

Testing CMOS Logic Gates for Realistic Shorts

Brian Chess
Anthony Freitas
F. Joel Ferguson
Tracy Larrabee

Department of Computer Engineering
University of California, Santa Cruz

Abstract

It is assumed that tests generated using the single stuck-at fault model will implicitly detect the vast majority of fault-causing defects within logic elements. This may not be the case. In this paper we characterize the possible shorts in the combinational cells in a standard cell library. The characterization includes errors on the cell outputs, errors on the cell inputs, and excessive quiescent current. The characterization provides input vectors to stimulate these errors. After characterizing the faults that occur due to possible electrical shorts, we compare the coverage of the logic faults using a single stuck-at test set and tests developed specifically to detect these shorts. We discuss the effectiveness of I_{DDQ} testing for these faults.

1 Introduction

A large percentage of the physical area of any complex CMOS logic chip is used to implement the individual logic gates; internal shorts in the logic gates may not be detected by single line stuck-at faults (SSA). The standard testing strategy, that of generating tests that detect all SSA faults, explicitly guarantees that signal lines can carry both logic values and that all logic gates will pass both values from any input pin, but this may not be enough to guarantee that there are no defects within the gates.

Other researchers have shown that some opens and shorts may not be detected by SSA tests [Wad78, GCV80]. In this paper we determine the relative probability of each defect that causes an electrical short between two nodes in a logic cell. The probability is derived using the layout of the cell and the types of defects that are likely.

Our analysis of faulty logic gates will be on a commercial standard cell library of static CMOS gates. The smallest gate is an inverter and the largest is a 6-input and-or-invert gate. We extract and analyze all possible shorts that can be caused by a spot defect causing a conductor to be deposited on the polysilicon layer, a metal layer, or an oxide layer separating the polysilicon and metal layers. These shorts will be modeled as zero ohm connections. This is consistent with reports that the resistance of over 70% of shorts in the metal layers are either less than 500 ohms or high enough not to cause a logic fault [RMBF92]. Two

important classes of electrical faults in the logic gate will not be considered in this paper: opens and gate oxide shorts. Opens in transistor nets may cause sequential behavior [Wad78, WNS87]. Gate oxide shorts are an important class of faults, but they should be modeled as resistive shorts, which is beyond the scope of this paper. Their behavior is well documented by Hawkins and Soden, Syrzycki, and Hao and McCluskey [HS86, Syr87, HM93].

The system that we use to extract, characterize, and test realistic shorts is illustrated in Figure 1. This system shares many components with the system used to generate tests for realistic bridges in the interconnect [FL91, CL94]. Like the previous system, we use Carafe to generate netlists and to extract realistic shorts. We will discuss Carafe in Section 2. Our characterization tool, CShort, characterizes the behavior of each short provided by Carafe. We discuss CShort in detail in Section 3. The use of the final component of the system, the Nemesis ATPG program, is described in Section 4.

2 Extracting realistic shorts with Carafe

Before we can discuss faulted cells, we must define a few terms. *Defects* are fabrication anomalies. We are only interested in local (spot) defects—defects affecting only a small portion of the cell. Local or spot defects are often the result of specks of contaminants on the IC or photolithography during manufacturing. Defects may cause changes in the transistor-level description of the circuit; this change in circuit description is called an *electrical fault*. Shorts are the only electrical faults that we consider in this paper. The change in the circuit's behavior is a *behavioral fault*. The changes in behavior can be detected as changes in logic function, excess propagation delay, or excess quiescent power supply current (or any combination of these).

We use Carafe [Jee91, JF93], an inductive fault analysis [SMF85] tool, to analyze individual cells. Carafe determines which shorts are likely to occur—the *realistic* shorts—based on the layout (physical design) of the circuit and the types of defects that occur during the chip's fabrication. We use the defect den-

