Test Set Embedding for Deterministic BIST Using a Reconfigurable Interconnection Network

Lei Li and Krishnendu Chakrabarty, *Senior Member, IEEE*

***Abstract*—We present a new approach for deterministic built-in self-test (BIST) in which a reconfigurable interconnection network (RIN) is placed between the outputs of a pseudorandom pattern generator and the scan inputs of the circuit under test (CUT). The RIN, which consists only of multiplexer switches, replaces the phase shifter that is typically used in pseudorandom BIST to reduce correlation between the test data bits that are fed into the scan chains. The connections between the linear-feedback shift-register (LFSR) and the scan chains can be dynamically changed (reconfigured) during a test session. In this way, the RIN is used to match the LFSR outputs to the test cubes in a deterministic test set. The control data bits used for reconfiguration ensure that all the deterministic test cubes are embedded in the test patterns applied to the CUT. The proposed approach requires very little hardware overhead, only a modest amount of CPU time, and fewer control bits compared to the storage required for reseeding techniques or for hybrid BIST. Moreover, as a nonintrusive BIST solution, it does not require any circuit redesign and has minimal impact on circuit performance.**

***Index Terms*—Embedded core testing, system-on-a-chip (SoC) testing, test application time, test-data volume.**

I. Introduction

HIGHER circuit densities and ever-increasing design complexity are placing a severe burden on the automatic test equipment (ATE) used to test today's integrated circuits (ICs). The integration of complex embedded cores in system-on-a-chip (SOC) designs is leading to a sharp increase in test-data volume, which requires significant investment in additional memory depth per ATE channel. ATE channel bandwidth is another limitation encountered in the testing of SOCs with high clock frequencies, enormous test-data volume, and a large number of I/O pins. In order to mitigate these problems, a number of techniques based on test-data compression, built-in self-test (BIST), and a combination of the two have been proposed in the literature.

In the test-data compression approach, a deterministic test set is compressed and stored in ATE memory. The compressed test set is transferred through ATE channels to the IC, where it is decompressed and applied to the circuit under test (CUT) by decoding hardware. Techniques based on statistical coding [13], [15], run-length coding [14], Golomb coding [6], frequency-directed run-length (FDR) coding [5], and variable-input Huffman coding (VIHC) coding [9], have been proposed to reduce test-data volume. Test-data volume-reduction techniques based on on-chip pattern decompression are also presented in [4], [8], [21], [27]–[29], and [32].

The resurgence of interest in test-data compression has also led to new commercial tools that can provide substantial compression for large industrial designs. For example, the OPMISR [3] and SmartBIST [17] tools from IBM and the TestKompress tool from Mentor Graphics [26] reduce test-data volume and testing time through the use of test-data compression and on-chip decompression.

In BIST solutions, test patterns are generated by an on-chip pseudorandom pattern generator, which is usually a linear feedback shift register (LFSR). BIST alleviates a number of problems related to test interfacing, e.g., limited-signal bandwidth and high pin count. A typical logic BIST architecture is shown in Fig. 1. In order to detect the random-pattern resistant faults and achieve complete coverage of single stuck-at faults, techniques based on test point insertion [7], [30], reseeding [2], [11], [22], [24], bit-flipping [34], bit-fixing [20], [31], [35], and weighted random pattern testing [33] have been proposed. Test-point insertion techniques require design changes to improve random pattern testability, such that 100% fault coverage can be achieved using a reasonable number of pseudorandom test patterns. The other BIST techniques are nonintrusive in that they typically apply a limited number of random patterns; for the remaining hard-to-test faults, deterministic test patterns are obtained by either controlling the state of the pattern generator [11], [18], [22], [24] or by altering the output of the pattern generator [31], [33]–[35]. A number of studies have also been reported recently on the use of logic BIST for large industrial circuits [12], [23].

Techniques based on the combination of data compression and BIST have also been developed recently [16], [19]. The hybrid BIST scheme presented in [16] applies weighted pseudorandom patterns to the circuit to achieve 100% fault coverage. The compressed weight set is stored on ATE and decompression is carried out using an on-chip look-up table. In [19], the seeds for the LFSR are compressed using statistical coding.

In this paper, we present a new deterministic BIST approach in which a reconfigurable interconnection network (RIN) is placed between the outputs of the LFSR and the inputs of the scan chains in the CUT. The RIN, which consists only of multiplexer switches, replaces the phase shifter that is typically

Manuscript received June 20, 2003; revised December 3, 2003. This research was supported in part by the National Science Foundation under Grants CCR-9875324 and CCR-0204077, and in part by a graduate fellowship from the Design Automation Conference. A preliminary version of this paper appeared in *Proceedings of the International Test Conference*, pp. 460–469, Charlotte, NC, Sept./Oct. 2003. This paper was recommended by Associate Editor S. Hellebrand.

The authors are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: ll@ee.duke.edu; krish@ee.duke.edu).

Digital Object Identifier 10.1109/TCAD.2004.831593

used in pseudorandom BIST to reduce correlation between the test-data bits that are fed into the scan chains. The connections between the LFSR and the scan chains can be dynamically changed (reconfigured) during a test session. In this way, the RIN is used to match the LFSR outputs to the test cubes in a deterministic test set. The control data bits used for reconfiguration ensure that all the deterministic test cubes are embedded in the test patterns applied to the CUT. The proposed approach requires very little hardware overhead, only a modest amount of CPU time, and fewer control bits compared to the storage required for reseeding techniques or for hybrid BIST. Moreover, as a nonintrusive BIST solution, it does not require any circuit redesign and has minimal impact on circuit performance.

