IEEE 6th Workshop on RTL and High Level Testing, pp. 24-30, July 2005.

Matrices of Multiple Weights for Test Response Compaction with Unknown Values

Thomas Clouqueur[1], Kewal K. Saluja[2], Hideo Fujiwara[1]
[1]Graduate School of Information Science, Nara Institute of Science and Technology, Japan.
[2]University of Wisconsin-Madison, USA.

Abstract

Occurrence of unknown states/values in scan chains in response to test vectors is a common phenomenon. This paper presents a method for designing matrices for linear test output compactors by using rows of multiple weights. The proposed method offers superior compaction performance provided that the unknowns are non-uniformally distributed among the scan chains. Our method increases the compaction ratio for multiple output compactors and reduces memory for single output compactors. The paper also provides an evaluation of the convolution and block based X-compactors based on the proposed method.

1 Introduction

As the number of scan cells in VLSI circuits increases, more and more scan chains are being used to control the scan chain length which is an essential factor for test application time. However, the number of scan chains can go well beyond the number of pins available to load and unload these scan chains, requiring compression (lossless) and compaction (lossy) to reduce test application time and required tester memory depth [13]. That compression/compaction can also deal with primary inputs and primary outputs by including them in scan chains.

Compression techniques can be used to reduce the amount of data stored on the tester and loaded on the chip to generate the test stimuli. As test cubes generated by ATPG tools have generally a low fill rate, high compression ratios can be obtained by designing schemes that generate the specified bits of the test stimuli, randomly filling all the unspecified bits. Some of the schemes developed for test stimuli compression include the statistical coding schemes [10], LFSR reseeding schemes [11] and parallel serial scan scheme [5].

On the scan output side, compaction techniques can be used to also reduce the amount of data uploaded from the chip and stored on the tester (or on chip in case of BIST) for comparison. Lossy compaction is possible because a compacted version of the output data is sufficient to decide if the chip is faulty or not and even perform diagnostic in some cases. The performance of a compactor is evaluated by the compaction ratio, the loss of fault coverage and the loss of fault diagnostic capability. Fault coverage can decrease if two or more errors cancel each other out during compaction, which is referred to as aliasing or error masking. In the same manner, the presence of unknown values in the scan chains (*X states*) can severely impact the performance of compactors. This is because X states can mask known states and therefore prevent some errors to directly reach the output of the compactor. That effect is referred to as *X masking*.

X states have various sources such as bus contentions, uninitialized states, inaccurate simulation models, and uncertainties during at-speed testing for delay faults. However, the sources of X states can be localized within a circuit and, therefore, affect some scan chains more than others. For example, observation of some industrial circuit showed that over 90% of the X states were produced by only 10% of the scan chains. Based on that observation, this paper proposes new test response compactors called multiple weight compactors. Multiple weight compactors can reduce the X masking, given that a small part of the scan chains produce a large part of the X states.

This paper is organized as follows. In Section 2, we give an overview of previous studies conducted on test response compaction, and, in Section 3 we introduce the main idea of the multiple weight compactors. Section 4 describes the error detection and diagnostic properties of the multiple weight compactors in the presence and in the absence of X states. Section 5 describes experimental results to evaluate the X masking properties of the multiple weight compactors. Section 6 specifies the number of scan chains observable as a function of the number of outputs and the number of memory elements. Finally, the paper concludes with Section 7.

2 Previous work

In general, test response compactors are characterized by their space and time compaction capability, their dependence with respect to the circuit or the test set, and their linearity or nonlinearity. A number of schemes were developed to compact a given test response [1, 2, 9, 12, 22]. They can achieve high compaction ratios with minimum impact on fault coverage, but they are test set dependent, which is undesirable when considering the design flow. Such schemes can be used for testing of IP cores as the knowledge of the circuits may not be required for compacting a given test response. Other schemes [3, 19] are only circuit dependent, thus limiting the design flow violation but still constraining that the compactor can be designed only once the circuit is fully designed. This study focuses on circuit independent compactors which design only requires coarse information about the circuit such as the number of outputs. Such compactors are usually linear to maximize the observability of the scan outputs, thus we also

restrain this study to linear compactors.