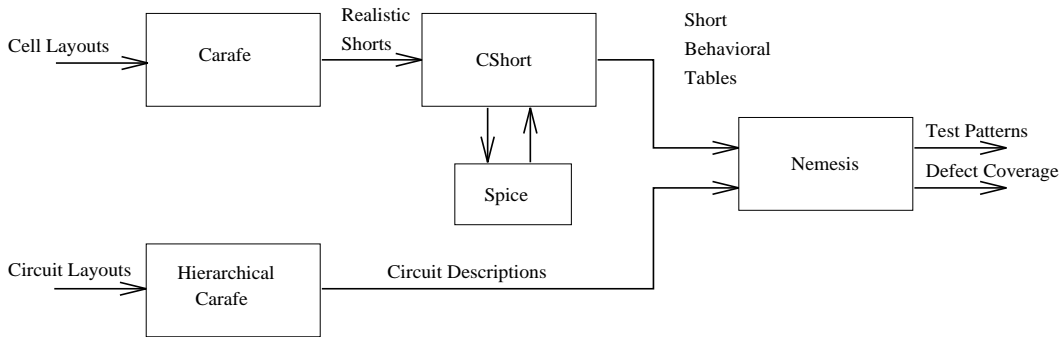


Figure 1: The test pattern generation system for shorts within standard cells

sities given by Feltham and Maly [FM91]. In addition to using the standard version of Carafe on the individual cells, we also use a hierarchical version of Carafe on the entire circuit under test. The hierarchical version is designed to create a gate-level description of the fault-free circuit instead of a transistor level description. We use this so we can do gate-level ATPG, which we will describe fully in Section 4.

Throughout this paper we use the term *short* to mean a Carafe extracted realistic short caused by a defect on the polysilicon layer, metal layer, or the oxide layers separating polysilicon and metal. For the experiments in this paper, we ran Carafe using the largest possible defect radius that did not short three parallel wires together.

For each short, there is a corresponding region wherein if the center of a defect would fall, the two nodes would be shorted together. The size of this region is the *critical area* of the short. Each critical area is multiplied by the relative defect density of the defect template that Carafe used to generate that critical area. This *weights* the critical area based on the probability of the defect occurring. Shorts in the fault list that produce equivalent circuits are combined, and the *weighted critical areas* (WCAs) are summed. The resulting WCAs determine the relative probability of each short.

Using each short’s WCA and the assumption that exactly one short occurs, we can determine the probability that a specific short, among all possible shorts, will occur, given that one short does occur. This is the weight of the fault (the WCA of the fault divided by the sum of the WCAs of all faults). In this manner, we can examine the defect coverage of a test set. The defect coverage of a test set is the sum of the weights of the faults detected by that test set. Because defect coverage is the sum of the conditional probabilities of the realistic shorts, it is a more accurate measure of the value of a test set than fault coverage.

3 Characterization of realistic shorts with CShort

Having determined which shorts are likely to occur, CShort characterizes each short’s effect on the standard cell by translating the short to the appropriate behavioral fault. Given a standard cell with

n inputs, CShort uses SPICE to simulate all 2^n input vectors in the presence of each realistic short predicted by Carafe. During simulation, all cell inputs are driven with standard size inverters from the cell library to provide a realistic simulation of the cell in a circuit.

CShort characterizes logic errors appearing on the inputs and the outputs of the cell, and it also measures excess power supply current for I_{DDQ} testing. In this paper we use a logic threshold of 2.3 Volts [CLR94] and a current threshold of .2mA [Gay93].

The first five columns of Table 1 show a portion of the fault-free truth table for a 4-input and-or-invert gate in a commercial standard cell library. The column labeled “Short value (voltage)” shows the digital value and voltage created by a short between the second input and the output of the gate. Only input combinations that produce conflicts are shown, and for each row we mark the entries that will be effected with a star(*). The short’s effect on the cell is not easily classifiable. It is not wired-AND or wired-OR: sometimes a discrepancy appears on the cell output, and sometimes a discrepancy appears on the input. In our experience, input combinations that cause logic errors always cause increased I_{DDQ} , and sometimes input combinations that do not cause any logic error cause increased I_{DDQ} .

3.1 Handling Feedback via CShort

Since a short may create a feedback loop, a formerly combinational cell may oscillate or gain the ability to hold state, although the requirements that must be met for a cell to oscillate are rarely fulfilled. In order for a cell to oscillate, the feedback loop must have odd inversion parity, stronger front transistors than back transistors, and a relatively long feedback path. Most cells fail in at least one of these requirements (often by not having a long enough feedback path).

