Test Set Reordering Using the Gate Exhaustive Test Metric

Kyoung Youn Cho and Edward J. McCluskey
Center for Reliable Computing
Department of Electrical Engineering
Stanford University, Stanford, CA
kycho@crc.stanford.edu

Abstract

When a test set size is larger than desired, some patterns must be dropped. This paper presents a systematic method to reduce test set size; the method reorders a test set using the gate exhaustive test metric and truncates the test set to the desired size.

To determine the effectiveness of the method, test sets with 1,556 test patterns were applied to 140 defective Stanford ELF18 test cores. The original test set required 758 test patterns to detect all defective cores, while the test set reordered using the presented method required 286 test patterns. The method also reduces the test application time for defective cores.

1. Introduction

Production testing cost is closely related to the test application time and the test set size. In order to reduce these values, test set reordering has been studied. The advantage of test set reordering is based on the following: 1) when a test set is too large to fit in a tester memory the patterns in the later part of the test set can be removed with only minimal impact on defect detection [Pomeranz 04]; 2) the defective chips fail early on a tester, reducing the test application time for the defective chips [Maly 86][Jiang 01][Pomeranz 04].

Experimental results using industrial chips demonstrated that there were big differences in the number of defect detection among single stuck-at fault (SSF) test patterns, and test patterns that detected large number of defective chips were distributed throughout the test set [Nigh 00]. Therefore, test set reordering is important to reduce test set size and to reduce test application time.

An optimal test set reordering method was presented assuming that the defect occurrence probability and the relationship between defects and faults were available [Maly 86]. However, this is not a feasible solution in modern semiconductor designs. Jiang and Vinnakota used the failing data from sample chips; they applied all groups of test patterns to a sample set of ICs, and used the failing data to reorder test sets [Jiang 01]. This method is not cost efficient because it needs a tester. The method also requires a representative sample set of chips because it assumes that faulty behavior doesn't change from lot to lot, which may or may not be true. In another approach, fault simulation for each test pattern is performed, and the number of single stuck-at faults detected by each test pattern is used to reorder test sets [Lin 01]. Other researchers presented their own criteria (or metrics) to select test patterns from an *N*-detect test set [Chao 04][Tian 05]. Test type reordering was also studied; functional test, IDDQ test, delay test, and stuck-at test sets were reordered, and the total test application times were compared [Butler 00].

A *test metric* is used to measure the thoroughness of test sets; when a test set is evaluated using a test metric, the thoroughness of the test set is usually calculated as a coverage value [McCluskey 04a, 04b]. It is also used to guide automatic test pattern generation (ATPG) tools to generate test patterns [McCluskey 04a, 04b]. A test metric can also be used to reorder generated test sets, which is addressed in this paper.

In this paper, a test set reordering method using the gate exhaustive test metric (GEM) is presented and experimental results on actual test chips are reported. The *GEM* evaluates the thoroughness of a test set by calculating the ratio of the number of gate input combinations observed by the test set to the number of observable gate input combinations; the ratio is defined as the *gate exhaustive coverage (GEC)* of the test set [Cho 05].

An *observed gate input combination* of an internal gate is an input combination applied to the gate inputs with the gate output being sensitized to at least one observation point such as a primary output or a scan flip-flop [Cho 05]. An *observable gate input combination* of an internal gate is an input combination that can be applied to the gate inputs with the gate output being sensitized to at least one observation point [Cho 05]. A *nonobservable gate input combination* of an internal gate is an input combination that cannot be applied to the gate inputs (known as "controllability don't care" [De Micheli 94]) or that cannot be sensitized to any observation point (known as "observability don't care" [De Micheli 94]) [Cho 05]. *Gate exhaustive simulation* identifies the gate input combinations observed by a given test set using a simulation of the circuit operation. A *gate exhaustive test set* is a test set that observes all observable gate input combinations at some observation points in a combinational circuit or a full scan sequential circuit [McCluskey 93][Cho 05].

The *SSF test metric* evaluates the thoroughness of a test set by calculating the percentage of target SSFs detected by the test set; the percentage is defined as the *SSF coverage* of the test set. The *SSF model* is a fault model in which only one node is stuck at logic-0 or logic-1 in the circuit under test [Eldred 59][Abramovici 90]. *SSF simulation* identifies the SSFs detected by a given test set using a simulation of the circuit operation.

