Synthesis of Single-Output Space Compactors for Scan-Based Sequential Circuits

Bhargab B. Bhattacharya, *Senior Member, IEEE*, Alexej Dmitriev, Michael Gössel, and Krishnendu Chakrabarty, *Senior Member, IEEE*

***Abstract*—This paper addresses the problem of space compaction of test responses of combinational and scan-based sequential circuits. In a general circuit, compaction of output space to a single output with zero-aliasing cannot always be achieved by earlier known approaches. In this work, it is shown that given a precomputed test set T, the test responses at the functional outputs of any arbitrary circuit-under-test (CUT) can be compacted to a *single periodic output*, with guaranteed *zero aliasing*. All the errors that are produced by T at the outputs of the CUT will also appear at the output of the compactor. The method is independent of the fault model and the structure of the CUT and uses only the knowledge of the test set T and the corresponding fault-free responses. A new concept of distinguishing outputs and a characteristic function is used to design the compactor. The test vectors in T are appropriately ordered to optimize the compactor logic, which to achieve zero-aliasing uses a test pattern counter to designate the sequence of test application and a special code checker. A design procedure is described to synthesize the compactor using logic synthesis tools, and relevant experimental results on hardware overhead for several benchmark circuits are presented. It is further shown that the overhead can be significantly reduced if the constraint of exact zero aliasing is slightly relaxed.**

***Index Terms*—Scan-based sequential circuits, space compaction, testing.**

I. Introduction

SPACE compaction, which refers to the problem of reducing a wide data stream to a narrow signature stream, is commonly used for test response compression. A typical space compaction scheme for a general circuit-under-test (CUT) is illustrated in Fig. 1. A CUT with n primary inputs $X = \{x_1, x_2, \ldots, x_n\}$, and m primary outputs $Y = \{y_1, y_2, \ldots, y_m\}$ is given, and the m-bit test responses for a test set T are to be compacted to a q-bit wide data stream, $q \ll m$. The *compaction ratio* (m/q) becomes maximum for a single-output space compactor, i.e., when $q = 1$. Space compaction often leads to loss of fault coverage due to *aliasing* that maps a faulty response to a fault-free signature of the CUT.

Earlier attempts toward compactor synthesis include structure-independent methods [2], [8], [12], structure-dependent methods [3]–[7], [9] and those based on coding theory [10]. Complete fault coverage and optimum compaction of test responses cannot be guaranteed for coder-based compactors. The structure-dependent methods, on the other hand, need a fault model and an internal description of the CUT. Their effectiveness for propagating errors produced by nonmodeled faults is unknown. Space compactors are also employed in core-based systems for reducing the volume of test response data in a modular built-in self-test (BIST) environment [20]. Typically, a complex system-on-a-chip (S-o-C) is built with several IP cores whose structural descriptions are not known to the system designer [1]. Consequently, their testing poses a new challenge, as fault simulation cannot be performed during system integration and the controllability and observability of the individual cores at the I/O pins of a S-o-C are low. Thus, structure-dependent space compactors are unsuitable for modern core-based systems. Design of space compactors that do not rely on the structural information of the CUT, and are independent of fault model, has become important with the emergence of core-based system [2], [8], [12].

Three major issues need to be addressed while designing a space compactor: 1) to maximize compaction ratio; 2) to minimize aliasing; and 3) to reduce design complexity and overhead. Multiple-input signature registers (MISR) are widely used for space compaction as they provide high compaction, low overhead, and negligible aliasing. It has been shown recently that exact zero-aliasing space compaction can be achieved with maximum compaction ratio (i.e., $q = 1$) for an arbitrary combinational CUT under a given test set, if the compactor, in addition to being fed by the outputs of the CUT, is also fed by the inputs to the CUT as in Fig. 2 [12]. Unfortunately, this approach is not applicable to a sequential circuit, as the secondary inputs (internal state lines) may not be accessible for synthesizing the compactor. Recently, parity trees [13] and MISR [14] have been shown to be useful for on-chip compaction of scan outputs for large industrial circuits and S-o-C cores. In general, zero-aliasing single-output space compaction for a scan-based sequential circuit is an important open problem.

In this paper, we analytically show that given any arbitrary CUT, the limitation imposed by an earlier logarithmic lower bound on q [8] for zero-aliasing can be overcome by designing a compactor that is fed by the outputs of the CUT and a test

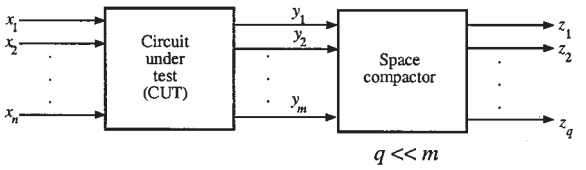

Fig. 1. Conventional space compaction scheme.

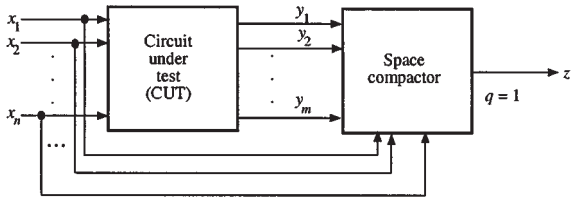

Fig. 2. Modified space compaction scheme using inputs.

pattern counter. For a combinational or a scan-based sequential circuit, we describe a procedure for synthesizing a *zero-aliasing* compactor that shrinks the functional outputs of the CUT to a *single periodic output* ($q = 1$). The outputs of the scan chains are observed without space compaction. Albeit we described the procedure for functional outputs, the method can be readily applied to space compaction of scan outputs as well, and in general to any set of m lines producing an m-bit wide response stream under a test sequence. We assume that the internal structure of the CUT is not known. The proposed method is independent of the fault model and uses only the information of the fault-free responses of the CUT to a precomputed test set T. All the errors produced by T at the outputs of the CUT appear at the compactor output.