Sequential behavior is more common than cell oscillation, and it is identified by CShort. If a feedback short from the input of a cell to the output of a cell occurs, and the feedback path has even inversion parity, there is a possibility that the cell will demonstrate sequential behavior. If these conditions are met, CShort applies six test vectors to determine if there is sequential behavior. If sequential behavior exists, these six

Unfaulted values					Short value (voltage)	Quiescent current
a1	a2	b1	b2	q		
0	0*	0	0	1	1 (3.16 V)	.8 mA
0	0	0	1	1*	0 (1.26 V)	.5 mA
0	0	1	0	1*	0 (1.26 V)	.5 mA
0	1*	1	1	0	0 (1.68 V)	.7 mA
1	0	0	0	1*	0 (1.31 V)	.7 mA
1	0	0	1	1*	0 (1.16 V)	.6 mA
1	0	1	0	1*	0 (1.16 V)	.6 mA
1	1	0	0	0*	1 (3.44 V)	.5 mA
1	1	0	1	0*	1 (3.44 V)	.5 mA
1	1	1	0	0*	1 (3.44 V)	.5 mA
1	1*	1	1	0	0 (1.50 V)	.7 mA

Table 1: Partial truth table for short between a2 and q in a 4-input gate

vectors allow us to identify three types: *dominant 0*, *dominant 1*, and *dominant output*.

Dominant 1 behavior occurs when the strongest 1 on the output is stronger than a 0 on the input; this differs from a wired-OR because all 1s in a wired-OR are stronger than all 0s (a wired-OR is a type of dominant 1). Similarly, dominant 0 behavior occurs when the strongest 0 on the output is stronger than a 1 on the input. The last of these behaviors, dominant output, occurs when both dominant 1 and dominant 0 occurs.

Six vectors are sufficient to identify a specific short as combinational, or one of the three sequential categories. The first three vectors first force the output to the strongest logic 0, then sensitize the path from the faulted input to the output node, and finally make a one bit change on the shorted input from logic 0 to logic 1 to determine if the logic 0 on the output dominates the logic 1 on the input. If the output remains a 0 by forcing the input to change to a logic 0, the faulted cell retains its original value (the input has a fixed strength because it is being driven by an inverter). The second three vectors work similarly by forcing the output to the strongest logic 1, then sensitizing the path from the input to the output, and finally making a one bit change from logic 1 to logic 0 on the input to determine if the strong logic 1 dominates the input’s logic 0.

CShort uses a SPICE simulation of the six vectors to classify the behavior of the faulted cell. Table 2 shows the six vectors, the type of behavior predicted by each of the sequential models, and the SPICE simulation results for the cell DEC1 in the CMOSN library. The unfaulted function of the DEC1 cell is $(IN1 \cdot IN2) + IN3$. From the SPICE results, we determine that a short between input IN3 and the output of the cell produces dominant 1 sequential behavior. Table 3 shows the CShort output that is generated for the DEC1 cell for this short. When the short value is listed as P in this table, the output is the logic value during the previous input.

Unfaulted values				Short value
IN1	IN2	IN3	q	
0	0	0	0	P
0	0	1	1	1
0	1	0	0	P
0	1	1	1	1
1	0	0	0	P
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Table 3: CShort output for DEC1 cell displaying dominant 1 behavior

3.2 CShort Results

As Hao and McCluskey observed, a discrepancy on a cell input may occur if the short joins an input and a node that is stronger than the input [HM91]. As Table 1 demonstrates, this is the situation when the faulted 4-input and-or-invert gate has inputs 0111. We will discuss the exploitation of cell input error detection in Section 4.

Figures 2 through 4 summarize properties of the faults found in the standard cell library we used for our simulations. All plots are sorted by total WCA per cell, that is, the plots are ordered from the smallest cell, on the left, to the largest cell, on the right; the first bar represents the same cell in all six graphs.

The first plot in Figure 2 shows the distribution of WCA for the cell library. There is about a six-to-one difference in the critical areas (and hence the cell area) of the smallest and largest cells. The second plot in Figure 2 shows the percentage of defects that are shorts to feedthroughs. Feedthroughs are nodes that go across the cell to carry signals from one channel the next.

The next two plots show the percentage of defects and faults that cannot be detected with logic tests since no input stimulates an error on an input or output. In these graphs, shorts to feedthroughs are considered undetectable. The first plot in Figure 3 shows the percentage of WCA that is undetectable. We call these faults, those that produce no errors for any input, undetectable at the cell level. These faults are also undetectable when embedded in a larger circuit.

