Better Models or Better Algorithms? Techniques to Improve Fault Diagnosis

The simple stuck-at fault model paired with a complex fault diagnosis algorithm is compared against the bridging fault model paired with a simple fault diagnosis algorithm to determine which approach produces the best fault diagnosis in CMOS VLSI circuits.

by Robert C. Aitken and Peter C. Maxwell

Failure analysis is an important task for continuous improvement of both the quality of shipped ICs and the underlying fabrication process. Fault diagnosis (also called location) can aid the failure analysis process by producing a list of candidate faults given a set of observed tester failures. These faults are then mapped to potential defect sites, allowing a failure analysis engineer to target a specific and manageable portion of the chip.

Improvements to fault diagnosis have tended to be either improvements in fault modeling[1,2,3,4] or improvements in diagnostic heuristics and algorithms.[5,6,7,8] In this paper we attempt to analyze the relative contributions of models and algorithms by comparing the diagnostic ability of the simple stuck-at fault model paired with a complex location algorithm to the complex and realistic bridging fault model paired with a simple location algorithm. These models and algorithms are compared both on known bridging defects from actual chips, and since the available sample of known bridging faults is small, on a larger sample of simulated bridging faults.

Only voltage testing is considered in this analysis. In addition, we employ a single fault model in all cases, both for simplicity and because in many of the cases of interest for diagnosis, single-site defects are more likely. A part that failed its functional package test, for example, probably contains only a single defect, or it would not have passed its numerous tests at the wafer level and its parametric tests at the package level.

We consider only full-scan circuits in this paper, since full-scan circuitry continues to be more amenable to diagnosis than nonscan or partial-scan circuitry. The main reason for this is that full scan does not require the fault models to predict future states accurately, since scan circuitry reloads the state after every test vector. This reduces the number of potential detections, and more significantly, reduces the dependency between potential detections, which can greatly complicate fault diagnosis. We find that about 10% of returned parts have failing scan chains and cannot be diagnosed, but for the remainder, the improvement in diagnostic accuracy is worth the costs of full-scan circuitry.

Fault Diagnosis Methodology

The fault diagnosis methodology is part of the overall failure analysis process. Not all failing chips are selected for failure analysis. Some typical candidates include chips that pass their component test but fail at board test, chips that fail component test in a similar fashion, and field returns. Fault diagnosis takes *failing test vectors** as input and returns a set of potential defect sites. Since these sites must be physically examined by a failure analysis engineer, it is important that there not be too many of them. Fig. 1 shows the tools we use and the files created during fault diagnosis.

* A given chip test consists of a set of hundreds or thousands of individual test vectors. On a given bad chip some of these vectors will produce outputs that fail the test, and it is these vectors that constitute the failing test vectors.

Bridging and Stuck-At Faults

The most common approach for modeling IC defects is the stuck-at fault model.[1] This model states that defective lines in a circuit will be permanently shorted to either the power supply (stuck-at 1) or ground (stuck-at 0). The model has been popular for test generation and fault simulation because it is simple to use and because a complete stuck-at test set thoroughly exercises the device under test (it requires that both logic values be observed on all lines in a circuit).

With the advent of CMOS integrated circuit technology, the connection between the stuck-at fault model and actual defects has become somewhat tenuous. This is less important from a test generation perspective, since tests for stuck-at faults tend to be excellent tests for other types of defects as well.[2] For diagnosis, however, an accurate fault model might be more important. In the accompanying article, we consider bridging,[3] which extends the stuck-at model by allowing a defective line to be shorted to any other line in the circuit, not just the power and ground lines. Unlike the simpler stuck-at model, there are numerous variations of the bridging fault model, depending on which bridges are considered (all possible versus layout-based), and how they are presumed to behave (wired AND, wired OR, dominant signal, analog, etc.). Our model[4] considers possible bridges extracted from layout and models their behavior according to the relative signal strengths of the driving transistors.

References

1. R.D. Eldred, "Test Routines Based on Symbolic Logical Statements," *Journal of the ACM,* Vol. 6, 1959, pp. 33-36.

2. T.W. Williams and K.P. Parker, "Design for Testability—A Survey," *Proceedings of the IEEE,* Vol. 71, January 1983, pp. 98-112.