We use the notion of *distinguishing outputs (columns)* in the response matrix and define a *characteristic function* to provide a unique signature to each of the fault-free responses to T. To ensure zero-aliasing, a counter that keeps track of test application is used to feed an additional combinational circuit C_{map} called *mapping logic*. In test mode, all the outputs of the CUT are compacted to a single periodic sequence of alternating 0s and 1s by using a comparator circuit and a special CMOS code checker. The scheme is general in nature and is applicable to any CUT whose fault-free responses to a test set are known. Since $q = 1$, the compactor achieves a maximum compaction ratio. Using logic synthesis tools, the proposed compactor is synthesized for several representative ISCAS benchmark circuits (including several large benchmarks) and experimental results on hardware overhead are provided. The overhead is observed to be relatively high, as our method attempts to achieve the maximum compaction with exact zero aliasing. However, the procedure is flexible in the sense that the overhead can be reduced if the condition of exact zero aliasing is slightly relaxed to allow a negligible amount of aliasing.

The rest of the paper is organized as follows. In Section II, we present a synthesis procedure for designing a zero-aliasing single-output space compaction for a sequential circuit. Related issues for minimizing hardware overhead are also discussed. Experimental results are presented in Section III. Conclusions and future directions appear in Section IV.

II. Two-Stage Compaction to a Single Periodic Output

Most of the conventional space compactors [3], [5], [7], [8] use a scheme as in Fig. 1. The lower bound on q for such a zero-aliasing space compactor is given by $q \geq \lceil \log_2(\alpha + 1) \rceil$, where α is the number of distinct fault-free responses of the CUT to the test set T, and therefore, in general, q cannot be made equal to 1, to preserve zero-aliasing [8]. However, as mentioned earlier, if the input (test) information is also used in addition to the outputs (responses) to the CUT to synthesize the compactor, as in Fig. 2, then zero aliasing can be achieved with $q = 1$ [12]. This may lead to increased interconnect area, and , for a sequential circuit, the input information corresponding to the internal state lines may not be accessible. To circumvent this, we propose a new scheme as described below.

We assume that for the given CUT C, a scan test set $T = \{t_1, t_2, \ldots, t_k\}$, $|T| = k$, and the corresponding fault-free output response vectors $Y(t_i)$, $1 \leq i \leq k$ are available. The width of each test vector t_i is thus determined by the number of primary and scan inputs. Let the set of all response vectors be represented by the *response matrix* RM $Y[t_i, y_j]$, $1 \leq j \leq m$. The test vectors are applied to the CUT in a certain sequence. A pattern counter with $\lceil \log_2 k \rceil$ bits is used the state which designates the particular test being applied to the CUT at that instant. The counter starts from the all-0 state and is incremented by the *test clock*, i.e., whenever the application of a test vector to the CUT is completed. The space compactor circuit is fed with the functional outputs $Y = y_1, y_2, \ldots, y_m$ of the CUT and the counter outputs $b_1, b_2, \ldots, b_{\lceil \log_2 k \rceil}$.

The proposed scheme of the compactor to achieve zero aliasing with a single output is shown in Fig. 3. If the CUT is fault-free, then the compactor will generate a *single* periodic (alternating) output z of 0s and 1s (e.g., 010 101$\cdots$), when the test vectors are applied to C in a predefined sequence.

The compactor operates in two stages. In the *first stage*, m functional outputs $y_1, y_2, \ldots, y_m$ are compacted to p outputs $z_1, z_2, \ldots, z_{p-1}, z_p$, where $\lceil \log_2(\alpha + 1) \rceil \leq p \leq m$. We design this stage based on a new concept of distinguishing lines (D) and characteristic function (Ch). The outputs $z_1, z_2, \ldots, z_{p-1}$ correspond to the set D, and z_p implements the characteristic function (see Fig. 3). The outputs of this stage feed the *second stage* compactor, that consists of the following four logic blocks:

1) a test pattern counter with $\lceil \log_2 k \rceil$ bits, whose state identifies the test vector currently in application;
2) an additional combinational logic circuit C_{map} (called mapping logic) driven by the counter state $b_1, b_2, \ldots, b_{\lceil \log_2 k \rceil}$, that generates matching functions $g_1, g_2, \ldots, g_{p-1}, g_p$, corresponding to the p outputs of the first-stage compactor;
3) a comparator block having at most p XOR gates, where the output of the ith gate G_i is given by $h_i = z_i \oplus g_i$, $1 \leq i \leq p$;
4) a self-checking CMOS code checker fed by the lines $h_1, h_2, \ldots, h_p$; it compacts a p-bit wide data stream to a one-bit wide periodic output stream at z. The input code of the checker consists of only two codewords 00$\cdots$0, and 11$\cdots$1, which are mapped to output codewords 0 and 1, respectively.

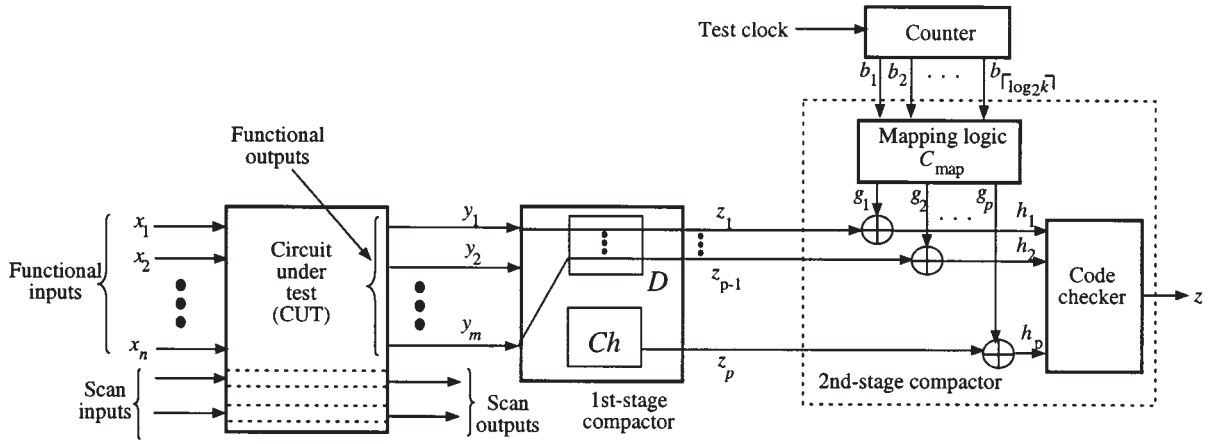

Fig. 3. Scheme of space compactor with a single periodic output z.

