Response Compaction for Test Time and Test Pins Reduction Based on Advanced Convolutional Codes

Yinhe Han Yu Hu Huawei Li Xiaowei Li Anshuman Chandra*
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100080
Graduate School of Chinese Academy of Sciences, Beijing, 100039
* *Synopsys, Inc., 700 E. Middlefield Rd., Mountain View, CA 94043, USA*
*E-mail: {*yinhes, lihuawei,lxw@ict.ac.cn*}* and anshuman.chandra@synopsys.com

Abstract

This paper addresses the problem of test response compaction. In order to maximize compaction ratio, a single-output encoder based on check matrix of a (n, n-1, m, 3) convolutional code is presented. When the proposed four theorems are satisfied, the encoder can avoid two and any odd erroneous bit cancellations, handle one unknown bit(X bit) and diagnose one erroneous bit. Two types of encoders are proposed to implement the check matrix of the convolutional code. Large number of X bits can be tolerated by choosing a proper memory size and weight of check matrix, which can also be obtained by an optimized input assignment algorithm. Some experimental results would verify the efficiency of the proposed optimized algorithm.

1. Introduction

The design of response encoder is a very well researched topic and a number of techniques have been presented in the literature [1-13]. These techniques can be divided into two categories: circuit function specific and circuit function independent. The techniques based on first category are discussed in [1-3]. The techniques based on second category use encoders based on coding theory. The technique presented in [4] first constructs a linear code encoder and discusses the bounds of some parameters in proposed encoder. The encoder (X-Compact) based on Hamming code is presented in [5] that can correct one error, and then is extended to detect any odd errors. The encoder presented in [8] is based on linear block codes to tolerate some unknown bits (X bits) in test response. The encoders in [4-7,11] are all implemented using XOR gates. A sequential encoder to implement linear block code is presented in [8].

The schemes in [9] and [10] present an encoder based on the parity check matrix of convolutional code. The convolutional compactor [9] can achieve much higher compaction ratio compared to X-Compact. Several issues on aliasing probability, X-Masking and diagnosis are discussed in this paper. A response compaction scheme based on *q*-compactor was presented in [10], which is also based on the convolutional code. The *q*-compactor proposed in [10] focuses on the single-output compactor and the full-diagnosis capability.

In this paper, we will develop and extend the work in [10]. The work presented in this paper can be considered as an extension of convolutional compactor in [9] [10] by relating it with the convolutional code theory and the linear code matrix-based encoder in [4].

The remainder of the paper is organized as follows: Section 2 introduces the background of convolutional code and presents some design theorems to construct the single-output encoder. The two styles of implementations of proposed encoder are

* This paper is supported by the National Natural Science Foundation of China (NSFC) under grant No.90207002 and 60242001 and supported by Basic Foundation of Institute of Computing Technology, CAS under grant No. 20036160. The work of Anshuman Chandra is supported by Synopsys Corporation.

Proceedings of the 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'04)
1550-5774/04 $ 20.00 IEEE

presented in Section 3. In Section 4, the theoretical analysis and an input assignment algorithm are presented to handle large number of X bits.

2. Construction of Advanced *(n, n-1, m, d)* Convolutional Code

Convolutional code was first introduced by Elias [15] in 1955 and is widely applied today in telecommunication systems, e.g., radio, satellite links, mobile communication [19]. A convolutioanl code can be denoted as *(n, k, m, d)*, where *n* is the number of parallel output information bits, *k* is the number of parallel input information bits at one time interval (cycle), *m* is the maximum number of memory elements in encoder, and *d* is the minimum distance of code words. It also can be characterized by an arbitrarily large generator matrix G_∞. Each *k*-input information during ∞ cycles can be mapped into ∞ *k-tuple* polynomials: $I=\{I_0(x), I_1(x), \ldots, I_\infty(x)\}$, $I_i(x)$ is a *k* degree polynomial. Then the "code word" $C=\{C_0(x), C_1(x), \ldots, C_\infty(x)\}$, where $C_i(x)$ is *a n-tuple* polynomial, is defined as: $C = I \bullet G_\infty$. The dot in this formulation denotes vector-matrix multiplication.