3. K.C.Y. Mei, "Bridging and Stuck-at Faults," *IEEE Transactions on Computers,* Vol. C-23, July 1974, pp. 720-727.

4. P.C. Maxwell and R.C. Aitken, "Biased Voting: A Method for Simulating CMOS Bridging Faults in the Presence of Variable Gate Logic Thresholds," *Proceedings of the International Test Conference,* 1993, pp. 63-72.

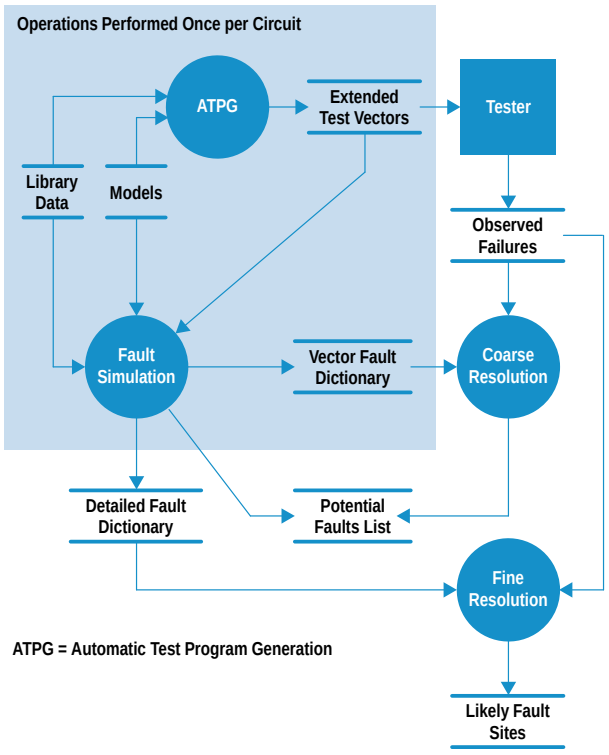


Fig. 1. Fault diagnosis tools and files.

Test Generation for Diagnosis. Scan vector sets for production test are usually optimized to achieve maximum single stuck-at fault coverage with a minimum number of vectors. This allows test costs to be reduced without compromising quality. A side effect of this process is that many faults tend to fail on the same set of vectors, making it difficult to distinguish between faults when using a vector fault dictionary,[9] which contains only failing test vectors. We improve the diagnostic resolution of production test vector sets by attempting to generate additional tests to distinguish between stuck-at faults which have identical failing behavior in the production set. This method seeks to maximize the diagnostic capability measures described in reference 10. The vectors produced from this effort are called an extended vector set.

The second shortcoming of production test sets is their reliance on the single stuck-at fault model. Stuck-at test sets can also be extended by vectors to target other faults such as bridging faults and transistor stuck-on/off faults. For the test sets used in our model, such faults were not explicitly targeted.

Fault Location Software. Once an extended vector set is available, it can be used in the diagnostic process. To avoid the large amounts of data associated with detailed fault dictionaries, we generate a vector fault dictionary, or fault coverage matrix (see Fig. 2). As with test vector generation, dictionary generation is performed only once for a given chip design.

The remaining steps are run on individual failing chips. Failing chips are run on the tester and their *observed failures* (test vectors and outputs where failures are observed) are logged. For chips with numerous failures, only the first few hundred failures are typically recorded. Diagnosis is a two-step process. Coarse diagnostic resolution is obtained by using the vector fault dictionary and the vectors from the observed failures to reduce the potential fault list from all faults to a manageable number.

The coarse resolution process eliminates the vast majority of faults from consideration. A more extensive fault simulation is then performed (using fast fault simulation techniques[11]) on the remaining faults to construct a detailed fault dictionary, showing not only which vectors are expected to fail, but also the scan elements and output pads where errors are expected to be seen. The output of this simulation is then compared with the tester data, and faults that match are sent on to failure analysis. This two-step process has been successful at reducing failure data and diagnosis time and predicting defect sites.

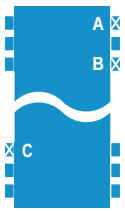
The success of the diagnostic software depends on the accuracy of the fault models and the ability of the algorithms to deal with unmodeled defects. The next section discusses the models and algorithms we selected for evaluation.