An *original test set* is a test set that has the sequence of test patterns generated by an ATPG tool. When the sequence of patterns of an original test set is changed, the test set is called a *reordered test set*.

A *surrogate fault model* has been used to represent defects when chip experiments are not available; a fault model that is not targeted during test set generation or test set reordering is used as a surrogate fault model. When the SSF model is used for test generation or test set reordering, the bridging fault model has been used as a surrogate fault model [Kapur 92][Tian 05]. This paper also discusses the correlation between "surrogate" fault coverage and the defect detection. The bridging fault model is evaluated as a surrogate fault model.

This paper is organized as follows: Section 2 presents the test set reordering method using the GEM; Section 3 reports the experimental results that support the effectiveness of the test set reordering method presented in this paper; Section 4 evaluates the bridging fault model as a surrogate fault model. Section 5 discusses the effectiveness of the GEM in detecting defective chips; Section 6 concludes the paper.

2. Test Set Reordering

Cho et al. [Cho 05] reported that a test set with a higher GEC detected more defective chips than a test set with a lower GEC [Cho 05]. The GEC was also shown to be correlated with silicon fallout better than the SSF coverage or the bridge coverage estimate [Guo 06]. In this paper, we present a test set reordering method using the GEM. The test set reordering flow and an example are presented in this section.

Gate exhaustive simulation is conducted on all test patterns without dropping the gate input combinations observed by other test patterns. After the simulation, one test pattern that yields the highest GEC is removed from the original test set and appended to the reordered test set. Then, all gate input combinations observed by the selected test pattern are removed from the list of gate input combinations observed by each test pattern in the original test set. The test pattern selection process finishes when all test patterns are reordered. The reordered test set maximizes the cumulative GEC for each additional test pattern. The *cumulative GEC of test pattern T* is the GEC calculated for the subset comprising all patterns up to and including test pattern T.

An example of test set reordering using the GEM is presented using the ISCAS85 c17 benchmark circuit illustrated in Fig. 1. A gate exhaustive test set for the benchmark circuit was generated using the procedure presented in [Cho 05]; the test set consists of 9 test patterns. Gate exhaustive simulation was conducted on the generated test patterns, and the gate input combinations observed by each test pattern are presented in Table 1. In Table 1, U1/01 denotes the gate input combination AB = 01 to gate U1.

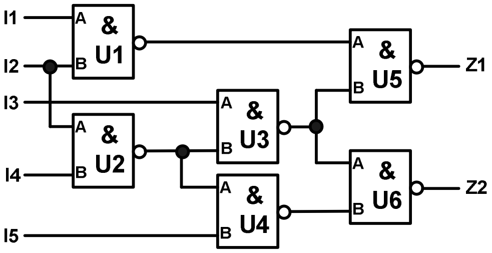

Figure 1. ISCAS85 c17 benchmark circuit

Table 1. Observed gate input combinations

Test pattern	Observed gate input combinations
t_1 (01000)	U1/01, U3/01, U4/10, U5/11, U6/11
t_2 (00110)	U2/01, U3/11 U5/10, U6/01
t_3 (11001)	U1/11, U2/10, U4/11, U5/01, U6/10
t_4 (00000)	U1/00, U3/01, U4/10, U5/11, U6/11
t_5 (11101)	U2/10, U5/00, U6/00
t_6 (01010)	U1/01, U3/00, U4/00, U5/11, U6/11
t_7 (00100)	U2/00, U3/11, U5/10, U6/01
t_8 (01111)	U1/01, U2/11, U3/10, U4/01, U5/11, U6/11
t_9 (10000)	U1/10, U3/01, U5/11, U6/11

In Table 1, test pattern t_8 observes the largest number of gate input combinations in the test set, so test pattern t_8 is selected first. Then, the gate input combinations observed by test pattern t_8 are removed from the list of the gate input combinations observed by other test patterns. One reordered test sequence is (t_8, t_3, t_7, t_4, t_6, t_5, t_9, t_2, t_1). During the process we can find that test pattern t_1 can be removed without reducing the GEC of the test set.

Table 2 shows the SSFs detected by each test pattern, in which U1/Z/1 means the stuck-at 1 fault on the pin Z of gate U1. One sequence of test patterns reordered using the SSF test metric is (t_2, t_8, t_3, t_9, t_1, t_4, t_5, t_6, t_7). During the process we can find that the test patterns t_1, t_4, t_5, t_6, and t_7 can be removed without impacting on the SSF coverage of the test set.