The rest of the paper is organized as follows. Section II presents an overview of related prior work. In Section III, we present the architecture of the proposed BIST scheme and describe the procedure for the synthesis of the RIN and the determination of the control bits. In Section IV, we present a probabilistic analysis of the test set embedding technique. In Section V, we describe a strategy for declustering the care bits in the test cubes to improve the efficiency of the proposed method. Experimental results and a comparison with related recent work are presented in Section VI. Finally, Section VII concludes the paper.

II. Related Prior Work

Most BIST techniques rely on the use of a limited number of pseudorandom patterns to detect the random-pattern-testable faults, which is subsequently followed by the application of a limited number of deterministic patterns to detect the random-pattern-resistant faults. Based on the mechanisms that are used to generate the deterministic patterns, logic BIST techniques can be classified into two categories: methods that generate deterministic patterns by controlling the states of the LFSR [11], [18], [22], [24], and techniques that modify the patterns generated by the LFSR [31], [33], [34].

LFSR reseeding is an example of a BIST technique that is based on controlling the LFSR state. LFSR reseeding can be static, i.e., the LFSR stops generating patterns while loading seeds, or dynamic, i.e., test generation and seed loading can proceed simultaneously. The length of the seeds can be either equal to the size of the LFSR (full reseeding) or less than the size of the LFSR (partial reseeding). In [18], a dynamic reseeding technique that allows partial reseeding is proposed to encode test vectors. An LFSR of length $r \geq s_{\max} + 20$, where $s_{\max}$ is the maximum number of specified bits in any deterministic test cube, is used to generate the test patterns. While the length of the first seed is r, the lengths of the subsequent seeds are significantly smaller than r. A set of linear equations is solved to obtain the seeds, and the test vectors are reordered to facilitate the solution of this set of linear equations.

A BIST pattern generator based on a folding counter is proposed in [11]. The properties of the folding counter are exploited to find the seeds needed to cover the given set of deterministic patterns. Width compression is combined with reseeding to reduce the hardware overhead. In [22], a two-dimensional test-data compression technique that combines an LFSR and a folding counter is proposed for scan-based BIST. LFSR reseeding is used to reduce the number of bits to be stored for each pattern (horizontal compression) and folding counter reseeding is used to reduce the number of patterns (vertical compression).

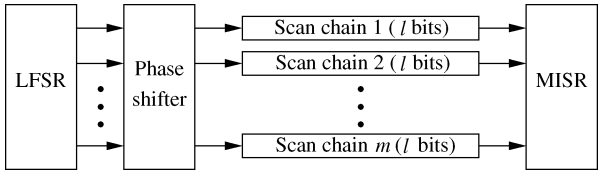


Fig. 1. Generic logic BIST architecture based on an LFSR, MISR, and phase shifter.

Bit-flipping, bit-fixing and weighted random BIST are examples of techniques that rely on altering the patterns generated by the LFSR to embed deterministic test cubes. In [16], a hybrid BIST method based on weighted pseudorandom testing is presented. A weight of 0, 1 or u (unbiased) is assigned to each scan chain in CUT. The weight sets are compressed and stored on the tester. During test application, an on-chip look-up table is used to decompress the data from the tester and generate the weight sets. A 3-weight weighted random scan-BIST scheme is discussed in [33]. The weights in this approach are 0, 0.5, and 1. In order to reduce the hardware overhead, scan cells are carefully reordered and a special ATPG approach is used to generate suitable test cubes.

III. Proposed Approach

In a generic LFSR-based BIST approach shown in Fig. 1, the output of the LFSR is fed to a phase shifter to reduce the linear dependency between the data shifted into different scan chains. The phase shifter is usually a linear network composed of exclusive-or gates. In the proposed approach, illustrated in Fig. 2(a), the phase shifter is replaced by an RIN that connects the LFSR outputs to the scan chains. The RIN consists of multiplexer switches and it can be reconfigured by applying appropriate control bits to it through the inputs $D_0, D_1, \ldots, D_{g-1}$. The parameter g refers to the number of configurations used during a BIST session and it is determined using a simulation procedure. The control inputs $D_0, D_1, \ldots, D_{g-1}$ are provided by a d-to-g decoder, where $d = \lceil \log_2 g \rceil$. A d-bit configuration counter is used to cycle through all possible 2^d input combinations for the decoder. The configuration counter is triggered by the BIST pattern counter, which is preset for each configuration by the binary value corresponding to the number of test patterns for the corresponding configuration. Although the elimination of the phase shifter may reduce the randomness of the pseudorandom patterns, complete fault coverage is guaranteed by the RIN synthesis procedure described later.

As shown in Fig. 2(b), the multiplexers in the RIN are implemented using tristate buffers with fully decoded control inputs. While the multiplexers can also be implemented in other ways, we use tristate buffers here because of their ease of implementation in CMOS. The outputs of the tristate buffers are connected

