A Novel Method to Improve the Test Efficiency of VLSI Tests

Hailong Cui and Sharad C. Seth [*]
Dept. of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115
(hlcui,seth)@cse.unl.edu

Shashank K. Mehta
School of Information Technology
Indian Institute of Technology, Bombay
shashank@it.iitb.ac.in

Abstract

This paper considers reducing the cost of test application by permuting test vectors to improve their defect coverage. Algorithms for test reordering are developed with the goal of minimizing the test cost. Best and worst case bounds are established for the performance of a reordered sequence compared to the original sequence of test application. SEMATECH test data and simulation results are used throughout to illustrate the ideas.

1 Introduction

The cost of VLSI testing can be divided into four parts: test preparation, test execution, test silicon, and test quality [12]. The decision about whether to adopt design for test (DFT) can affect all the four parts, as indeed, many other aspects of design and test. Previous cost analyses have primarily focused on the silicon cost and the benefits derivable from automation of DFT [1, 3, 12]. However, once the decision to adopt DFT has been made and test vectors have been generated, test execution time remains as the only source of test cost optimization. We focus on this aspect of optimization in this paper.

We call the failing vector of a defective device as the test length of the device. An optimized test execution should feature a short average test length. Efforts to reduce the average test length can be classified according to how much actual tester data is available.

(1) No tester data is available. This is the case in the early phase of the design cycle prior to fabrication. Gate level design and chip layout information is available for test optimization.

[*]Contact author, phone (402) 472-5003, fax: 472-7767. This material is based upon work supported by the National Science Foundation Grant No. 9806795.

(2) Incremental test information is available as in the normal test process of fabricated chips. In this case, test vectors (or test) are applied only to the devices that did not fail previously.

(3) Full pass/fail information is available for every vector of each test, and for each device. This information is expensive and collected only for a small number of devices for failure mode analysis (FMA).

Other test data related to fault coverage and yield modeling may be used to optimize the test as well.

Test optimization based on item (1) includes test set compaction techniques, such as reverse order test compaction [8]. Test compaction methods do not take into account the likelihood of defects and may be subjected to further optimization with the techniques presented in this paper.

Test optimization based on item (3) was proposed in [6]. This method however, requires full knowledge of defect occurrence probability and the relationship between modeled faults and physical defects. To acquire this information, significant feedback is needed from the manufacturing process. Special simulation tools, such as yield simulator [11], may also be needed ,which makes it difficult to accommodate the idea into commercial testers. The results may not be accurate since there are potentially a huge number of defects and the fault defect relationship can be complex.

In this paper we focus on test optimization based on item (2) above. As normal testing proceeds, the tester data provides valuable information about defect occurrence and coverage that can be exploited to improve the efficiency of the test patterns. This information is not available during early design cycle and, unlike the FMA data, is routinely available during the production phase. Since it only requires the information during normal test process, the method proposed in this paper is much easier to implement in commercial testers as compared to the method proposed in [6].

2 Test Cost and Efficiency

We assume that the cost of testing a device is proportional to its test length. The execution cost of test T is then closely related to the rising of the chip fallout curve. To see this, assume there are N chips and the test set consists of M vectors. Let V_i be the number of vectors applied to chip $i, i = 1...N$. For a good chip, $V_i = M$ and for a bad chip, $1 \leq V_i \leq M$. Let F_j be the total number of chip failures up to and including vector j, then the *total test cost*, ignoring the constant of proportionality, can be computed in two different ways:

$$\sum_{i=1}^{N} V_i = \sum_{j=0}^{M-1} (N - F_j) \tag{1}$$

where $F_0 \triangleq 0$. Each item in the summation corresponds to the $N - F_j$ chips that have gone though the $j + 1$ tests. The summation represents the area above the test failure curve as shown in Figure 1. Hence we also refer to the test failure curve as the test cost curve and a reduction in the area above the cost curve would translate into a reduction of test cost for test T.