A. Design of the First-Stage Compactor

The goal of this stage is to design a simple compactor that first compacts a sequence of m-bit wide test responses at the functional outputs of the CUT to a p-bit wide compacted sequence, where $\lceil \log_2(\alpha+1) \rceil \leq p \leq m$. Thus, to all distinct fault-free responses to the CUT, we assign a unique signature of p bits that can distinguish each response vector $Y(t_i)$ from all other distinct vectors in RM, as well as from all the vectors not contained in RM. To achieve this, we use the notion of *distinguishing columns in RM* and define a special Boolean function called *characteristic function*.

Given the response matrix RM $Y[t_i, y_j]_{k\times m}$, we first choose a minimum set of columns (called *distinguishing columns*) to form a reduced submatrix $Y'_{k\times(p-1)}$, $(p-1) \leq m$, such that for any pair of test vectors t_i, t_l: $Y'(t_i) \neq Y'(t_l) \iff Y(t_i) \neq Y(t_l)$. In other words, each row vector (called *distinguishing vector*) in the reduced matrix $Y'_{k\times(p-1)}$ identifies the vector in the original RM uniquely, distinguishing it from all other distinct rows of RM. The columns of the response matrix actually corresponds to the primary outputs of the CUT. These $(p-1)$ output lines from the CUT corresponding to the distinguishing columns are called *distinguishing outputs or lines* (D); they directly form the outputs $z_1, z_2, \ldots, z_{p-1}$ of the first-stage compactor.

The pth output of the first-stage compactor $z_p(y_1, y_2, \ldots, y_m)$, called the *characteristic function* is given by

$$z_p = \begin{cases} 1, & \forall \text{ response vectors to tests } t_i,\ 1 \leq i \leq k; \\ d, & \text{(don't care), if the distinguishing vector} \\ & \text{in } \{y_1, y_2, \ldots, y_m\} \text{ is different from all} \\ & \text{those in RM;} \\ 0, & \text{otherwise.} \end{cases}$$

In the limiting case, when $p-1=m$, $z_p = 0$.

Thus, z_p becomes 1 for all fault-free response vectors (in RM), and 0 for everything else $\notin$ RM that cannot be distinguished from the vectors in RM by its distinguishing vector alone. The remaining combinations (whose distinguishing vector is different from those in RM) may be set to *don't cares* for logic optimization. The function z_p can be synthesized following the above Boolean description.

TABLE I
RESPONSE VECTORS AND THEIR DISTINGUISHING COLUMNS

Response vectors	CUT outputs						Distinguishing columns (1st-stage compactor outputs)		
	y_1	y_2	y_3	y_4	y_5	y_6	$y_1(=z_1)$	$y_2(=z_2)$	$y_6(=z_3)$
$Y(t_1)$	0	0	0	1	1	0	0	0	0
$Y(t_2)$	0	1	0	1	1	1	0	1	1
$Y(t_3)$	1	0	0	0	1	1	1	0	1
$Y(t_4)$	0	0	0	1	1	0	0	0	0
$Y(t_5)$	0	1	1	0	0	0	0	1	0
$Y(t_6)$	1	1	1	0	1	1	1	1	1
$Y(t_7)$	1	0	0	0	0	0	1	0	0

Example 1: Consider a CUT C with six outputs $y_1, y_2, \ldots, y_6$ whose fault-free response vectors $Y(t_i)$ for seven tests $t_1, t_2, \ldots, t_7$, are shown in Table I. In this example, the response vectors $Y(t_1)$ and $Y(t_4)$ are identical. The distinguishing vector for each distinct response vector can be obtained by choosing columns y_1, y_2, and y_6. These lines directly form the compactor outputs z_1, z_2, and z_3 respectively (see Fig. 4). The characteristic function z_4 is described in Table II. For each of the distinct response vectors, z_4 is set to logic 1. Since, there are six distinct distinguishing vectors, the cubes corresponding to the remaining two ($001ddd$, $110ddd$) are set as don't cares (d). The logic value of z_4 for all the remaining input combinations is set to 0. Thus, the six outputs of the CUT are compacted to four outputs z_1, z_2, z_3, and z_4.

Lemma 1: Given a CUT C with m functional outputs $y_1, y_2, \ldots, y_m$, and the fault-free responses to a test set T, the first-stage compactor with p outputs $z_1, z_2, \ldots, z_{p-1}, z_p$, $\lceil \log_2(\alpha+1) \rceil \leq p \leq m$, designed as above, propagates all errors produced by T in C.

Proof: For every correct response $Y(t_i)$ corresponding to a test t_i, the characteristic function at the compactor output will be 1, and the distinguishing outputs will show the signature of the concerned fault-free response vector. If an error occurs, the erroneous response vector $Y^f(t_i)$ must correspond to either another distinct vector in RM, or a vector $\notin$ RM. In the first case, the error will propagate to the compactor output through at least one bit of the distinguishing vectors. In the second case, if the distinguishing vector of $Y^f(t_i)$ disagrees with those in RM,

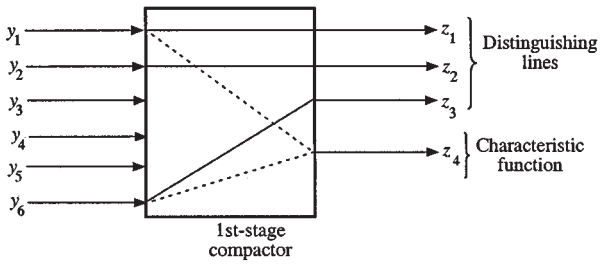

Fig. 4. Distinguishing lines and characteristic function for Example 1.

TABLE II
CHARACTERISTIC FUNCTION z_4 FOR EXAMPLE 1

Distg. columns			Remaining columns			
y_1	y_2	y_6	y_3	y_4	y_5	z_4
0	0	0	0	1	1	1
0	1	1	0	1	1	1
1	0	1	0	0	1	1
0	0	0	0	1	1	1
0	1	0	1	0	0	1
1	1	1	1	0	1	1
1	0	0	0	0	0	1
0	0	1	d	d	d	d
1	1	0	d	d	d	d
Everything else						0

the error is carried through it, else the characteristic function assumes logic 0. Hence, the compactor is zero aliasing for all errors produced by T in C. The bounds on p follow easily. □

Remark 1: The above compactor has $p-1$ distinguishing lines and at most one output for the characteristic function. No logic synthesis effort is required for the first $p-1$ set of functions, as they correspond to certain outputs of the CUT. Only the characteristic function z_p may have to be synthesized to design the compactor.