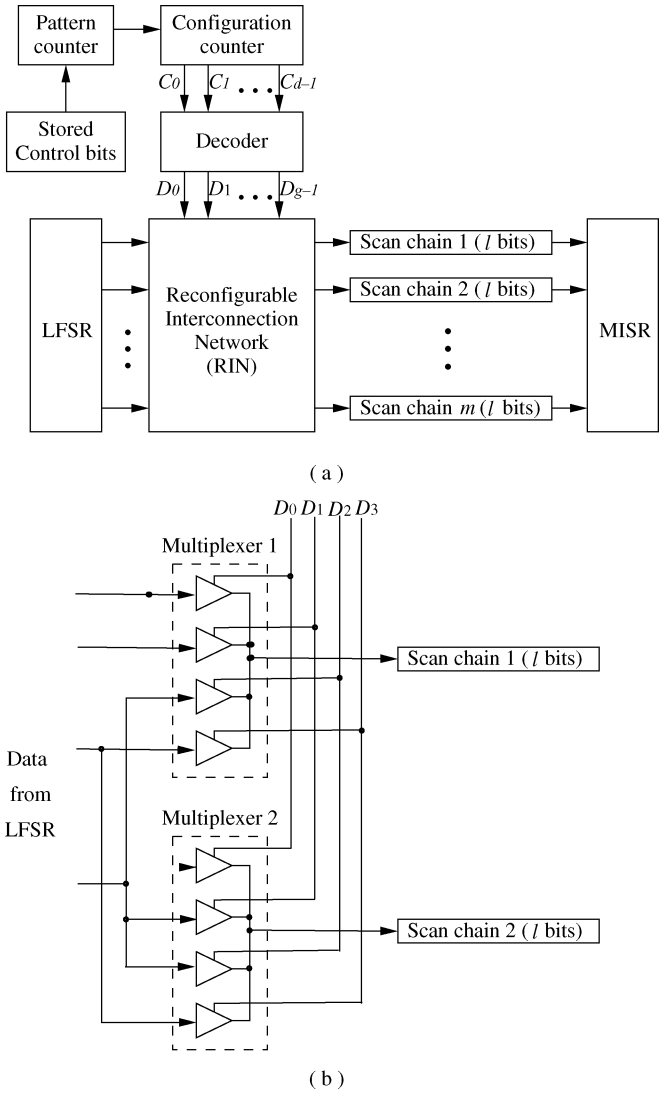


Fig. 2. (a) Proposed logic BIST architecture. (b) RIN for $m = 2$ and $g = 4$.

at the output of the multiplexer. Each input I_i of a multiplexer is connected to the input of a tristate buffer, which is controlled by the corresponding control signal. While the number of multiplexers can be at most equal to the number of scan chains, in practice, it is sometimes smaller than the number of scan chains because not all scan chains need to be driven by different LFSR cells. The number of tristate gates in each multiplexer is at most equal either to the number of configurations or to the number of LFSR cells, whichever is smaller. Once again, in practice, the actual number of tristate gates is smaller than this upper limit.

We next describe the test-application procedure during a BIST session. First, the configuration counter is reset to the all-0 pattern, and the pattern counter is loaded with the binary value corresponding to the number of patterns that must be applied to the CUT in the first configuration. The pattern counter is decremented each time a test pattern is applied to the CUT. When the content of the pattern counter become zero, it is loaded with the number of patterns for the second configuration and it triggers the configuration counter, which is incremented. This leads to a corresponding change in the outputs of the decoder, and the RIN is reconfigured appropriately. This process

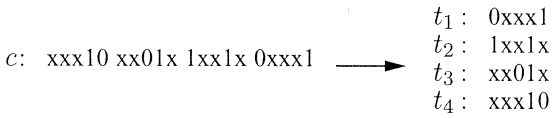


Fig. 3. Illustration of converting a test cube to multiple scan-chain format ($m = 4, l = 5$).

continues until the configuration counter passes through all g configurations. The total number of test patterns applied to the CUT is therefore $\sum_{i=1}^{g} n_i$, where n_i is the number of patterns corresponding to configuration i, $1 \leq i \leq g$. The BIST design procedure described next is tailored to embed a given set of deterministic test cubes in the sequence of $\sum_{i=1}^{g} n_i$ patterns.

During test application, pseudorandom patterns that do not match any deterministic test cube are also applied to the CUT. These pseudorandom patterns can potentially detect nonmodeled faults. However, these patterns increase the testing time. A parameter called $\mathrm{MaxSkipPatterns}$, which is defined as the largest number of pseudorandom patterns that are allowed between the matching of two deterministic cubes, is used in the design procedure to limit the testing time. We first need to determine for each configuration, the number of patterns as well as the interconnections between the LFSR outputs and the scan chains. We use the simulation procedure described next to solve this problem.

We start with an LFSR of length L, a predetermined seed, and a known characteristic polynomial. Let $T_D = \{c_1, c_2, \ldots, c_n\}$ be the set of deterministic test cubes that must be applied to the CUT. The set T_D can either target all the single stuck-at faults in the circuit, or only the hard faults that cannot be detected by a small number of pseudorandom patterns. As illustrated in Fig. 3, each deterministic test cube c in the test set is converted into the multiple scan-chain format as a set of m l-bit vectors $\{t_1, t_2, \ldots, t_m\}$, where m is the number of scan chains and l is the length of each scan chain. The bits in a test cube are ordered such that the least significant bit is first shifted into the scan chain. We use $\mathrm{Conn}_j^{(i)}$ to denote the set of LFSR taps that are connected to the scan chain j in configuration i, where $i = 1, 2, \ldots, g$, $j = 1, 2, \ldots, m$. The steps of the simulation procedure are as follows.

1) Set $i = 1$.
2) Set $\mathrm{Conn}_j^{(i)} = \{1, 2, \ldots, L\}$ for $j = 1, 2, \ldots, m$, i.e., initially, each scan chain can be connected to any tap of the LFSR.
3) Driving the LFSR for the next l clock cycles, we obtain the output of the LFSR as a set of L l-bit vectors $\{O_k | k = 1, 2, \ldots, L\}$, where vector O_k is the output stream of the kth flip-flop of the LFSR for the l clock cycles.
4) Find a test cube c^* in T_D that is compatible with the outputs of the LFSR under the current connection configuration $\mathrm{Conn}_j^{(i)}$, i.e., for all $j = 1, \ldots, m$, there exists $k \in \mathrm{Conn}_j^{(i)}$ such that t_j^* is compatible with O_k, where c^* has already been reformatted for m scan chains as a set of vector $\{t_1^*, t_2^*, \ldots, t_m^*\}$. (A vector $u_1, u_2, \ldots, u_r$ and a vector $v_1, v_2, \ldots, v_r$ are mutually compatible if for any i, $1 \leq i \leq r$, one of the following holds: 1) $u_i = v_i$ if they