We mentioned the degree of generator matrix G_∞ can be arbitrarily large in principle if the degree of input polynomial is arbitrarily large. However, in any constrain length: *m* cycles, the relation of affected code words are same. This leads us to define a *m-truncation* of polynomial matrix. The in-side information polynomial $I_m=\{I_0(x), I_1(x), \ldots, I_m(x)\}$ contains $m \times k$ binary bits, and the code word C_m contains $m \times n$ bits. The encoding mapping from *I* to *C* can be represented as $C_m = I_m \bullet G_m$, where the sub-matrix G_m is a truncation of the generator matrix of G_∞. The check matrix H_∞ of convolutional code is defined as: $G_\infty \bullet H_\infty^T = 0$. If the *m-truncation* generator matrix is used, the corresponding truncation check matrix is: $G_m \bullet H_m^T = 0$, where H^T_m is a $(m \times (n-k), m \times n)$ matrix. In H_m, each entry h_i is a *(n-k, n)* matrix. The H_m is determined by the first *n* columns since the other columns can be looked as a shift version of the first *n* columns. We define the matrix $H=\{h^T_1, h^T_2, h^T_3, \ldots, h^T_m\}^T$ as the basic check matrix.

From the previous literature, some relations between the linear code and convolutional code are found:

Lemma 1 [19]. *The linear code (n, n-k, d) has the minimum distance d iff any d-1 columns in the check matrix H* $(k \times n)$ *are linearly independent.*

Lemma 2 [19]. *The convolutional code (n, k, m, d) can be looked as a linear code whose generator matrix is* G_m *and check matrix is* H_m.

Saluja [4], Mitra [5] and Patel [6] give the design methods of compactor based on linear code. As said in Lemma 2, the convolutional code can be looked as a linear code whose generator matrix is G_m. So some design theorems of other linear code can be used to construct the convolutional code. The *(2m, 2m-1, m)* WA code [19] is presented when referring to the construction procedure of hamming code. WA code can correct one error. Massey [16] and Justesen [18] construct some convolutional codes based on design theorems of cyclic codes and Reed-Solomon codes. Recently, relations between convolutional code and MDS codes [18] are used to construct convolutional codes.

In this paper, we will present a simple method to construct the *(n, n-1, m, d)* convolutional code. Some definitions are introduced firstly:

Definition 1: To two *m-tuple* polynomials: $R_1(x)$ and $R_2(x)$, the relation

$R_1(x) \cong R_2(x)$ iff.

(I) $\exists \sigma \mid R_1(x) = x^\sigma \times R_2(x), \quad (0 \le \sigma \le m-1-deg(R_2(x))$ or

(II) $\exists \sigma \mid R_2(x) = x^\sigma \times R_1(x), \quad (0 \le \sigma \le m-1-deg(R_1(x))$

Proceedings of the 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'04)
1550-5774/04 $ 20.00 IEEE

is an *equivalent* relation in GF(2), where *deg(.)* is the maximum degree of non-zero coefficient in a polynomial.

Based on the definitions of "equivalent", we can get the following theorems. All of these theorems can be proved by referring Saluja [4], Mitra [5] and Patel [6]:

Theorem 1. *To the convolutional code (n, n-1, m, d), in the basic check matrix H, if no two column polynomials are equivalent, then the minimum distance of this code:* $d \geq 3$.

Theorem 2. *To the convolutional code (n, n-1, m, d), if the basic check matrix H is constructed based on theorem 1 and the weight of each column polynomial is odd, then the* H_m*-based encoder can guarantee to detect two and any odd errors in test response within m cycles.*

Theorem 3. *One error with an X bit produced in test response is guaranteed to be detected by the* H_m*–based encoder if theorem 1 is satisfied and the weight of each column polynomial is equal.*

Theorem 4 *The* H_m*–based encoder can locate one error from the output slices of scan chains if theorem 1 is satisfied.*

The theorem 2, 3, 4 can be looked as extensions of the three properties in [9] [10]. Observing these three theorems, though the theorem 3 gives the capacities of handling the X bits, it is inefficiency when facing the practical scenes and needed to be extended. All of these will be discussed in the following sections of this paper.