Fault Models and Algorithms

Improvements in the diagnosis process can be obtained by improving the fault models and the heuristics of the algorithms. Although previous work[5,6,7,8,12] has concentrated on the heuristics of algorithms, some results are available for diagnosis using more complex fault models,[3] often using I_{ddq}.*[2,13] Waicukauski et al[12] reported that their diagnosis method would work on bridging faults. Pancholy et al[14] developed a simple test chip to examine the behavior of stuck-at and transition faults on silicon. Millman et al[15] examined the relationship between simulated bridging faults and stuck-at fault dictionaries, using an early version of the "voting model"[8] for bridging simulation. Our work builds on these results by examining the behavior of actual bridging defects on silicon and performing simulation using biased voting,[4] which is an extension of the voting model that takes logic gate thresholds into account.

We examine the relative effectiveness of two approaches to diagnosis: the simple stuck-at fault model with a complex

* I_{ddq} is the quiescent drain current.



A, B, C = Circuit Outputs Where Failures Are Observed

Contents of Vector Fault Dictionary

Type of Fault	Failing Test Vectors
F1	1, 3, 5
F2	1, 2
F3	1, 3, 5

Contents of Detailed Fault Dictionary

Type of Fault	Failing Vectors + Circuit Outputs
F1	1-A 3-A, B 5-C
F2	1-A, B 2-C
F3	1-A, B, C 3-A 5-A, B

Fig. 2. The contents of a vector fault dictionary versus a detailed fault dictionary, which contains more data.

diagnostic algorithm and the more complex but realistic bridging fault model with a simple diagnostic algorithm. Using the stuck-at model and the simple algorithm together typically provides very limited success (although this is dependent on the manufacturing process). Clearly, using the complex bridging fault model with the complex algorithm would likely to be the most successful, but would give little insight into whether the algorithm or the model is the greatest contributor.

Stuck-at Fault Model. The stuck-at fault model continues to be the most commonly used model for test generation, fault simulation, and fault diagnosis. The model is simple and simulation using it is fast. This is important for dictionary-based diagnosis, especially if dynamic dictionary construction is used. Because of the model's ubiquitous nature, off-the-shelf CAD tools can often be used for much of the diagnostic process.

Bridging Fault Model. We use the biased voting model,[4] which is able to predict accurately the electrical behavior of a wide variety of bridging defects in standard cell CMOS circuits by considering the drive strength and logic thresholds of the circuit elements in the neighborhood of the bridge. Our implementation of the method runs approximately two to three times slower per fault than stuck-at simulation, but is still considerably faster than Spice, which it attempts to emulate. Use of a realistic bridging fault model requires the extraction of likely fault locations from the layout, which in turn requires an inductive fault analysis tool.[3] It is important that potential bridges be extracted between adjacent layers, as well as within layers.

Simple Diagnosis Method. The simple diagnosis method finds the best match between observed failure data and the predicted failure data. A failing output predicted by simulating a given fault for a given test vector matches the observed behavior when a failure that was observed on the tester appears at the same output for the same test vector. A fault is removed from consideration if it predicts a failure point at a vector (or vector/output pair for a detailed fault dictionary) where no failure was observed. This is equivalent to a somewhat relaxed fault list intersection algorithm.[8] In general, the best matches are chosen, and the actual number of matches depends on experience and the fault model being used. For our experiments, faults were only selected if they were able to predict all failures, which is the strictest selection rule.

Extended Diagnosis Method. The extended diagnosis method is similar to the bit-partial intersection method,[8] which extends the simple diagnosis method by not excluding faults whose simulation predicts failures that were not observed on the tester. Instead, the incorrect predictions for these faults are considered along with their correct predictions (i.e., simulated failures which were also observed on the tester). A final likelihood for each fault is determined by a weighted combination of the two measures, which is the extent to which the fault's predictions match the observed data. In general, a correct prediction is given a substantially higher weight than an incorrect prediction. For example, fault F3 in Fig. 3 (which has three correct predictions and one incorrect) has a higher likelihood than F2 (which has two correct predictions and zero incorrect).

It is possible to use an even more complex algorithm for diagnosis, such as the effect-cause method.[5] However, destructive scan (flip-flop outputs toggle during scan) on our example circuit precluded examining transition behavior, and no implementation of the algorithm was available for our experiments. Finally, it is easy to construct cases in which such algorithms can be misled by realistic bridging behavior, so the results are likely to be similar to those we obtained.

Experimental Results

Our experimental vehicle is a small, full-scan ASIC (nine thousand gates) implemented in a 1-μm process. Two experiments were conducted. In the first, parts with known bridging defects were diagnosed using the two approaches described above, and the results were compared with the known cause. In the second experiment, simulated bridging defects were diagnosed using the extended diagnosis method the stuck-at fault model.