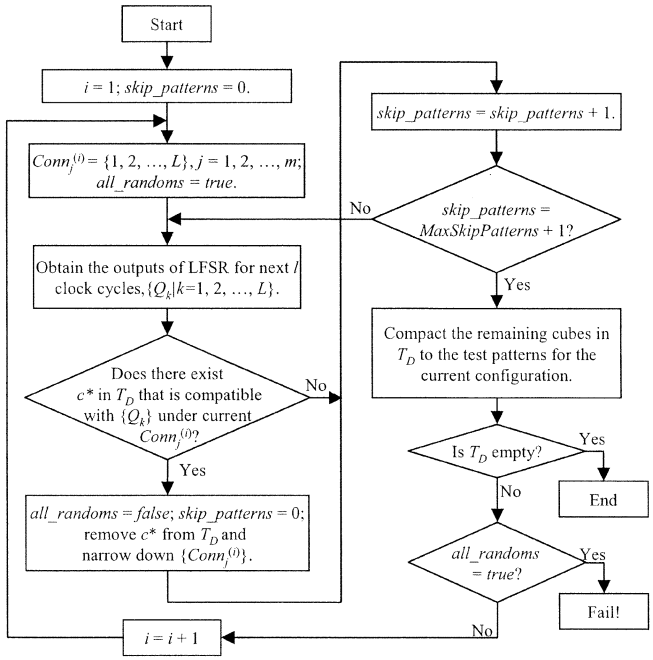


Fig. 4. Flowchart illustrating the simulation procedure.

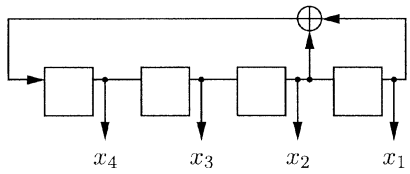
are both care bits; 2) u_i is a don't-care bit; and 3) v_i is a don't-care bit.)

5) If no test cube is found in Step 4, go to Step 6 directly. Otherwise, remove the test cube c^* found in Step 4 from T_D, and narrow down the connection configuration as follows. For each $j = 1, 2, \ldots, m$, let $\mathbf{U} \subset \mathrm{Conn}_j^{(i)}$ such that for any $k \in \mathbf{U}$, O_k is not compatible with t_j^*. Then set $\mathrm{Conn}_j^{(i)} = \mathrm{Conn}_j^{(i)} - \mathbf{U}$.
6) If in the previous $\mathrm{MaxSkipPatterns}+1$ iterations, at least one test cube is found in Step 4, then go to Step 3. Otherwise, the simulation for the current configuration is concluded. The patterns that are applied to the circuit under this configuration are those that are obtained in Step 3.
7) Match the remaining cubes in T_D to the test patterns for the current configuration, i.e., if any test vector in T_D is compatible with any pattern for the current configuration, remove it from T_D.
8) If no pseudorandom pattern for the current configuration is compatible with a test cube, the procedure fails and exits. Otherwise, increase i by 1, and go to Step 2 to begin the iteration for the next configuration until T_D is empty.

Fig. 4 shows a flowchart corresponding to the above procedure, where the variable $skip\_patterns$ is used to record the number of continuous patterns that are not compatible with any deterministic test cube, and $all\_randoms$ is used to indicate if all the patterns for the current configuration are pseudorandom patterns.

An example of the simulation procedure is illustrated in Fig. 5. A four-bit autonomous LFSR with characteristic polynomial $x^4 + x + 1$ is used to generate the pseudorandom patterns. There are four scan chains and the length of each scan chain is four bits. The parameter $\mathrm{MaxSkipPatterns}$ is set to 1. The output of the LFSR is divided into patterns p_i, $i = 1, 2, \ldots$. Each pattern consists of four four-bit vectors. The procedure that determines the connections is shown as Steps Init to f. Step Init is the initialization step in which all the connections $\mathrm{Conn}_j^{(1)}$, $j = 1, 2, 3, 4$ are set to $\{1, 2, 3, 4\}$. In Step a, the first pattern p_1 is matched with the test cube c_1, and the connections are shown for each scan chain: Scan chain 1 can be connected to x_1 or x_4, both Scan chains 2 and 3 can only be connected to x_2, Scan chain 4 can be connected to x_1, x_2, or x_4. In Step c, none of the cubes is compatible with p_3. When neither p_5 nor p_6 matches any cubes in Step e, the iterations for the current configuration are terminated. The patterns that are applied to the CUT in this configuration are $p_1, p_2, \ldots, p_6$. We then compare the remaining cube c_4 with the six patterns and find that it is compatible with p_2. So, c_4 is also covered by the test patterns for the current configuration. Thus, the connections for this configuration are: Scan chain 1 is connected to x_4, both Scan chains 2 and 3 are connected to x_2, Scan chain 4 is connected to x_1. Since p_5 and p_6 are not compatible with any deterministic cubes, the number of patterns for this configuration is set to four. If there are test cubes remaining to be matched, the iteration for the next configuration starts from p_5.

