

## Acknowledgements

The authors thank Joel Ferguson, J. Alicia Grice, Alvin Jee, Haluk Konuk, Rich McGowen, and Carl Roth for technical contributions. This work was supported by the Semiconductor Research Corporation under Contract 90-DJ-141 and The National Science Foundation under grants MIP-9011254 and MIP-8907380.

## References

- [1] 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.
- [2] 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.
- [3] J. M. Acken and S. D. Millman. Fault model evolution for diagnosis: Accuracy vs precision. In *Proceedings of the Custom Integrated Circuits Conference*, 1992.
- [4] 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.
- [5] B. Chess and T. Larrabee. Bridge fault simulation strategies for CMOS integrated circuits. In *Proceedings of Design Automation Conference*, pages 458–462, 1993.
- [6] 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.
- [7] F. J. Ferguson and J. P. Shen. A CMOS fault extractor for inductive fault analysis. *IEEE Transactions on Computer-Aided Design*, 7(11):1181–1194, November 1988.
- [8] 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.
- [9] 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.
- [10] T. Larrabee. Test pattern generation using boolean satisfiability. *IEEE Transactions on Computer-Aided Design*, pages 6–22, January 1992.
- [11] W. Maly, M.E. Thomas, J.D. Chinn, and D.M. Campbell. Double-bridge test structure for the evaluation of type, size and density of spot defects. Technical Report CMUCAD-87-2, Carnegie Mellon University, SRC-CMU Center for Computer-Aided Design, Dept. of ECE, February 1987.
- [12] 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.
- [13] J.P. Shen, W. Maly, and F.J. Ferguson. Inductive fault analysis of MOS integrated circuits. *IEEE Design and Test of Computers*, 2(6):13–26, December 1985.
- [14] J. A. Waicukauski, E. B. Eichelberger, D. O. Forlenza, E. Lindbloom, and T. McCarthy. Fault simulation for structured VLSI. *VLSI Design*, VI:20–32, 1985.

```

foreach feedback bridge fault (FBF)
  front wire unfaulted value = 0, fault block output = 1
  if test generation is successful (sequential behavior must be prevented)
    FBF is covered, move to the next fault
  else front wire unfaulted value = 1, fault block output = 0
    if test generation is successful (sequential behavior must be prevented)
      FBF is covered, move to the next fault
    else if FBF is a feedback fault with no fanout
      FBF is undetectable, move to the next fault
    else back wire unfaulted value = 0, fault block output = 1
      if test generation is successful (oscillation must be prevented)
        FBF is covered, move to the next fault
      else back wire unfaulted value = 1, fault block output = 0
        if test generation is successful (oscillation must be prevented)
          FBF is covered, move to the next fault
        else FBF is undetectable

```

Figure 5: Pseudo-code for ATPG for bridge faults that may induce feedback

Circuit	Feedback		No
	FWF	FNF	Feedback
C0432	846	53	684
C0499	1,152	36	1,598
C0880	581	57	2,634
C1355	1,635	179	2,610
C1908	1,752	104	2,879
C2670	1,193	213	12,295
C3540	3,421	205	12,857
C5315	3,188	342	36,885
C6288	10,562	206	11,077
C7552	3,990	479	49,294

Table 3: Breakdown of circuit bridge fault statistics

number of PBFs is small compared to the number of faults, and in fact, only 309 different PBFs were used in the entire MCNC implementations of the ISCAS-85 benchmarks.

We can see that the number of feedback bridge faults is a significant percentage of the number of bridge faults, and we could not expect to achieve useful fault coverage if we did not produce accurate tests for the feedback bridge faults. The number of feedback bridge faults with no fanout are important, because Lemma 1 says that they can only be detected with the discrepancy placed on the front wire. This means that there are a significant number of faults that can never be tested with the discrepancy placed on the back wire.