1) Synthesis of Characteristic Function Using Counter States: For efficient logic synthesis, it may sometimes be more convenient to implement the characteristic function z_p in a slightly different fashion. From the input description of z_p, we remove the variables corresponding to distinguishing columns and then to each row $Y'(t_i)$ of the reduced RM, we add the counter state $b_1, b_2, \ldots, b_{\lceil \log_2 k \rceil}$ corresponding to the test t_i. The truth table is defined in a similar way as before. It is easy to show that this modified scheme also preserves zero aliasing. We illustrate this technique by the following example.

Example 2: The fault-free response matrix for the ISCAS-89 circuit s349 is shown in Table III. The circuit has 11 functional outputs, and the size of the test set is 13 [18]. The distinguishing columns (#5) are shown in Table IV. In Table V, we define the input set of the characteristic function by removing five distinguishing columns, then adding four columns corresponding to the counter bits b_1, b_2, b_3, and b_4. The truth table is described as in the earlier case.

Remark 2: The above design using counter states ensures zero aliasing even if the set of distinguishing columns is replaced by any arbitrary subset of columns of RM. Thus, the size of the total input set of variables defining z_p can be appropriately chosen to fit in available logic synthesis tools. The use of a counter in defining the characteristic function provides

TABLE III
RESPONSE MATRIX OF s349

	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}
$Y(t_1)$	1	1	1	1	1	1	1	1	1	1	0
$Y(t_2)$	0	0	1	1	1	0	0	1	0	0	1
$Y(t_3)$	1	1	0	1	1	1	0	0	0	1	0
$Y(t_4)$	1	1	1	1	0	0	0	1	0	0	1
$Y(t_5)$	0	0	0	0	1	1	1	1	0	1	0
$Y(t_6)$	0	0	0	0	0	0	0	1	0	1	0
$Y(t_7)$	1	0	1	0	1	0	1	1	0	1	0
$Y(t_8)$	1	1	1	1	0	0	0	0	1	1	0
$Y(t_9)$	0	1	0	1	1	1	1	1	0	1	0
$Y(t_{10})$	0	0	1	0	0	1	1	1	0	1	0
$Y(t_{11})$	0	1	0	1	0	1	0	1	0	1	0
$Y(t_{12})$	1	1	0	1	0	0	1	1	0	1	0
$Y(t_{13})$	1	0	1	0	1	0	0	1	0	1	0

TABLE IV
DISTINGUISHING COLUMNS

	y_1	y_2	y_5	y_7	y_8
$Y(t_1)$	1	1	1	1	1
$Y(t_2)$	0	0	1	0	1
$Y(t_3)$	1	1	1	0	0
$Y(t_4)$	1	1	0	0	1
$Y(t_5)$	0	0	1	1	1
$Y(t_6)$	0	0	0	0	1
$Y(t_7)$	1	0	1	1	1
$Y(t_8)$	1	1	0	0	0
$Y(t_9)$	0	1	1	1	1
$Y(t_{10})$	0	0	0	1	1
$Y(t_{11})$	0	1	0	0	1
$Y(t_{12})$	1	1	0	1	1
$Y(t_{13})$	1	0	1	0	1

TABLE V
DESCRIPTION OF THE CHARACTERISTIC FUNCTION USING COUNTER STATES FOR s349

	CUT outputs						Counter bits				Charact. function
	y_3	y_4	y_6	y_9	y_{10}	y_{11}	b_1	b_2	b_3	b_4	
$Y(t_1)$	1	1	1	1	1	0	0	0	0	0	1
$Y(t_2)$	1	1	0	0	0	1	0	0	0	1	1
$Y(t_3)$	0	1	1	0	1	0	0	0	1	0	1
$Y(t_4)$	1	1	0	0	0	1	0	0	1	1	1
$Y(t_5)$	0	0	1	0	1	0	0	1	0	0	1
$Y(t_6)$	0	0	0	0	1	0	0	1	0	1	1
$Y(t_7)$	1	0	0	0	1	0	0	1	1	0	1
$Y(t_8)$	1	1	0	1	1	0	0	1	1	1	1
$Y(t_9)$	0	1	1	0	1	0	1	0	0	0	1
$Y(t_{10})$	1	0	1	0	1	0	1	0	0	1	1
$Y(t_{11})$	0	1	1	0	1	0	1	0	1	0	1
$Y(t_{12})$	0	1	0	0	1	0	1	0	1	1	1
$Y(t_{13})$	1	0	0	0	1	0	1	1	0	0	1
	d	d	d	d	d	d	1	1	0	1	
	d	d	d	d	d	d	1	1	1	0	d
	d	d	d	d	d	d	1	1	1	1	
Remaining combinations											0

us options for a flexible design by choosing the size of p. In the limiting case, no distinguishing lines may be used, i.e., the second-stage compactor is not required. All the m outputs of the CUT and the counter bits will be used to determine the characteristic function z_p, and single-output compaction will be achieved. The other extreme case arises when all m outputs serve as distinguishing lines and feed the second-stage compactor. In this case, the first-stage compactor is not needed. An example of the latter appears while analyzing the circuit s298 (see Table X).

2) Reduction of Hardware Overhead: The first-stage compactor designed as above, provides zero aliasing. However, hardware overhead can be significantly reduced if a small amount of aliasing is tolerated. For example, the counter bits may be eliminated from the input description of the characteristic function z_p, after removing the distinguishing columns, to reduce cost. It can be shown that the resulting aliasing probability is very small since aliasing occurs only if the error pattern matches a pattern from the fault-free responses for the bit positions that are not included in the set of distinguishing lines. This condition is unlikely to be satisfied for most realistic errors. The exact computation of the aliasing probability will depend on the underlying error model.

Furthermore, since most of the errors are propagated through z_p, the probability of error propagation through the distinguishing lines alone is very small for large m. Thus, the number of lines feeding the second-stage compactor can be reduced by using a linear compactor (XOR tree), with slight increase in aliasing [5]. This, in turn, reduces the cost of C_{map} and the checker. Similarly, if synthesis cost of the characteristic function z_p turns out to be too high, its input set can be reduced by eliminating the distinguishing lines, or by precompacting them using a linear compactor. The cost of z_p can also be reduced in many cases by removing a subset of CUT outputs from the input set of z_p and including them in the second-stage compactor, at the expense of slightly increasing the overhead of mapping logic. Hence, the method provides design flexibility with a tradeoff between overhead and aliasing, while achieving $q = 1$. This is discussed later in context to Table XI.