Known Bridging Defects. The sample chips for this experiment came from two categories. Three chips had metal-to-metal bridging faults inserted with a focused ion beam (FIB). Two others were parts that failed at board test, and for which subsequent failure analysis revealed bridging defects as the root cause. The experiments were conducted so that the person running the diagnostic tools did not know the defect locations.

Both diagnosis methods described above are able to rank faults based on their ability to predict observed failures. For the example in Fig. 3, the simple diagnosis method would rank the faults F2, F1, with F3 being excluded (F2 and F1 have no wrong predictions and F2 has more correct predictions than F1), while the extended diagnosis method would rank them F3, F2, F1 (order is based on the number of correct predictions).

Since a failure analysis engineer usually does not have time to investigate a large number of defect sites, we declare a diagnosis to be misleading when at least nine simulated faults are assigned by the particular diagnosis method to have a higher likelihood than the actual fault of predicting observed failures because their simulated (or predicted) behavior closely matches observed behavior. We wanted to compare the two diagnosis methods by examining the number of misleading diagnoses.

Observed Failures* = A, B, C

Type of Fault	Predicted Failures**	Matches to Observed Failures	
F1	A	A	Correct
		0	Wrong
F2	A, B	A, B	Correct
		0	Wrong
F3	A, B, C, D	A, B, C	Correct
		D	Wrong

*Failures Produced by Applying Failing Test Vector Set 1 to a Failing Chip on the Tester (see Fig. 2)

**Failures Produced by Applying Failing Test Vector Set 1 to Fault Models

Fig. 3. The relationship between failing test vectors, observed failures, and the predicted failures produced by applying the failing vectors to a particular fault model.

The results of using these diagnostic methods on bridging faults are summarized in Table I. The entries under the fault model columns are the number of cells that have been predicted to have potential defects after performing fine resolution using the model in question. The extended diagnosis method which was used with the stuck-at fault model was able to identify the correct cell as the most likely defect for chips 2 and 3. For chip 4, the algorithm identified twelve other locations as being likely fault locations before it found the correct fault on the thirteenth try. For chip 1, on the other hand, faults on a total of 15 cells (including the actual defective one) predicted the observed failures correctly. However the stuck-at model on chip 1 also predicted failures on other outputs where no such failures were observed, which shows a danger in relying on a subset of failing outputs to create a dictionary of likely fault sites (see Fig. 1). Many of the faults on these 15 cells were equivalent,* so no stuck-at diagnosis could distinguish between them. Both chip 4 and chip 1 are thus misleading diagnoses because in both cases greater than nine faults match the observed failures better than faults at the actual defect site.

Table I
Results for Known Bridging Defects

Chip	Actual Defect	Type*	Fault Model	
			Bridging	Stuck-at
1	FIB bridge	Nonfeedback	1	15
2	FIB bridge	Feedback	1	1
3	FIB bridge	Feedback	1	1
4	Bridge	Feedback	1	13
5	Bridge	Feedback	2	5
6	FIB open	—	1	1
7	Open	—	2	2

FIB = Focused ion beam.

* See Fig. 4.

The bridging model was successful in each case using the simple diagnosis method. An unobserved failure was never predicted and all observed failures were predicted in each case. For chip 5, two bridges matched the observed behavior. The second possibility was also in the immediate vicinity of the defect.

We also analyzed two chips with known open failures (the final two chips in Table I). These failures can also be diagnosed to the correct location by both methods, showing that the bridging method did not produce misleading results in these cases. The actual rate of misleading diagnoses when the bridging method is used with nonbridging defects has not yet been determined primarily because of lack of data.

* Equivalent means that the same faulty behavior is caused by two different faults of the same kind.

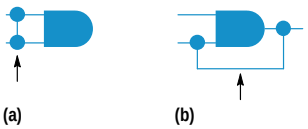


Fig. 4. (a) A nonfeedback fault. (b) A feedback fault.

