity of some of the defect behaviors. The number of times that SLAT produced a small diagnosis was surprising (4 multiplets or fewer on 13 of 20 trials), but in all cases iSTAT was able to improve this resolution, in some cases dramatically.

Another observation is that it was difficult to create gate faults that looked like anything other than (possibly-intermittent) stuck-at faults on the gate outputs. The inability to create truly "complex" gate fault behaviors most likely has to do with pattern-dependent fault detection, since output faults on gates can often swamp faults on the gate inputs unless enough tests with the right logic values are applied. The same is true for the bridging faults where, while the maximum multiplet size is 4 (both faults on both nodes detected), the algorithm mostly produced 1- to 2-fault multiplets.

5. Finding Meaning (and Models) in Multiplets

The main problem with the diagnoses returned by most per-test diagnosis schemes is one of interpretation. The end product of these algorithms can often be a large collection of sets of faults (multiplets), any of which can be used to explain the observed faulty behavior. If you show even a single multiplet, consisting of several faults or nodes, to a failure analysis engineer, the likely response is "But what does this *mean*?"

The SLAT method does make an attempt to make its diagnoses more understandable. The SLAT authors propose the construction of *splats*, which are sets of (apparently) equivalent nodes common to all multiplets. But, this only serves to identify fault equivalencies; each multiplet must still be investigated as a possible defect scenario. A more ambitious analysis method, called "SLAT Plus", was recently proposed [BarBha01]. This method analyzes logic-value relationships across all nodes of the circuit during observed failures, in an attempt to infer possible bridging defects. That work, however, is preliminary, and involves a different and more extensive type of analysis than is proposed here.

The WL algorithm also makes a simple attempt to interpret its results, by classifying a diagnosis into one of three categories. Class I is an exact match with a single stuck-at fault. Class II is a match on all failing tests with a single stuck-at fault but not all passing tests. Class III indicates multiple stuck-at faults that match only inexactly, a category that consists of a wide range of defects.

The POIROT algorithm attempts something of a compromise: it decomposes the matching operation into single tests, but also applies a set of pre-built signatures for certain fault models in addition to the stuck-at model. It is, in fact, much like the WL algorithm with the addition of bridging fault and net fault candidates. Since it explicitly targets these additional models, it doesn't require any interpretation of its results when one of the more specific candidates (bridge or net fault) is implicated. However, by relying on a a-priori set of candidates, it suffers from the candidate selection problem. For example, there are $\binom{n}{2}$ possible two-line bridging faults in a circuit, where n is the number of signal lines and can be quite large. There are also $2n$ individual stuck-at faults, and $O(n)$ open faults. The result is that, with only 3 candidate types considered, the POIROT algorithm can quickly become infeasible for large circuits.

The purpose of this section is to find a way to discover meaning in multiplets. The idea is to analyze each multiplet in a diagnosis to determine whether the component faults are in some way related to one another, or if they appear to be simply a collection of random faults. In the first case, an algorithm should then be able to infer a defect mechanism; in the second case, either the meaning escapes (due to unmodeled behavior) or perhaps the circuit behavior really is the result of a collection of unrelated defects.

But how can candidate faults be related to each other, and a meaning extracted from the observed behavior? The traditional answer for explaining defective behavior has been the use of fault models. The stuck-at fault model, various bridging fault models, and the transition fault model are all examples of using abstractions to simplify what can be complex defect behaviors. These fault models have the advantage of being relatively easy to understand and (with some translation) identify as part of failure analysis.

It seems intuitive, then, to interpret multiplets by correlating them with common fault models, calculating for every multiplet a correlation score for each fault model. A high correlation score implicates a likely defect scenario for that multiplet. A low correlation score for every candidate multiplet in a diagnosis indicates either that the defect is not well represented by any of the fault models, or that the defect consists of multiple fault instances.

5.1 Plausibility Metrics

To judge this correlation, the most natural scoring, mathematically speaking, is the *plausibility* of a match between a multiplet and a fault model, or the upper probability limit that a multiplet represents an instance of a particular fault model. For each multiplet, the proposed iSTAT analysis algorithm computes a plausibility score for each fault model, with a maximum score of 1.0 (complete agreement of faults to defect assumptions) and a minimum score of 0.0 (no agreement). A description of each fault model considered and the details of the plausibility calculations follow.