B. Design of the Second-Stage Space Compactor

The second-stage compactor receives $z_1, z_2, \ldots, z_{p-1}, z_p$ as well as the counter state $b_1, b_2, \ldots, b_{\lceil \log_2 k \rceil}$ as inputs, and produces a single periodic output (a string of alternating 0s and 1s) at z, on application of the test vectors in a certain sequence. A similar technique known as output data modification was used earlier to achieve time compaction [11].

Let the test set T be arbitrarily partitioned into two disjoint subsets T_0 and T_1 of almost equal size ($T = T_0 \cup T_1$, $T_0 \cap T_1 = \varnothing$, $|T_0| \geq |T_1| \geq 1$, $|T_0| - |T_1| \leq 1$). Let T_S denote the test sequence $\{t_{01}, t_{11}, t_{02}, t_{12}, \ldots\}$, where $t_{0i} \in T_0$, and $t_{1l} \in T_1$. Thus, T_S is formed by choosing consecutive vectors alternatingly from T_0 and T_1, until all tests in T are exhausted.

1) Design of Mapping Logic C_{map}: The mapping logic C_{map} (see Fig. 3) generates matching outputs g_i, $1 \leq i \leq p$, which are given by

$$g_i = z_i \oplus h_i, \quad 1 \leq i \leq p,$$

where

$$h_1 = h_2 = \cdots = h_p = \begin{cases} 0, & \forall \text{ test } t_i \in T_0 \\ 1, & \forall t_i \in T_1 \end{cases}$$

and

$$h_i = d(\text{don't care}), \qquad \text{otherwise.} \tag{1}$$

The above condition implies that if a test vector $t \in T_0 (t \in T_1)$ is applied to the fault-free CUT, the logic values at the input lines $(h_1, h_2, \ldots, h_m)$ to the code checker will be simultaneously 0 (1), and thus, on application of the test sequence T_S, these inputs receive alternatingly all-0 and all-1. Hence, it follows that $\forall i$, $1 \leq i \leq p$, the output function g_i of C_{map} is given by

$$g_i(t) = \begin{cases} z_i(t) \oplus 0 = z_i(t), & \forall t \in T_0 \\ z_i(t) \oplus 1 = \overline{z_i}(t), & \forall t \in T_1 \\ = d(\text{don't care}), & \text{otherwise.} \end{cases} \tag{2}$$

Since the counter keeps track of the test sequence T_S, and the values of $z_i(t)$ for $t \in T$ are known at the first-stage compactor output from RM, (2) can be readily used to synthesize the mapping logic C_{map} if a suitable partition of T into T_0 and T_1 is determined. For the $(2^{\lceil \log_2 k \rceil} - k)$ additional counter states that do not correspond to the k tests, we set the logic values to g_i, $1 \leq i \leq p$ as don't cares. Logic synthesis tools such as ESPRESSO [16] and SIS [17] can now be used to synthesize C_{map}.

2) Determination of Test Sequence T_S: We use a simple heuristic to determine the test ordering with the goal of reducing the overhead of C_{map}. We implement this by reducing the number of 1s in the truth-table description of C_{map}, which may reduce the cost of its implementation.

Let $Z(Y(t))$ denote the vector $\{z_1, z_2, \ldots, z_p\}$ at the output of the first-stage compactor corresponding to the response vector $Y(t)$ to test t. Let $N_1(Z) = \sum_{i=1}^{p} z_i$ denote the Hamming weight, i.e., the number of 1s in the vector $Z(Y(t))$.

We order $(<)$ the test vectors ts with respect to nondecreasing values of their $N_1(Z)$, i.e., $N_1(Z(Y(t_i))) < N_1(Z(Y(t_j)) \Rightarrow t_i < t_j$, and $t_i < t_j \Rightarrow N_1(Z(Y(t_i))) \leq N_1(Z(Y(t_j)))$

$$\text{We choose,} \quad T_0 = \left\{t^1, t^2, \ldots, t^{\lceil k/2 \rceil}\right\}$$
$$T_1 = \left\{t^{\lceil k/2 \rceil + 1}, \ldots, t^k\right\}$$

where $\lceil k/2 \rceil$ is the smallest integer larger than or equal to $k/2$.

This makes $|T_0| = |T_1|$, if k is even, and $|T_0| = |T_1| + 1$, if k is odd, and the length of the test sequence T_S would be $(|T_0| + |T_1| = |T| = k)$. Hence, the choice of such a partition will imply that the total number of 1s in the output description of C_{map} as given by (1) and (2) is minimum.

We now give an example to illustrate the design process.

Example 3: Consider the circuit of Example 1. The response matrix, distinguishing columns, and the characteristic function are shown in Tables I and II earlier. Table VI shows ordering of test vectors in terms of the nondecreasing sequence of the Hamming weights of their compacted outputs z_1, z_2, z_3, z_4, from which the sets T_0, and T_1 are determined. In Table VII, the test

TABLE VI
ORDERING OF TEST VECTORS

Tests	1st-stage compactor outputs				$N_1(Z)$	Test set partition
	z_1	z_2	z_3	z_4		
t_1	0	0	0	1	1	
t_4	0	0	0	1	1	T_0
t_5	0	1	0	1	2	
t_7	1	0	0	1	2	
t_2	0	1	1	1	3	
t_3	1	0	1	1	3	T_1
t_6	1	1	1	1	4	

TABLE VII
TEST SEQUENCING AND THE REQUIRED OUTPUTS OF C_{map}

Test sequence	1st-stage compactor outputs				Required checker inputs				Required outputs of C_{map}			
T_S	z_1	z_2	z_3	z_4	h_1	h_2	h_3	h_4	g_1	g_2	g_3	g_4
t_1	0	0	0	1	0	0	0	0	0	0	0	1
t_2	0	1	1	1	1	1	1	1	1	0	0	0
t_4	0	0	0	1	0	0	0	0	0	0	0	1
t_3	1	0	1	1	1	1	1	1	0	1	0	0
t_5	0	1	0	1	0	0	0	1	0	1	0	1
t_6	1	1	1	1	1	1	1	1	0	0	0	0
t_7	1	0	0	1	0	0	0	0	1	0	0	1