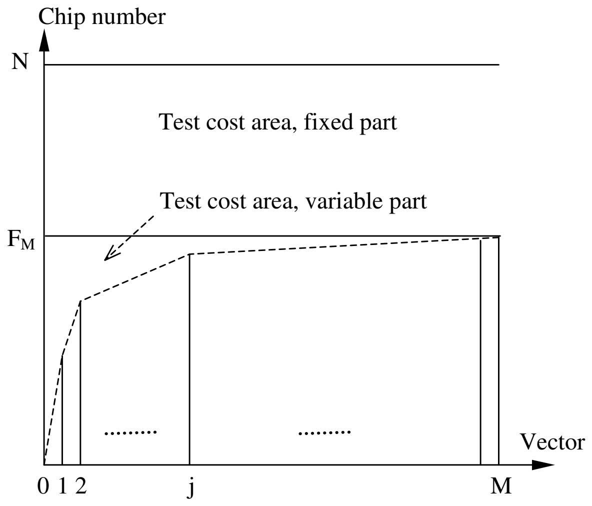

Figure 1. The test failure curve as the test cost curve. The cost area includes two parts: the fixed part for the good chips, and the variable part for the defective chips. The latter varies for different order of test-vector application.

Based on the test cost, we define the *efficiency* of a test as:

$$E(T) = \frac{\sum_{j=0}^{M-1} F_j}{M \cdot F_M} \tag{2}$$

The efficiency corresponds to the area between x-axis and the chip failure curve as a fraction of the area between the line $y = F_M$ and the x-axis, as shown in Figure 1. The efficiency ranges between $\frac{1}{M}$ (when all defective parts fail at the last vector) and 1 (when all defective parts fail right at the the first vector). For the same test set, a high efficiency test would have a chip failure curve with a sharp rise. Note that the test efficiency is independent of the empirical yield.

We know that the cost curve depends on the defect coverage and the defect occurrence probability. To simplify analysis, defects are often assumed to have equal occurrence probability [2, 9, 13]. Under this assumption, the cost curve would depend only on the defect coverage.

3 Improving Test Efficiency

Given test failure data with the *original sequence* of test-vector application, in this section, we derive ways of reordering the test vectors so as to produce a test sequence with a higher efficiency. Further, we derive lower and upper bounds of improvement in the cost curve over the original sequence.

Assume that the number of defects occurring on a chip follows a negative-binomial distribution [10]. Then the probability that a chip will fail at defect coverage f, following the work in [2], would be:

$$P(f) = 1 - (a \cdot f + 1)^{-b} \tag{3}$$

Let the incremental defect coverage of vector v_i be $\triangle f_i$ and of vector v_{i+1} be $\triangle f_{i+1}$. Let the probabilities of new detection (of defective chips) for vector v_i and v_{i+1} be $p_{new,i} = P(f + \triangle f_i) - P(f)$ and $p_{new,i+1} = P(f + \triangle f_i + \triangle f_{i+1}) - P(f + \triangle f_i)$ respectively. Then, by the convexity property of Equation 3

$$p_{new,i} > \left.\frac{dp}{df}\right|_{f+\triangle f_i} \cdot \triangle f_i$$

and

$$p_{new,i+1} < \left.\frac{dp}{df}\right|_{f+\triangle f_i} \cdot \triangle f_{i+1}$$

Thus, if vector v_{i+1} detects more failures than vector v_i (indicating $p_{new,i} \leq p_{new,i+1}$) then

$$\left.\frac{dp}{df}\right|_{f+\triangle f_i} \cdot \triangle f_i < \left.\frac{dp}{df}\right|_{f+\triangle f_i} \cdot \triangle f_{i+1}$$

Therefore, $\triangle f_i < \triangle f_{i+1}$, i.e. v_{i+1} has more incremental defect coverage than vector v_i and a swap of the positions of vector v_i and v_{i+1} is justified. Note that for the empirical values to converge to the true means, a sufficiently large sample of tester data must be collected before attempting reordering of test vectors.

