

# Using Multiple Compacted Responses to Diagnose Scan Response Errors during Testing

Steven S. Lumetta\*  
Department of ECE  
University of Illinois  
lumetta@uiuc.edu

## Abstract

*Scan test vector and response volume are becoming problematic, and in industrial designs are complicated by the presence of unknown values in test responses. Recent work has addressed this problem by devising X-tolerant codes that allow both compaction of test responses and guaranteed detection of errors despite the presence of unknown response values. The X-MISR scan compaction architecture [Mitra04] shows how random codes can be generated on-chip and used as X-tolerant codes to provide a single testing architecture that can be tuned to the needs of the chip design and the ability to remove unknowns in test responses without change to the architecture itself, yet provides several orders of magnitude of test response compaction. The architecture can take advantage of compaction in both space and time.*

*In this paper, we address the problem of using the compacted test response from such a stochastic system to identify the error syndrome of the response without scanning the whole response out of the chip. Effectively, this technique allows testing to focus on problematic areas of a chip by simply running an erroneous test vector a small number of times during testing (the compacted response contains sufficient information to identify such vectors). As the codes used during each run are random, the compacted responses provide independent information about the error syndrome of the chip, often allowing the scan cell or cells in error to be identified uniquely. Some types of diagnostic data can thus be gathered during testing at a small cost in additional test response volume (two to six runs typically suffice).*

---

\*Paper based on work performed in Sept. and Oct. 2004 and funded in part by National Science Foundation grant ACI-99-84492-CAREER. The work was also graciously supported by Intel Corporation. The content does not necessarily reflect the position or the policy of either organization. The author wishes to thank Subhasish Mitra, Ed McCluskey, and Janak Patel for their support and for their helpful comments.

## 1 Introduction

Scan design for testability (DFT) techniques are now used in whole or part for most chip designs, but the increasing complexity and area of these designs has raised concerns about both test time and data volume. A substantial body of work has addressed these issues, providing methods both for compaction of test vectors (e.g., Illinois Scan [Hamzaoglu99]) as well as test responses. A number of Built-in Self Test (BIST) techniques can also help with these problems, either directly by covering some of the testing on-chip, or indirectly through the reuse of mechanisms such as Multiple Input Signature Registers, which can also be used to compact scan test responses.

One major problem with compaction of scan test responses has been the presence of unknown values, denoted X's, in the responses. These unknowns often arise from difficulties in modeling circuit elements, inaccuracies of simulation, and interactions between clock domains. Until recently, the primary means of dealing with X's was to ignore (i.e., not compare) any output affected by them. In the last few years, a number of groups have proposed methods for handling X's more effectively.

This paper builds on the X-tolerant compaction architecture presented in [Mitra04], which uses stochastic coding techniques to provide a testing architecture that can be tuned to match both the necessary level of X tolerance and the demands of the testing environment without changing the architecture.

In this paper, we consider the problem of identifying which bits from a single test response are in error, and which are correct, without sacrificing test volume compaction. The core idea is to leverage the stochastic nature of the X-tolerant testing architecture to obtain independent samples of the error bits. Initially, chips are tested using a standard set of test vectors. The base testing architecture, called an X-MISR, provides enough information in its compacted output to distinguish failing test vectors.

When many chips fail for certain vectors, the manufacturer can then extend the test set to include multiple copies of the failing vectors, each of which provides an independent random sample of the scan cell errors in a given chip. A small number of these random samples usually suffices to uniquely identify the scan cell or cells in error, allowing this identification at only a fraction of the volume required to scan out the whole response vector, and without using a separate, post-test diagnosis process for every chip. In practice, this technique allows a manufacturer to diagnose large numbers of failing chips rapidly through a small increase in test time and test response volume. The test input volume need not actually increase, as the samples merely involve repeating the test inputs of interest.

The remainder of the paper is organized as follows. In the next section, we provide more information on background materials, particularly on the X-MISR architecture that we build upon in this paper. Section 3 describes a simple algorithm for diagnosis and provides simulation results to illustrate the algorithm’s effectiveness. We discuss possible extensions to the algorithm in Section 4, and present our conclusions in Section 5.