The leftmost bar in plot 1 of Figure 3 is a large inverter with 56% undetected defects. If we look at the same cell in plot 2 of Figure 2, we see that 56% of the WCA is due to shorts involving feedthroughs, so all of the undetected shorts for this cell involve feedthroughs. The percentage of faults that are undetectable are shown in the second plot in Figure 3. If all faults were equally likely these two bar graphs would be identical. We can see that the percentage of undetectable faults is often not a good estimate of the percentage of undetectable defects. We expect coverage estimates based on defects rather than faults to be a better predictor of test escapes.

The first plot of Figure 4 shows the percentage of defects that have errors on an input node for at least

vec	IN1	IN2	IN3	Unfaulted	Dom-1	Dom-0	Dom-out	SPICE
1	0	1	0	0	0	0	0	0 (0.00 V)
2	0	1	0	0	0	0	0	0 (0.00 V)
3	0	1	1	1	1	0	0	1 (5.00 V)
4	1	1	1	1	1	1	1	1 (5.00 V)
5	0	1	1	1	1	1	1	1 (5.00 V)
6	0	1	0	0	1	0	1	1 (4.33 V)

Table 2: Six vectors characterizing short in DEC1 cell (dominant 1 behavior)

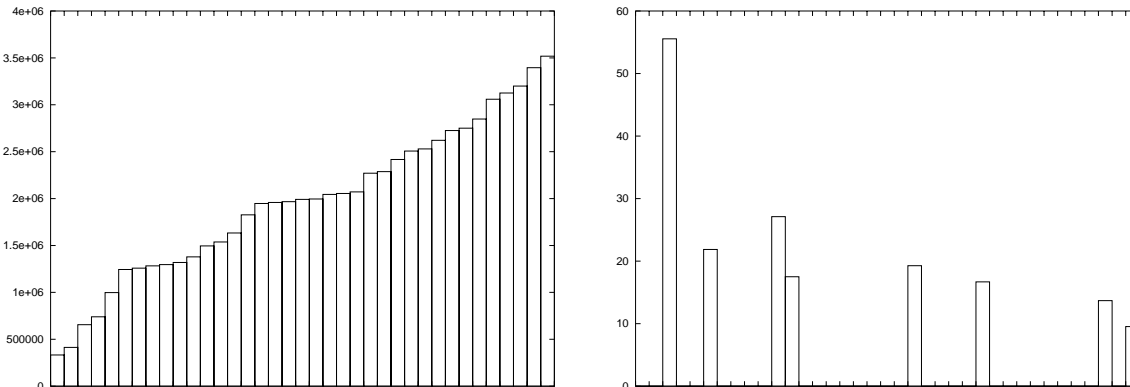


Figure 2: WCA distribution & % WCA due to feedthroughs for each cell

one test vector. Since errors on inputs can be used to detect faults, this provides additional opportunity for coverage. The second plot of Figure 4 shows the functionally detectable WCA for each cell. The bottom segment of the bar for each cell shows the percentage of defects that can be detected by errors on the outputs. The dashed segment shows the defects that can only be detected by input errors.

As can be seen from Figure 4, for many cells we must detect errors on the inputs to get more than 70% defect coverage. Alternatively, we can use another test method such as I_{DDQ} to achieve a higher defect coverage.

4 Test generation for realistic shorts via Nemesis

Nemesis, our test pattern generation system, can now simulate and generate tests for faults within cells. Before this addition, Nemesis produced tests for single stuck-at faults, tests for interconnect bridge faults, and tests to detect excessive I_{DDQ} in the presence of transistor stuck-on faults and bridge faults [Lar92, FL91, CL94].

4.1 Observing a faulted cell in a circuit

It is easy to detect when the output of a single NAND gate is stuck-at 1, just as it is easy to detect if an isolated cell is defective, but testing an entire circuit containing that NAND gate, or a faulted cell, may be more difficult. Once the cell is embedded in a circuit, the constraints required in order to stimulate

and propagate the fault may be contradictory or very difficult to obtain. As we will see in Section 4.3, many testable faults inside cells are untestable as logic faults when they are components in a larger circuit.