3.1 The Swap Algorithm

The Swap algorithm employs the idea just presented for reordering of test vectors. Before stating the algorithm, we introduce the necessary definitions.

M, as before, is the number of vectors and S_i is the set of new defective chips that fail at vector v_i in the original sequence. Clearly, for $i \neq j, S_i \cap S_j = \phi$. Let $|(S_i)|$ be the number of elements in S_i.

S_i's are the elements of interest in the following discussion. Define $\Omega = \{S_i | i = 1...M\}$, then, associated with each vector v_i, there is a high bound set $H_i \subseteq \Omega$ and a low bound set $L_i \subseteq \Omega$. H_i contains the S_i's denoting chips that can *potentially fail* at vector v_i. L_i contains the S_i's denoting chips that *must fail* at vector v_i.

At the beginning $H_i = L_i = \{S_i\}$. For any set $X \subseteq \Omega$ define function $f(X)$ as:

$$f(X) = \sum_{S_i \in X} |(S_i)| \tag{4}$$

which gives total number of chips in set X. Therefore $f(H_i)$ gives the number of chips which can potentially fail at vector v_i and $f(L_i)$ gives the number of chips which must fail at vector v_i.

```
1. Initialization:
      for $i = 1$ to $M$ do
      BEGIN
            $H_i = L_i = \{S_i\}$ ;
      END
2. Swap Process:
      for $j = M - 1$ downto 1 do
      BEGIN
            for $i = 1$ to $j$ do
            BEGIN
                  if $f(H_i) < f(L_{i+1})$ then
                  BEGIN
                        swap_vector($v_i$, $v_{i+1}$);
                        $H_i^{old} = H_i$;
                        $H_i = H_i^{old} \cup H_{i+1}$;
                        $L_i = L_{i+1}$;
                        $L_{i+1} = \phi$;
                        $H_{i+1} = H_i^{old}$;
                  END
            END
      END
```

Figure 2. The Swap Algorithm

The Swap algorithm (Figure 2) is based on comparing the values of $f(H_i)$ and $f(L_{i+1})$ at each step. If $f(H_i) < f(L_{i+1})$ then vectors v_i and v_{i+1} are swapped, i.e. $v_i^{new} = v_{i+1}$ and $v_{i+1}^{new} = v_i$. The condition for swapping ensures that v_i^{new} will detect more defective chips than the old vector v_i.

After the swap, it is also necessary to adjust the high and low bound sets associated with the swapped vectors. The basic idea behind this adjustment is that when a vector is swapped to appear earlier in the test sequence, it is guaranteed still to fail all the defective chips it did before the swap; in addition, it may fail some (or all) of the defective chips failed by the other vector. Hence the low bound set of v_{i+1}^{new} is the empty set ϕ; the low bound set of v_i^{new} and the high bound set of v_{i+1}^{new} remain unchanged; but the high bound set of v_i^{new} is the union of the high bound sets of the two vectors.

The Swap algorithm produces a new sequence of test vectors. When this test sequence is applied to the same chips again, the cost curve will rise more quickly than the old sequence and will have a higher test efficiency. The fault coverage curve for the new sequence is also expected to rise faster as indicated by the simulation results in section 4.

The Swap algorithm is similar to the bubble sort. The only difference is that after each swap, the key values for comparison are changed in the swap algorithm while in the bubble sort they remain the same. This means the final sequence does not necessarily correspond to a sort according to the key values S_i's.

