

On Evaluating Competing Bridge Fault Models for CMOS ICs

Brian Chess Carl Roth
Tracy Larrabee

Computer Engineering, University of California, Santa Cruz 95064

Abstract

We compare the accuracy, speed and applicability to test generation of existing bridge fault modeling solutions. We identify some previously undiscussed anomalous circuit behaviors, and describe the extent to which they affect bridge fault simulation and testing. Finally, we present a system for evaluating bridge fault models in a test generation environment, and we present an experiment that provides an assessment of how defect coverage can be affected by a generating and checking model.

1 Introduction

In the search for IC quality, realistic defect testing is becoming increasingly important. In order to obtain the relatively modest quality level of 200 defective parts per million (DPM) in a circuit with a 90% yield requires that the test set detect 99.8% of the manufacturing defects [WB81, MB88]. The majority of spot defects in current MOS technologies cause changes in the circuit description that result in shorts [FM91]. Tests that cover 100% of the testable single stuck-at faults may not adequately cover shorts [MG91, FL91].

Until recently, people had to guess at the wire pairs for which to generate tests because the total number of pairs is prohibitively large,

and methods had not yet been developed to determine which of the possible shorts were important. Carafe, an inductive fault analysis tool [Jee91, JF93], extracts the relative probabilities of extra-conductor defects and missing conductor defects. Circuit faults extracted by Carafe that result in a short are referred to here as *realistic bridge faults*. For the purposes of this paper we consider only shorts involving gate inputs and outputs. The use of Carafe reduces the number of bridges to be considered from $O(n^2)$ to $O(n)$.

Carafe reports the likelihood of occurrence for each fault it extracts. This likelihood indicates how probable the fault is to occur relative to all of the other faults in the list. When we generate tests for bridge faults, we can report not only what percentage of the realistic bridge faults we test, but what percentage of the realistic bridge defects we test. The defect coverage should be much more indicative than the fault coverage when it comes to relating test quality to defects per million parts shipped (DPM) [MB88].

In the rest of this paper we will discuss various bridge fault models, how accurately they predict faulted circuit behavior, and how the choice of a bridge fault model affects ATPG effectiveness. We suggest a method for creating high-coverage tests when using imperfect models, and we present an experiment show-

ing our method is workable.

2 Survey of bridge fault models

A variety of techniques are used to model realistic bridge faults in ICs. Each fault model makes a set of assumptions about the circuits and faults being modeled; these assumptions make the models tractable, but they can lead to inaccurate prediction of faulty behavior. We will describe the regions of accuracy and inaccuracy of each model we discuss.

When the idea of test generation for bridges was new, the assumption that the bridges caused wired-AND or wired-OR behavior was fairly good. In the dominant technologies of the time (such as TTL), bridges did create wired logic. However, wired logic does not accurately reflect the behavior of CMOS circuits [AM91, FL91, MG91]. In fact, the actual CMOS behavior is far enough from wired-logic, that we must deal with voltages all the way from power to ground—including voltages that may be interpreted by some gate inputs as a logic 0 and some as a logic 1; this is known as the Byzantine Generals problem [AM92].

The wired-logic model (wired-AND or wired-OR) is the easiest model to implement for simulation and test pattern generation. It has the benefit of speed; the wired-logic model closely resembles the single-stuck-at-fault model with a few additional constraints [AM85]. The wired-logic model assumes that one of either the `nmos` or `pmos` networks will always determine the circuit's result. It is relatively easy to present examples in real circuits where this is not the case [FL91, GP92].

A better model would assume that the circuit value at the fault site is described in general by a Boolean function of the inputs to the gates driving the bridged wires. This two-

component model can be derived in a number of ways—two notable methods are analog simulation [FL91, RP93] and the voting model [Ack88, AM91].

Two component simulation works well at modeling the upstream components from the fault site, but fails to take into account the possible sensitive behavior of downstream components. This oversight can be dealt with in two ways. An optimistic model assumes that the bridge value is always digitally resolvable (in which case it might not always be correct). A pessimistic model describes the fault behavior with an incomplete Boolean function, where some of the bridge's behavior falls within a *gray region* within which the model fails to give an answer. Both of these approaches have been implemented in bridge fault simulators and test pattern generators [MG91, FL91].