Fault equivalence is much less common with realistic bridging faults than with stuck-at faults. Equivalent faults simplify test generation but complicate diagnosis because the nodes involved may be widely dispersed on the actual chip. The relative occurrence of equivalent faults is shown in Table II for all faults modeled in the circuit. The unique row shows the number of faults that behaved differently from all other faults in the test set. The vast majority of bridging faults behaved this way, but only 28% of stuck-at faults did, even though the test set was designed to maximize this behavior. The misleading row shows the number of faults with inherently misleading behavior, in that they were identical to at least nine other faults. Almost one sixth of the stuck-at faults belonged to this category, compared with less than 2% of bridging faults. This implies a substantial inherent misleading diagnosis rate even for stuck-at defects when the stuck-at model is used. The row labeled other represents faults that had between two and nine fault equivalencies. In total, the 25345 bridging faults exhibited 21540 failing behaviors, for an average of 1.18 faults per behavior, while the corresponding stuck-at figure was 2.04 for 21010 actual faults.

Table II
Equivalent Fault Behavior

Modeled Behavior	Bridging		Stuck-at	
	Number	Percentage*	Number	Percentage*
Unique	19404	76.6	5967	28.4
Misleading	412	1.6	3261	15.5
Other	1724	6.8	1086	5.1
Total	21540	85.0	10314	49.1

* As a percentage of actual faults (bridging faults = 25345 and stuck-at faults = 21010).

Simulated Bridging Defects. The data in the previous section shows that misleading diagnoses of bridging defects can occur with the stuck-at model, but are insufficient to allow a rate to be calculated. Rate is a measure of the probability of a misleading diagnosis. To get a better idea of the rate, the following simulation experiment was performed.

We selected 200 bridging faults at random from the set extracted for the circuit. Of these, 78 were feedback faults and 122 were nonfeedback. These faults were then simulated and a detailed fault dictionary obtained for each. The faults could not be used as observed failure files directly because they contained potential detection information.** Failure files were generated by assuming that 0%, 10%, 50%, 90%, and 100% randomly selected potential detections would result in errors. Our experience suggests that in practice this number is around 10% for our simulation method.

Coarse diagnosis (vectors only) was then performed on these files. Four stuck-at faults were considered for correctly describing the fault (stuck at 0 or 1 on either of the bridged nodes). One of these faults, which predicted the most observed failures, was then selected as the best match. All other faults that predicted at least as many failures as the best match were noted. Of these, the faults with no more than the number of failing predictions of the best match were included as candidates. This is meant to represent a

** In a simulation, a potential detection is a situation in which it is hard to prove that the fault was detected (see "Potential Detection" on page 115).

Table III
Results after Coarse Resolution

Category	Potential Detection (%)	Faults Remaining (%) 1-4	5-9	10-19	20-49	50+	Misleading Diagnoses (%)
Nonfeedback	0	46.7	15.6	5.7	11.5	20.5	37.7
	10	45.1	13.1	6.6	10.7	24.6	41.8
	50	41.8	13.1	8.2	8.2	28.7	45.1
	90	43.4	13.9	4.9	11.5	26.2	42.6
	100	42.6	13.9	5.7	11.5	26.2	43.4
Feedback	0	41.0	11.5	10.3	11.5	25.6	47.4
	10	35.9	14.1	15.4	5.1	29.5	50.0
	50	35.4	12.5	14.6	8.3	29.2	52.1
	90	33.3	14.1	14.1	7.7	30.8	52.6
	100	35.9	12.8	14.1	6.4	30.8	51.3
Bridge (self)	10	85.9	6.4	5.1	0.0	2.6	7.7

kind of optimal selection process, which is able to stop when it reaches the correct defect. In reality, the selection process would likely include other faults to guarantee that the correct fault is selected.

The coarse resolution results are summarized in Table III. As an example of interpreting the entries in the table consider the case in which 10% of predicted potential detections were considered to result in hard detections. For nonfeedback faults, 45% of the faults were resolved to fewer than five sites after coarse resolution, and the rate of misleading diagnoses was almost 42%. The average CPU time required for coarse resolution on an HP 9000 Model 735 workstation was 24 seconds for each feedback fault and 22 seconds for each nonfeedback fault. It seems evident that feedback faults result in a higher rate of misleading diagnoses than nonfeedback faults. This is not unexpected, since the behavior of feedback faults is more complex and their connection to stuck-at behavior is more limited.

As a reference point, one experiment (the last entry in Table III) was performed using the bridging model to attempt to diagnose the predicted bridging faults. In this case, the data for the feedback faults (which are more difficult to diagnose than nonfeedback faults) with a 10% rate of potential detections was diagnosed using a detailed bridging fault dictionary. The misleading diagnostic rate after coarse resolution was 7.7%. Most of these were cases where very few failures were observed. The average CPU time for coarse resolution was 38 seconds for this experiment.