A. Single or intermittent stuck-at fault

This case is trivial: if the multiplet consists of a single fault candidate, it will be classified as a stuck-at or intermittent stuck-at fault on a single node. While this is a simple classification, many defect types mimic intermittent stuck-at faults. Depending upon the test set, bridging faults, gate faults, open faults and transition faults could all look like stuck-at faults. In the SLAT paper, the authors found that 37% of the defects they diagnosed looked like stuck-at faults, which is not inconsistent with our industrial experience of diagnosing actual failures. So, this defect class is likely to be a catch-all for many defects that aren't activated multiple times by the test set.

> **Plausibility**: 1.0 if multiplet is size 1; 0.0 otherwise.

B. Node/transition fault

If a multiplet consists of two fault candidates of opposite polarity on the same node, it is classified as a *node fault*. The most likely defects for this scenario are a dominance bridging fault, a transition fault, or some open faults.

Plausibility: 1.0 if multiplet is size 2, and faults involve the same node; 0.0 otherwise.

C. Net fault

If examination of the netlist determines that most or all of the component faults of a multiplet are the branches or stem of a common net, then it can be identified as a *net fault*. This type of fault was proposed by the authors of the POIROT system to cover open defects that affect nets with fanout.

Plausibility: 1.0 if multiplet is size 2 or greater, and all faults are on the same net (including fanout); if size 3 or greater, percentage of faults on the same net; 0.0 if multiplet is size 1.

D. Gate fault

If we find by examining either the faultlist or the circuit netlist that most or all of the faults in a multiplet involve a common gate or standard cell, then it will be classified as a *gate fault*. Some possible defects that could look like intermittent faults on a gate's outputs and inputs are transistor stuck-on or stuck-off, internal shorts, clocking problems, or some other logic error.

Note that since gate faults are a superset of node faults, any multiplet that gets a node fault score of 1.0 will also get a gate fault score of 1.0. While this classification is slightly redundant, it does reflect the fact that any defect on a node can also reasonably be attributed to its connected gates.

Plausibility: 1.0 if multiplet is size 2 or greater, and all faults are on ports of the same gate; if size 3 or greater, percentage of faults on the same gate; 0.0 if multiplet is size 1.

E. Two-line bridging fault

The identification of a two-line bridging fault relies on a multiplet containing faults on two nodes. Also, due to the nature of two-line shorts, tests that detect faults having opposite polarity should fail, and tests that detect faults of the same polarity should pass.

Plausibility: if multiplet is size 2, 3, or 4, and all faults are on (exactly) two nodes, then combine a) percentage of common tests for faults of opposite polarity that fail, with b) percentage of common tests for faults of same polarity that pass[4]; 0.0 otherwise.

[4] In previous publications, we have referred to these metrics as "required vector" and "restricted vector" scores, respectively.

F. Path/path-delay fault

If netlist examination find that the component faults of a netlist can be found on a single path by tracing back from failing outputs, then the defect is classified as a *path fault*. The as-yet unproven assumption is that path-delay faults can be identified in this manner.

Plausibility: 1.0 if multiplet is size 2 or greater and all faults exist on a path from an output to an input; if size 3 or greater, percentage of faults on the same path; 0.0 if all faults are on the same node, gate or net, or if multiplet is size 1.

These plausibility calculations were designed so that the information they require could be determined during the normal iSTAT algorithm operations of limited path tracing and fault simulation. For the bridging fault model, these calculations are a subset of those that a normal bridging fault diagnosis algorithm would perform; the same is true for node and net faults. At this stage, however, no specific fault simulation is done, other than normal STAT-based stuck-at simulation. These calculations, then, are a sort of "first-order" model-based diagnosis on the multiplet candidates, and the plausibility numbers express how reasonable it is to pursue a more intensive diagnosis for any fault model.

5.2 Proximity Metrics

The plausibility calculations for the models that involve electrical shorts could be significantly improved if information about physical proximity of the faults is available. For a traditional stuck-at fault, the implication is that a signal line is shorted to power or ground; whether this is plausible depends upon the proximity of a supply wire to the signal line. Similarly, the plausibility of a two-line bridging fault is highly dependent upon the proximity of the two lines. This information, however, is not normally used during traditional fault diagnosis, which usually only works with netlist information and test data, and so it was not included in the calculations specified above or in the experiments described below.

But the issue of proximity raises an interesting avenue of fault interpretation. Not all correlations or fault relationships can be expressed by traditional fault models. There are complicated defect scenarios that affect isolated areas of a die, such as large spot defects, physical damage, or poor localized implantation [NighVal98]. No current fault model could properly capture such a scenario, even though a STAT-type diagnosis might implicate faulty circuit nodes in the area of the defect.