3. Implementations of H_m*-based* Encoder

The implementation of check matrix H_m of a *(n, n-1, m, d)* convolutional code can be consisted by some modulo-2 adders and shift registers. Two styles of encoders can implement this check matrix. The type I encoder is non-intrusive in which the inputs of encoder are only connected into the outputs of scan chains. The type II encode is intrusive in which the inputs of encoder must be connected into several last stages of scan chains. "Intrusive" and "Non-intrusive" indicate whether the encoder lies on the internal scan cells. "Intrusive" lies on the internal scan cells and "Non-intrusive" is independent with the internal scan cells. The Figure 2 and Figure 3 show examples of intrusive and non-intrusive styles. The basic check matrix *H* of these two encoders is:

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Figure 1. An example of the basic check matrix *H*

Figure 2 is the non-intrusive implementation of H_m. It is consisted by some XOR gates and registers. The XOR gates compose an XOR tree. XOR tree is synthesis based on H_m. Each row in *H* represents an output in XOR tree and each column represents an input. In type I, only the outputs of scan chains are needed to connected into encoder. It suitable to core-based design since it does not need to know the internal DFT information of third-part cores.

Proceedings of the 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'04)
1550-5774/04 $ 20.00 IEEE

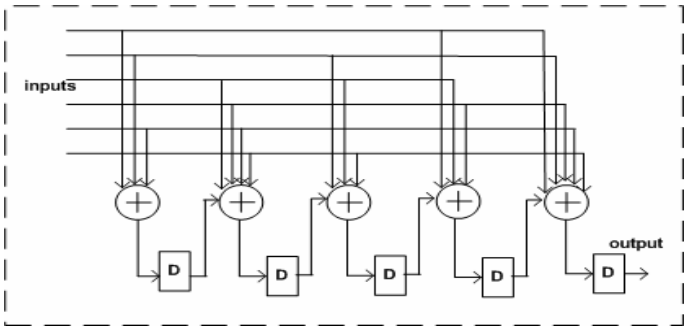

Figure 2. Type I, Non-Intrusive Implementation of Encoder

The Figure 3 shows the intrusive implementation of encoder. The inputs of encoder are connected to the last *m* stages of scan chains. The connection of *m* stages of scan cells in a scan chain corresponds to a column in *H*. Because this implementation reuses the memory elements of scan chains, it is a linear combinational compactor.