In order to correctly resolve the gray region left by the pessimistic model, it is necessary to take additional circuit information into account. Depending on the input thresholds of the downstream components, a voltage in the gray region may be interpreted as different logic values: this is known as the Byzantine Generals problem [AM92]. We can address the Byzantine Generals problem by characterizing cell inputs and assigning analog input threshold values, which can then be compared to the analog bridge value to determine the behavior of the downstream gates. This technique for simple threshold determination has been implemented in bridge fault simulators using extensions of the voting model and using analog simulation [MA93, RP93]. The simple threshold characterization model assumes that the analog effect of the bridge fault only propagates one level downstream from the fault site. However, this assumption is not always valid, and predicting the correct behavior of the fault may require a more in-depth simulation or an analysis of a larger region of the circuit.

A more general model assumes that the analog behavior induced by the fault extends for a certain distance beyond the fault site, after which the circuit behavior is digitally resolvable. The EPROOFS simulator [GP92] implements this within a mixed-mode simulator, where a SPICE-like analog simulation of the region around the fault site is incorporated into a digital simulation of the rest of the circuit. EPROOFS provides correct answers for many cases, especially those involving feedback or the need for analysis of a larger region of the circuit, when previous models would have failed.

EPROOFS uses a promising approach, but it is slow. Further, there is currently no test pattern generator that implements such a sophisticated model. The only acknowledged model more accurate is full-circuit analog simulation, which is unwieldy for circuits of any significant size.

Even full analog simulation might not always give correct answers. Individual instances of real circuits are not identical, and simulations of them are not expected to be accurate to the level of precision that a simulator such as SPICE might suggest. This is not usually a problem; digital circuits are designed to operate correctly in the presence of the small fabrication and environment variations. But these behaviors can become significant in the presence of bridge faults because the circuit is not necessarily behaving in a digital manner. A minute voltage difference at the fault site might mean the difference between a logic 1 and a logic 0 downstream. Such discrepancies can be unintentional, due to the inherent inaccuracies of a model, or they can be uncontrollable due to differences in the circuits themselves. Even the most accurate models available might not correctly predict the behavior of a real circuit.

3 Accurately modeling bridge faults

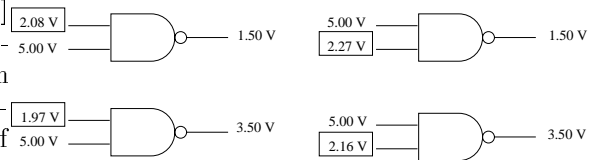


Figure 1: Simple determination of input thresholds

Figure 1 illustrates the input thresholds for a NAND gate. Here, we determine the logic 0 and logic 1 thresholds for each input when the other inputs are at non-controlling values. We see that for the top input, voltages greater than 2.08 V result in an output voltage below 1.50 V (input logic 1), while voltages less than 1.97 V result in an output voltage above 3.50 V (input logic 0). A similar analysis for the second input results in a logic 1 threshold of 2.27 V and a logic 0 threshold of 2.16 V.

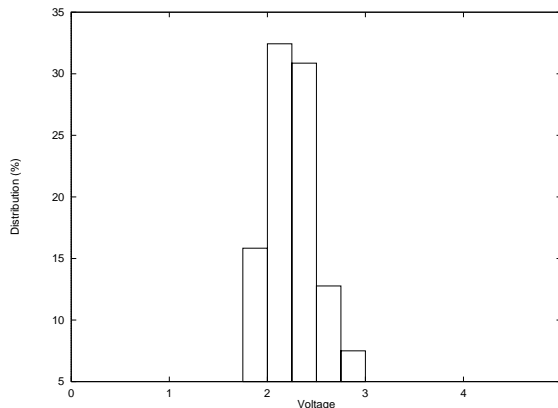


Figure 2: Distribution of input threshold voltages

Analysis for other cells places all input thresholds between 1.80 V and 2.90 V, with

more than 60% of the inputs having thresholds between 2.0 V and 2.5 V. Table 1 gives the logic 0 threshold for some of the combinational gates in a commercial standard cell library.

For the cell library we examined, voltages between 1.75 V and 3.0 V are intermediate because of the range of input threshold voltages in the same circuits. In Figure 2 we display the range of cell input thresholds.