LFSR:



Output of LFSR:

	...	p_6	p_5	p_4	p_3	p_2	p_1
x_1 :	...	1100	1000	1111	0101	1001	0001
x_2 :	...	0110	0100	0111	1010	1100	1000
x_3 :	...	1011	0010	0011	1101	0110	0100
x_4 :	...	0101	1001	0001	1110	1011	0010

Test cubes:

c_1	c_2	c_3	c_4
00xx	0xxx	xx11	xx11
1xx0	xx1x	x10x	1xxx
10xx	01xx	x1x0	01xx
x0xx	11xx	10xx	x001

Determination of connections:

	Init)	a) p_1: c_1	b) p_2: c_3
$Conn_1^{(1)}$:	(1,2,3,4)	(1, 4)	(4)
$Conn_2^{(1)}$:	(1,2,3,4)	(2)	(2)
$Conn_3^{(1)}$:	(1,2,3,4)	(2)	(2)
$Conn_4^{(1)}$:	(1,2,3,4)	(1, 2, 4)	(1, 4)

c) p_3: none	d) p_4: c_2	e) p_5, p_6: none	f) p_2: c_4
(4)	(4)	(4)	(4)
(2)	(2)	(2)	(2)
(2)	(2)	(2)	(2)
(1, 4)	(1)	(1)	(1)

Fig. 5. Illustration of the simulation procedure.

IV. Probabilistic Analysis of Test-Set Embedding

In the test set embedding technique described in Section III, the number of configurations directly determines the hardware overhead for the RIN as well as the storage requirement. In this section, we use probabilistic analysis to determine the average number of configurations for a test set with a given fraction of care bits. This analysis helps us to estimate the number of configurations for test set embedding, without having to simulate the LFSR and check for compatibilities. We use the terminology introduced in Section III, and assume that the care bits are uniformly distributed in the test set.

Let $T_D = \{c_1, c_2, \ldots, c_n\}$ be the set of deterministic test cubes that must be applied to the CUT. Suppose that each deterministic test cube c in the test set is converted to the multiple scan-chain format as a set of m l-bit vectors $\{t_1, t_2, \ldots, t_m\}$, where m is the number of scan chains and l is the length of each scan chain. Driving the L-bit LFSR for l clock cycles, we obtain the output of the LFSR as a set of L l-bit vectors $\mathbf{O} = \{O_k | k = 1, 2, \ldots, L\}$, where vector O_k is the output stream of the kth flip-flop of the LFSR for the l clock cycles. We also obtain N LFSR output sets $\{\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_N\}$ by driving the LFSR for Nl clock cycles in each configuration. In other words, N is the number of test patterns provided by the LFSR in a single configuration.

Let p be the probability that any given bit in the test set is a care bit, i.e., it is specified as either 0 or 1. Since an LFSR outputs 0s and 1s with equal probability, the probability that any bit of the test set is compatible with an output bit of the LFSR is simply $(1-p)+p/2 = 1-p/2$. Let $\mathcal{E}_1$ denote the event that t_i is compatible with O_k, where $1 \le i \le m$ and $1 \le k \le L$. The probability $P[\mathcal{E}_1]$ of event $\mathcal{E}_1$ is simply $(1-p/2)^l$.

Let $\mathcal{E}_2$ denote the event that t_i is compatible with $\mathbf{O}_j$, where $1 \le i \le m$ and $1 \le j \le N$. It can be easily seen that t_i is compatible with $\mathbf{O}_j$ if it is compatible with at least one vector in the corresponding LFSR output set $\{O_k | k = 1, 2, \ldots, L\}$. Therefore, the probability of event $\mathcal{E}_2$ is given by

$$\begin{aligned} p_2 &= P[\mathcal{E}_2] \\ &= 1 - \left(1 - \left(1 - \frac{p}{2}\right)^l\right)^L. \end{aligned} \tag{1}$$

Next, let $\mathcal{E}_3$ denote the event that a test cube $c \in T_D$ is compatible with $\mathbf{O}_j$, $1 \le j \le N$. This implies that each of the m vectors derived from c is compatible with $\mathbf{O}_j$, therefore

$$\begin{aligned} p_3 &= P[\mathcal{E}_3] \\ &= \left(1 - \left(1 - \left(1 - \frac{p}{2}\right)^l\right)^L\right)^m. \end{aligned} \tag{2}$$

Let $\mathcal{E}_4$ denote the event that a given test cube $c \in T_D$ is compatible with at least one LFSR output set among the N elements $\{\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_N\}$. It can be easily seen that

$$\begin{aligned} p_4 &= P[\mathcal{E}_4] \\ &= 1 - (1 - p_3)^N. \end{aligned} \tag{3}$$

We next determine the probability p_{1N} that at least one test cube in T_D is compatible with the N LFSR output sets

$$p_{1N} = 1 - (1 - p_3)^{Nn}.$$

This implies that q_{0N}, the probability that no test cube in T_D is covered by the N LFSR output sets, is simply $(1-p_3)^{Nn}$.

Next, we consider the covering of two test cubes with N LFSR output vector sets in the same configuration. Assuming that test cube c_1 contains $\{t_{11}, t_{12}, \ldots, t_{1m}\}$, c_2 contains $\{t_{21}, t_{22}, \ldots, t_{2m}\}$, $\mathbf{O}_1 = \{O_{1k} | k = 1, 2, \ldots, L\}$, and $\mathbf{O}_2 = \{O_{2k} | k = 1, 2, \ldots, L\}$, we define the following additional events:

1) $\mathcal{E}_5$: t_{1i} is compatible with O_{1k} and t_{2i} is compatible with O_{2k};
2) $\mathcal{E}_6$: t_{1i} is compatible with $\mathbf{O}_1$ and t_{2i} is compatible with $\mathbf{O}_2$;
3) $\mathcal{E}_7$: c_1 is compatible with $\mathbf{O}_1$ and c_2 is compatible with $\mathbf{O}_2$;
4) $\mathcal{E}_8$: c_1 and c_2 are compatible with at least one pair of LFSR output vector sets in the N sets $\{\mathbf{O}_1, \mathbf{O}_2, \ldots, \mathbf{O}_N\}$.