Table 4 shows the number of bridge faults covered, proved untestable, or aborted by our system. For the ten circuits, we cover an average of 99.39% of the faults. We fail to generate tests for or prove untestable very few of the faults. Table 4 also shows the time in seconds on a Digital Equipment Corporation Alpha

Circuit	Number of Faults			Time (Secs.)
	C	U	A	
C0432	1,552	30	1	22
C0499	2,785	1	0	6
C0880	3,263	9	0	7
C1355	4,415	9	0	19
C1908	4,684	41	10	141
C2670	13,577	110	14	231
C3540	16,333	96	9	944
C5315	40,324	85	6	158
C6288	21,786	59	0	471
C7552	53,470	268	25	1,237

Table 4: Test pattern generation coverage

3000/400 necessary to achieve the reported coverage. This time is comparable to the time it takes us to process the single stuck-at faults for the same circuits. For all ten circuits, the bridge fault ATPG system takes an average of 4/3 the time per fault as the single stuck-at system takes, but for most circuits (including four of the five largest circuits), the time per processed bridge fault is less than the time for processed single stuck-at fault. This shows that realistic bridge fault ATPG is a viable alternative to single stuck-at ATPG.

## 5 Conclusions

We have presented a method for modeling realistic bridge faults, a theoretical foundation for generating combinational tests that correctly handles feedback bridge faults, and a test pattern generator that accurately generates tests for all realistic bridge faults. Our system is complete and practical, and it has been used to generate tests that cover at least 98.0% of the realistic bridge faults and an average of 99.4% of the realistic bridge faults in the MCNC layouts of the ISCAS-85 benchmark circuits.

If a discrepancy is to be propagated from the back wire, the feedback influenced region is a subsection of the faulted region. Analysis of the region consists of applying the PBF to faulted circuit values.

If a discrepancy is to be propagated from the front wire, the feedback influenced region is disjoint from the faulted region. Analysis of the feedback region involves propagating the complement of the unfaulted value of the back wire, and applying the PBF to the resulting values.

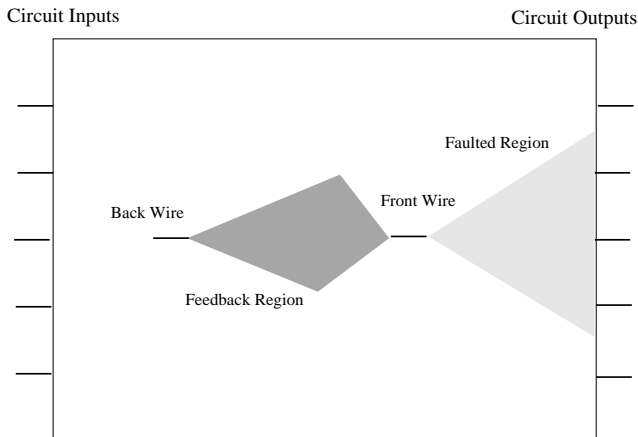


Figure 4: Feedback region for front wire discrepancy

### 3 ATPG

Our ATPG system generates test patterns for realistic bridge faults by using Carafe to determine the realistic bridge faults, using analog simulation to find the change in logical behavior for each fault, and then generating tests detecting all realistic bridge faults using the Nemesis ATPG system [10]. In the following sections we will describe the ATPG process in more detail.

#### 3.1 Simulator

Unlike previously published bridge fault simulators, our method of bridge fault simulation does not associate bridge faults with wires; instead, wires are allowed to “remember” if a fault can be propagated to a primary output [5]. After an attempt to propagate a discrepancy from a wire is made, a field in the wire’s data structure is set to reflect the success or failure of the propagation. If a fault further down the fault list introduces a discrepancy onto the same wire, it can immediately be determined whether or not the discrepancy can be propagated.