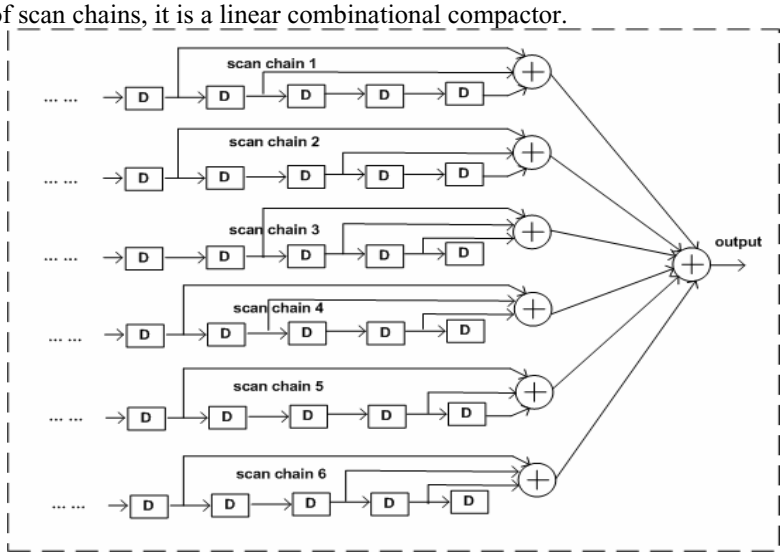

Figure 3. Type II, Intrusive Implementation of Encoder

4. Design for X bits Tolerance

There are some undetermined states in the scan chains whose outputs are not known during simulation, which are also called unknown bits or X bits (X's) [5, 12]. Theorem 3 guarantees to handle one X bit. However, this is not sufficient for real-life circuits. Finding an expensive convolutional code can help us improve this capacity. However, as shown in [6], it is not always necessary. This paper will moves the idea from "eliminate X bits" to "tolerate X bits". This means that we can improve the detection probability of errors with X-Masking through other simple methods. The detection probability: *P* is proposed in [6] and we redefine it as the detection probability of one error in presence of X bits.

An analysis of the detection probability to H_m*-based* encoder using stochastic theory is presented in the following paragraph. The check matrix H_m of *(n, n-1, m, d)* convolutional code is a *(m,* $m \times n$ *)* matrix. The weight of each column polynomial is same and equal to ω. We assume that each entry in H_m assigned 1 is equal probability and equal with $P_1 = \omega / m$ (though 1s are not randomly distributed when they are constrained by theorem 1 and

theorem 2). The input information (response data of scan chains) is *I*, and the check word of output is *C*, then: $C = H_m \times I^T$, where *I* is a $m \times n$ -bit word and *C* is a *m-bit* check word. One error is injected in the first *n* bits of *I*. Assuming the density of X bits in response data is P_X, if the X bits are randomly distributed, then P_X is the probability of one bit assigned X in response data. We use $\overline{P}_j$ to denote the un-detection probability of injected error only through observing the *j-th* bit C_j in *C*. The C_j can be calculated as:

$$C_j=[h_1,..., h_j, 0,...,0] \times I^T$$

$[h_1,..., h_j, 0,...,0]$ is the j^{th} row in H_m. Thus, only the X bits which locates in first $j*n-1$ bits can mask injected error in the calculation of C_j. The $\overline{P}_j$ can be calculated as:

$$\overline{P}_j = P_1 \times \{1-[P_X \times (1-P_1) + (1-P_X)]^{j \times n-1}\} + (1-P_1)$$

It can be reduced as:

$$\overline{P}_j = 1 - P_1 \times (1 - P_1 \times P_X)^{j \times n-1}$$

Thus, the probability of the error when it is not detected through observing all the *m* bits of check word is:

$$\overline{P} = \prod_{j=1}^{m} [1 - P_1 \times (1 - P_1 \times P_X)^{j \times n-1}]$$

Then the detection probability of the single error is:

$$P = 1 - \{\prod_{j=1}^{m} [1 - P_1 \times (1 - P_1 \times P_X)^{j \times n-1}]\}$$

If we use P_u represents "the detection probability of one error with u X bits", then combining $P_1 = \frac{\omega}{m}$, $P_X = \frac{u}{mn}$ and the formulation of *P*, P_u can be calculated as:

$$P_u = 1 - \{\prod_{j=1}^{m} [1 - (\frac{\omega}{m}) \times (1 - \frac{\omega}{m} \times \frac{u}{mn})^{j \times n-1}]\}$$

If there are *e* errors, then the detection probability of these errors in presence of *u* X bits is:

$$P_u^e = 1 - \{\prod_{j=1}^{m} [1 - (\frac{\omega}{m}) \times (1 - \frac{\omega}{m} \times \frac{u}{mn})^{j \times n-1}]\}^e$$