The example demonstrates that the sequence of test patterns reordered using the GEM is different from the sequence reordered using the SSF test metric.

Table 2. Detected single stuck-at faults

Test pattern	Detected single stuck-at faults
t_1 (01000)	Z1/1, I1/1, U3/Z/0, I3/1 Z2/1, I5/1
t_2 (00110)	U5/B/1, Z1/0, I2/1, U3/Z/1, U2/A/1, U2/Z/0, U6/A/1, Z2/0
t_3 (11001)	U1/Z/1, Z1/0, I2/0, U2/Z/0, I4/1, U4/Z/1, Z0/0
t_4 (00000)	Z1/1, U3/Z/0, I3/1, Z2/1, I5/1
t_5 (11101)	Z1/0, U2/Z/0, I4/1, Z2/0
t_6 (01010)	Z1/1, I1/1, U3/Z/0, Z2/1
t_7 (00100)	U5/B/1, Z1/0, U3/Z/1, U2/Z/0, U6/A/1, Z2/0
t_8 (01111)	Z1/1, I1/1, I2/0, U3/B/1, U3/Z/0, U2/Z/1, U4/A/1, Z2/1
t_9 (10000)	U1/B/1, Z1/1, I2/1, U3/Z/0, I3/1, Z2/1, I5/1

The basic procedure of test set reordering is similar to that of test set compaction [Hamzaoglu 00], but they have different goals. The goal of test set compaction is to reduce test set size without any impact on the target fault coverage; the goal of test set reordering is to reduce test set size with minimal impact on defect detection (or the target fault coverage). In the test set compaction using the GEM, only t_1 can be removed because this pattern doesn't observe any unique gate input combination that other test patterns don't observe; the order of test patterns doesn't matter in test set compaction.

3. Experimental Results

In this paper, a gate exhaustive test set is used as an original test set, but any test set can be used as an original test set. This test set was selected because it was the most effective test set we had tested with respect to test escapes and test set size. The gate exhaustive test set was generated using the static pattern fault model provided by Cadence Encounter Test Design Edition [Cadence 03] with maximal compaction option. The detailed method of gate exhaustive test set generation was presented in [Cho 05].

In this experiment, the Stanford ELF18 test chips [Mitra 04] were used. The ELF18 test chips were fabricated using the Philips 0.18 micron Corelib technology and V_{dd} is 1.8V. More than 70,000 test chips were manufactured; each test chip contains 6 R.E.A.L. digital signal processor cores. One ELF18 core contains 13 scan chains; the total number of scan flip-flops is 2,290; the total number of gates for one core is 53,732.

Table 3 reports the test length, the SSF coverage, the GEC, and the test generation time of the gate exhaustive test set for the ELF18 core. The original gate exhaustive test set is denoted as **ge_org**.

Table 3. Original test set information: the ELF18 core

Test set	Test length	SSF coverage	GEC	Test generation time [sec]
ge_org	1,556	99.9%	99.1%	2,311

The original test set was reordered using the method presented in Sec. 2 so that the cumulative GEC for each additional test pattern is maximized (**ge_gem**). To compare the effectiveness of the presented reordering method, the original test set was also reordered so that the cumulative SSF coverage for each additional test pattern is maximized (**ge_ssf**). Table 4 describes the test sets compared in this paper.

Table 4. Test sets compared in this paper

Test set	Description
ge_org	A gate exhaustive test set generated by the Cadence Encounter Test Design Edition ATPG tool
ge_ssf	A test set reordered from **ge_org** to maximize the cumulative SSF coverage for each additional test pattern
ge_gem	A test set reordered from **ge_org** to maximize the cumulative GEC for each additional test pattern

Test set reordering was conducted on a Sun-Fire-V240 workstation running the Solaris 9 operating system. The main memory size was 4 GBytes. Table 5 reports the execution time for each test set reordering. The execution time includes SSF and gate exhaustive simulation time for **ge_ssf** and **ge_gem**, respectively.

Table 5. Execution time to reorder test sets

Test set	Execution time [hh:mm]
ge_org	0:00
ge_ssf	7:19
ge_gem	15:23

Figure 2 shows the difference in the cumulative SSF coverage between **ge_ssf** and **ge_gem**. The figure illustrates that **ge_ssf** has higher cumulative SSF coverage than **ge_gem** for almost all test patterns.