Example: Assume we have a test sequence of 8 vectors $(v_1, v_2, ..., v_8)$ with the incremental detections of $6, 13, 5, 9, 10, 2, 3, 4$ chips respectively. For the first inner loop, since $f(H_1) = 6, f(L_2) = 13$, vector v_1, v_2 are swapped. Now $H_1 = \{S_1, S_2\}, L_1 = \{S_2\}$and $H_2 = \{S_1\}, L_2 = \phi$. Next, vector v_1 in the $2nd$ position is compared with the $3rd$ vector v_3. Since now $f(H_2) = 6$ and $f(L_3) = 5$, no swap is made. Then the $3rd$ and $4th$ vector are compared, as so on. After the first round of the inner loop, the new sequence would be $v_2, v_1, v_4, v_5, v_3, v_7, v_8, v_6$. In the second round, vector v_1 in the $2nd$ position, is first swapped with v_4, then compared and swapped with v_5. No other changes are made thereafter and the final sequence is $v_2, v_4, v_5, v_1, v_3, v_7, v_8, v_6$.

3.2 Performance of Any New Sequence

For any reordered sequence we need to know how well it will perform. This can be done by rerunning the test on the same chips with the reordered sequence. However, we show that it is possible to determine the best and worst cases of the cost curve for any reordered sequence without rerunning the test. First we need an algorithm to determine the high and low bound sets of vectors for any sequence.

Suppose $(v_{i_1}, v_{i_2}, ..., v_{i_m})$ is the new sequence where i_j indicates the original position of the j-th vector. For vector v_{i_j}, v_{i_k} with $j < k$, if $i_j < i_k$, v_{i_j} comes before vector v_{i_k} in the original sequence, therefore, $S_{i_k} \notin H_{i_j}$ since v_{i_k} is applied after v_{i_j} in both sequence. We indicate this by saying there is no *swap* event between v_{i_j} and v_{i_k}.

If $i_j > i_k$ we know vector v_{i_k} was applied originally before v_{i_j} but in the new sequence it is applied after v_{i_j}, therefore v_{i_j} can potentially detect failures of vi_k. Hence, $S_{i_k} \in H_{i_j}$ and the lower bound set L_{i_k} is set to ϕ since potentially vector v_{i_k} may not detect any failure in the new sequence. In this case we say there is a *swap* event between v_{i_j} and v_{i_k}.

```
1. Initialization:
        for $t = 1$ to $M$ do
        BEGIN
                $H_t = \{S_{i_t}\}$ ;
        END
2. The high bound:
        for $j = 1$ to $M - 1$ do on the new sequence
        BEGIN
                for $k = j + 1$ to $M$ do on the new sequence
                BEGIN
                        if ( $i_j > i_k$ )
                                $H_j = H_j \cup \{S_{i_k}\}$;
                                $L_k = \phi$;
                END
        END
```

Figure 3. Algorithm to determine high and low bound sets of an arbitrarily reordered sequence

The algorithm, given in Figure 3 first initializes the high bound set of each vector in the new sequence with the S_i's in the original sequence, then it examines each vector v_{i_j} with all the following vectors v_{i_k} with $k > j$. Only in case there is a *swap* event, i.e. $i_j > i_k$, we set $H_j = H_j \cup S_{i_k}$ and $L_{i_k} = \phi$.

Once we have the high and low bound for each vector, the performance of the new sequence in the best case can be derived by checking the high bound set of the vectors, i.e. each S_x is assigned to be detected by the k_{th} vector of the new sequence where $k = min(\{t|S_x \in H_t\})$. Similarly, for the performance in worst case, each S_x is assigned to be detected by the j_{th} vector where $j = max(\{t|S_x \in H_t\})$. For the worst case, it can be readily seen that each vector detects its original failure set, i.e. vector v_{i_j} detects and only detects the chips in S_{i_j}.

One natural question is, if we do a sort of the vectors based on the incremental failures of the original test sequence, how would the new sequence perform? First, as already indicated, the sort sequence may differ from the swap sequence. Second, in the worst case, each vector in the new sequence detects its original chip set, therefore the sequence based on sorting will have the best performance in the worst case over all other sequences.