The probabilities associated with the above events are as follows:

$$\begin{aligned} P[\mathcal{E}_5] &= \left(1 - \frac{p}{2}\right)^{2l} \\ P[\mathcal{E}_6] &= 1 - \left(1 - \left(1 - \frac{p}{2}\right)^{2l}\right)^L \\ p_7 &= P[\mathcal{E}_7] \\ &= \left(1 - \left(1 - \left(1 - \left(\frac{p}{2}\right)\right)^{2l}\right)^L\right)^m \\ P[\mathcal{E}_8] &= 1 - (1 - p_7)^{\binom{N}{2}}. \end{aligned} \tag{4}$$

Let p_{2N} be the probability that at least two test cubes in T_D are covered by the N LFSR output vector sets. It therefore follows that:

$$p_{2N} = 1 - (1 - p_7)^{\binom{N}{2}\binom{n}{2}}.$$

Similarly, let p_x be the probability that x test cubes are compatible with x given LFSR output vector sets. It can be easily seen that

$$p_x = \left(1 - \left(1 - \left(1 - \left(\frac{p}{2}\right)\right)^{\mathrm{xl}}\right)^L\right)^m$$

and p_{xN}, the probability that at least x test cubes in T_D are covered by the N LFSR output vector sets, is given by

$$p_{\mathrm{xN}} = 1 - (1 - p_x)^{\binom{N}{x}\binom{n}{x}}.$$

From the formulas for p_{xN} and $p_{(x+1)N}$, we can derive the probability q_{xn} that the N LFSR output vector sets can cover exact x test cubes

$$q_{\mathrm{xn}} = p_{\mathrm{xN}} - p_{(x+1)N}$$

where $x = 1, 2, \ldots, \min(n, N)$. Hence, the average number of

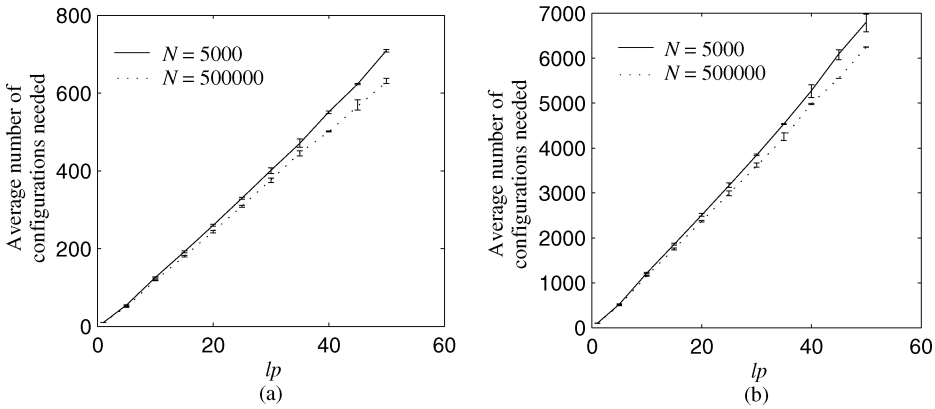


Fig. 6. Average number of configurations needed versus lp for (a) $n = 100$ and (b) $n = 1000$.

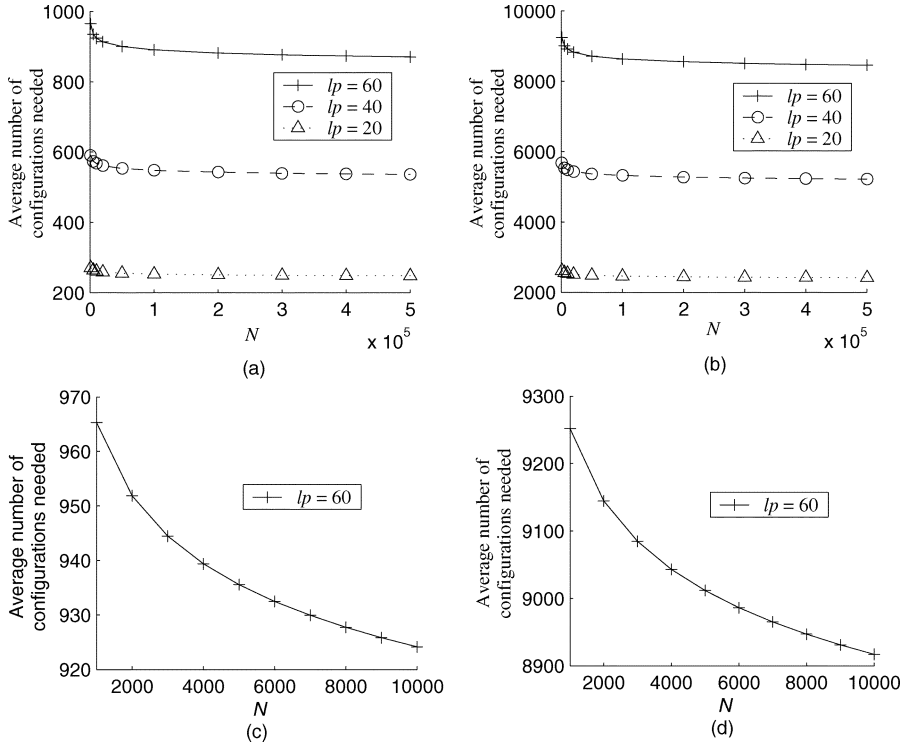


Fig. 7. Average number of configurations needed versus N for (a) $n = 100$, (b) $n = 1000$, (c) $n = 100$, $1000 \leq N \leq 10\,000$, and (d) $n = 1000$, $1000 \leq N \leq 10\,000$.

test cubes that the N LFSR output vector sets can cover is simply $z = \sum_{i=0}^{\min(n,N)} iq_{\text{in}}$, and the average number of configurations needed to embed the test set T_D is, therefore, n/z.