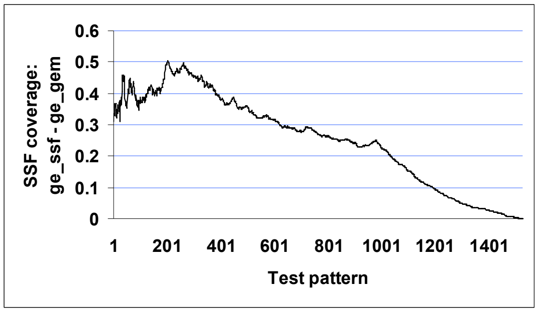

Figure 2. Cumulative SSF coverage comparison: ge_ssf – ge_gem

Figure 3 illustrates the difference in the cumulative GEC between **ge_gem** and **ge_ssf; ge_gem** has higher cumulative GEC than **ge_ssf** for almost all test patterns.

The test sets (**ge_org**, **ge_ssf**, and **ge_gem**) were applied to the ELF18 test cores. In the experiment, each core was tested independently and 140 defective cores that failed the original test set were tested. The test application stops at the first failure, and the first failing cycle was used to calculate the first failing pattern number for each defective core. The pattern number of the first test pattern is 1.

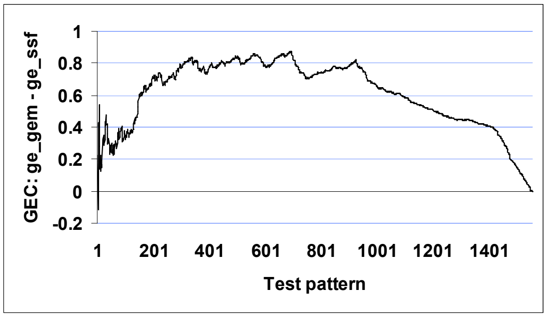

Figure 3. Cumulative GEC comparison: ge_gem – ge_ssf

Figure 4 illustrates the difference in the first failing pattern numbers between **ge_org** and **ge_gem** for each core. The figure demonstrates that **ge_gem** detects many tested cores much earlier than **ge_org**. For some tested cores, **ge_org** detects the cores earlier than **ge_gem**, but the pattern number difference is less than 100.

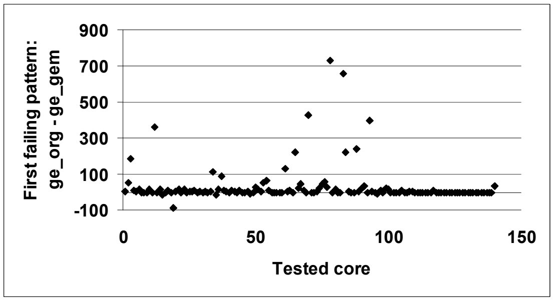

Figure 4. Difference in the first failing pattern numbers: ge_org – ge_gem

Figure 5 describes the difference in the first failing pattern numbers between **ge_ssf** and **ge_gem** for each core. Some tested cores are detected much earlier by **ge_gem** than **ge_ssf**.

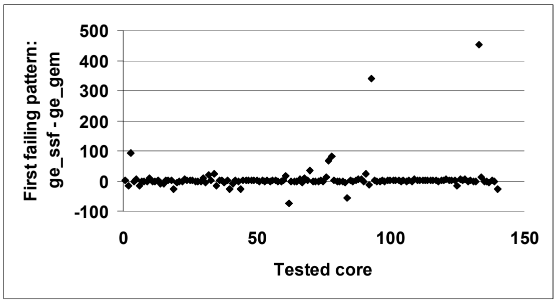

Figure 5. Difference in the first failing pattern numbers: ge_ssf – ge_gem

Table 6 reports the maximum numbers and the average numbers of the first failing patterns for each test set. The maximum number shows how many test patterns can be removed without any impact on defect detection; the average number represents the test application time for the defective cores. The table demonstrates that more patterns can be removed from **ge_gem** than **ge_ssf** with minimal impact on defect detection. This is important when the test set size must be reduced because it doesn't satisfy the required limits; when tester memory size or test application time is limited, test set size must be reduced by dropping some test patterns.

Table 6. The statistics of first failing test pattern number

Test set	**ge_org**	**ge_ssf**	**ge_gem**
Maximum	758	739	286
Average	44.09	18.86	12.69

Figure 6 describes the number of test escapes when only the test patterns up to the number shown in the horizontal axis are applied. In other words, the figure shows the number of defective cores that escape the test set when the latter patterns of the test sets are eliminated. The figure demonstrates that **ge_gem** detects more defective cores than **ge_org** or **ge_ssf** when the test sets are truncated to the same size.