In many cases, errors on cell inputs are not observable, but Hao and McCluskey observed that if the the input wire fans out, as it does in Figure 5, the discrepancy may be detectable [HM91]. Our investigations have underlined just how important this effect is, and how important repercussions following from the effect are. We will be discussing this in detail throughout the rest of this paper. By the same token, if a discrepancy appears on both a cell input and the cell output, two discrepancies may converge and mask the fault, as shown in Figure 6. A discrepancy on a faulted cell input can, when considering cell output discrepancies, make a cell that appears to be untestable, testable. Likewise, input discrepancies may combine with output discrepancies so that a fault is untestable. A discrepancy on an input wire can affect the detectability of 48% of the WCA in the layouts of the ISCAS-85 benchmark circuits [BF85]. Faults representing 15% of the WCA in the commercial layouts of the ISCAS-85 benchmark circuits cannot be tested without justifying discrepancies from the cell’s input to a circuit output.

Because of discrepancies on the inputs to faulted cells, it is also possible for a short to cause a feedback path external to the faulted cell. This situation is similar to the feedback created by a bridge fault in the interconnect, and can be analyzed similarly [CL93].

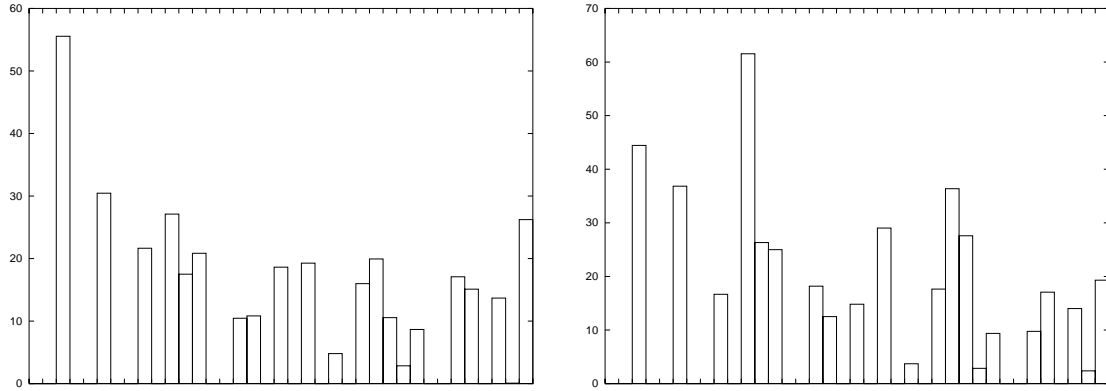


Figure 3: % cell-undetectable defects & faults for each cell

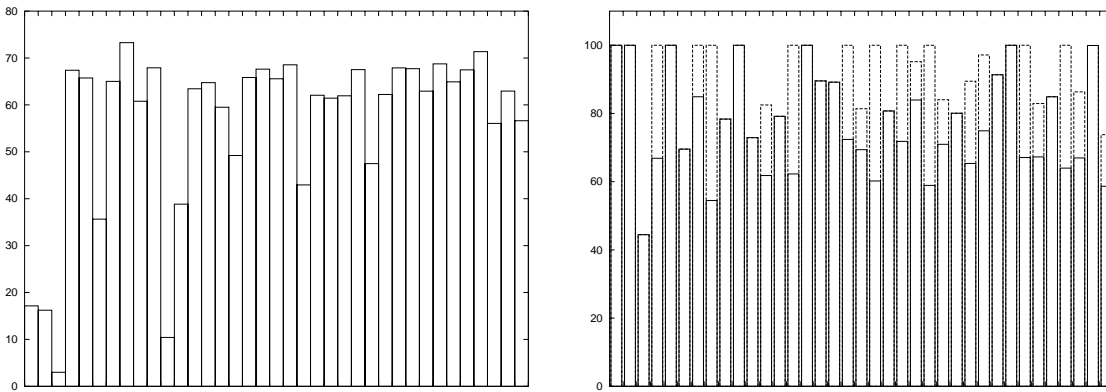


Figure 4: % WCA for faults with input errors & functionally detectable WCA for each cell