## 2 Background

A variety of mechanisms have been suggested to address the problem of dealing with unknown values in test responses. Methods such as that proposed in [Naruse03] use on-chip linear feedback shift registers (LFSRs) with carefully selected seeds along with hardware support to force unknown response values to 0 or 1. When the number of unknowns is small, it is possible to use standard erasure coding techniques [Patel03], although with current automatic test equipment, this approach prevents an immediate decision on chip failure by a bit line test processor.

In this paper, we use the X-code approach introduced by [Mitra02]. An X-code is a binary linear code mapping a large number of test response bits, some of which are X’s, into a small number of compacted response bits. As with any code, an X-code is designed to ensure that errors in a few input bits always produce errors in some output bits. Outputs that depend on X’s in the input can not be compared with an expected response, however, and the key difference between an X-code and a standard code is that the X-code always propagates errors to at least one output not affected by the X’s.

The first few papers on this subject [Mitra02, Rajski03, Wang03, Wohl03] proposed codes that could only handle a small number of X’s. A deterministic code can

be guaranteed to produce an error for some fixed maximum number of X’s and fixed maximum number of errors. However, the numbers of bits thus guaranteed is typically small, and codes that guarantee the X-code property for large numbers of bits are practically infeasible.<sup>1</sup>

X-codes are much more powerful when used to provide only a probabilistic guarantee. The X-tolerant MISR design introduced in [Mitra04] makes use of this fact by randomly generating low-density parity check codes on-chip and using these codes as X-codes. In an X-MISR, random weighting logic [Eichelberger91] is used to generate codes such that any given input affects an output bit with the specified probability. For simplicity, we consider only inverse powers of two as weights, as these weights are easily generated (using LFSRs and AND gates). Like a MISR, the X-MISR codes and accumulates bits cycle by cycle into a set of flip-flops used to hold a signature. Unlike a MISR, these bits are affected only by the inputs (as specified by the random code), thus X’s do not propagate from one output to another. This propagation of outputs renders the original MISR designs ineffective for compacting responses with X’s. The X-MISR incorporates this LFSR-like mixing as part of random bit generation, thus obtaining its benefits before the bits are polluted with X’s.

By varying the number of cycles of accumulation, the X-MISR can be tuned to trade off between compacted response volume and error detection capabilities. Figure 1 illustrates the error detection capabilities of an X-MISR. The architectures used for the figure have 248-bit signatures, and each line in the figure corresponds to an architecture with the specified bit weighting. For example, for the line with weight 1/32, each input is expected to affect about eight of the output bits ( $248/8=7.75$ ). The points along each line represent a set of operating points for the architecture, determined by the number of cycles over which each signature is accumulated and the X-density of the test responses. The horizontal axis gives the escape probability—the chance that no error is detected—for a defect that produces three errors in a response (a slightly pessimistic assumption: more errors are likely in practice, and are easier to detect). The vertical axis is normalized to represent the number of bits of X-MISR output necessary per X in the uncompactd test response. An industrial design with moderate effort for eliminating X’s has around 0.02% X’s in its responses. In this case, one can calculate the size of the compacted output in terms of the uncompactd output by multiplying the vertical axis by 0.0002. This value is the inverse of the compaction ratio,

<sup>1</sup>While theoretically possible, such codes provide vanishingly small compaction benefits, and are thus useless in practice.