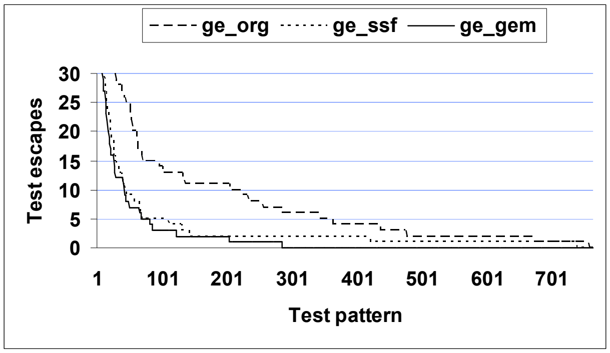

Figure 6. Test escapes for the test sets when the test sets are truncated

In addition to that, the test set reordering method can reduce the test application time. However, the amount of reduction depends on the yield of the fabrication process. Equation 1 calculates the average number of test patterns (*AN*) applied to each chip when test application stops at the first failure. *TL* is the total test length, *Y* is the yield of the fabrication process, and *AF* is the average number of the first failing patterns for defective chips. When the yield is 0.5, the average numbers of test patterns applied to each chip for the **ge_org**, **ge_ssf**, and **ge_gem** are 800, 787, and 784, respectively.

$$AN = TL \times Y + AF \times (1 - Y) \qquad (1)$$

In combinational circuits, test set reordering may affect defect detection of test sets because of sequence dependent defects [McCluskey 04]. The ELF18 core is a full scan circuit, which means that the internal state of the circuit when a test pattern is applied is more determined by the scan shift-in operation than the previous test pattern [Ma 99]. In the application of the three test sets, sequence dependent behavior was not considered.

4. Surrogate Fault Model

In order to compare the effectiveness of the test sets in detecting un-modeled faults or defects, surrogate fault models have been used. In this paper, the bridging fault model is evaluated as a surrogate fault model. The target bridging faults were extracted from the layout data of the ELF18 design and used when the bridging fault simulation was performed. The bridging fault model used in the simulation was the victim-aggressor model [Acken 83] illustrated in Fig. 7; the logic value on the aggressor node changes the logic value on the victim node. The detection of a bridging fault depends on the driving force of gates. In this paper, however, it is assumed that the bridging fault is detected whenever the stuck-at-0 (1) fault on the victim node is detected with logic-0 (1) being assigned to the aggressor node. For example, if the stuck-at-0 fault on the victim node in Fig. 7 (a) is detected with logic-0 being assigned to the aggressor node, the bridging fault is considered to be detected. Bridging fault simulation was conducted using the Synopsys TetraMAX [Synopsys 05].

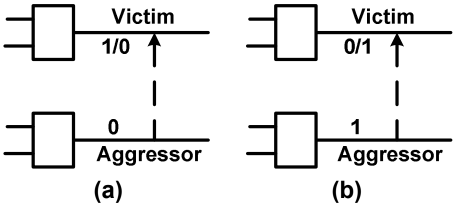

Figure 7. Bridging fault models: (a) Logic-0 on the aggressor node changes logic-1 on the victim node; (b) Logic-1 on the aggressor node changes logic-0 on the victim node

Figure 8 illustrates the difference in the cumulative bridging fault coverage between **ge_ssf** and **ge_gem**. The figure shows that the cumulative bridging fault coverage of **ge_gem** is almost similar to that of **ge_ssf**. Comparing Figs. 6 and 8 demonstrates that there is not a meaningful correlation between the number of detected cores and the cumulative bridging fault coverage in the ELF18 experiments; in other words, the bridging fault model is not a useful surrogate fault model in the ELF18 experiments.

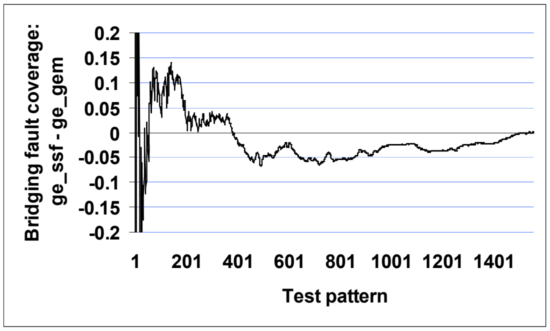

Figure 8. Comparison of cumulative bridging fault coverage: ge_ssf – ge_gem

5. Effectiveness of the the GEM

In this section, the effectiveness of the GEM over the SSF test metric is presented.