Coarse resolution is only the first part of the diagnostic process, but the number of faults remaining after it occurs determines the time required to process the remaining faults. Fine resolution was performed on the faults remaining after coarse resolution. The results are shown in Table IV, which is identical in structure to Table III. The average CPU time required for fine resolution was 146 seconds for each feedback defect and 147 seconds for each nonfeedback fault.

It is interesting to note that in some cases the rate of misleading diagnoses actually increased after fine resolution. This may be because of a particular situation that occurs for some bridging faults. Sometimes vectors that fail closely match one of the stuck-at faults at one node of the bridge, while the actual failing signals propagate from another fault site. A typical example is shown in Fig. 5, in which the buffer is able to overdrive the NAND gate, so failing vectors can occur whenever the two nodes have opposing values. In this example, the buffer drives zero most of the time, and a stuck-at one failure on the buffer output matches the failing vectors extremely well. Since the buffer is always dominant, failures never propagate from that site and fine resolution would disregard that defect. In these cases, the coarse resolution process was modified to pick at least one fault from each site. This typically increased the number of faults remaining after coarse resolution by a factor of three or four. Note that the number of such defects is nontrivial, since global signals and buses tend to be driven with large buffers, and these signals pass or cross many others.

The faults that are difficult to diagnose using the extended algorithm and stuck-at fault model tend to share one of two characteristics: large fanout and/or similar drive strengths. In the first case, the so-called "Byzantine general"[16] behavior causes problems, where faults propagate along some fanout branches but not others. With reconvergence, this can cause failures where none would happen with a stuck-at fault on the stem and the reverse. In the second case, when neither gate in the bridge is dominant, the fault effects are dispersed and less closely tied to either of the bridged nodes.

The final row of Table IV again refers to using the bridging model to diagnose itself. As expected, misleading diagnoses are virtually nonexistent at this point, although this result is itself somewhat misleading, since the modeled defects are the same as those being diagnosed. Fine resolution took an average of 90 seconds per defect for this experiment. There were typically many fewer faults to resolve than with the

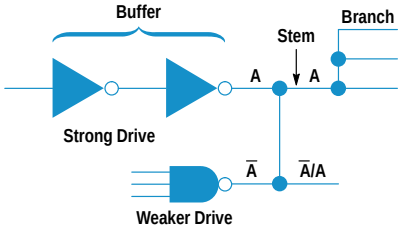


Fig. 5. Dominant bridging fault.

Table IV
Results after Fine Resolution

Category	Potential Detection (%)	Faults Remaining (%) 1-4	5-9	10-19	20-49	50+	Misleading Diagnoses (%)
Nonfeedback	0	43.4	14.8	8.2	13.1	20.5	41.8
	10	45.1	13.9	5.7	10.7	24.6	41.0
	50	42.6	14.8	11.5	12.3	18.9	42.6
	90	37.7	11.5	15.6	13.9	21.3	50.8
	100	36.9	14.8	13.1	12.3	23.0	48.4
Feedback	0	32.1	19.2	11.5	15.4	21.8	48.7
	10	29.5	16.7	14.1	24.4	15.4	53.8
	50	29.5	19.2	14.1	17.9	19.2	51.3
	90	35.9	19.2	11.5	15.4	17.9	44.9
	100	29.4	16.7	14.1	14.1	25.6	47.4
Bridge (self)	10	92.3	6.4	1.3	0.0	0.0	1.3

stuck-at model, which more than compensated for longer simulation time per fault.

Conclusion

We examined the occurrence of misleading diagnoses for an extended diagnosis algorithm with a stuck-at fault model and a simple algorithm with a realistic bridging fault model. A diagnosis was declared to be misleading when at least nine simulated faults were assigned a higher likelihood than the actual fault of predicting observed failures because their predicted behavior closely matched observed behavior.

In the results from the actual chips and the simulation experiments, it appears that the extended diagnosis algorithm, when used with the stuck-at fault model, results in a rate of about 40% of misleading diagnoses for realistic bridging faults. This is much higher than the rate obtained when using a simple diagnosis algorithm with a bridging fault model. This suggests that fault model improvement may be more beneficial than algorithm improvement in producing diagnostic success. Similar rates are seen for known bridging defects and for those simulated using the biased voting model,[4] although in the former case the sample size is too small for any conclusions. This work is continuing as a joint project between HP's Design Technology Center and HP's Integrated Circuit Business Division's quality group.