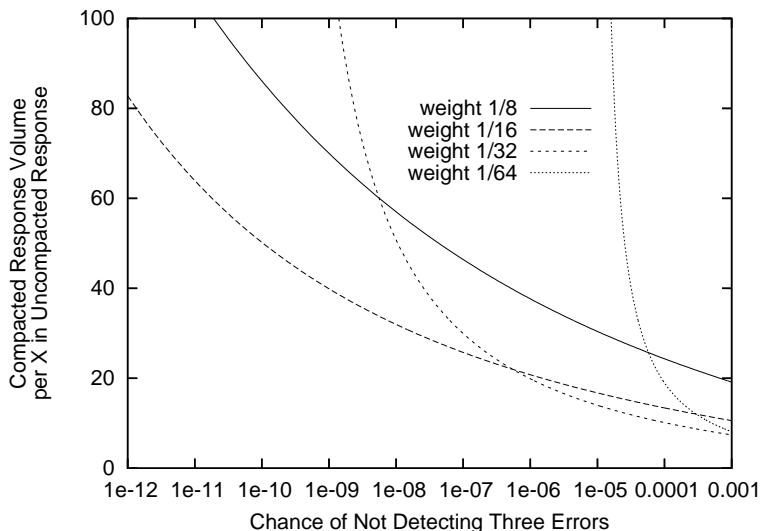


Figure 1: X-MISR error detection capabilities.

thus compaction ratio is calculated by dividing 5000 by the number on the vertical access, and ranges from 50 at the top of the figure to over 650 at the lower right. For more carefully managed designs, with lower X densities, compaction ratios can be an order of magnitude larger.

### 3 Diagnosing Bit Errors

In this section, we describe the problem of diagnosing bit errors in test response using the output of an X-MISR. We begin with some simple mathematical analysis and discussion of the nature of the problem, then show how a simple algorithm can be used fairly effectively in practice. In the next section, we discuss potential avenues for extending the simple algorithm's capabilities using both the X-MISR structure and a slightly more detailed mathematical treatment.

Assume that a given test response contains  $x$  X's and  $t$  errors, and consider the probability  $b_t$  that any given output bit from an X-MISR with weight  $w$  is observed as an error. We assume that these errors are reproducible, i.e., that running the same test vector through the circuit under test produces the same response and the same error syndrome. The X-MISR architecture can also be used to diagnose probabilistic errors, but we leave such work for the future. Due to the random nature of the X-MISR codes, each output in a compacted response can be treated as an independent random variable, and the probability  $b_t$  for an output does not depend on the state of any other output. For an output bit to show an error, it must not

be affected by any of the X's, and must be affected by an odd number of the errors (an even number produces error cancellation). We can thus write

$$b_t = \sum_{\substack{i=1 \\ i \text{ odd}}}^t \binom{t}{i} w^i (1-w)^{t-i+x}$$

The equation is essentially just the odd binomial terms up to  $t$  for weight  $w$  summed together and multiplied by  $(1-w)^x$ , which is the probability that none of the X's affects an output.

Now consider the problem of trying to invert a random code in order to identify the errors in the input. Obviously, from an information theoretic point of view, it is impossible to guarantee that such a process succeeds without making some assumption about the number of bits in error. Long before reaching this limit, however, we run into more practical difficulties. Intuitively, the compacted response for a test response that contains too many errors is just random bits. In terms of the equation above, as  $t$  grows, the sum of the odd binomial terms converges to about 1/2, thus roughly half of the bits unaffected by X's are observed to be in error. In this regime, it is not practically viable to invert the code, and one must fall back on full scan for diagnosis. However, it is easy to detect when one is in such a regime by simply counting the number of output errors, and the actual practical bound on diagnosis depends on the weight in a fairly elegant way. Consider Figure 2, which shows the relationship between the weight of the code and the fraction of the bits in error as

a function of the number of error bits in the input. The horizontal axis is normalized by multiplying the number of input error bits by the weight of the code, and the vertical axis is normalized by considering only output bits not affected by X's (equivalent to setting  $x$  to 0). Viewed in this way, the fraction of error bits is relatively independent of the weight. In other words, we can improve our ability to diagnose errors by lowering the weight of the code. As we will see, when the number of errors is fewer than about  $1/2$  the inverse of the weight, error diagnosis is fairly easy. With errors numbering around the inverse of the weight, diagnosis is hard. Beyond that limit, it is probably not worthwhile.