To detect more SSFs, at most one controlling value is applied to a gate. For example, if the gate input combination (00) is applied to a two-input NAND gate, all the sensitization paths through the gate are blocked; this will reduce the number of detected SSFs. However, this gate input combination may be effective in detecting some types of defects. Let us consider the circuit illustrated in Fig. 9. Applying the (00) combination to NAND gate K assigns a strong logic-1 to the gate output; the strong logic-l may change the logic value on the node connected by bridging defects. The bridging defects may not be detected, if (01) or (10) combination is applied to the inputs of gate K.

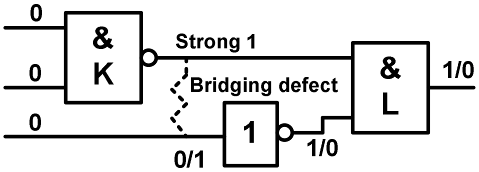

Figure 9. Bridging defect detection

Another example that supports the effectiveness of the GEM is explained using the circuit presented in Fig. 10. In SSF test generation, the stuck-at 1 fault on the output of gate L will be propagated to Z1 because it is an easy path to propagate the effect of the SSF; however, applying the (001) combination to the inputs of gate L will block the sensitization path to Z1 at gate M, and propagate the fault signal to Z2 through another path, increasing the defect detection probability by detecting the SSF with different conditions from the SSF test generation.

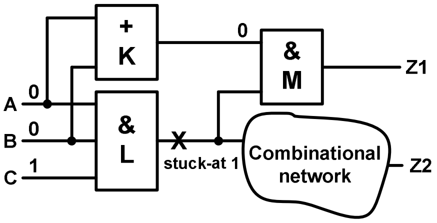

Figure 10. Propagation of fault signal

Applying some gate input combinations to a gate may detect specific types of defects inside the gate that are not guaranteed to be detected by detecting only SSFs. Let us consider the transistor level implementation of a two-input AND gate with the bridging defect between internal node N and the gate output as described in Fig. 11. When the PMOS transistors M1 and M2 turn on simultaneously, logic-1 may be assigned to the gate output in stead of logic-0 because of the strong logic-1 assigned to internal node N, detecting the bridging defect; however, the bridging defect may not be detected by applying (01) or (10) combination because the NMOS transistor M6 may drive the output to logic-0. However, the detection depends on the relative driving strength of transistors. The examples demonstrate that observing more gate input combinations increases the possibility of defect detection.

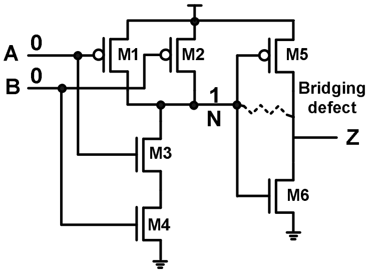

Figure 11. A transistor level implementation of two-input AND gate with a bridging defect

6. Conclusions

This paper presents a test set reordering method; the method reorders a test set using the gate exhaustive test metric and truncates the test set to the desired test set size. The reordering was applied to a test set generated with a maximal compaction and reordering supported by an ATPG tool. Experimental results using the Stanford ELF18 test chips demonstrated that the test set reordered using the gate exhaustive test metric could be truncated with less impact on defect detection than the test set reordered using the SSF test metric. The method also reduces test application time for defective chips more than the method using the SSF test metric.

The test set reordering method presented in this paper is cost efficient because it can be implemented using computer simulation without any experiment on a tester.

The results also demonstrate that the bridging fault model is not a useful surrogate fault model in the ELF18 experiments. We propose the GEM as a test metric that can be used to measure the thoroughness of test sets.

Acknowledgments

The research was supported by NSF under contract number CCR-0098128 and LSI Logic. The ELF18 test chip experiments are supported by Philips Semiconductors.

The authors would like to express their thanks to Bill Price, Stefan Eichenberger, and Fred Bowen of Philips Semiconductors, Erik Volkerink, Intaik Park, and Francois-Fabien Ferhani of Stanford CRC for their support for the ELF18 experiments.

References

[Abramovici 90] Abramovici, M., M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, New York, 1990.