With respect to space and time compaction capabilities, compactors can be divided into three classes: *infinite memory compactors*, *finite memory compactors*, and *zero memory compactors*. Each class of compactors can combine some space compaction with time compaction, zero memory compactors being purely space compactors. When a scan output sequence is fed to an infinite memory compactor, the signature produced is dependent on all the output values, no matter how long the sequence is. Thus, these compactors are used to achieve very high compaction ratios in the time dimension. They are frequently used, for example in BIST environments, and well developed techniques for infinite time compaction include linear feedback shift register (LFSR), multiple input shift register (MISR), and counting based techniques. Note that, while performing high time compaction, a MISR also performs space compaction when the number of memory elements used is smaller than the number of scan chains. When producing signatures, compactors should propagate errors occurring at the scan outputs. Although linear compactors have good signal propagation property, some errors can cancel each other during compaction, possibly leading to a drop in fault coverage. That effect, referred to as *error masking* (or aliasing) can be marginalized in MISR designs by increasing the size of the shift register. However, the loss of information due to compaction still impacts severely the fault diagnostic based on compacted outputs. Similarly, the presence of X states in the scan chains also affects the compactor performance. The impact of X states, referred to as *X masking*, is severe because each X states in the output sequence multiplies by two the number of possible signatures corresponding to the fault free behavior. That makes signature analysis impossible since sequences of test usually contain more X states than the size of the shift register. A solution to that problem is to mask the X states occurring at the scan outputs before feeding them to the compactor. The solutions proposed for masking the X states are either test set dependent [18] or require some extra inputs [4, 16]. Generally, the masking circuit not only masks X states but also some known states, resulting in a possible loss of fault coverage of different classes of faults [23]. In this work, we study compactors assuming that the scan outputs contain X states. That environment occurs either when no mask is used to remove X states or when the mask only removes part of the X states. Since infinite memory compactors have limited X tolerance, we focus on zero memory compactors and finite memory compactors.

Zero memory compactors are combinational circuits that merely achieve space compaction. Some zero memory compactors, namely Saluja-Karpovsky compactors, are based on check matrices of error correcting codes [21]. Their error masking properties and diagnostic capabilities depend directly on the distance of the code corresponding to the check matrix used. In the presence of X states, these compactors can be used with post-processing of the compacted output to compare it with a set of possible compacted outputs for the fault free circuit, each possible compacted output corresponding to a different realization of the X states [17]. Other compactors named X-compactors avoid that post-processing by building compaction matrices guaranteeing that a given number of errors are directly observable, i.e. by simple comparison with the fault free compacted output, in the presence of a given number of X states at the scan output [14]. Zero memory compactors are limited in the compaction ratio attainable by the relative small number of scan chains. Indeed, much higher compaction ratio can usually be obtained by compacting more values at a time. Finite memory compactors can improve the compaction ratio of a zero memory compactors by expending compaction in the time domain. When a scan output sequence is fed to a finite memory compactor of time depth d, the signature produced at a given time t depends on the output values between time $t - d$ and t. For example, a MISR for which the signature is checked and reset every d cycles is a finite memory compactor. Recently, two block compactors were proposed, namely *convolutional compactor* [7, 20] and *block compactor* [24]. They are able to provide one output compaction while preserving low error masking and X masking. Also, when compared with the X-compact scheme for equal compaction ratio, they demonstrate lower X masking probability. Another finite memory compactor produces *X tolerant signatures* by generating random rows for the linear operation and compacting in the time dimension [15].

3 Main idea

This study focuses on three linear compactors: X-compact, convolutional compactor and block compactor. X-compactors can be described with a binary matrix having n rows, each corresponding to a scan chain and m columns, m being the number of compactor outputs. The entry in row i and column j of the compaction matrix is "1" if and only if scan chain i is connected to the XOR tree feeding output j for zero memory compactor (or memory element j for finite memory compactors). Otherwise, the entry is "0". Similarly, block compactors of depth d can be described with a binary matrix with $d.n$ rows and $d.m$ columns. Each row is associated to one of the last d cells of the scan chains, the set of $d.n$ such cells forming an input data block as described in Figure 1. The matrix compacts the input data block into an output data block of size $d.m$, which is then scanned out in d cycles through the m outputs as the next input data blocked is formed during regular shifting of the scan chains. Note that the same model using data blocks is valid for X-compact. Then, a data block consists of scan cells located at the scan chain outputs and observed during a single scan-out cycle, i.e. depth $d = 1$. Note also that Figure 1 suggests that block compactors are intrusive of the scan chains. However, there exist non-intrusive designs for which the compactor inputs are merely the scan chain outputs. Of course, these designs have extra memory elements within the compactor.