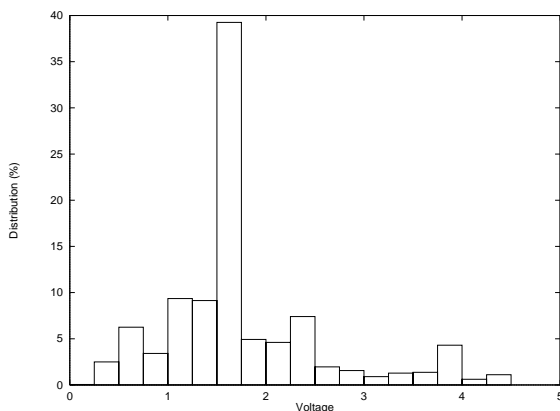


Figure 3: Distribution of bridge voltages

Gate	1st Input	2nd Input	3rd Input
NOT	2.36	-	-
NAND	2.02	2.21	-
NOR	2.58	2.81	-
AND	2.25	1.96	-
OR	2.16	2.37	-
XOR	1.83	2.17	-
AOI21	2.29	2.47	1.96

Table 1: Logic 0 input thresholds for sample gates

Figure 3 is the result of simulating 1000 random patterns against commercial layouts of the ISCAS-85 benchmark circuits [BF85].

These simulations show that indeterminate voltages are not uncommon in the presence of a bridge fault in these circuits. When the two wires involved in a realistic bridge fault were forced to different values, a SPICE simulation of the two components driving the bridge was initiated. The resulting graph shows that the wired-AND model will accurately predict about 70% the bridge values at the fault site. A fault model that combines the wired-AND and wired-OR models will correctly predict about 80% of the bridge values. It seems unlikely that the needed defect coverage can be met without a fault model that accurately addresses the 20 % of the bridge values between 1.75 V and 3.0 V.

Developing a solution to the Byzantine Generals problem raises an interesting issue: What precision should we assume for our circuit analysis? We certainly should not assume precision beyond that afforded us by our circuit simulator. For example, if we use SPICE we can't report more decimal places of accuracy than SPICE does. But when it comes to trusting the accuracy of the simulator itself, we must consider two additional sources of inaccuracy.

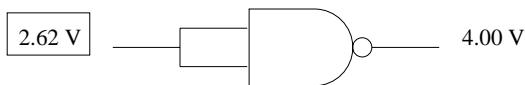


Figure 4: Inadequate simple gate thresholds

First, detailed characterization of a particular region of the circuit may not yield a correct analysis of the entire circuit. For example, if we perform SPICE simulations of the NAND gate in Figure 1, the resulting threshold values lead to an incorrect analysis of the circuit in Figure 4. These sorts of configurations might seem contrived, but they are created in some technology mapped circuits. The simple method of determining thresholds in Figure 1

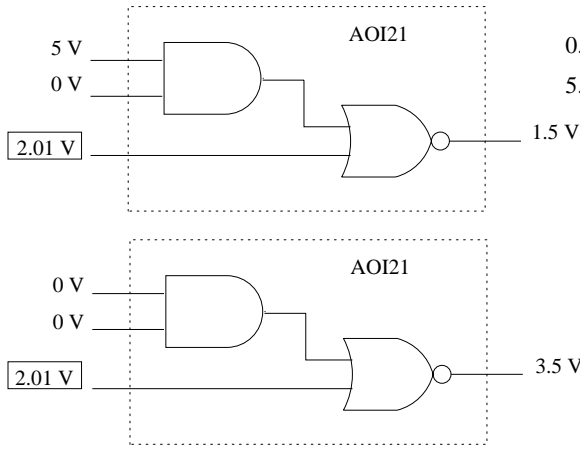


Figure 5: More inadequate simple gate thresholds

will also be inaccurate if there are multiple sets of inputs that sensitize the gate output, as in Figure 5. The threshold of the third input from Table 1 should cause the output of the lower gate to be a logic 0, but SPICE analysis shows this isn't the case.

Second, the circuit that we simulate may not be a close enough match to the circuit that is being tested. If the simulation of a vector results in a bridge voltage of 2.100 V and the threshold of a downstream component is 2.099 V, should we attempt to predict the output of the downstream component? What if the bridge voltage is 2.10 V and the component threshold is 2.09 V? The correlation between the simulated circuit and the physical circuit breaks down well before the end of the precision reported by SPICE.