TABLE VIII
ASSIGNMENT OF COUNTER STATES AND THE TRUTH TABLE OF C_{map}

Test sequence	Assigned counter state			Outputs of C_{map}			
T_S	b_3	b_2	b_1	g_1	g_2	g_3	g_4
t_1	0	0	0	0	0	0	1
t_2	0	0	1	1	0	0	0
t_4	0	1	0	0	0	0	1
t_3	0	1	1	0	1	0	0
t_5	1	0	0	0	1	0	1
t_6	1	0	1	0	0	0	0
t_7	1	1	0	1	0	0	1
—	1	1	1	d	d	d	d

TABLE IX
REDUCING COST OF C_{map} BY CHOOSING A DIFFERENT T_S

Test sequence	Assigned counter state			Outputs of C_{map}			
T_S	b_3	b_2	b_1	g_1	g_2	g_3	g_4
t_7	0	0	0	1	0	0	1
t_2	0	0	1	1	0	0	0
t_4	0	1	0	0	0	0	1
t_6	0	1	1	0	0	0	0
t_5	1	0	0	0	1	0	1
t_3	1	0	1	0	1	0	0
t_1	1	1	0	0	0	0	1
—	1	1	1	d	d	d	d

sequence $T_S = \{t_1, t_2, t_4, t_3, t_5, t_6, t_7\}$ is shown that is obtained by choosing consecutive vectors alternatingly from T_0 and T_1. The corresponding outputs of C_{map} to satisfy the desired checker inputs are shown in column 4. Table VIII shows the assignment of counter states to the tests and the truth-table describing the outputs of C_{map}. The Boolean functions for these outputs are thus given by $g_1 = b_2b_3 + b_1\overline{b}_2\overline{b}_3$; $g_2 = b_1b_2 + \overline{b}_1\overline{b}_2b_3$; $g_3 = 0$; $g_4 = \overline{b}_1$.

Remark 3: Construction of T_S may have a strong influence on the cost of C_{map}. For instance, in Table IX, another test sequence $T_S = \{t_7, t_2, t_4, t_6, t_5, t_3, t_1\}$ is chosen from the sets T_0, T_1. Realization of this truth table leads to a cheaper implementation of C_{map}: $g_1 = \overline{b}_2\overline{b}_3$; $g_2 = \overline{b}_2b_3$; $g_3 = 0$; $g_4 = \overline{b}_1$.

3) Comparator and the Code Checker: The comparator block consists of at most p XOR gates. If an output function of the mapping logic is constant, the corresponding XOR gate is not needed.

We use a special CMOS gate to implement the code checker as shown in Fig. 5. As observed in [15], this code checker can admit large fan-in. In test mode, all the p inputs to the checker would be either 0 ($\forall t \in T_0$) or 1 ($\forall t \in T_1$). In the first case, all the p-MOS (n-MOS) transistors at the first level of the CMOS gate will be ON (OFF), and in the second case, they will be OFF (ON). Therefore, the p-bit vector $00\cdots0$ ($11\cdots1$) arriving at the input of the checker will be compacted to $0(1)$ at the output z. Therefore, application of T_S to the CUT will generate a periodic output sequence $010\,101\cdots$ at z, in the absence of any error. Further, this checker is self-checking. The two codewords ($00\cdots0$, $11\cdots1$) are sufficient to detect all s-a-0, s-a-1 faults at the inputs/outputs of the checker, and all stuck-open faults in the transistors. All single transistor stuck-on faults except those in the inverter, are also detected by these two vectors.

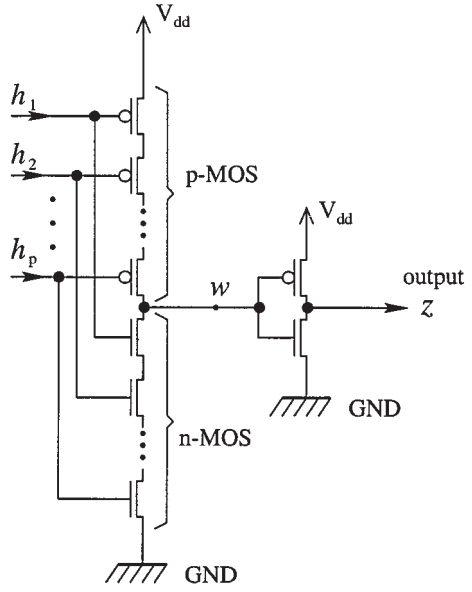

Fig. 5. CMOS implementation of a self-checking code checker.

4) Zero-Aliasing Space Compaction: By Lemma 1, all the errors produced by T in the CUT are propagated to the outputs $Z = \{z_1, z_2, \ldots, z_{p-1}, z_p\}$ of the first-stage compactor. Design of C_{map} according to (1) and (2) ensures that $\forall t \in T_0(T_1)$, the checker inputs $h_1 = h_2 = \cdots = h_p = 0(1)$, if and only if the vector Z is error free. When an error changes the vector Z under a test t, two cases may arise. If it causes the inputs to the checker to receive a vector consisting of 0s and 1s, then at the first level of CMOS gate, both the paths from V_{dd} to w, and w to GND (Fig. 5) will be open. The logic value at w will be floating and will show the previous value. On the other hand, if the error inverts all the checker inputs simultaneously, (i.e., $11\cdots1$ instead of $00\cdots0$, or vice-versa), then also the output at z becomes opposite to its expected value. In both the cases, therefore, the alternating nature (periodicity) of the output at z is lost.

From the above discussion, the next theorem follows.

Theorem 1: Given a test set T for an arbitrary CUT C, the proposed design yields a single-output zero-aliasing space compactor for all errors produced by T in C.