Though we can't get the further form of this equation, we can see that the detection probability is closely related with the memory size and weight. Their relations are analysed in Figure 4 and Figure 5. In the Figure 4, four cases are presented. In first three case designs, the detection probabilities of encoders increase with increasing of weight. However, large weight isn't always a good choice. In the forth case, *P* of encoder reaches maximum point(71.5%) only when weight is equal to 9. The analysis of Figure 4 tells that that the weight must be picked up carefully which affects the detection probability significantly. To four cases of in Figure 4, the weight in the range of [7~13] is appreciated. The curves in Figure 4 also indicate which time a DFT technique is needed to be used to bound X generates. When the X-density is about 0.15%（u=10), if we select proper weight, the X-Masking probability can be reduced to 0.01%. It can be accepted in the practical scene. However, when X-density increase to 0.78% (u=50), the minimal X-Masking should reach 29%. It is too large to accept. At this time, some DFT logics must be augmented to reduce X density to an acceptable level.

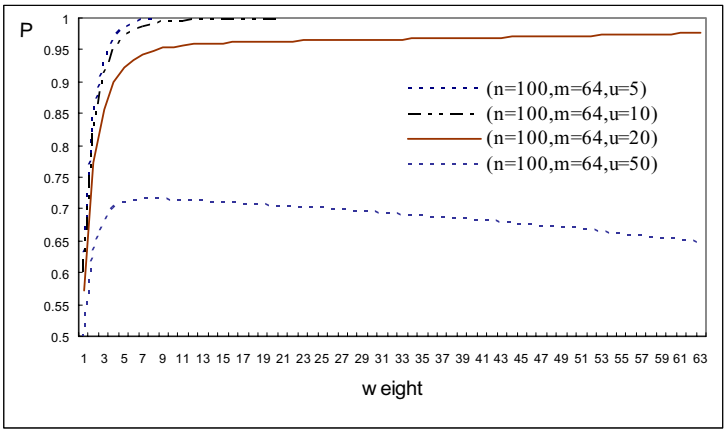

Figure 4. The detection probability vs. weight and X-density

The Figure 5 analyses the relation between the detection probability and memory size. This relation is simple and monotone. The larger memory size always comes to good.

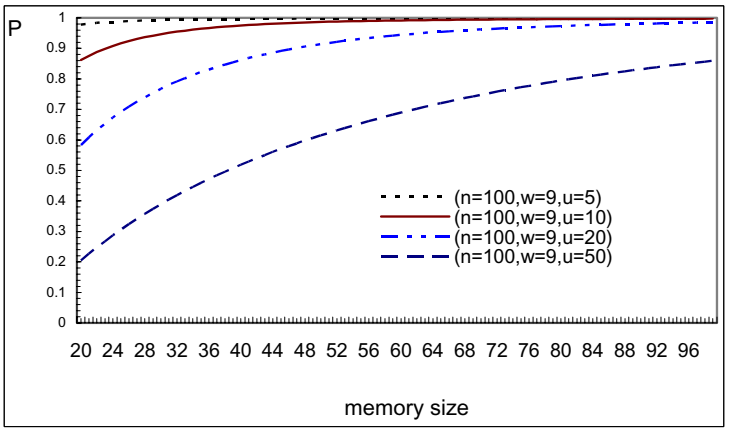

Figure 5. The detection probability vs. weight and X-density

In the above analysis procedure, we assume that the distribution of X bits is random. However, many observations of X bits in industrial chips tell us that the X bits in test response are largely clustered. This means that the majority of X bits are produced by only a small number of scan chains. The detailed experimental results are reported in [9].

The probability of one error masked by X bits is also related with the time that the X bits reside in the memory elements. This time is determined by the column polynomial of basic check matrix. Let's examine the type I encoder. If one X bit is spreaded into the registers which are near the output, the time of X bits in memory elements are short. These X bits have little force to the erroneous signature. Seeing an example, if one X bit is produced in scan out pin I_i, and it is spreaded into 0^{th}, 1^{th}, 2^{th} registers(the column polynomial of I_i in H is $C_i(x)=x^0+x^1+x^2$). Thus, these spreaded X bits will be shifted out during 3 cycles (the 0^{th} register is nearest to the output). If the column polynomial of I_i is $C'_i(x)=x^3+x^4+x^5$, then the spreaded X bits needs 6 cycles to shift out. It is obvious that C_i is superior to $C'_i(x)$. We define the "force" of an input I_i: τ_i to denote the maximum shift-out time when a single X-bit is injected in this input. It is calculated as: $\tau_i = \sum_{l=0}^{l=m-1} (d)\,[C_i^l x^d \mid C_i^l = 1]$, C_I^l is the coefficient of degree l in *i-th* column polynomial C_i, τ_i represents the sum of non-zero coefficient degree

in C_i. Thus, the input with a smaller τ_i can help us to reduce X-Masking probability when the X-density of it is large.