We now describe a simple and efficient algorithm that can be used to diagnose errors in a test response from the errors in the compacted test response. The code matrix is also necessary, of course, and can be obtained in practice by simulating the pseudo-random number generation logic.

The algorithm works as follows. We begin by assigning each input bit a score. For each output bit to which an input bit contributes (via the code), it scores either +1 or -1. Output bits that are in error give +1, and output bits not in error give -1. An input bit with the highest score is chosen, the error syndrome is XOR'd with the input bit's contributions, and the process repeats until the residual error syndrome becomes exactly zero or a fixed number of input bits have been selected without reaching zero, in which case we give up (our simulations give up as soon as they pick an input not in the actual set in error).

For codes with less random structure (and, unfortunately, more overhead, making them inappropriate for our purposes), this algorithm has been used to guarantee error correction in linear time [Sipser96]. In our context, the algorithm runs in time proportional to the product of the test response size and the compacted response size. For any given architecture, the output size is fixed, of course. In practice, the algorithm takes a negligible amount of time on a modern microprocessor.

Although the algorithm is simple and fast, it is also fairly effective, which we now show through simulation results. For all simulations, we used codes that compacted test responses with 80,000 bits into 248 bits, a compaction ratio of 322. Weights varied from  $1/8$  to  $1/64$ ; the practical values for error detection with this X-MISR architecture, as shown in Figure 1, are  $1/16$  and  $1/32$ . As all output bits are independent, we measure using blocks of 31 bits, or  $1/8$  of the full output to account for the impact of X's. With weight  $1/32$  and 16 X's, for example, roughly  $5/8$  of the compacted response is not affected by X's. If the same test vector is put through the chip twice under these

conditions, we have roughly  $5/4$  of a full compacted response for the purposes of diagnosis. Each data point was calculated using 10,000 separate random error patterns to evaluate the probability of successful diagnosis using the algorithm.

Figure 3 shows the probability of success as a function of the number of 248-bit test responses available for diagnosis. The left graph corresponds to an architecture with weight  $1/32$ , and the right graph to an architecture with weight  $1/8$ . The curves in the left graph are for a number of error bits between 12 and 20. For more or fewer errors, the curves look similar, but the horizontal spacing becomes wider for more error bits, an effect apparent in the right graph. Notice that the lower weight code (left graph) at first requires more bits to obtain the same effectiveness; this effect reflects the possibility that an input bit affects too few output bits to be identified by the algorithm. When comparing between weights, however, it is also important to recognize the impact of X's: while the 16 X's expected with the 80,000-bit test response magnify the horizontal axis of the left graph by a factor of 1.67, the same number of X's magnifies the right graph's axis by a factor of 8.47, as only about 12% of the outputs are not affected by X's with a weight of  $1/8$ .

In most cases, enough samples should be taken to guarantee that the simple algorithm works with some probability for a maximum number of errors in the test response. For a range of weights, we calculated the minimum number of response bits necessary for the simple algorithm to succeed in all 10,000 trials, thus providing a probability of success of at least 99.99%. The results appear in Figure 4. The graph on the left shows the absolute number of responses needed, i.e., assuming that all bits are useful for the diagnosis. The graph on the right assumes that the input contains 16 X's and scales the vertical axis appropriately based on the fraction of each compacted response affected by one or more X's. The small bumps in the curves are due to simulations with probabilities near 99.99% success: when a simulation returned 9,999 successful trials, it was not counted, and sometimes produced a bump. These bumps will go away with more trials (or a lower probability).