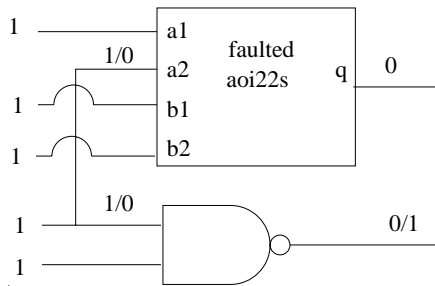


Figure 5: A short can cause a detectable discrepancy on a cell input

Figure 7 illustrates a situation in which analysis of the cell alone is not enough to predict the behavior of the cell in the circuit. The function of the short depends of the external logic of the feedback path. For example, if we apply the vector 0111 to the circuit in Figure 7, it will cause a sensitized path between input b1 and the output. However, once the path is sensitized, it will violate the requirements for sensitization, which will cause the output to change, and once again the path will be sensitized: SPICE simulation shows an

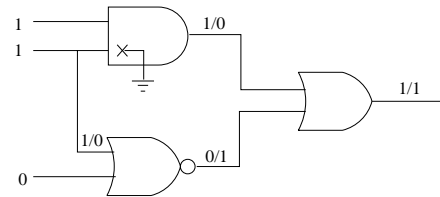


Figure 6: A short where the input and output discrepancy mask each other

oscillation between 1.83 Volts and 3.75 Volts along the feedback path. This is quite a different result than the constant 1.68 Volts predicted by Table 1.

4.2 The ATPG process

Test pattern generation for cell shorts begins by creating a fault list. Each cell in the circuit contributes a set of faults, derived by Carafe and CShort, to that list. With the fault list, Nemesis can generate either logic tests or I_{DDQ} tests. In either case, Nemesis attempts to generate tests only for non-trivial faults, that is, faults that join wires that should not be connected. It is possible (perhaps as the result of technology mapping) that a short inside a cell joins two

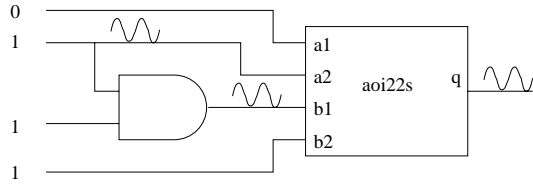


Figure 7: Errors on inputs can create large feedback loops

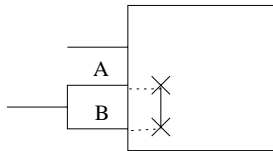


Figure 8: Wires A and B are shorted inside the cell, but are equivalent

wires that are connected on the outside of the cell. Figure 8 shows such an example. This situation is not very common, though it does occur in our commercial layouts of five of the ten ISCAS-85 benchmark circuits.

After deleting all trivial shorts from the fault list, Nemesis removes from the fault list all faults that are untestable at the cell level (even though we cannot generate tests for these faults, it is important for CShort to pass them to Nemesis, so that their WCA can be added to the total WCA for defect coverage calculations). With the remaining faults Nemesis performs pseudo-random Parallel Pattern Single Fault Propagation (PPSFP) fault simulation [WEF⁺85] for either logic or I_{DDQ} tests. When PPSFP simulation is finished, algorithmic generation is applied for the remaining faults on the fault list.

To generate I_{DDQ} tests, Nemesis attempts to stimulate the faulted cell with one of the input sets reported by CShort as guaranteed to cause excess quiescent current. When generating logic tests, Nemesis first performs some preliminary analysis of each wire in the circuit.

Nemesis determines the answers to two questions:

1. Can a discrepancy on a cell input be detected on this wire?
2. Will a discrepancy on a cell input be able to feed back into the cell?

Figure 9 shows the configurations of the four possible answers to these questions. The Type 1 configuration shows that a discrepancy on Wire A cannot be used to detect any shorts in the faulted cell. The Type 2 configuration shows a case where Wire A can potentially carry a discrepancy that can be justified to a circuit output. Finally, in the Type 3 and 4 configurations, Wire A can be used to detect shorts, and it may cause feedback, so we must explicitly deal with both potential complications.