An additional type of correlation, then, would be useful for interpreting a multiplet: *the physical proximity* of

the component faults. This proximity can be calculated from an analysis of the layout or artwork files. When faced with a set of multiplets in a diagnosis, the proximity measure would tell the failure analysis engineer how localized the faults for each multiplet are in silicon. Given the limits to how much area physical investigation can reasonably cover, a high physical proximity correlation could very well be the most valuable information to an FA engineer, more valuable perhaps than any fault model.

Another type of proximity measure that would be interesting for multiplet analysis is *logical proximity*, or the number of gates or cells that separate the set of faults in the multiplet. This information would be easier to calculate than physical proximity, since it can be determined from the same netlist file used for fault tracing and simulation. Some of this proximity information is captured in the node, net, and gate fault classes, but some more complicated defects may involve several gates. In any case, both the logical and physical proximity measures could indicate how related a particular set of faults in a multiplet are, which may help in limiting the search for root cause to an area of the die or to an area of functional logic.

These proximity calculations were not performed as part of the experiments reported in this paper. However, they are likely to play an important part in continuing research on fault interpretation, and so remain an area of further development.

6. Experimental Results – Multiplet Classification

Table 2 gives the results of multiplet classification on the same simulated defects from Table 1. For each simulated defect, the plausibility of the top (correct) multiplet is calculated vis-à-vis each defect class.

As expected, some of the defects are classified as stuck-at faults simply because the multiplet size is 1. For the bridging and gate faults that are classified as stuck-at, the result is highly dependent on the test set – if the tests don't activate the other faults or fault polarities, then these defects will look like stuck-at faults.

Generally speaking, a fault that received a 0.0 plausibility score for all defect classes was a case of multiple unrelated stuck-at faults. It is possible, however, for two unrelated stuck-at faults to get a non-zero bridging fault score, as happened with defect #3. For that defect, the stuck-at faults in the multiplet are of opposite polarity, and all vectors common to the two fault signatures fail, so there is nothing in this behavior that is inconsistent with a two-line bridging fault. On the other hand, the component faults for defects #2 and #4 are completely inconsistent with a bridging fault. In either case, this analysis can only judge the consistency of the behavior with a bridging fault; it would take either layout analysis, or a bridging-fault diagnosis algorithm, or both, to judge whether the bridging fault is actually a good explanation for the behavior

Defect No.	Simulated Defect	Single Stuckat	Node Fault	Net Fault	Gate Fault	2-Line Bridge	Path Fault
1	Single stuck-at fault	**1.0**	0.0	0.0	0.0	0.0	0.0
2	2 independent stuck-at faults	0.0	0.0	0.0	0.0	0.0	0.0
3	2 independent stuck-at faults	0.0	0.0	0.0	0.0	**1.0**	0.0
4	2 interfering stuck-at faults	0.0	0.0	0.0	0.0	0.0	0.0
5	3 interfering stuck-at faults	0.0	0.0	0.0	0.0	0.0	**0.67**
6	4 stuck-at faults, 3 interfering	0.0	0.0	0.0	0.0	0.0	**0.75**
7	Two-line wired-OR bridge	0.0	0.0	0.0	0.0	**1.0**	0.0
8	Two-line wired-AND bridge	0.0	0.0	0.0	0.0	**1.0**	0.0
9	Two-line wired-AND bridge	0.0	0.0	0.0	0.0	**1.0**	0.0
10	Two-line wired-XNOR bridge	0.0	0.0	0.0	0.0	**1.0**	0.0
11	Two-line dominance bridge	**1.0**	0.0	0.0	0.0	0.0	0.0
12	Two-line dominance bridge	**1.0**	0.0	0.0	0.0	0.0	0.0
13	Net fault (3 branch stuck-at faults)	0.0	0.0	**1.0**	0.0	0.0	0.0
14	Net fault (3 branch stuck-at faults)	0.0	0.0	**1.0**	0.0	0.0	0.0
15	Gate replacement (OR to AND)	**1.0**	0.0	0.0	0.0	0.0	0.0
16	Gate replacement (OR with NOR)	0.0	**1.0**	0.0	**1.0**	0.0	0.0
17	Gate replacement (MUX - NAND)	0.0	0.0	0.0	**1.0**	0.0	0.0
18	Gate output inversion	**1.0**	0.0	0.0	0.0	0.0	0.0
19	Multiple logic errors on one gate	**1.0**	0.0	0.0	0.0	0.0	0.0
20	Multiple logic errors on one gate	0.0	0.0	0.0	**1.0**	0.0	0.0

Table 2. Results from correlating top-ranked multiplets to different fault models.