The results in Figure 4 show that the simple algorithm can be used with high probability without incurring much overhead. Specifically, if the weight  $1/32$  design is chosen, the X-MISR architecture can operate at compaction ratios of over 320 for most test responses, and can perform focused testing on problem areas to provide fast diagnosis at a cost of a few extra test responses. Note that by simply repeating each test stimulus six times, we retain a compaction ratio over 50 yet effectively guarantee that any

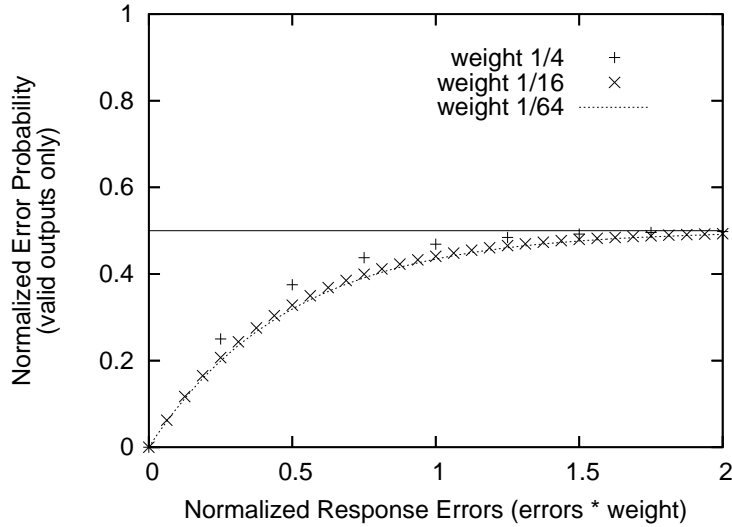


Figure 2: Expected fraction of compacted response bits observed in error as a function of the number of uncompactd response bits actually in error.

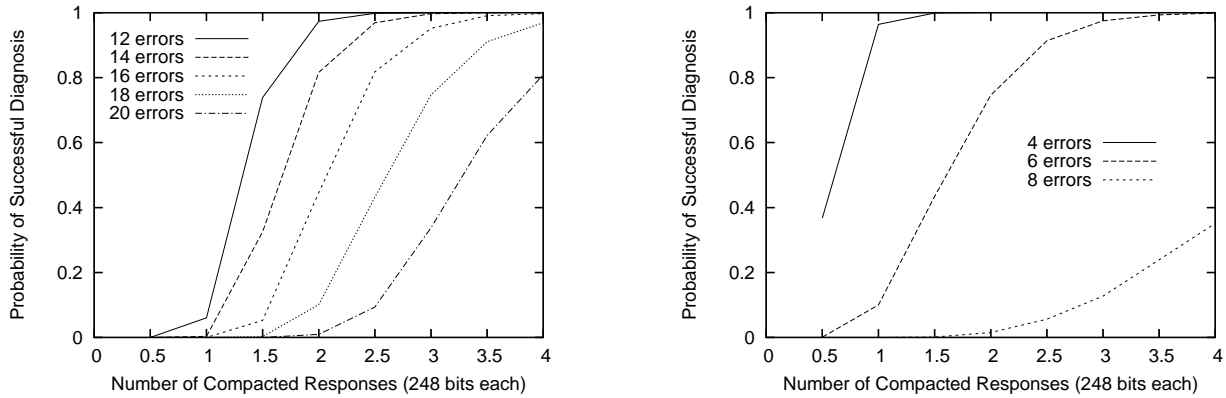


Figure 3: Chance of successful diagnosis using a simple algorithm. The left graph uses weight 1/32, and the right 1/8.

error pattern of up to about 13 bits can be diagnosed without further work. In practice, additional diagnostic testing is likely to be more focused, and the overall compaction ratio will stay over 300 while allowing defects with even more widespread impact in terms of errors to be diagnosed accurately.

#### 4 Extending the Results

This section considers how the efficacy of the simple algorithm for error diagnosis can be extended, first by leveraging the structure of the X-MISR architecture, then by considering more sophisticated mathematical reasoning.