Our bridge fault simulation is done in parallel and is modeled after the PPSFP simulator of Waicukausi et al. [14]. Note that, given the PBF for a bridge, the bridge value for each of the parallel patterns is evaluated in the same fashion as that of any other gate (each of which can perform an arbitrary combinational function). In parallel bridge fault simulation, faults can be propagated from both of the wires involved in the bridge. We can do this because the fault block does not introduce a discrepancy on both of the

Circuit	Number of Faults		PBFs
	Stuck-At	Bridge	
C0432	564	1,583	118
C0499	786	2,786	36
C0880	966	3,272	156
C1355	1,882	4,424	58
C1908	1,246	4,735	117
C2670	2,237	13,701	191
C3540	3,185	16,483	254
C5315	4,865	40,415	247
C6288	8,748	21,845	33
C7552	6,291	53,763	253

Table 2: Number of faults and PBFs for each circuit

wires for any input pattern: the discrepancy is always on one wire or the other. This means that each bit-slice in the pair of faulted and fault-free wire values may represent a discrepancy on one wire or the other, but not both. A wire is placed on the simulation event queue if its faulted and fault-free values differ in any bit-position—regardless of whether the difference represents a value propagated from the fault on the first wire or the second wire. If the two bridged wires share a significant number of downstream components, the number of individual component simulations can be as little as half the number required by simulators that associate bridge faults with wires.

#### 3.2 Generator

Before describing our generator, it is important to distinguish between two types of feedback faults. If, for a particular fault, every path from the back wire to a primary output goes through the front wire, the fault is a *feedback fault with no fanout*. If some but not all of the paths from the back wire to a primary output go through the front wire, the fault is a *feedback fault with fanout*. As noted earlier, it is a consequence of Lemma 3 that a feedback fault with no fanout can only be detected with the discrepancy placed on the front wire. Pseudo-code for our feedback bridge fault test generator is shown in Figure 5.

For each attempt to generate a test for a bridge fault, we enforce constraints on unfaulted values for all wires, on faulted values for wires in between the discrepancy and any circuit output, and, for feedback bridge faults, on feedback-influenced values.

### 4 Results

Table 1 shows the number of stuck-at faults, the number of realistic bridge faults, and the number of Primitive Bridge Functions associated with the bridge faults for the MCNC layouts of the ISCAS-85 benchmark circuits [4]. There are 3 to 9 times as many bridge faults as there are stuck-at faults for the given circuits with the MCNC layouts.

Table 3 divides bridge faults into two categories: bridge faults that are capable of producing feedback and bridge faults that are not capable of producing feedback. Feedback bridge faults are subdivided into two groups: feedback bridge faults with fanout (FWF) and feedback bridge faults with no fanout (FNF). The

the test. We will examine the previous state of the bridge by cases:

First, assume that the previous value on the bridge is the same as the unfaulted value on the back wire. This means that all of the wires that are inputs to the fault block carry the same values as they do in the unfaulted circuit. With these inputs we know that the PBF assigns a discrepancy to the back wire.

Now assume that the value on the bridge previous to the application of the test is different than the unfaulted value on the back wire. The discrepancy that the test was designed to introduce is already present, so the previous state will not prevent a discrepancy from being placed on the back wire.  $\square$

Notice that Lemma 2 does not guarantee that the test will succeed, or that the circuit will not oscillate.

**Lemma 3.** If a potential test for a feedback bridge fault introduces a discrepancy on the back wire *without causing the output of the PBF to be dependent on the value on the back wire*, propagation of the discrepancy cannot be masked by oscillation.

In order to oscillate, the inputs to the PBF must change sufficiently to require the PBF to change the value on the bridge. If the output of the PBF cannot change, then the circuit cannot oscillate.  $\square$

Note that if all paths from the back wire to circuit outputs go through the front wire, it is impossible to propagate a fault from the back wire without affecting the value on the front wire; for this type of fault, it is impossible to create an oscillation-free test that places a discrepancy on the back wire.

**Lemma 4.** If a potential test for a feedback bridge fault introduces a discrepancy on the front wire *independent of the value on the back wire*, stimulation of the fault cannot be blocked by sequential behavior.