Feedback bridge faults add their own complications, but they comprise too large a percentage of the realistic circuit defects for the issue of feedback to be ignored. Feedback bridge faults comprise between 10 and 60% of the total realistic bridge faults for our commercial layouts of the ISCAS-85 benchmark circuits

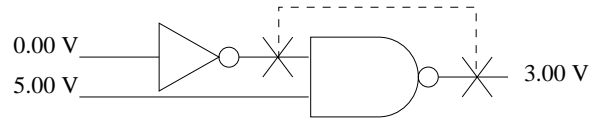


Figure 6: A non-oscillating feedback bridge fault

[CL93].

For wired logic, the issue of a bridge fault creating feedback is easy to deal with. When we consider the wired-AND model, if the fault appears on the back wire, and it sensitizes the front wire, the circuit may oscillate [AM85]. All we need to do is disallow tests in which the front wire is sensitized to the back wire. Similar arguments can be made for wired-OR tests invalidated by sequential behavior. More general arguments can be made for any possible Boolean bridge function [CL93].

These methods of dealing with feedback share a common characteristic: each of them analyzes the potential feedback path, and if it is sensitized, they reject the test. Instead of rejecting these tests, a more accurate model might find that the existence of a sensitized feedback path may actually allow the defect to be detected. For example, although a cursory analysis might find that the circuit in Figure 6 oscillates, a SPICE simulation suggests that the output will be stable, and rejecting the test is inappropriate.

An additional complication is introduced by the Byzantine Generals problem in the presence of feedback. It is well known that a feedback bridge fault can turn a formerly combinational circuit into a sequential circuit [AM92]. If the bridge can be accurately modeled in the purely digital domain, the sequential behavior can be modeled as a latch at the fault site. If the Byzantine Generals problem is considered, it is possible for the fault to hold a non-digital state, as Figure 7 demonstrates. If a digital latch is inserted at the fault site, the output

of the XOR gate will be forced to 0 regardless of the value held by the latch. This is because the inputs to the AND and OR gates are not allowed to make separate interpretations of the bridge value. If the latch is allowed to hold an arbitrary analog value, the resulting sequential circuit can hold the non-digital stable state.

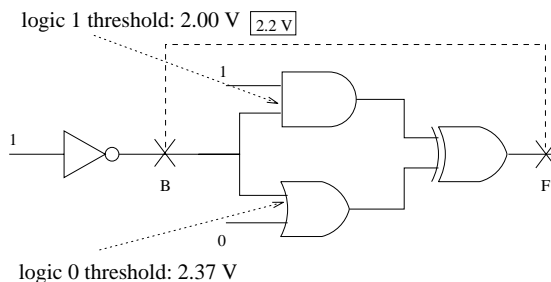


Figure 7: Non-digital stable state

The number of problems we have presented cannot be solved at the same time in a timely fashion. If we wish to generate the best possible tests in the least amount of time, we need to decide which of these problems we must address at each stage of our test generation process and what benefit we are thereby gaining.

4 Fault Models in ATPG

ATPG is always done with a fault model in mind. We suggest that bridge fault ATPG should be done with *two* fault models in mind.

Usually a sole fault model is chosen because it is feasible to incorporate that model into the test generation system. Because test pattern generation is exponential in the worst case, we tend to settle for a model that vastly simplifies the behavior of the potential defects. Unfortunately, the model that is used to present defect coverage statistics is usually the same model incorporated into the ATPG system. That is, the reported performance of the test pattern generation system is directly tied to the

strengths and weaknesses of the model that is used not only to generate the tests, but to grade the tests.

As we discussed previously, there are simulators that evaluate the accuracy of test vectors independently from ATPG systems, but the results are presented without taking into account the attempted goal of the provided test set. The simulator may tell you that 98% of the bridges of interest are detected by a given test set, but how closely does this match the behavior of the actual circuit? There may be faults declared tested that are not predicted correctly by the model. We may even test for 98% of the bridges, but the actual bridges we tested in the real circuit may not be the ones we thought we were testing for. While the test engineer might not care how we produced that test set as long as we delivered what we promised, if we deliver by accident, we may not always be so lucky. We need to be alerted to cases when a test does not detect the behavior predicted by the fault model.

In addition, the ATPG tool probably declared that some bridges are untestable; does this mean that they are untestable in the general sense, or untestable under the model used by the ATPG system? Faults that are declared untestable by a given model may in fact be testable, and there might exist another more accurate model that would have allowed us to generate a test. We need to be alerted to cases when a fault declared untestable by one model can be tested by another.