Fig. 6(a) and (b) show the average number of configurations needed to embed the test set with $n = 100$ and $n = 1000$, respectively. Since the number of specified bits per scan chain for a test cube affects the number of configurations, we also vary lp, the average number of specified bits per scan chain for a test cube, from 1 to 50 and compute the results for two different values of N. For each value of lp, ten pairs of (l, p) values corresponding to $p = 0.01, 0.02, \ldots, 0.10$ are used to compute the average number of configurations needed. The graphs are drawn by joining points corresponding to the mean obtained over the ten values of p. The minimum and the maximum values are shown in the figures as a vertical bar for each point. For determining the average number of configurations, we assumed the number of scan chains $m = 32$ and the length of the LFSR $L = 64$. The average number of configurations needed tends to increase linearly with lp. For larger values of lp, the number of patterns generated in each configuration N affects the number of configurations; this dependence is less pronounced for smaller values of lp. The effect of N on the number of configurations is shown in Fig. 7.

In Table XII of Section VI, we list the number of configurations predicted by the analysis. The analytical results closely match the experimental results. The match is less pronounced for other sets of experiments; nevertheless, the analysis provides a useful prediction for the effectiveness of the proposed method.

V. Declustering the Care Bits

The simulation procedure to determine the number of patterns and the connections for each configuration can sometimes fail to embed the test cubes in the LFSR sequence. This can happen if MaxSkipPatterns is too small, or the test cubes are hard to match with the outputs of the LFSR. During our experiments, we found that it was very difficult to embed the test cubes for the s38417 benchmark circuit. On closer inspection, we found that the care bits in some of the test cubes for s38417 are highly clustered, even though the percentage of care bits in T_D is small. When these test cubes are converted into a multiple-scan-chain format, most of the vectors contain very few care bits, but a few vectors contain a large number of care bits. These vectors with many care bits are hard to embed in the output sequence of the LFSR.

In order to embed test cubes with highly clustered care bits, we propose two declustering strategies. The first is to reorganize the scan chains such that the care bits can be scattered across many scan chains, and each scan chain contains only a few care bits. Another strategy is based on the use of additional logic to interleave the data that are shifted into the different scan chains. The first strategy requires reorganization of the scan chains, but it does not require extra hardware overhead. Care needs to be taken in scan-chain redesign to avoid timing closure problems. The interleaving method does not modify the scan chains, but it requires additional hardware and control mechanisms.

The method of reorganization of scan chains is illustrated in Fig. 8. As shown in the figure, before the reorganization, all the care bits of the given test cube are grouped in the second vector, which is hard to match with the output of LFSR. After the reorganization, the care bits are scattered across all the vectors, and the largest number of care bits in a vector is only two. This greatly increases the probability that this vector can be matched to an output pattern of the LFSR. Note that the concept of reorganization of scan chains is also used in [11]. However, the reorganization used in [11] changes the scan-chain structure and makes it unsuitable for response capture—a separate solution is needed in [11] to circumvent this problem. In our approach, the basic structure of the scan chains is maintained and the usual scan-test procedure of pattern shift-in, response capture, and shift-out can be used.

The scan cells in the CUT can be indexed as $c_{i,j}$, $i = 0, 1, \ldots, m-1$, $j = 0, 1, \ldots, l-1$, where m is the number of scan chains and l is the length of a scan chain. Note that we start the indices from 0 to facilitate the description of the scan-chain reorganization procedure. The ith scan chain consists of the l scan cells $c_{i,j}$, $j = 0, 1, \ldots, l-1$. We use $c^*_{i,j}$ to denote the reorganized scan cells, in which the ith scan chain consists of the l scan cells $c^*_{i,j}$, $j = 0, 1, \ldots, l-1$. For each $j = 0, 1, \ldots, l-1$, the m cells $c_{0,j}, c_{1,j}, \ldots, c_{m-1,j}$ constitute a vertical vector. The reorganized scan cell structure is obtained by rotating each such vertical vector upwards by d positions, where $d = j \bmod m$, i.e., $c^*_{i,j} = c_{k,j}$, where k is given by $k = (i + d) \bmod m$.

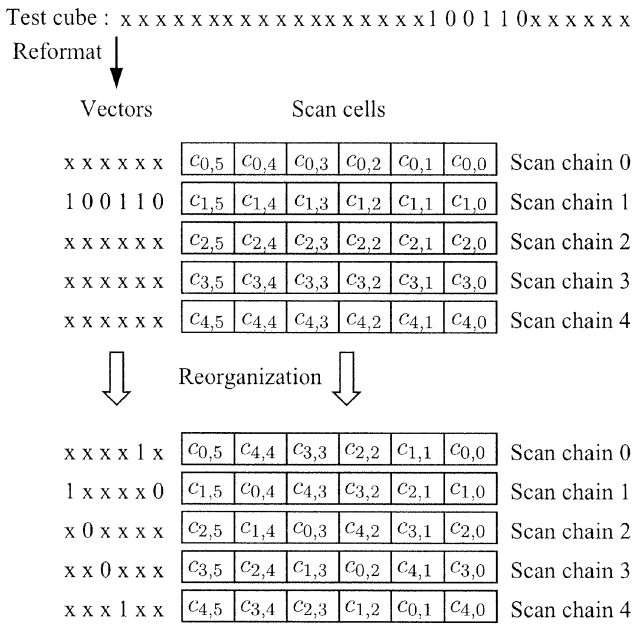


Fig. 8. Illustration of the reorganization of scan chains.