Consider a feedback bridge fault before the application of a test. Since we are applying a single combinational test, the value on the bridge before the application of the test is unknown. If the test does not rely on this unknown to have a particular value, the test is not dependent on that unknown.  $\square$

We now observe that the method for preventing oscillation is the same as the method for preventing sequential behavior—if a discrepancy can be propagated from the back wire without altering the inputs to the PBF such that the PBF changes the value on the bridge, then neither oscillation or sequential behavior will prevent a test regardless of which wire carries the fault. This observation leads to our main theorem:

**Test Guarantee Theorem for feedback bridge faults** . If a test creates a situation in which the result

of propagating either Boolean value from the back wire causes the PBF to assign the same value to the bridge, the test will not be invalidated because of feedback.

Lemma 1 and Lemma 2 show that the cases addressed by Lemmas 3 and 4 are the only situations in which feedback can affect a test. Lemma 3 and Lemma 4 show that if the output of the PBF is independent of the value on the back wire, then the test will not be invalidated by oscillation or sequential behavior.  $\square$

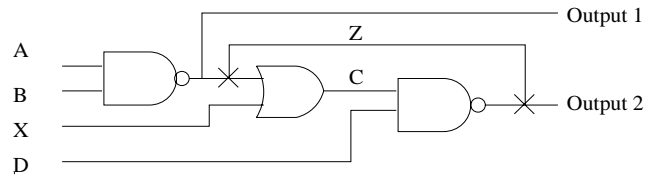


Figure 2: A feedback bridge fault that might oscillate

The Test Guarantee Theorem requires that the feedback loop created by the bridge be broken. We note that this requirement may not be satisfied simply by stipulating that the back wire not sensitize the front wire in the unfaulted circuit; the back wire must not be allowed to sensitize the output of the fault block. We can use Figure 2 to illustrate a situation in which the back wire does not sensitize the front wire in the unfaulted circuit but does sensitize the output of the fault block. When we use the PBFs from Table 1, the potential test  $ABXD = 1100$  will be rejected if we use the SPICE-derived PBF because it may cause the circuit to oscillate, but if we use the wired-AND PBF, the circuit could never oscillate.

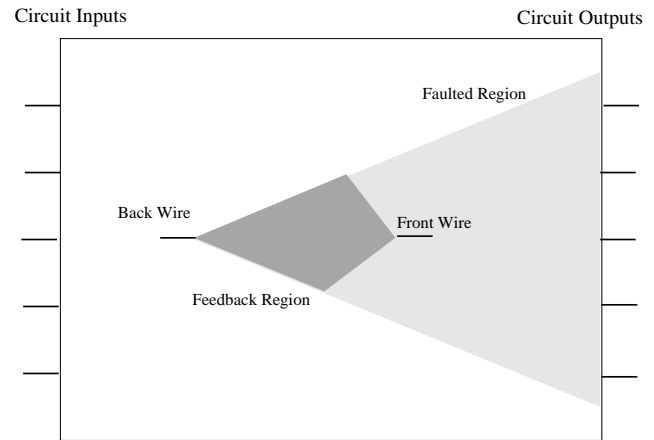


Figure 3: Feedback region for back wire discrepancy

Enforcing the additional constraints imposed by the Test Guarantee Theorem involves an analysis of the *feedback-influenced* region of the circuit. A wire is said to be feedback-influenced if it is on any path between the two bridged wires.

There are two sources of inaccuracy concerning the translation of the circuit faults provided by the fault extractor to the changes in logic function provided by the PBF generator. We present these next and show afterward why neither of them should significantly affect the results shown later in this paper.

One potential inaccuracy is the lack of a specific threshold voltage for converting the voltages derived from analog simulation to logic values for the PBF [3]. Gates, and even different inputs to the same gate, have different logic threshold values. The lowest input threshold we have found in the MCNC cell library is 1.8 volts; the highest input threshold is 2.8 volts. The largest threshold range for different inputs to a single gate is 0.68 volts. The result is a voltage range that can be interpreted by some inputs as a logic 0 and by other inputs as a logic 1, which leads to many PBF truth table entries being indeterminate. Therefore, in the analysis given later in the paper, we will use a single logic threshold — the voltage in which  $V_{in} = V_{out}$  for the smallest inverter in the cell library. This will cause the test pattern generation results to be incorrect for a small number of bridge faults, but the general results will not change significantly because the errors favor neither optimism nor pessimism in ATPG.