It may be noted that most of the errors propagate through the line corresponding to the characteristic function z_p. However, $\forall t \in T$, the logic value at the output z_p becomes 1, if no error

TABLE X
HARDWARE OVERHEAD OF THE COMPACTOR WITH $q = 1$, AND ZERO ALIASING

Circuit	Number of inputs/outputs/ flip-flops	Circuit area	Test length $\|T\|$	Number of distingu. lines	Percentage area overhead		
					Charact. function	Mapping logic C_{map}	Total
c499	41/32/0	616	52	2	10.06	1.62	11.69
c3540	50/22/0	2934	84	11	16.90	15.20	32.10
c5315	178/123/0	4369	37	7	41.31	3.20	44.51
c7552	207/108/0	6095	73	9	42.46	5.23	47.69
c6288	32/32/0	4800	12	4	3.54	0.56	4.10
s298	3/6/14	300	23	6	0	20.33	20.33
s344	9/11/15	329	13	5	16.41	7.29	23.70
s349	9/11/15	333	13	5	15.32	10.21	25.53
s1423	17/5/14	1460	20	4	1.10	2.47	3.57
s9234.1	36/39/211	8815	105	26	8.03	14.36	22.39
s35932	35/320/1728	35181	12	43	6.76	1.09	7.85
s38417	28/106/1636	38790	68	26	10.74	2.6	13.34

TABLE XI
REDUCED OVERHEAD FOR $q = 1$, WITH SMALL ALIASING

Circuit	Percentage area overhead		
	Charact. function	Mapping logic C_{map}	Total
c3540	9.17	9.30	18.47
c5315	14.47	4.32	18.79
c6288	3.00	0.56	3.56
c7552	15.10	6.20	21.30
s344	5.17	7.29	12.46
s349	3.90	10.21	14.11
s1423	0	2.47	2.47

is present. A stuck-at one fault at the output line z_p will therefore never be detected by any test in T, and its presence could have a catastrophic masking effect on all the errors propagating through it. To overcome this, we add one additional vector t' in T that produces a response vector not in RM, to enforce a 0 at the output z_p, prior to the synthesis procedure.

III. EXPERIMENTAL RESULTS

The proposed method for designing zero-aliasing single-output space compactors was implemented on a SUN Ultra-5/Solaris workstation and applied to several ISCAS-85 and ISCAS-89 benchmark circuits. In each case, we used compact test sets either generated by the MINTEST program [18] or by Hansen and Hayes [19]. A simple heuristic was used to determine a minimal set of distinguishing columns. First, ESPRESSO [16] was used for two-level synthesis of the characteristic function and the mapping logic. This was followed by multilevel minimization using SIS [17]. Since these tools are unable to handle very large designs, we partitioned the outputs for the larger benchmark circuits in order to ensure that the synthesis scripts ran to completion. For example, we partitioned the 320 functional outputs of s35932 into ten partitions of 32 outputs each.

The experimental results for zero-aliasing space compaction with $q = 1$ are shown in Table X. We report the hardware overhead for five combinational and seven sequential circuits (including two of the larger benchmarks). These figures include the area for the mapping logic and the characteristic function. The area due to the code checker, pattern counter, and comparators is not included since these are generic functional blocks that are not tailored to any CUT and can be reused across the complete design. The hardware overhead is related to the test length and the nature of response matrix. For example, the area overhead is only 4.10% for c6288 and 7.85% for s35932. Both these circuits possess very compact test sets. However, the overhead is high for some circuits, e.g., c5315 and c7552, as they have a large test set and a relatively small silicon area. Further, they have a large number of outputs to be compacted to a single stream. It may be noted that the first stage alone is a zero-aliasing compactor with p lines, where p is the number of distinguishing lines plus 1. For example, using only the first stage, 320 outputs of s35932 can be compacted to 44 lines, with no loss of test information.

In order to reduce overhead, several techniques in various combinations as discussed earlier in Section II can be employed. They do not guarantee exact zero aliasing; nevertheless, an almost negligible amount of aliasing may be preferable if the compactor area is reduced significantly. This is indeed the case for many circuits. For example, Table XI shows that the area overhead for s344 is reduced from 23.70% to 12.46%. The design still allows compaction to a single output regardless of the size of the output space.

IV. CONCLUSION

We have shown analytically that the functional outputs of a scan-based sequential circuit can be compacted to a single periodic output with guaranteed zero aliasing. A new synthesis procedure is then described for designing the space compactor. The proposed technique relies only on the knowledge of the fault-free responses of the CUT to a precomputed test set. It does not require access to a gate-level model or any other form of structural information, either for fault simulation, test generation, or for calculating error propagation probabilities. It may, therefore, find applications to space compaction in embedded cores. For example, it would be useful in enhancing test access mechanisms (TAM) for IP cores. Several techniques are

used in industry to access embedded cores in an S-o-C [1]. Two industrial TAM designs recently used in Philips are based on multiplexing and distribution architectures [21]. The number of I/O pins required for TAM can be significantly reduced if space compaction is employed at the outputs of the core. Reduction of TAM outputs allows more pins to be used as TAM inputs, thereby reducing testing time. Similarly, space compaction may also improve TAM distribution architecture. While the method can be used to design zero-aliasing compactors, the flexibility inherent in the synthesis procedure allows the designer to reduce hardware overhead by tolerating a small amount of aliasing. The proposed compactor can be improved in a number of ways. We are currently investigating on the relationship between the compactor area and the partition of the set of CUT outputs for optimizing mapping logic and characteristic function. We expect the area of the compactor to decrease significantly if a suitable partition as well as a matching reordering of the test sequence is found. Although zero-aliasing single-output compaction has been shown here to be theoretically possible, a low-overhead improved practical design based on this approach is needed for a viable silicon solution.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for many constructive comments that helped considerably to improve the presentation of the paper.

REFERENCES

[1] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded core based system chips," in *Proc. Int. Test Conf.*, 1998, pp. 130–143.

[2] M. Gössel, E. Sogomonyan, and A. Morosov, "A new totally error propagating compactor for arbitrary cores with digital interfaces," in *Proc. VTS*, 1999, pp. 49–56.

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

[4] M. Seuring, M. Gössel, and E. Sogomonyan, "A structural approach for space compaction for concurrent checking and BIST," in *Proc. VTS*, 1998, pp. 354–361.

[5] K. Chakrabarty and J. P. Hayes, "Test response compaction using multiplexed parity trees," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 1399–1408, Nov. 1996.

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

[7] K. Chakrabarty, B. T. Murray, and J. P. Hayes, "Optimal zero-aliasing space compaction of test responses," *IEEE Trans. Comput.*, vol. 47, pp. 1171–1187, Nov. 1998.

[8] M. Seuring and K. Chakrabarty, "Space compaction of test responses for IP cores using orthogonal transmission functions," in *Proc. VTS*, 2000, pp. 213–219.

[9] J. Savir, "On shrinking wide compressors," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 1379–1387, 1995.