Many wires are of Type 1. If a fault is only visible on inputs of Type 1, we know the fault can never be

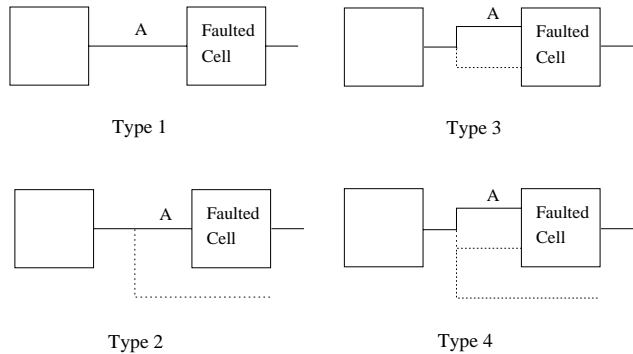


Figure 9: Four possible wire configurations with cell shorts

propagated to a circuit output, and we can mark it as untestable and remove it from the fault list.

Nemesis does not currently analyze faults that might be invalidated by feedback, so we do not use wires of Type 4 to propagate discrepancies, and even if a discrepancy appears on a Type 4 wire that is an input to the faulty cell, Nemesis will not propagate the discrepancy. Similarly, it is possible that faults visible only on inputs that are wires of Type 3 can be detected by propagating a discrepancy back through the faulted cell, but because our system does not analyze faults that might be invalidated by feedback, we do not accept such tests.

For algorithmic test generation, Nemesis attempts to stimulate the faulty cell with an input combination that will create a discrepancy on the cell output or on a cell input that will not create feedback. Hence it does not attempt to generate tests with discrepancies on cell inputs of Type 3 or 4. After a discrepancy is generated by stimulating the cell, Nemesis attempts to propagate the discrepancy to a circuit output.

4.3 Nemesis Results

Circuit Name	Faults			Defect Cover
	Cover	Untest	Fail	
C432	1,861	140	20	90.02
C499	3,199	0	8	78.09
C880	3,217	2	5	93.95
C1355	5,014	74	45	97.60
C1908	5,029	29	20	83.17
C2670	7,747	201	98	88.88
C3540	10,996	163	127	91.54
C5315	18,128	115	55	90.54
C6288	23,893	581	242	97.73
C7552	22,549	234	220	87.60

Table 4: ATPG for logical detection of shorts

We present the results of test pattern generation in Table 4. The table shows the number of faults covered, proved untestable, and failed for logic tests generated for each of the ISCAS-85 circuits as implemented, together with the defect coverage. Table 5 shows the

Circuit Name	Faults Covered	% Defect Coverage
C432	1,861	90.02
C499	3,153	77.21
C880	3,208	93.71
C1355	5,014	97.60
C1908	4,993	82.60
C2670	7,765	89.02
C3540	10,971	90.97
C5315	18,121	90.52
C6288	23,893	97.73
C7552	22,612	87.76

Table 5: Coverage of shorts with a SSA test set

cell-fault coverage of a complete SSA test set. The reason the defect coverages seem low compared to the ratio between the covered faults and the total number of faults is that shorts to feedthroughs are not considered as Nemesis faults (because we do not model them), but their WCA is included in the total WCA).

We can see that generating logic tests specifically for cell-shorts may or may not provide enough additional coverage to make it worth targeting cell faults specifically.

However, our guaranteed defect coverage using logic tests of either kind is around 90%, on average, which is too low. A majority of the undetected WCA represents shorts to feedthroughs, so our coverage will not improve much unless we model, simulate, and generate tests for shorts to feedthroughs. We could ignore all the WCA related to feedthroughs and show the coverage of only the faults characterized by CShort, but we are interested in reporting the guaranteed coverage of all of the Carafe-extracted faults and not just the faults we model. Correct modeling of feedthroughs is left for future work.

Circuit Name	Faults			Num vecs	Defect Cover
	Cover	Untest	Fail		
C432	2,021	187	0	30	95.27
C499	3,359	997	0	84	86.60
C880	3,335	228	0	34	96.28
C1355	5,161	0	0	81	100.00
C1908	5,315	993	0	92	90.64
C2670	8,303	693	19	41	95.20
C3540	11,846	735	4	67	96.11
C5315	19,188	1,383	0	61	96.08
C6288	24,424	34	0	32	99.81
C7552	24,037	3,088	18	71	93.81

Table 6: ATPG for detection of shorts via I_{DDQ}

In Table 6, we present the number of faults covered, proved untestable, and failed for I_{DDQ} tests generated for each of the ISCAS-85 circuits, together with the realistic defect coverage obtained. Table 7 pro-