Another source of inaccuracy affects only feedback bridge faults. Because our PBFs are generated by analog simulation of representative non-feedback bridge faults, simulation results may be inaccurate when applied to feedback bridge faults. In general, feedback makes the front node “weaker” than predicted. This reduces the chance that the fault causes oscillation or sequential behavior, since this can only occur if the front node wins the conflict. In this paper we assume that all vectors that may cause oscillation or sequential behavior do not detect the fault, so the fault coverage figures presented later are likely to be conservative.

## 2 The Test Guarantee Theorem

We are interested in combinationally detecting bridge faults—that is, detecting each fault with only one pattern; we will not be considering multi-vector stimulation or propagation approaches.

When discussing feedback bridge faults, it is useful to refer to the two bridged wires by their locations in the circuit. Take any path that goes from a circuit input to a circuit output and contains the two bridged wires. The *back* wire is the wire closest to the circuit inputs on this path, and the *front* wire is the other bridged wire.

Combinationally detecting a non-feedback bridge fault involves stimulating the fault and propagating a discrepancy to a circuit output. A fault is stimulated if applying the PBF to the unfaulted circuit produces a value for the bridge that is different from the unfaulted value of one of the two bridged wires. A bridge fault with feedback may create a sequential circuit, in which case the state of the circuit may prevent stimulation of the fault, or a stimulated fault may cause oscillation, which may prevent a tester from detecting a discrepancy at the circuit outputs.

Most approaches for testing feedback bridge faults

check for tests invalidated by oscillation or sequential behavior by analyzing the inversion parity between the two bridged wires [1, 12]. Previous successful bridge fault test pattern systems—notably that of Millman and Garvey [12]—generate a test as if there is no feedback and then check to make sure that feedback will not invalidate the test. But oscillation and sequential behavior do not need to be prevented by performing a check after test pattern generation. Further, independence of previous state and the absence of oscillation can be established as a requirement for test generation without analysis of the inversion parity between the bridged wires. Before introducing the theorem from which we derive our test pattern generation system, we present four lemmas showing how sequential behavior or oscillation can affect potential tests.

**Lemma 1.** If a potential test for a feedback bridge fault produces unfaulted circuit values such that the application of the PBF places a discrepancy on the front wire, propagation of the discrepancy cannot be masked by oscillation.

In order for a test for a feedback bridge fault to be invalidated by oscillation, the value on the bridge must repeatedly feed back to the inputs to the PBF and cause the output of the PBF to change. We will examine the value on the bridge by cases:

First, assume that the value on the bridge is also the unfaulted value of the back wire. This means that none of the inputs to the fault block are different from the values in the unfaulted circuit, therefore the PBF will immediately assign a discrepancy to the front wire—no oscillation will occur.

Now assume that the value on the bridge is different than the unfaulted value on the back wire. If this causes the value on the bridge to change, it will become the unfaulted value of the back wire. As shown above, if the value on the bridge is equal to the unfaulted value of the back wire, the bridge cannot oscillate.

By placing the fault on the front wire, the feedback path introduced by the bridge is forced to be inactive. Without feedback, oscillation cannot occur.  $\square$

Notice that Lemma 1 does not guarantee that the test will succeed, or that the test will not be invalidated by sequential behavior.

**Lemma 2.** If a potential test for a feedback bridge fault produces unfaulted circuit values such that the application of the PBF places a discrepancy on the back wire, stimulation of the fault cannot be blocked by sequential behavior.