Convolutional compactors can be described using the model presented in Figure 2. For convolutional compactors, the matrix has n rows each connected to a scan chain output and M columns, M being the number of memory elements in the compactor. The memory elements form m chains connected to the outputs and scanned out in $d = \lceil \frac{M}{m} \rceil$ cycles.

The error masking and X masking properties of X-compact, convolutional compactors and block compactors depend essentially on the properties of their matrices. The three compactors use matrices with rows of equal and odd weight, the weight being the number of ones. For X-compact and block compactors, such single weight matrices guarantee to detect 1, 2 or any odd number of errors occurring within the same data block if all matrix rows are

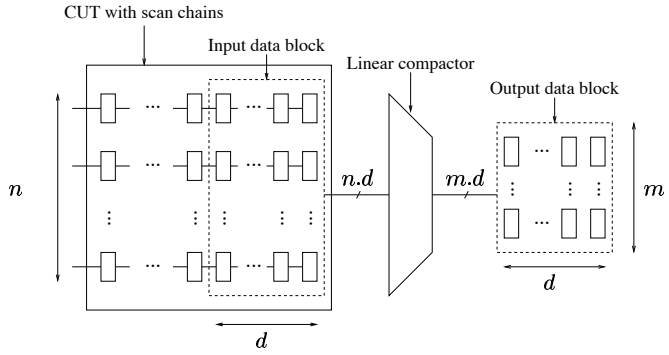

Figure 1: X-compactor and Block compactor.

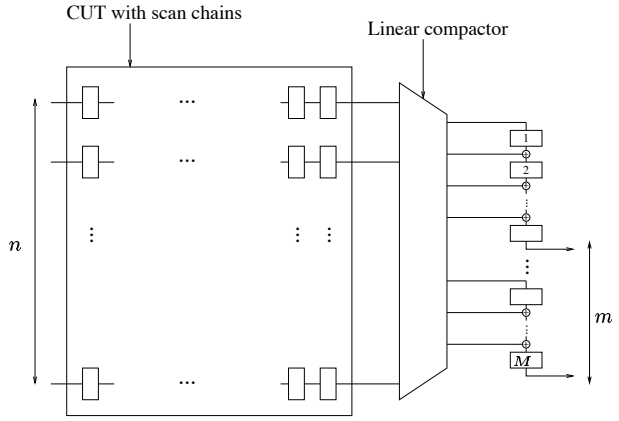

Figure 2: Convolutional compactor.

different. The same property holds for convolutional compactors if rows are different and one row is not a shifted version of another. Also, errors can be at the same time or at different scan cycles. For X-compact and block compactors, in the presence of one X state in a data block, the matrices of equal weight guarantee detection of any single error. Convolutional compactors with single weight matrix can also detect any single error in presence of one X states produced at the same or at a different scan cycle.

It was observed in the study of convolutional compactors that matrices with low weight outperformed matrices with higher weight when scan chains produced a large number of X states. But the opposite observation was made when scan chains produced only few X states. Indeed, increasing the weight makes scan chains outputs propagate to more compactor outputs, and become more observable. However, increasing the weight also increases the number of compactor outputs that are made unobservable by each X state produced. Another observation we mentioned in the introduction is that the X state distribution among the scan chains is not uniform in general, i.e. a majority of the X states is produced by a small fraction of the scan chains.

Therefore the main idea of this study is to build compactors using matrices with multiple weights. Rows of small weight are assigned to scan chains producing many X states, while rows of larger weight are assigned to scan chains producing few X states. The multiple weight compactors considered are based on X compactors, convolutional compactors and block compactors. Note that multiple weight compactors were used in pioneer work on compaction [8, 21] and in compactors with random rows [15], but those schemes do not consider the distribution of X values when assigning the rows to the scan chains. Also, a method to build multiple weight matrices considering the X value distribution has been proposed recently for convolutional compactors [6]. In this paper, we generalize that idea to other types of compactors and study the general properties of multiple weight compactors. In particular, we prove that, in the absence of X states, the error masking is identical for the compactors with rows of multiple weights and of single weight. The diagnostic capability is also proven identical for both types of compactors. In the presence of X states, we describe the X masking properties of multiple weight compactors and we evaluate X masking probability through simulation. Finally, we also show that multiple weight compactors can achieve higher compaction ratio than single weight compactors.