Following the previous example, the sequence for sort is v_2,v_5,v_4,v_1,v_3,v_8,v_7,v_6. Now, in the best case, v_2 detects $f(H_1) = |S_1| + |S_2|$ failures. Similarly, v_5 will detect all failures in S_3, S_4, and S_5; v_4, v_1, and v_3 will not detect any failures as their high bound sets are already covered by the earlier test vectors; v_8 will detect all failures in S_6, S_7, and S_8; v_7 and v_6 will again not detect any failures. In the worst case the failure sets are given simply by the S_i corresponding to each v_i.

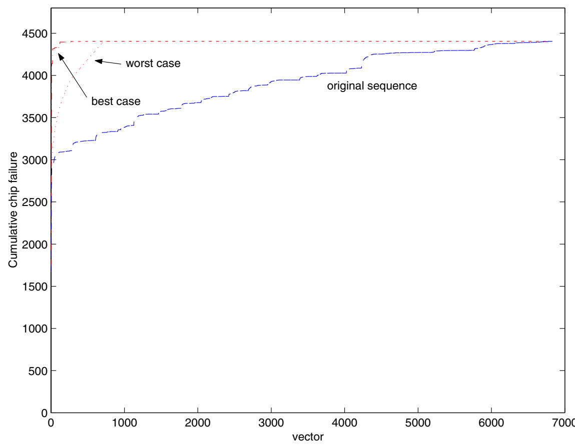

Figure 4. For the SEMATECH test data the sorted sequence shows much better performance than the original sequence even in the worst case.

With Equation 1, the test efficiency in the best and worst cases can then be computed for any new sequences. We derived new swap and sort sequences for the SEMATECH scan-test data [7]. The efficiency of the original test is 88.8%. For the swap algorithm, $98.8\% \leq E(swap) \leq 99.3\%$, which is an 11.3% to 11.8% increase in efficiency. Similarly, for the sort algorithm, $98.9\% \leq E(sort) \leq 99.9\%$, i.e., an 11.4% to 12.5% increase over the original sequence. It can be seen that the sort sequence is slightly better than the Swap sequence in the worst case. The best case for the sort algorithm, however, might be too optimistic unless the faults are highly-clustered (see Section 4). Figure 4 shows the performance of the sort sequence in the best and worst cases for SEMATECH scan-test data.

The results establish upper and lower bounds for the cost curve. The actual curve for any sequence is expected to lie between these two bounds. However, the SEMATECH data does not provide enough information to compute the actual curve for an arbitrarily reordered sequence. Hence a simulation study, described in the next section, was conducted to understand the factors that determine the placement of the actual curve with respect to the best and worst cases.

4 A Simulation Study

The simulation was based on a scan version of the benchmark circuit $s38417$. Atalanta [4] is used as the ATPG tool and Hope [5] is used as the fault simulator. This scan version circuit contains 31180 single stuck-at faults, and 165 faults are aborted. As defect level is not an issue here we ignore the aborted faults. Atalanta generated 1210 deterministic vectors for the non-aborted faults. This sequence of vectors is hereafter referred to as the original sequence.

The elements of interest in the simulation include the deterministic vectors, chips, defects that may happen on a chip, and faults caused by the defects. As the defects cannot be known exactly, we simply assume a large number (5 times the number of faults) for them.

The overall setup of the simulation is as follows. For each chip, we determine if it is defective by a Bernoulli trial according to the yield. If it is defective, the number of defects occurring on it is determined according to a negative binomial distribution and this number of defects are randomly selected to occur on this chip. A predetermined relationship between faults and defects is established by assuming a Poisson distribution, with mean c, for the number of faults caused by a defect [9]. Finally, by fault simulation we determine whether or not a defective chip is detected by a vector. Further details of the simulation process are being omitted because of space limitations.