As mentioned earlier, the X-MISR architecture works in a manner similar to a MISR: it calculates the contributions of bits coming out of the scan chains cycle by cycle and accumulates them into a signature register. Once a specified number of cycles have passed, the X-MISR emits a signature for comparison and starts accumulating a new signature. The number of cycles between signatures is configurable, and is normally set based on the X-density in order to trade off between response volume and error detection capability. However, it can also be changed to reduce the number of X's and errors in a given signature. For example, rather than compacting 80,000 bits into 248, we can break the 80,000 into four blocks of 20,000 and emit a 248-bit signature for each block. No changes need

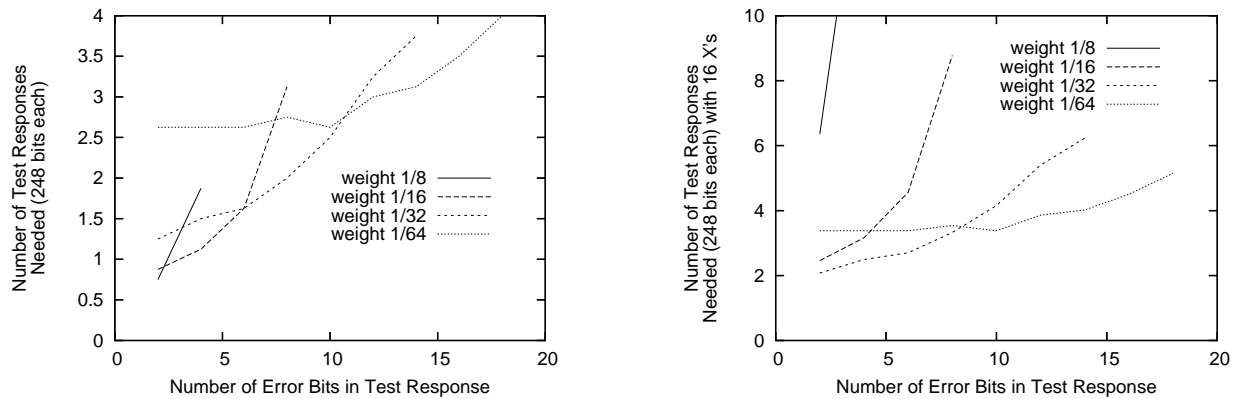


Figure 4: Number of compacted test responses required to successfully diagnose a given number of test response error bits. The left graph is an absolute number, and the right is scaled to account for 16 X's in the test response.

be made to the X-MISR other than to change the rate at which it emits signatures. As a result of this change, however, we reduce the expected number of errors in each signature by a factor of four (statistically speaking; in practice, errors are likely to be clustered) while also reducing the number of X's in each block, which raises the number of output bits available for the diagnosis algorithm. Thus, while an architecture with weight 1/32 might not scale beyond about 20 or 30 error bits when used to compact full test responses, using this approach might extend its capabilities to diagnose as many as 80 to 100 bits, while still providing compaction relative to full scan.

The second opportunity for improvement lies in taking advantage of the statistics of random variables. Looking again at Figure 2, we see that the expected number of errors in the compacted test response varies rapidly as a function of the number of errors in the uncompact test response when the number of error bits is smaller than the inverse of the weight  $w$ . We can thus use this information to make a fairly accurate maximum likelihood estimate of the number of error bits, and can use this information to help predict the error pattern. Similarly, we can differentiate input bits in error from those not in error by examining the frequency with which an input (uncompact response) bit contributes to the outputs in error (via the code matrix) and the frequency with which it contributes to the outputs not in error. Figure 5 illustrates this relationship for a code of weight 1/64. The flat line in the middle is the chance that an input bit not in error contributes to an output. Here the chance is simply  $w$ , as the input bit has no impact on whether or not the output shows an error. In contrast, the fraction of output errors to which an input error bit contributes is much higher (the vertical axis of the figure is normalized to  $w$ ), as shown in the

top line of the figure. Similarly, the fraction of non-error outputs to which an input error bit contributes is much lower, as shown by the bottom line. By counting these two groups carefully, we should be able to more quickly and effectively diagnose the original error bits. The simple algorithm is taking advantage of this phenomenon already, albeit perhaps not as well as is possible. Unfortunately, these two probabilities are not independent, and we have not yet succeeded in producing any substantial improvements over the simple algorithm by using these results. Nevertheless, they hold some promise for future work.