[Acken 83] Acken, J. M., “Testing for Bridging Faults (Shorts) in CMOS Circuits,” *Proc. Design Automation Conf.*, pp. 717-718, 1983.

[Butler 00] Butler, K. M. and J. Saxena, “An Empirical Study on the Effects of Test Type Ordering on Overall Test Efficiency,” *Proc. Intl. Test Conf.*, pp. 408-416, 2000.

[Cadence 03] Encounter Test Design Edition User Guide, Cadence Design Systems, Inc., Oct. 2003.

[Chao 04] Chao, C.-T., L.-C. Wang, and K.-T. Cheng, “Pattern Selection For Testing Of Deep Sub-Micron Timing Defects,” *Proc. Design Automation & Test in Europe*, pp. 1060-1065, 2004.

[Cho 05] Cho, K. Y., S. Mitra, and E. J. McCluskey, “Gate Exhaustive Testing,” *Proc. Intl. Test Conf.*, paper 31.3, 2005.

[De Micheli 94] De Micheli, G., *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.

[Eldred 59] Eldred, R. D., “Test Routines Based on Symbolic Logical Statements,” *Journal of the ACM*, vol. 6, no. 1, pp. 33-36, 1959.

[Guo 06] Guo, R., *et al.*, “Evaluation of Test Metrics: Stuck-at, Bridge Coverage Estimate and Gate Exhaustive,” *Proc. VLSI Test Symp.*, pp. 66-71, 2006.

[Hamzaoglu 00] Hamzaoglu, I. and J. H. Patel, “Test Set Compaction Algorithms for Combinational Circuits,” *IEEE Trans. on CAD*, vol. 19, no. 8, pp. 957-963, Aug. 2000.

[Jiang 01] Jiang, W. and B. Vinnakota, “Defect-Oriented Test Scheduling,” *IEEE Trans. VLSI Systems*, vol. 9, no. 3, pp. 427-438, Jun. 2001.

[Kapur 92] Kapur, R., J. Park, and M. R. Mercer, “All Tests for a Fault are Not Equally Valuable for Defect Detection,” *Proc. Intl. Test Conf.*, pp. 762-769, 1992.

[Lin 01] Lin, X, J. Rajski, I. Pomeranz, and S. M. Reddy, “On Static Test Compaction and Test Pattern Ordering for Scan Designs,” *Proc. Intl. Test Conf.*, pp. 1088-1097, 2001.

[Ma 99] Ma, S., I. Shaik, and R. S. Fetherston, “A Comparison of Bridging Fault Simulation Methods,” *Proc. Intl. Test Conf.*, pp. 587-595, 1999.

[Maly 86] Maly, W., “Optimal Order of the VLSI IC Testing Sequence,” *Proc. Design Automation Conf.*, pp. 560-566, 1986.

[McCluskey 93] McCluskey, E. J., “Quality and Single-Stuck Faults,” *Proc. Intl. Test Conf.*, p. 597, 1993.

[McCluskey 04a] McCluskey, E. J., et al., “ELF-Murphy Data on Defects and Test Sets,” *Proc. VLSI Test Symp.*, pp. 16-22, 2004.

[McCluskey 04b] McCluskey, E. J., “Digital IC Testing for Art Historians and Test Experts,” ICCD’04 presentation, http://crc.stanford.edu/iccd_ks.pdf.

[Mitra 04] Mitra, S., E. H. Volkerink, E. J. McCluskey, and S. Eichenberger, “Delay Defect Screening using Process Monitor Structures,” *Proc. VLSI Test Symp.*, pp. 43-52, 2004.

[Nigh 00] Nigh, P. and A. Gattiker, “Test Method Evaluation Experiments & Data,” *Proc. Intl. Test Conf.*, pp. 454-463, 2000.

[Pomeranz 04] Pomeranz, I. and S. M. Reddy, “On Maximizing the Fault Coverage for a Given Test Length Limit in a Synchronous Sequential Circuit,” *IEEE Trans. Computers*, vol. 53, no. 9, pp. 1121-1133, Sep. 2004.

[Synopsys 05] TetraMAX ATPG User Guide, Synopsys, Inc., Jan. 2005.

[Tian 05] Tian, Y., M. R. Grimaila, W. Shi, and M. R. Mercer, “An Optimal Test Pattern Selection Method to Improve the Defect Coverage,” *Proc. Intl. Test Conf.*, Paper 31.2, pp. 762-770, 2005.