As we have shown earlier, the model used to simulate a circuit may never exactly reflect the behavior of an actual circuit. We must perform a cost-benefit analysis and use the best model we can for the cost. ATPG is such a time-consuming task that we, of necessity, use a simple model with only a basic correspondence to actual circuit behavior. In some sense, we cannot afford to use the more realistic models for ATPG. We could, however,

check the ATPG tests with a more expensive—and thus more accurate—model. This *checking model* need not be the most expensive and accurate available; it does need to have increased accuracy in areas in which the generating model is deficient.

In order to avoid rejecting a model out of hand due to isolated inaccuracies, we must keep track of the assumptions that were made in the formulation of the model. These assumptions give rise to cases in which the assumptions are not valid (and the model should not be trusted). By characterizing the *reliable* and *unreliable* regions of operation of a model, a confidence estimate, or quality value, can be assigned to the results of the model for a given input.

The requirement for more accurate ATPG is not necessarily a more accurate model. What is required is a checking mechanism for the model, similar to the confidence estimate described above. With a confidence estimate, an ATPG system can assess the quality of the tests it generates, and this heuristic can guide it to generate more reliable tests in general. For example, if the test generator can choose between placing 0.5 V on the bridge and placing 2.5 V on the bridge, the reliability of the test will increase if the bridge voltage is 0.5 V—well away from any input threshold.

Abstracting the fault model inside a circuit simulator fails to do this, because the simulator and the test pattern generator are completely independent. What is required is a better integration of the simulator, the model, and the test pattern generator so that global knowledge about the model’s strengths and weaknesses can be exploited to generate more accurate tests.

5 A sample experiment

To illustrate the potential benefit of a checking model in an ATPG system, we present a sample experiment that compares the fault coverage of a model with and without independent verification. We used the Nemesis ATPG system [Lar92] on the ISCAS-85 benchmarks. For the generating model, we use the simplest bridge fault model available: wired-AND. For the checking model, we use two-component simulation. We discussed the strengths and weaknesses of these models in Section 2. Both models reject tests that could be invalidated because of feedback.

Table 2 shows the results of our two experiments. The *defects verified* column shows the results when we generated a complete set of tests and then ran the set through a simulator using the two-component simulation checking model. The *defects checked* column shows the results when we generated each test and then used the two-component simulation model to verify and prune the tests. This gives the test pattern generation system another opportunity to generate a test that covers the fault if the two-component simulation model disallows the test.

With coverage improvements ranging from 0.04 to 3.49 percent, the additional coverage obtained by the use of the generating-checking pair affords us coverage superior to that obtained by using wired-AND alone (although still inferior to that of tests generated using the two-component model [CL94]). This improvement requires a much smaller programming effort than that required to generate tests with the two-component model because wired-AND test generation is very similar to stuck-at test generation [AM85]. Simulation is conceptually simpler than test generation, and the effort required to create a sophisticated simulator is far below that required to create an equally sophisticated test generator

Circuit	% defects verified	% defects checked	improve
C432	92.27	95.76	3.49
C499	97.63	98.33	0.70
C880	96.14	97.42	1.28
C1355	95.78	95.82	0.04
C1908	95.32	97.03	1.71
C2670	96.46	97.31	0.85
C3540	97.13	97.84	0.71
C5315	98.84	99.00	0.16
C6288	94.56	95.37	0.81
C7552	97.62	97.69	0.07

Table 2: Wired-AND verified & checked with two-component simulation

[CL93].

The time spent in generating the tests for the generating-checking pair is more than that spent to generate tests for wired-AND, by factor of two to four, and it takes about half of the time that it takes to generate tests using the two-component model. In this situation we have the ability to generate tests using the two-component model (paying the attendant costs), but as checking models become more sophisticated, this comparison is no longer feasible: it is not currently possible to use SPICE for test generation.

The results of this experiment would be more interesting if the checking model could modify the generating model as problems with the generating model were detected. For example, when the two-component simulation model determined that for a particular pair of gates and a specific set of inputs, the bridge did not create wired-AND behavior, then the generating model would no longer attempt to sensitize the fault with that input. By extending this idea to more sophisticated checking models, we have the potential for creating an

inexpensive test generator that produces very accurate tests.