## 5 Conclusions

This paper described a simple algorithm that can be used with an stochastic test response compaction architecture to identify a scan cell or cells in error in a test response through a small number of compacted response samples. If this technique is employed with a stochastic compactor capable of tolerating unknown values in test responses, such as the X-MISR design, the diagnosis can also be performed in an X-tolerant manner. By employing our approach during testing, manufacturers can obtain rapid and precise failure information for a large number of chips without extensive additional diagnosis processes, and without adding substantial overhead to the testing process. The X-MISR design also allows further improvement over the base algorithm by splitting errors into separate blocks, thus reducing both the number of errors and unknowns that affect each compacted signature, and extending the capabilities of the method. Statistical results also suggest that the simple algorithm can be improved through the use of a more detailed analysis, but doing so is left for future work.

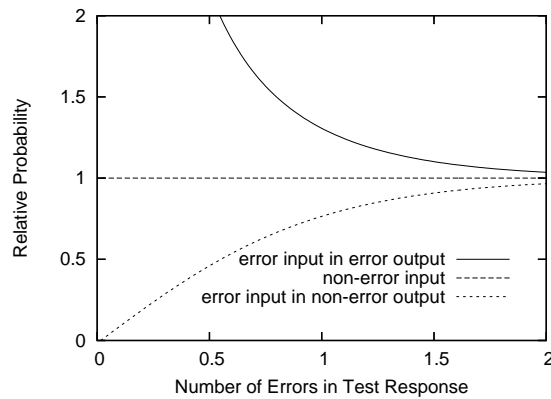


Figure 5: Probability differences between error and non-error inputs. The flat line in the middle shows the probability that an input bit contributes to any given output bit (equal to  $w$ ). The top and bottom lines show the relative probabilities that an input (uncompacted response) bit contributes to an output bit in error (top) or not in error (bottom).

## 6 References

[Eichelberger91] E. B. Eichelberger, E. Lindbloom, J. Waicukauski, T. W. Williams, “Structured Logic Testing,” *Prentice-Hall*, 1991.

[Hamzaoglu99] I. Hamzaoglu, J. H. Patel, “Reducing Test Application Time for Full Scan Embedded Cores,” in *Proc. of the Fault-Tolerant Comp. Symp.*, pp. 260-267, June 1999.

[Mitra02] S. Mitra, K. S. Kim, “X-Compact: An Efficient Response Compaction Technique for Test Cost Reduction,” in *Proceedings of the International Test Conference*, pp. 311-320, October 2002.

[Mitra04] S. Mitra, S. S. Lumetta, M. Mitzenmacher, “X-Tolerant Signature Analysis,” *Proceedings of the International Test Conference*, October 2004.

[Naruse03] M. Naruse, I. Pomeranz, S. M.Reddy, S. Kundu, “On-Chip Compression of Output Responses with Unknown Values using LFSR Reseeding,” in *Proceedings of the International Test Conference*, pp. 1060-1068, October 2003.

[Patel03] J. H. Patel, S. S. Lumetta, S. M. Reddy, “Application of Saluja-Karpovsky Compactors to Test Responses with Many Unknowns,” in *Proc. of the 21st IEEE VLSI Test Symposium*, pp. 107-112, April 2003.

[Rajski03] J. Rajski, J. Tyszer, C Wang, S. M. Reddy, “Convolutional Compaction of Test Responses,” in *Proceedings of the International Test Conference*, pp. 745-754, October 2003.

[Sipser96] M. Sipser, D. A. Spielman, “Expander Codes,” *IEEE Transactions on Information Theory*, 42(6):1710-1722, 1996.

[Wang03] C. Wang, S. M. Reddy, I. Pomeranz, J. Rajski, J. Tyszer, “On Compacting Test Response Data Containing Unknown Values,” in *Proc. International Conf. on Computer-Aided Design*, pp. 855-862, 2003.

[Wohl03] P. Wohl, L. Huisman, “Analysis and Design of Optimal Combinational Compactors,” in *Proc. IEEE VLSI Test Symposium*, pp. 101-106, 2003.