4.1 Simulation Results

Simulations were done to compare the performance of the sort sequence vs. the original sequence for two cases of fault clustering: high ($c = 20$) and low ($c = 2.8$). The results show (Figure 5) that the actual curves are well above the worst case in both the cases and well below the best case for low fault clustering. Both fault

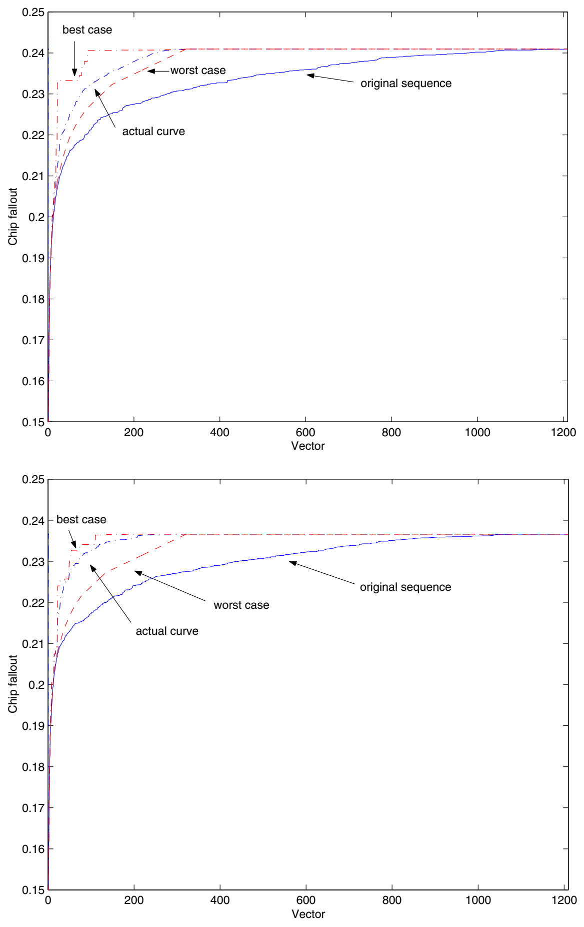

Figure 5. Simulation results for low clustering (top, $c = 2.8$) and high clustering (bottom, $c = 20$) of faults. Actual cost curves for the sort sequence are well above the original curve in the worst case, and may approach the curve in the best case in high fault clustering case.

and defect coverages show significant improvement over the original sequence as shown in Figure 6 and Figure 7. In the high fault clustering case, the improvement of defect coverage is even higher (the figure is omitted due to space limitations). This confirms our analysis at the beginning of Section 3 that the reordered sequence would have sharper defect coverage increase.

Based on the above discussion, a test optimization procedure is proposed: (1) Chips are tested with the original test order derived from the test generator. (2) After G chips have been tested, the test vectors are sorted according to the number of their chip failures. And the new order are applied to the next G chips.

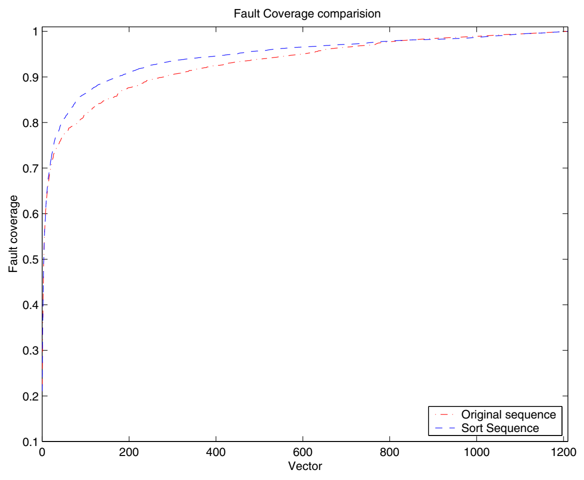

Figure 6. Simulation results: improvement of the fault coverage curve for the sort sequence ($c = 2.8$).