4 Compactor properties

In this section, we analyze the error detection and diagnostic capabilities of multiple weight compactors based on X-compact and block compactor schemes. The X-compactors and block compactors are analyzed simultaneously by considering compaction of data blocks. Convolutional compactors have similar properties than X-compactors and block compactors but the errors and X states considered are not restrained to belong to a data block. Also, the matrices used for convolutional compactors need the additional constraint that one row is not a shifted version of another row. The analysis of error detection and diagnostic capabilities are conducted assuming first that the scan chains do not produce X states. Then the presence of X states is considered.

4.1 Properties in the absence of X states

First, we look at cases for which no X states are produced in a data block. These properties were previously derived [8, 21] and are repeated here for clarity of the presentation.

Property 1: A multiple weight compactor detects errors from any one or two scan cells in a data block if the rows of its matrix are nonzero and different.

Note that, for convolutional compactors, the data blocks do not have fixed positions and a given error is guaranteed to be detected if the condition stated above holds for any data block to which the error belongs. That observation holds for all the properties stated in this section.

Property 1 simply states that any error can propagate to the compacted outputs if the rows are nonzero, and two errors cannot cancel each other if two rows cannot add up to zero, i.e. two rows are different. That property corresponds to well known results from error correcting code theory since a check matrix with nonzero and different rows corresponds to a Hamming code (shortened or not) of distance three [21].

Property 2: A multiple weight compactor detects errors from any one, two, or k scan cells, with k odd, in a data block if the rows of its matrix are nonzero, different, and all the entries of one column of its matrix are "1".

Such a matrix corresponds to an augmented Hamming code with overall parity check, which has distance four. Therefore, detection of 1, 2, and any odd number of errors is guaranteed. Note that this property is identical to the one derived for the single weight compactors previously proposed. In general, multiple weight matrices can be built using check matrices of error correcting codes of any given distance. The code needs to be chosen so that its check matrix has enough rows of each weight desired.

Property 3: A multiple weight compactor detects errors from any one, two, or k scan cells, with k odd, in a data block if the rows of its matrix are nonzero, different, and have odd (but not necessary equal) weight.

Adding the condition that all the weights are odd gives again identical detection performance for multiple weight and single weight compactors. That property derives from the fact that errors cancel two by two, therefore if an odd number of errors are propagated an odd number of times, they cannot fully cancel out. Note that detection of four errors can also be guaranteed by choosing the rows in a manner such that four rows cannot add up to zero [20].

Regarding error diagnosis ability, the following property holds.

Property 4: A multiple weight compactor uniquely identifies any single erroneous cell in a data block if the rows of its matrix are nonzero and different.

Property 4 also relates to error correcting code theory, the matrix corresponding to a code of distance three. The result of Property 4 is again equivalent to the diagnostic capability of single weight compactors.

4.2 Properties in the presence of X states

When using rows of multiple weights, it is possible that one row r_1 with high weight covers a row r_2 with low weight, i.e. the entry in any column of r_1 is "1" whenever the entry in the same column of r_2 is "1". In such a case, an X state produced by the scan cell associated with r_1 will make the scan cell associated with r_2 unobservable. Therefore, no error detection is guaranteed in the presence of X states for multiple weight compactors when considering all the scan cells together. Note that single error detection in the presence of single X state is guaranteed by single weight compactors. Nevertheless, some properties hold for multiple weight compactors when considering groups of scan cells separately. For simplicity of the presentation, we assume that only two different weights are used in the compactor matrix, namely w_l for weight low and w_h for weight high.

Property 5: A multiple weight compactor using weights w_l and w_h detects any single erroneous cell in a data block, provided that an X state is produced by one of the scan cell associated with a row of low weight in the data block, if the rows of its matrix are nonzero and different.

Indeed, a row of low weight cannot cover another row of low weight (since the rows are different) and cannot either cover a row of large weight.

Property 6: A multiple weight compactor using weights w_l and w_h detects any single erroneous cell in a data block associated to a row of high weight, provided that an X state is produced by any scan cell in the data block, if the rows of its matrix are nonzero and different.

Indeed, a row of high weight cannot be covered by any other row in the matrix.

Property 7: A multiple weight compactor using weights w_l and w_h with $w_h > 2w_l$ detects any single erroneous cell in a data block associated to a row of high weight, provided that two X states are produced by any two scan cells associated with rows of low weight in the data block, if the rows of its matrix are nonzero and different.