A second and somewhat counterintuitive result has also emerged from this work. In at least some cases, additional information can impede diagnosis. It was observed that vector dictionaries are sometimes better at diagnosing bridging faults than detailed fault dictionaries, particularly when the faults are of the one-gate-dominates variety. Additional work is underway to better characterize this behavior.

Acknowledgments

The authors gratefully acknowledge assistance from Jeff Schoper, Bob Shreeve, and others in the collection of chip data for this work.

References

1. J.M. Acken, *Deriving Accurate Fault Models,* Technical Report CSL-TR-88-365, Stanford University, Computer Systems Laboratory, October 1988.
2. R.C. Aitken, "A Comparison of Defect Models for Fault Location with I_{ddq} Measurements," *Proceedings of the International Test Conference,* September 1992, pp. 778-787.
3. A. Jee and F.J. Ferguson, "Carafe: A Software Tool for Failure Analysis," *Proceedings of the International Symposium for Testing and Failure Analysis,* November 1993, pp. 143-149.
4. P.C. Maxwell and R.C. Aitken "Biased Voting: A Method for Simulating CMOS Bridging Faults in the Presence of Variable Gate Logic Thresholds," *Proceedings of the International Test Conference,* October 1993, pp. 63-72.
5. J. Rajski and H. Cox, "A Method of Test Generation and Fault Diagnosis in Very Large Combinational Circuits," *Proceedings of the International Test Conference,* September 1987, pp. 932-943.
6. M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design,* W.H. Freeman and Co., New York, 1990.
7. P.G. Ryan, K. Davis, and S. Rawat, "A Case Study of Two-Stage Fault Location," *Proceedings of the International Reliability Physics Symposium,* March 1992, pp. 332-337.
8. P. Kunda, "Fault Location in Full-Scan Designs," *Proceedings of the International Symposium for Testing and Failure Analysis,* November 1993, pp. 121-127.
9. G. Ryan, W.K. Fuchs, and I. Pomeranz, "Fault Dictionary Compression and Equivalence Class Computation for Sequential Circuits," *Proceedings of the International Conference on Computer-Aided Design,* November 1993, pp. 508-511.

Potential Detection

A logic simulation will produce one of three values for a driven circuit output: 0, 1, and X, where X is unknown (the simulator cannot predict the value). The situation in which a circuit produces a known value, say 0, in a fault-free simulation but an unknown value, X, in the simulation of a fault, is called a potential detection. It is called potential detection because it will be detected if the unknown value is 1 on an actual faulty chip, but it will not be detected if the unknown value is 0. The following table summarizes these detectability conditions.

Good Value	Faulty Value 0	1	X
0	N	D	P
1	D	N	P
X	N	N	N

D = Detected
N = Not detected
P = Potentially detected

10. E.M. Rudnick, W.K. Fuchs, and J.H. Patel, "Diagnostic Fault Simulation of Sequential Circuits," *Proceedings of the International Test Conference,* October 1992, pp. 178-186.

11. J.A. Waicukauski, E.B. Eichelberger, D.O. Forlenza, E. Lindbloom, and T. McCarthy, "A Statistical Calculation of Fault Detection Probabilities by Fast Fault Simulation," *Proceedings of the International Test Conference,* November 1985, pp. 779-784.

12. J.A. Waicukauski and E. Lindbloom, "Failure Diagnosis of Structured VLSI," *IEEE Design and Test,* Vol. 6, no. 4, August 1989, pp. 49-60.

13. S. Chakravarty and M. Liu, "Algorithms for Current Monitor-Based Diagnosis of Bridging and Leakage Faults," *DAC-92 Proceedings,* June 1992, pp. 353-356.

14. A. Pancholy, J. Rajski and L. McNaughton, "Empirical Failure Analysis and Validation of Fault Models in CMOS VLSI," *Proceedings of the International Test Conference,* September 1990.

15. S. Millman, E.J. McCluskey and J. Acken, "Diagnosing CMOS Bridging Faults with Stuck-At Fault Dictionaries," *Proceedings of the International Test Conference,* September 1990, pp. 860-870.

16. L. Lamport, R. Shostak, and M. Pease, *The Byzantine Generals Problem,* Technical Report 54, Computer Science Laboratory, SRI International, March 1980.