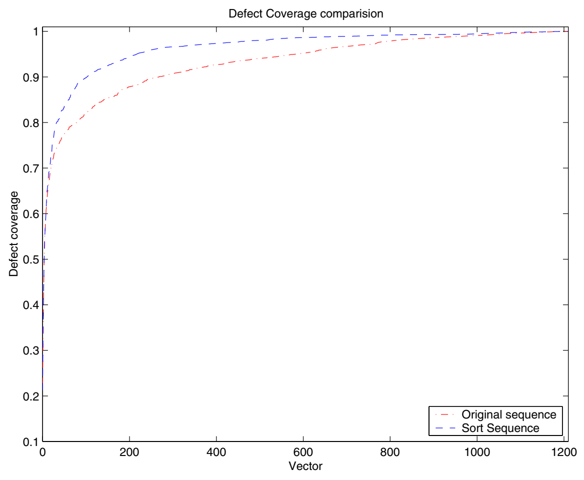

Figure 7. Simulation results: improvement of defect coverage curve for the sort sequence ($c = 2.8$).

5 Conclusion

We analyzed test-vector reordering as a way to reduce the test cost and established the best and worst case performance obtainable by a reordered test sequence. Test efficiency was defined as a yield-independent measure of performance under reordering. For the SEMATECH scan tests, it was shown that the test efficiency could be improved by at least 10% by reordering. While the results are shown for scan tests, they apply equally to non-scan circuits as long as independent test sub-sequences are considered for reordering.

Acknowledgment: The authors wish to thank Mr. Phil Nigh of IBM, Burlington, Vermont, for providing the SEMATECH data.

References

[1] T. Ambler, B. Bennetts, H. Bleeker, and G. O'Donnell. The economics of design for test: Applications clarify the cost of test. *Evaluation Engineering*, 33(11), pp. 22–23, 26, 1994.

[2] J. T. de Sousa and V. D. Agrawal. Reducing the complexity of defect level modeling using the clustering effect. *Proc. Meeting on Design Automation and Test in Europe*, pp. 640–644, 2000.

[3] T. Gheewala, C. E. Stroud, D. J. Burns, N. E. Donlin, and C. C. Packard. Acceptance barriers confronting DFT and BIST. *Proc. IEEE Int. Test Conf.*, pp. 1111–1116, 1991.

[4] H. K. Lee and D. S. Ha. On the generation of test patterns for combinational circuits. Technical Report 12, Dept. of Electrical Eng., Virginia Polytechnic Institute and State University, 1993.

[5] H. K. Lee and D. S. Ha. Hope: An efficient parallel fault simulator for synchronous sequential circuits. *IEEE Trans. on Computer-Aided Design*, 15(9), pp. 1048–1058, 1996.

[6] W. Maly. Optimal order of the VLSI IC testing sequence. *Proc. 23rd ACM/IEEE Design Automation Conf.*, pp. 560–566, 1986.

[7] P. Nigh, W. Needham, K. Butler, P. Maxwell, R. Aitken, and W. Maly. So what is an optimal test mix? a discussion of the SEMATECH methods experiment. *Proc. IEEE Int. Test Conf.*, pp. 1037–1038, 1997.

[8] I. Pomeranz, L. Reddy, and S. Reddy. ROTCO: A reverse order test compaction technique. *Proc. of IEEE EURO-ASIC Conf.*, pp. 189–194, Sept. 1992.

[9] S. C. Seth and V. D. Agrawal. Characterizing the LSI yield equation from wafer test data. *IEEE Trans. Computer-Aided Design.*

[10] C. H. Stapper, F. M. Armstrong, and K. Saji. Integrated circuit yield statistics. *Proc. of the IEEE*, 71(4), pp. 453–470, 1983.

[11] D. M. H. Walker. *Yield simulation for integrated circuits.* Kluwer Academic Publishers, 1987.

[12] S. Wei, P. K. Nag, R. D. Blanton, A. Gattiker, and W. Maly. To DFT or not to DFT? *Proc. IEEE Int. Test Conf.*, pp. 557–566, 1997.

[13] T. W. Williams. Test length in a self-testing environment. *IEEE Design & Test of Computers*, 2(2), pp. 59–63, 1985.