Circuit Name	Faults Covered	Num vecs	Defect Coverage
C432	1,951	12	91.95
C499	3,198	19	83.13
C880	3,172	14	91.09
C1355	4,858	13	94.15
C1908	5,107	23	87.20
C2670	8,137	16	93.21
C3540	11,583	25	93.79
C5315	18,945	23	94.82
C6288	24,377	24	99.51
C7552	23,588	23	91.39

Table 7: I_{DDQ} coverage with a bridge I_{DDQ} test set

vides a good opportunity for comparison: it shows the I_{DDQ} cell-fault coverage of a complete I_{DDQ} test set for realistic interconnect bridge faults [FL91]. We can see that generating I_{DDQ} tests specifically for cell-shorts provides additional coverage for an increase in the number of tests.

5 Summary

In this paper we presented a method and software that, using the realistic shorts generated by Carafe, can give a DC characterization of all shorts in any small combinational logic cell. The characterization includes logic errors appearing on the inputs and outputs of the cell, and excess I_{DDQ} , for each of the 2^n possible input vectors. In addition, it identifies any sequential behavior induced by the short as one of three types.

With this software we characterized the combinational cells of a commercial standard cell library. Our characterization showed several surprising results. For many cells, a significant portion of the logic-fault causing defects can only be detected by considering the errors that appear on the input of the gate or with I_{DDQ} testing. Also, for some cells 20% of the defects not involving feedthroughs are undetectable without I_{DDQ} testing.

We have analyzed the effect of placing a faulted cell in a circuit, including pointing out the previously unreported potential for feedback complications once you examine a faulted cell in a complete circuit. We have pointed out that a cell-short can create a feedback path external to the faulted cell, and that input errors effect the testing of a very large number of the cell shorts that we examined. Taking input errors from cell shorts into account is important for accurate testing of faults within cells. We have described a complete system for testing cell faults and reported on its performance on benchmark circuits implemented with a commercial library.

Acknowledgements

We thank Braden Carter for work on our implementation, and Haluk Konuk and Rob Aitken for technical

discussions. This work was supported by the Semiconductor Research Corporation under Contract 93-DJ-315 and the National Science Foundation under grants MIP-9158490 and MIP-9158491.

References

- [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.
- [CLR94] B. Chess, T. Larrabee, and C. Roth. On evaluating competing bridge fault models for CMOS ICs. In *Proceedings of the 1994 VLSI Test Symposium*, page to appear. IEEE, 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.
- [Gay93] Rick Gayle. The cost of quality: Reducing ASIC defects with IDDQ, at-speed testing, and increased fault coverage. In *Proceedings of International Test Conference*, pages 285–292, 1993.
- [GCV80] J. Galiay, Y. Crouzet, and M. Vergnault. Physical versus logical fault models in MOS LSI circuits: Impact on their testability. *IEEE Transactions on Computers*, C-29(6):527–531, June 1980.
- [HM91] H. Hao and E.J. McCluskey. Resistive shorts within CMOS gates. In *Proceedings of International Test Conference*, pages 292–301. IEEE, 1991.
- [HM93] Hong Hao and Eduward J. McCluskey. Analysis of gate oxide shorts in CMOS circuits. *IEEE Transactions on Computers*, 42(12):1510–1517, December 1993.
- [HS86] C.F. Hawkins and J.M. Soden. Reliability and electrical properties of gate oxide shorts in CMOS ICs. In *Proceedings of International Test Conference*, pages 443–451. IEEE, 1986.
- [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.
- [RMBF92] R. Rodríguez-Montanés, E.M.J.G. Bruls, and J. Figueras. Bridging defects resistance measurements in a CMOS process. In *Proceedings of International Test Conference*, pages 892–899. IEEE, 1992.
- [SMF85] 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.
- [Syr87] Marek Syrzycki. Modelling of spot defects in MOS transistors. In *Proceedings of International Test Conference*, pages 148–157. IEEE, September 1987.
- [Wad78] R.L. Wadsack. Fault modeling and logic simulation of CMOS and MOS integrated circuits. *Bell System Technical Journal*, 57(5):1449–1474, May-June 1978.
- [WEF+85] 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.
- [WNS87] B.W. Woodhall, B.D. Newman, and A.G. Sannuli. Empirical results on undetected CMOS stuck-open failures. In *Proceedings of International Test Conference*, pages 166–170. IEEE, 1987.