Based the above discussions, following column polynomials generating and input assignment algorithm is presented to achieve the low X-Masking probability:

H- Synthesis and Input assignment algorithm:

(1) Order the outputs of scan chains based on the X-density produced in these outputs; Then, get the ordered outputs set S.

(2) Divide S into two sub-set S_1 and S_2. The sum of X-density of outputs in S_1 exceed a threshold (such as 80%).The other outputs are in S_2.

(3) Select a small ω (such as 3 or 5). Generate the column polynomials with small τ. Order these polynomials and connect inputs corresponding to these polynomials with outputs in S_1.

(4) Select a large ω (such as 9, 11 or larger). Generate the column polynomials random. Order these polynomials and connect inputs corresponding to these polynomials with outputs in S2.

Some experimental results using this algorithm are listed in Table 1. To S_1, we select 3 as ω. To S_2, we select 11 as ω. Case 1, 2 have fewer X bits; Case 5, 6 have massive X bits, and Case 3, 4 are between them. Total 10^8 erroneous bits are random injected and the number of masked erroneous bits are reported in the table. In these cases, the X bits are clustered in some inputs of encoder. Two measurements are conducted for each case. To case 1, 3, 5, main X bits are clustered in one input, we find the X-Masking probability can be reduced drastically when using proposed algorithm. In case 2, 4, 6, the X-Masking probability is also reduced. To see the worst case (case 6), the X-Masking probability is reduced from 3% into 0.05%. It now can be acceptable.

Table 1. The X-masking probability reduction after the proper inputs assignment

Cases	Encoder	X-Density	X Bits Distribution	Random Inputs Assignment	Proper Inputs Assignment
Case 1	(n=100, m=64, d=3)	0.0025%	I_1(90%), other(10%)	369 257	26 22
Case 2	(n=100, m=64, d=3)	0.0025%	I_1(30%), I_2(25%), I_3(10%), I_4(15%), I_5(5%), other(15%)	3267 4657	649 256
Case 3	(n=100, m=64, d=3)	0.025%	I_1(90%), other(10%)	1846 1432	127 308
Case 4	(n=100, m=64, d=3)	0.025%	I_1(30%), I_2(25%), I_3(10%), I_4(15%), I_5(5%), other(15%)	10654 9786	1775 1042
Case 5	(n=100, m=64, d=3)	0.25%	I_1(90%), other(10%)	54123 47899	915 843
Case 6	(n=100, m=64, d=3)	0.25%	I_1(30%), I_2(25%), I_3(10%), I_4(15%), I_5(5%), other(15%)	302796 434523	6832 5091

5. Conclusions

In this paper, we presented a test response compaction technique based on a convolutional code *H_m-based* encoder. The proposed encoder is a single-output encoder, therefore the maximum compaction ratio can be obtained. If the design theorems presented are satisfied, the encoder can benefit from aliasing-resistance, diagnosis and handling massive X bits. The proposed encoder is suited for SOC designs whose a small area overhead penalty can greatly enhance handling of X-bits.

The experimental results show that the weight and memory size of code are two very important parameters to optimize the performance of the encoder. The large memory size is always better without considering the area overhead. If an optimized weight and memory size are selected, the X-Masking probability can be reduced drastically and to an acceptable level.

6. Acknowledgement

The author would like to thank the anonymous reviewer of DFT 2004 to indicate that the proposed code is Wyner-Ash Codes. This comment helps us to go through our paper and improve the theory foundation.