6 Conclusions and future research

In order to generate quality tests with high defect coverages, we need to consider and correctly handle a large variety of faulty circuit behaviors—and as we have demonstrated, some of them can be difficult to simulate. We have shown that regardless of the model used to represent the circuit behavior, there will always be situations modeled incorrectly—either due to systematic deficiencies in the model or peculiarities of a given circuit. By characterizing when a model correctly predicts circuit behavior and when it does not, we can generate higher quality tests.

Quality tests must not only be accurate, but also timely. In the search for an optimal cost-benefit tradeoff, we suggest using a simple generating model and a more sophisticated checking model. This allows us to improve the accuracy and quality of our generated tests while allowing us to characterize and improve our generating model.

Acknowledgements

For technical discussions, data, and analysis, the authors thank the testing group of UC Santa Cruz—most notably Joel Ferguson, Haluk Konuk, Rich McGowen, and Alvin Jee. This work was supported by the Semiconductor Research Corporation under Contract 92-DJ-315 and the National Science Foundation under grant MIP-9011254.

References

[Ack88] J. M. Acken. *Deriving Accurate Fault Models*. PhD thesis, Stanford Univer-

- sity, Department of Electrical Engineering, September 1988.
- [AM85] M. Abramovici and P. R. Menon. A practical approach to fault simulation and test generation for bridging faults. *IEEE Transactions on Computers*, C-34:658–663, 1985.
- [AM91] J. M. Acken and S. D. Millman. Accurate modeling and simulation of bridging faults. In *Proceedings of the Custom Integrated Circuits Conference*, pages 17.4.1–17.4.4, 1991.
- [AM92] J. M. Acken and S. D. Millman. Fault model evolution for diagnosis: Accuracy vs precision. In *Proceedings of the Custom Integrated Circuits Conference*, 1992.
- [BF85] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinatorial benchmark circuits and a target translator in fortran. In *International Symposium on Circuits and Systems*. IEEE, June 1985.
- [CL93] B. Chess and T. Larrabee. Bridge fault simulation strategies for CMOS integrated circuits. In *Proceedings of Design Automation Conference*, pages 458–462, 1993.
- [CL94] B. Chess and T. Larrabee. Generating test patterns for bridge faults in CMOS ICs. In *Proceedings of European Test Conference*, 1994.
- [FL91] F. J. Ferguson and T. Larrabee. Test pattern generation for realistic bridge faults in CMOS ICs. In *Proceedings of International Test Conference*, pages 492–499. IEEE, 1991.
- [FM91] D. Feltham and W. Maly. Physically realistic fault models for analog CMOS neural networks. *IEEE Journal of Solid-State Circuits*, 26(9):1223–1229, September 1991.
- [GP92] G. Greenstein and J. Patel. EPROOFS: a CMOS bridging fault simulator. In *Proceedings of International Conference on Computer-Aided Design*, pages 268–271. IEEE, 1992.
- [Jee91] A. Jee. Carafe: An inductive fault analysis tool for CMOS VLSI circuits. Technical Report UCSC-CRL-91-24, University of California at Santa Cruz, Computer Engineering Department, February 1991.
- [JF93] A. Jee and F. J. Ferguson. Carafe: An inductive fault analysis tool for CMOS vlsi circuits. In *Proceedings of the IEEE VLSI Test Symposium*, page to appear, 1993.
- [Lar92] T. Larrabee. Test pattern generation using boolean satisfiability. *IEEE Transactions on Computer-Aided Design*, pages 6–22, January 1992.
- [MA93] P. Maxwell and R.C. Aitken. Biased voting: a method for simulating CMOS bridging faults in the presence of variable gate logic thresholds. In *Proceedings of International Test Conference*, pages 63–72. IEEE, 1993.
- [MB88] E.J. McCluskey and F. Buelow. IC quality and test transparency. In *Proceedings of International Test Conference*, pages 295–301. IEEE, 1988.
- [MG91] S.D. Millman and J.P. Garvey. An accurate bridging fault test pattern

generator. In *Proceedings of International Test Conference*, pages 411–418. IEEE, 1991.

[RP93] J. Rearick and J. Patel. Fast and accurate CMOS bridging fault simulation. In *Proceedings of International Test Conference*, pages 54–62. IEEE, 1993.

[WB81] T.W. Williams and N.C. Brown. Defect level as a function of fault coverage. *IEEE Transactions on Computers*, C-30(12):987–988, December 1981.