7. Analysis of Multiple Faults

By correlating multiplets to individual fault classes, the classification procedure implicitly invokes the single fault assumption, which is that the observed behavior can be attributed to a single fault mechanism. But, one of the strengths of per-test approaches is that they should be able to implicate the components of multiple simultaneous defects.

The signal for the likely presence of multiple faults is low plausibility scores for all fault classes. This would indicate that the multiplets don't match up well with any single fault scenario and the behavior may be due to multiple faults. There are several ways, then, to re-analyze the candidates to infer multiple fault groups.

Some of the fault classes define partial correlation scores, and for these classes a non-zero score might indicate that some of the faults in a multiplet fit the defect scenario. These are the path, net, and gate fault classes, and if a multiplet gets an imperfect but non-zero score for any of these classes, the faults that do correlate well can be separated and the rest of the faults re-analyzed to infer the presence of a second defect.

Another way to infer multiple defects is by applying the proximity measures introduced in the last section. Groups of individual faults that have high mutual proximity imply a high probability that they are related in a single defect mechanism. These proximity measures can be used to identify likely groups of faults, which can then be re-analyzed to correlate with the set of fault classes.

Finally, some of the fault classes have a defined cardinality, or a certain number of expected individual fault components. These are the two-line bridge fault class and the node class, and any multiplet that contains exactly two faults (node fault) or two, three or four faults (bridge fault) will automatically get a plausibility score of 0 for these classes. If multiple defects are involved, however, the multiplet could contain a viable node or bridge candidate mixed in with other candidate faults.

For these two fault classes, an exhaustive search has to be performed for large multiplet sizes. The case of node faults is simple: unless two faults of opposite polarity on the same node (e.g. A-sa-0 and A-sa-1) are contained in the multiplet, there is no evidence for a node (or transition) fault. For bridging faults, given a multiplet of k otherwise-uncorrelated stuck-at faults, there are $\binom{k}{2}$ possible bridging candidates. For most multiplets, this is an easily-handled number: for multiplets of size 20 it is 190 candidates, for size 50 it is 1,225 candidates, and for size 100 it is less than 4,950 candidates.

8. Conclusion

Per-test fault diagnosis techniques are a surprisingly effective and yet simple way to diagnose complex fault behaviors. We have presented several ways to further improve per-test fault diagnosis results by ranking candidates, considering more test results, and classifying multiplets into likely defect classes. Our experimental results on simulated defects indicate that our algorithm provides accurate and small diagnoses on a variety of defect types, and does a good job of interpreting multiplets as understandable fault models.

An additional observation is that a main advantage of STAT diagnosis is that it solves one of the major problems of model-based diagnosis: the problem of candidate extraction or selection. By using a STAT-based diagnosis algorithm that can identify likely fault components, and a way to translate the components into candidate fault models, model-based algorithms can be applied on much-reduced candidate spaces and produce much more precise diagnoses.

References

[WaiLin89] J. Waicukauski and E. Lindbloom. Failure diagnosis of structured VLSI. *IEEE Design and Test of Computers*, pages 49-60, August 1989.

[EicLin91] E. Eichelberger, E. Lindbloom, J. Waicukauski and T. Williams. **Structured Logic Testing**. Prentice Hall, New Jersey, 1991.

[VenDru00] S. Venkataraman, S. Drummonds. POIROT: A Logic Fault Diagnosis Tool and Its Applications. *Proceedings of the International Test Conference*, pages 253-262, IEEE, 2000.

[BarHea01] T. Bartenstein, D. Heaberlin, L. Huisman, D. Sliwinski. Diagnosing Combinational Logic Designs Using the Single Location At-a-Time (SLAT) Paradigm. *Proceedings of the International Test Conference*, pages 287-296, IEEE, 2001.

[Sha76] G. Shafer**. A Mathematical Theory of Evidence**. Princeton University Press, Princeton, New Jersey, 1976.

[Tor38] S.C. Tornay. **Ockham: Studies and Selections**. Open Court Publishers, La Salle, IL, 1938.

[BarBha01] T. Bartenstein, J. Bhawnani. SLAT Plus: Work in Progress. 2nd International IEEE Workshop on Yield Optimization and Test, Nov. 1-2, 2001.

[NighVal98] P. Nigh, D. Vallett, A. Patel, J. Wright, F. Motika, D. Forlenza, R. Kurtulik, W. Chong. Failure Analysis of Timing and I_{DDQ}-only Failures from the SEMATECH Test Methods Experiment. *Proceedings of the International Test Conference*, IEEE, pages 43-52, 1997.