[10] S. M. Reddy, K. K. Saluja, and M. G. Karpovsky, "A data compression technique for built-in self-test," *IEEE Trans. Comput.*, vol. C-37, pp. 1151–1165, Sept. 1988.

[11] Y. Zorian and V. K. Agarwal, "Optimizing error masking in BIST by output data modification," *J Electron. Testing: Theory Applicat.*, vol. 1, pp. 59–71, 1990.

[12] B. B. Bhattacharya, A. Dmitriev, and M. Gössel, "Zero aliasing space compaction using a single periodic output and its application to testing of embedded cores," in *Proc. Int. Conf. VLSI Design*, 2000, pp. 382–387.

[13] G. Hetheringten, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic BIST for large industrial designs," in *Proc. Int. Test Conf.*, 1999, pp. 358–367.

[14] H. Lang, J. Pfeiffer, and J. Maguire, "Using on-chip test pattern compression for full scan SoC designs," in *Proc. ETW*, 2000, pp. 47–52.

[15] S. Kundu, E. S. Sogomonyan, M. Gössel, and S. Tarnick, "Self-checking comparator with one periodic output," *IEEE Trans. Comput.*, vol. 45, pp. 379–380, Mar. 1996.

[16] R. K. Brayton, G. D. Hatchel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Boston, MA: Kluwer, 1984.

[17] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, A. Saldanha, H. Savoj, P. R. Stephan, and A. L. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Electronic Res. Lab., Tech. Rep. UCB/ERL M92/41, 1992.

[18] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithm for combinational circuits," in *Proc. ICCAD*, Nov. 1998, MINTEST-[Online]. Available: http://www.crhc.uiuc.edu/IGATE, pp. 283–289.

[19] M. C. Hansen and J. P. Hayes, "High-level test generation using physically-induced faults," in *Proc. VTS*, 1995, pp. 20–28.

[20] J. Rajski and J. Tyszer, *Arithmetic Built-In Self-Test for Embedded Systems*. Englewood Cliffs, NJ: Prentice Hall, 1998, pp. 49–50.

[21] J. Aerts and E. J. Marinissen, "Scan chain design for test time reduction in core-based IC's," in *Proc. Int. Test Conf.*, 1998, pp. 448–457.

Bhargab B. Bhattacharya (SM'94) received the B.Sc. degree in physics from the Presidency College, Calcutta, India, in 1971, the B.Tech. and M.Tech. degrees in radiophysics and electronics, and the Ph.D. degree in computer science, all from the University of Calcutta, in 1974, 1976, and 1986 respectively.

Since 1982, he has been on the faculty of the Indian Statistical Institute, Calcutta, where he became Full Professor in 1991. He had been with the Department of Computer Science and Engineering, University of Nebraska-Lincoln, from 1985 to 1987 and 2001–2002, as a Visiting Professor. In the summers of 1998 through 2000, he visited the Fault-Tolerant Computing Group, Institute of Informatics, at the University of Potsdam, Germany. His research interests include logic synthesis, testing, and physical design of VLSI circuits, graph algorithms, and image processing. He has published more than 100 papers in archival journals and refereed conference proceedings. Currently, he is collaborating with Intel Corporation, USA, and IRISA, France, for development of image processing hardware and technology mapping tools for designing reconfigurable processors.

Dr. Bhattacharya is a Fellow of the Indian National Academy of Engineering. He served on the conference committees of the International Test Conference (ITC), the Asian Test Symposium (ATS), the VLSI Design and Test Workshop (VDAT), the International Conference on Advanced Computing (ADCOMP), and the International Conference on High-Performance Computing (HiPC). For the International Conference on VLSI Design, he served as Tutorial Co-Chair in 1994, as Program Co-Chair in 1997, and as General Co-Chair in 2000.

Alexej Dmitriev graduated in electrical engineering from the Sankt-Petersburg Railway University, Russia, in 1994.

Since 1995, he has been with the Fault-Tolerant Computing Group at the University of Potsdam, Germany. His research interests include design of fault-tolerant circuits, error detection, and output compaction. He has published several research papers in these areas.

Michael Gössel received the Dr.rer.nat. degree in physics from the University of Jena and the Dr.sc.nat. degree in computer science from the University of Dresden, Germany.

From 1969 to 1991, he was with the Institute of Cybernetics of the Academy of Sciences, Berlin. From 1992 to 1996, he was the Head of the Fault-Tolerant Computing Group of the Max Planck Society at the University of Potsdam. Since 1994, he has been a Full Professor of computer architecture and fault-tolerance at the University of Potsdam, Germany. He has published over 140 research papers in the fields of automata theory, nonlinear systems, parallel memories, image processing, and fault-tolerant computing. He is the author and coauthor of eight books, including (with S. Graf) *Error Detection Circuits* (New York: McGraw-Hill, 1993), (with B. Rebel and R. Creutzburg) *Memory Architecture and Parallel Access* (New York: Elsevier 1994), and (with V. and Vl. Saposhnikov) *Self-Dual Discrete Systems* (in Russian) (Energoatomisdat, 2001).

Dr. Gössel is an Editor of the *Journal of Electronic Testing: Theory and Applications* (JETTA).

Krishnendu Chakrabarty (S'92–M'96–SM'00) received the B. Tech. degree from the Indian Institute of Technology, Kharagpur, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively, all in computer science and engineering.

He is now an Assistant Professor of Electrical and Computer Engineering at the Duke University, Durham, NC. He is also a Mercator Visiting Professor at the University of Potsdam, Germany. His current research projects are in system-on-a-chip test, embedded real-time operating systems, distributed sensor networks, and architectural optimization of microelectrofluidic systems. He has published over 90 papers in archival journals and refereed conference proceedings, and he holds a U.S. patent in built-in self-test.

Dr. Chakrabarty is a recipient of the National Science Foundation Early Faculty (CAREER) award and the Office of Naval Research Young Investigator award. He is an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, an Editor of the *Journal of Electronic Testing: Theory and Applications* (JETTA), and the Guest Editor of a special issue of JETTA on system-on-a-chip testing. He was also a guest editor in 2001 of a special issue of the *Journal of the Franklin Institute* on distributed sensor networks. He is a member of ACM and ACM SIGDA and a member of Sigma Xi. He serves as Vice Chair of Technical Activities in IEEE's Test Technology Technical Council and is a member of the program committees of several IEEE/ACM conferences and workshops.