Indeed, the condition $w_h > 2w_l$ prevents any two rows of low weight to cover a row of high weight. Note that such a property does not hold for single weight compactors.

5 Experimental results

The properties described in the presence of X states show that in some cases more X states can be supported by the multiple weight compactors than by the single weight compactors, and vice versa. However, better X masking performance is possible from multiple weight compactors if the cases in favor of these compactors are more frequent than cases against them. That can be achieved when the X states are not uniformly distributed among the scan cells by associating scan cells producing many (resp. few) X states to rows of low (resp. high) weight. The purpose of this section is to evaluate the possible gain in X masking through simulation.

We looked at the three compactors, X-compact, convolutional compactors and block compactors, in their original single weight version and their modified multiple weight version. We chose to evaluate X masking probability for compactors with 1600 input scan chains and 16 outputs to mimic a simulation conducted in the study of convolutional compactors [20]. For the convolutional and block compactors, the number of memory elements is 32 so that the depth is 2. The weight used for unmodified compactors is 7 while the two weights used in the modified compactors are 3 and 7. Three scenarios are simulated. In the first scenario (S1), the X states are assumed to be uniformly distributed among the scan cells and unmodified compactors are used. In the second scenario (S2), it is assumed that 90% of the X states are produced by 10% of the scan chains and unmodified compactors are used. In the third scenario (S3), it is again assumed that 90% of the X states are produced by 10% of the scan chains but modified compactors are used. For each scenario, the average portion of scan cells producing X states considering all the scan cells (named X probability) varies from 0.01% to 1%. For each case, X masking is measured by the average percentage of scan cells unobservable during compaction, including scan cells that produced an X state. The results are obtained by Monte Carlo simulation.

	X probability (%)						
	0.01	0.02	0.05	0.1	0.25	0.5	1
S1	0.069	0.27	1.7	7.0	36	77	98
S2	0.041	0.15	0.93	3.9	23	60	93
S3	0.041	0.099	0.37	1.2	7.4	28	71

Table 1: Percentage of scan cells masked with X-compact

The results for X-compaction are presented in Table 1. Considering the single weight compactor, i.e. scenarios S1 and S2,

the results show that with the non uniform distribution of X states, X masking is smaller than with uniform distribution. That result may be surprising but it is a consequence of the method used to construct the matrix. The matrix has 1600 rows when the total number of rows of weight 7 possible is $\binom{16}{7} = 11440$, therefore not all the possible rows are used. The rows chosen follow a lexicographic order such that the first 160 rows that are assigned to scan chains producing many X states have entry "1" in their first four columns. This similarity between rows reduces the effect of X masking. Note that choosing rows in a different manner may have other advantages such as guaranteeing detection of four errors. To evaluate the impact of row selection, we repeated the simulation of X-compaction with rows randomly selected. The results are presented in Table 2. They show that matrices with rows not following the lexicographic order have higher X masking. Furthermore, negligible difference was observed between uniform and non uniform distribution of X states for the unmodified compactor.

	X probability (%)						
	0.01	0.02	0.05	0.1	0.25	0.5	1
S1	0.065	0.27	1.9	8.2	42	84	99
S2	0.067	0.26	1.9	7.9	42	83	99
S3	0.027	0.066	0.29	1.1	8.6	36	82

Table 2: Percentage of scan cells masked with randomly selected X-compact

Table 1 and 2 show that a substantial reduction in X masking is achieve for scenario 3, corresponding to multiple weight X compactor. Note that in general smaller levels of X masking are obtained for multiple weight matrices built using the lexicographic order. However, in scenario 3, the matrices built using lexicographic order only perform better for high levels of X probability. In the remaining results regarding the convolutional and block compactor, lexicographic order was used to build the matrices.

	X probability (%)						
	0.01	0.02	0.05	0.1	0.25	0.5	1
S1	0.040	0.15	0.99	4.1	22	54	83
S2	0.036	0.13	0.78	3.0	14	36	64
S3	0.040	0.11	0.46	1.5	8.1	25	56

Table 3: Percentage of scan cells masked with convolutional compactor

The results presented in Table 3 and 4 demonstrate that multiple weight convolutional and block compactors generally have smaller X masking probability than corresponding single weight schemes. In fact scenario 3 always results in less masking except when they are very few X states produced. In such a case, the benefit gained by Property 7 is reduced because it is very unlikely that two X states are produced in the same data block. However, a single X state produced by a scan cell associated with a row of high weight can still mask a scan cell associated with a row of low weight. However, note that when very few X states are produced, the probability of X masking is very low and may have no impact on fault coverage.