7. References

1. A. Ivanov, B. Tsuji, Y. Zorian, "Programmable Space Compactors for BIST", *IEEE Transaction on Computer,* No.12, pp. 1393-1405, Dec.1996.
2. K. Chakrabarty, B. T. Murray, and J. P. Hays, "Optimal zero-aliasing space compaction of test responses", *IEEE Transaction on Computer*, vol. 47, No. 11, pp.1171-1187, Nov. 1998.
3. B. B. Bhattacharya, A. Dmitriev, M.Goessel, "Zero-Aliasing Space Compaction of Test Responses Using a Single Periodic Output", IEEE Transaction on Computers, VOL.52, NO. 12, pp. 1646-1651, Dec. 2003.
4. K. K. Saluja and M. Karpovsky, "Testing computer hardware through data compression in space and time", in *Proc. of International Test Conference,* pp.83-88, 1983.
5. S. Mitra and K. S. Kim, "X-Compact: An Efficient Response Compaction Technique", *IEEE Transactions on CAD&CS*, 2004, 23(3):421-432.
6. J. H. Patel, S. S. Lumetta, S. M. Reddy, "Application of Saluja-Karpovsky Compactors to Test Responses With Many Unkowns", in *Proc. of 21st VLSI Test Symposium,* pp. 107-112, 2003.
7. S. S. Lumetta, S. Mitra, "X-Codes: Error Control with Unknowable Inputs", in *Proc. of the International Symposium on Information Theory*, pp. 102, 2003.
8. C. Wang, S. M. Reddy, I. Pomeranz, J. Rajski, J. Tyszer, "On Compacting Test Response Data Containing Unknown Values", in *Proc. of International Conference on CAD*, pp. 855-862, 2003.
9. J. Rajski, J. Tyszer, C. Wang, S. M. Reddy, "Convolutional compaction of test responses", in *Proc. of International Test Conference,* pp. 745-754*,* 2003.
10. Y. Han, Y. Xu, H. Li, X. Li, A. Chandra, "Test Resource Partitioning Based on Efficient Responses Compaction for Test Time and Tester Channels Reduction", in *Proc. of 12th Asian Test Symposium ,* pp440-445, 2003.
11. P. Wohl, L. Huisman, "Analysis and design of optimal combinational compactors", in *Proc. of 21st VLSI Test Symposium,* pp.101-106, 2003.
12. I. Pomeranz, I. S. Kundu, and S. M. Reddy, "On Output Response Compression in the Presence of Unknown Output Values", in *Proc. of Design Automation Conference,* pp. 255-258, 2002.
13. P. Wohl and J. A. Waicukauski, T. W. Williams, "Design of Compactors for Signature-Analyzers in Built-IN SELF-TEST", in *Proc. of International Test Conference,* pp. 54-63, 2001.
14. C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, A. Ferko, B. Keller, D. Scott, Koenemann, and T. Onodera, "Extending OPMISR beyond 10x scan test efficiency", *IEEE Design and Test,* 19(5), pp. 65-73, 2002.
15. P. Elias, "Coding for Noisy Channels", *in IRE International Convention Record*, pt. 4, pp. 37-46, 1955.
16. J. L. Massey, D. J. Costello, and J. Justesen, "Polynomial Weights and Code Constructions", *IEEE Transaction on Information Theory*, No.1, pp. 101-110, 1973.
17. J. Justesen, "New Convolutional Code Constructions and a Class of Asymptotically Good Time-Varying Codes", *IEEE Transaction on Information Theory*, 19(2), pp. 220-225, 1973.
18. J. Rosenthal, J. M. Schumacher and E. V. York, "The behavior of convolutional codes", in the report BS-R9533 of National Research Institute for Mathematics and Computer Science in the Netherlands
19. S. Lin, D. J. Costello, "Error Control Coding: Fundamentals and Applications", Published by Prentice Hall, October 1, 1982.