In order for a test for a feedback bridge fault to be invalidated by sequential behavior, the previous state of the bridge has to affect the state of the bridge after the application of

# Generating Test Patterns for Bridge Faults in CMOS ICs

Brian Chess

Tracy Larrabee

Department of Computer Engineering,  
University of California, Santa Cruz 95064

## Abstract

We describe a system for generating accurate tests for bridge faults (with or without feedback) in CMOS ICs. We present the Test Guarantee Theorem, which allows for accurate test generation for feedback bridge faults via topological analysis of the feedback-influenced region of the faulted circuit (without the need for any post-test verification or explicit examination of inversion parity). We describe our test pattern generation system’s treatment of feedback bridge faults in detail and report on the system’s performance.

## 1 Introduction

In the search for IC quality, bridge fault testing is becoming increasingly important. Defect simulation experiments have shown that the majority of spot defects in MOS technologies cause changes in the circuit description that result in bridges [13, 7, 11]. In order to reduce the defects per million of shipped ICs, we must determine both which defects are likely to occur and the behavior of a faulty circuit with such defects. Tests that cover 100% of the testable single stuck-at faults may do a poor job of covering bridge faults [12, 6]. Given this information, we must have an accurate and efficient method for generating tests that detect the likely-to-occur faults.

To find meaningful statistics concerning ATPG, we produce a list of *realistic bridge faults*—bridge faults that could be caused by a single defect connecting two gate outputs. Carafe, the fault extraction program that we use, considers the layout of the circuit and lists the nodes that are adjacent on the same conducting layer of the circuit or that cross each other on layers separated by a single layer of insulating material [8, 9].

Before generating tests for the realistic bridge faults, we must accurately model the change in logic function caused by the bridge. Because we are interested in defects in CMOS ICs, we cannot use a “wired-logic” fault model [6, 2]. Instead, we replace a portion of the unfaulted circuit with a *fault block* in the faulted circuit. The function of the fault block depends on the behavior of the bridged components in the chosen technology. We call the logic function of the fault block the *Primitive Bridge Function* (PBF) of the bridge fault [5]. Figure 1 shows how a bridge fault between the outputs of two NAND gates creates a fault block.

The PBF of the fault block can be determined, with varying degrees of accuracy, from resistive modeling

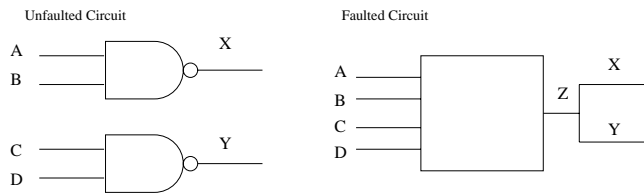


Figure 1: A bridge fault creates a fault block.

of the transistors [6], circuit analysis of the “vote” of transistors in series and in parallel [2], the tabular method used by Millman and Garvey [12], or an analog circuit analysis of the two gates. Table 1 shows three possible PBFs for the introduced fault block from Figure 1. The column labeled  $Z_{\text{WAND}}$  shows the fault block output if the technology in question follows the wired-AND model, the column labeled  $Z_{\text{WOR}}$  shows the fault block output if the technology in question follows the wired-OR model, and the one labeled  $Z_{\text{SPICE}}$  shows the fault block output derived from circuit analysis of the CMOS standard cell components from the Microelectronics Center of North Carolina (MCNC).

A	B	C	D	$Z_{\text{WAND}}$	$Z_{\text{WOR}}$	$Z_{\text{SPICE}}$
0	0	1	1	0	1	1
0	1	1	1	0	1	0
1	0	1	1	0	1	0
1	1	0	0	0	1	1
1	1	0	1	0	1	0
1	1	1	0	0	1	0

Table 1: Three possible PBFs

We chose to do two-component analog circuit simulation for each type of bridge but not each bridge (without considering the presence of feedback or the logic thresholds of downstream components). We assume a circuit simulation of each bridge fault is unnecessary because the use of a standard cell library ensures that every bridge fault between two 2-input NAND gates will share the same PBF. We construct the PBF for a fault block by simulating every input combination for which the two gate outputs will be different.