	X probability (%)						
	0.01	0.02	0.05	0.1	0.25	0.5	1
S1	0.035	0.13	0.82	3.6	21	57	91
S2	0.025	0.082	0.47	2.0	13	39	79
S3	0.033	0.083	0.35	1.2	7.9	28	66

Table 4: Percentage of scan cells masked with block compactor

When comparing the X-compact, convolutional and block compactor following lexicographic order, the general trend is that X masking is lowest for the block compactors when X probability is low and for convolutional compactors when X probability is high. However, X masking is lowest for X-compact in one case: scenario 3 with X probability of 0.025%. Furthermore, we observe that for low X probability and scenario 3, the X-compact scheme not following lexicographic order has lowest X masking. The effect of row selection on X masking requires further investigation to be studied in parallel with the effect of row selection on error detection.

6 Compactor dimensions

In this section, we analyze the compaction capability of multiple weight compactors and compare it with the capability of single weight compactors. The compaction capability of a compactor can be measured by the maximum number of scan chains supported for a given number of outputs m and a given number of memory elements M when considering finite memory compactors. Let $w_i, 1 \leq i \leq p$ be p different weights used to build the rows of the compactor matrix. For multiple weight X-compactor, the maximum number of scan chains supported, named N_x, is given by the following equation.

$$N_x = \sum_{i=1}^{p} \binom{m}{w_i} \tag{1}$$

For multiple weight convolutional compactor, the maximum number of scan chains supported, named N_c, is given by the following equation.

$$N_c = \sum_{i=1}^{p} \sum_{j=1}^{m} \binom{M-j}{w_i - 1} \tag{2}$$

For multiple weight block compactor, let $d = \lceil \frac{M}{m} \rceil$ be the depth of the compactor. Then, the maximum number of scan chains supported, named N_b, is given by the following equation.

$$N_b = \sum_{i=1}^{p} \frac{1}{d} \binom{M}{w_i} \tag{3}$$

In every case, the maximum number of scan chains supported by a single weight compactor would be only one of the term in the

sums specified. Therefore, the maximum number of scan chains supported by the multiple weight compactor is obvious greater. That property can be used to achieve higher compaction ratio for the X-compaction scheme. Since single weight convolutional compactors and block compactors are able to compact any number of scan chains into a single output, compaction ratio cannot be improved. However, multiple weight compactors can in some cases reduce the number of memory elements necessary to achieve single output compaction.

7 Conclusion

In this paper, multiple weight versions of X-compactors, convolutional compactors and block compactors were introduced. While preserving error detection and diagnostic capability of the single weight compactors in the absence of X states, they showed to reduce the X masking of the single weight compactors when the distribution of X states among the scan chains is nonuniform. Furthermore, multiple weight compactors can support more scan chains for a given number of outputs and given number of memory elements in the compactor.

Although only two different weights were used in the experiment presented in this paper, more weights can be considered. In general, observing the distribution of X states among the scan chain can result in a choice of several weights, smallest weight for the scan chains producing most X states, then increasing weights for scan chains producing decreasing portion of X states. Another observation is that some of the scan chains may produce more errors than others or errors resulting from hard to detect faults. In such case, it might be advantageous to associate a row of high weight with such scan chains to improve fault coverage.

An observation made during this study is that X masking depends on the row selection during the matrix construction. It appeared that matrices following lexicographic order have smaller X masking. However, as mentioned, selecting rows without following lexicographic order can have other advantages such as preventing four error cancellation. The tradeoff between error masking and X masking in the row selection needs to be further studied for both single weight and multiple weight compactors.

Acknowledgments

This work was supported in part by 21st Century COE Program and in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research B(2)(No. 15300018) and the grant of JSPS Research Fellowship (No. L04509).

References

[1] B. Bhattacharya, A. Dmitriev, M. Gossel, and K. Charkrabarty, "Synthesis of single-output space compactors for scan-based sequential circuits," *IEEE Trans. on CAD*, vol. 21, pp. 1171–1179, October 2002.

[2] K. Chakrabarty, "Zero-aliasing space compaction using linear compactors with bounded overhead," *IEEE Trans. on CAD*, vol. 17, pp. 452–457, May 1998.

[3] K. Chakrabarty and J. P. Hayes, "Zero-aliasing space compaction of test responses using multiple parity signatures," *IEEE Trans. on VLSI Systems*, vol. 6, pp. 309–313, June 1998.

[4] V. Chickermane, B. Foutz, and B. Keller, "Channel masking synthesis for efficient on-chip test compression," in *Proc. ITC*, pp. 452–461, 2004.

[5] I. Hamzaoglu and J. H. Patel, "Reducing test application time for full scan embedded cores," in *Proc. of FTCS*, pp. 260–267, 1999.

[6] Y. Han, Y. Hu, H. Li, X. Li, and A. Chandra, "Response compaction for test time and test pins reduction based on advanced convolutional codes," in *Proc. Symp. on Defect and Fault Tolerance in VLSI Systems*, pp. 298–305, 2004.

[7] Y. Han, Y. Xu, H. Li, X. Li, and A. Chandra, "Test resource partitioning based on efficient response compaction for test time and tester channels reduction," in *Proc. ATS*, pp. 440–445, 2003.

[8] M. Hsiao, "A class of optimal minimum odd-weight-column sec-ded codes," *IBM Journal of Research and Development*, vol. 14, pp. 395–401, July 1970.

[9] A. Ivanov, B. Tsuji, and Y. Zorian, "Programmable bist compactors," *IEEE Trans. on Computers*, vol. 45, pp. 1393–1404, December 1996.

[10] A. Jas, J. Ghosh-Dastidar, and N. Touba, "Scan vector compression/decompression using statistical coding," in *Proc. VTS*, pp. 114–120, 1999.

[11] B. Koenemann, "Lfsr-coded test patterns for scan designs," in *Proc. European Test Conference*, pp. 237–242, 1991.

[12] Y. K. Li and J. P. Robinson, "Space compression methods with output data modification," *IEEE Trans. on CAD*, vol. 6, pp. 290–294, March 1987.

[13] E. McCluskey, D. Burek, B. Koenemann, S. Mitra, J. Patel, J. Rajski, and J. Waicukauski, "Test data compression," *Design & Test of Computers, IEEE*, vol. 20, pp. 76–87, Mar-Apr 2003.

[14] S. Mitra and K. S. Kim, "X-compact an efficient response compaction technique," *IEEE Trans. on CAD*, vol. 23, pp. 421–432, March 2004.

[15] S. Mitra, S. S. Lumetta, and M. Mitzenmacher, "X-tolerant signature analysis," in *Proc. ITC*, pp. 432–441, 2004.

[16] M. Naruse, I. Pomeranz, S. M. Reddy, and S. Kundu, "On-chip compression of output responses with unknown values using lfsr reseeding," in *Proc. ITC*, pp. 1060–1068, 2003.

[17] J. H. Patel, S. S. Lumetta, and S. M. Reddy, "Application of saluja-karpovsky compactors to test responses with many unknowns," in *Proc. VTS*, pp. 107–112, 2003.

[18] I. Pomeranz, S. Kundu, and S. M. Reddy, "On output response compression in the presence of unknown output values," in *Proc. DAC*, pp. 255–258, 2002.

[19] B. Pouya and N. A. Touba, "Synthesis of zero-aliasing elementary-tree space compactors," in *Proc. VTS*, pp. 70–77, 1998.

[20] J. Rajski, J. Tyszer, C. Wang, and S. M. Reddy, "Convolutional compaction of test responses," in *Proc. ITC*, pp. 745–754, 2003.

[21] K. K. Saluja and M. Karpovsky, "Testing computer hardware through data compression in space and time," in *Proc. ITC*, pp. 83–88, 1983.

[22] O. Sinanoglu and A. Orailoglu, "Efficient construction of aliasing-free compaction circuitry," *IEEE Micro*, vol. 22, pp. 82–92, Sept-Oct 2002.

[23] Y. Tang, H.-J. Wunderlich, H. Vranken, F. Hapke, M. Wittke, P. Engelke, I. Polian, and B. Becker, "X-masking during logic bist and its impact on defect coverage," in *Proc. ITC*, pp. 442–451, 2004.

[24] C. Wang, S. M. Reddy, I. Pomeranz, J. Rajski, and J. Tyszer, "On compacting test response data containing unknown values," in *Proc. ICCAD*, pp. 855–862, 2003.