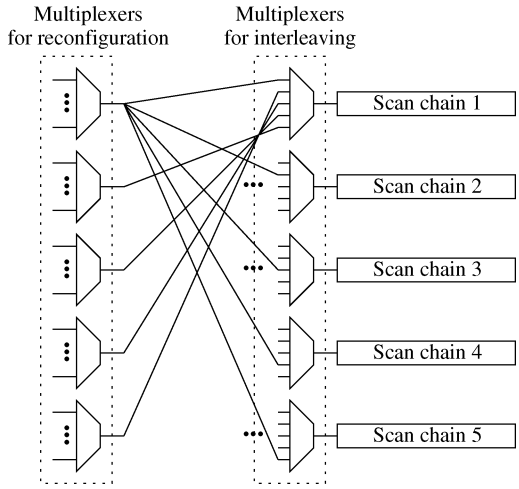


Fig. 9. Illustration of interleaving of the inputs of scan chains.

An alternative method for declustering, based on the interleaving of the inputs to the scan chains, is shown in Fig. 9. We insert an extra stage of multiplexers between the outputs of the RIN and the inputs of the scan chains. From the perspective of the RIN, the logic that follows it, i.e., the combination of the multiplexers for interleaving and the scan chains, is simply a reorganized scan chain with an appropriate arrangement of the connections between the two stages of multiplexers. For a CUT with m scan chains, m multiplexers are used for reconfiguration, and m multiplexers are inserted for interleaving. Each of the multiplexers used for interleaving has m inputs, which are selected in ascending order during the shifting in of a test pattern, i.e., the first input is selected for the first scan-clock cycle, the second input is selected for the second scan clock cycle, and so on. After the mth input is selected, the procedure is repeated

with the first input. We use A_i to denote the output the ith multiplexers for reconfiguration and $B_{i,j}$ to denote the jth input of the ith multiplexers for interleaving, where $i, j = 1, 2, \ldots, m$. The interleaving is carried out by connecting the inputs of the multiplexers for interleaving with the outputs of multiplexers for reconfiguration such that

$$B_{i,j} = \begin{cases} A_{i-j+1}, & \text{if } i \geq j \\ A_{i-j+1+m}, & \text{if } i < j. \end{cases}$$

In order to control the multiplexers for interleaving, an architecture similar to the control logic for the RIN can be used. However, for the interleaving, we do not need any storage and the pattern counter. A bit counter counting up to $m-1$, (where m is the number of scan chains) is used to replace the configuration counter. The bit counter is reset to 0 at the start of the shifting in of each pattern, and it returns to 0 after counting to $m-1$.

Consider the test cube shown in Fig. 8. After adding the second stage of multiplexers and connecting the inputs of the multiplexers for interleaving with the outputs of the multiplexers for reconfiguration, as shown in Fig. 9 (only the connections related to the first RIN multiplexer are shown for clarity), the output of the first multiplexer for reconfiguration should match with "xxxx1x," the same string as that in scan-cell reorganization method. Note that the above reorganization and interleaving procedures yield the same set of test cubes.

VI. Experimental Results and Discussion

In this section, we present experimental results for the seven largest ISCAS'89 circuits and for test cubes for two production circuits from IBM. We use three sets of test cubes T_D for the large ISCAS'89 circuits. The first set of test cubes are obtained from the Mintest ATPG program [10] without dynamic compaction, and by targeting all the irredundant single stuck-at faults. The other two test sets are the same test sets used in [11]. The second set of the test cubes are obtained without an initial pseudorandom pattern application, and they target all irredundant faults. The third set of the test cubes for the random-pattern-resistant faults is obtained after 10 000 pseudorandom patterns are applied to the circuits. We carried out experiments for each of the ISCAS'89 circuits with 32 scan chains, and also ran experiments for the larger circuits with 64 scan chains. In all of these experiments, we used a 64-bit primitive-polynomial LFSR with a fixed randomly-generated seed as the pseudorandom pattern generator. For simplicity of presentation, we assume that the circuits have balanced scan chains. When the scan chains are unbalanced, we can view them as being balanced through the addition of dummy scan cells.

Tables I and II present the results on test set embedding where T_D is obtained using the Mintest program. We use a value of 5000 for the $\mathrm{MaxSkipPatterns}$ parameter for this set of experiments. The fifth column shows the total number of configurations needed to embed T_D. The total number of patterns applied to the circuit is listed in the sixth column. The testing time in clock cycles is obtained as the product of the total number of patterns and $(l+1)$, where l is the length of scan chains. We implemented the RIN architecture containing the multiplexers for reconfiguration, the decoder and the configuration counter using the lsi_10k library of Synopsys design compiler to estimate the hardware overhead. Using the wire load model for the lsi_10k library, we designate the normalized area for a unit-length of wire to be 0.2 (assuming that the area of an inverter is 1 unit) to take into account the area of interconnects. The hardware area for each of the ISCAS'89 benchmark circuits is obtained in the same way. The percentage hardware overhead is obtained from the ratio of the area of BIST hardware to the area of the CUT. The pattern counter is not included in the calculation of the BIST hardware overhead because it is required for any scan-BIST scheme. The encoding efficiency shown in the tables is the ratio of the number of care bits in the test set to the amount of storage needed. The CPU times, listed in the last column, correspond to the simulation time required to embed the deterministic test cubes using a Sun Blade 1000 workstation with a 750-MHz UltraSPARC-III CPU and 1 GB of memory. The results in Table I show that only a small number of control bits (at most a few hundred) are required for test set embedding. The hardware overhead of the RIN, the decoder, and the configuration counter are also very small, less then 7% for five circuits, and only 5.34% on average. Similar results are presented in Table II.

As indicated in Tables I and II, we were unable to embed the test cubes fbits in these test cubes are highly clustered. As a result, it is difficult to match these cubes to the patterns obtained from the LFSR. We therefore considered scan-chain reorganization to obtain experimental results for s38417. Tables III and IV show the result obtained with scan-chain reorganization for the seven largest benchmark circuits with 32 and 64 scan chains, respectively.

Tables V and VI present experimental results on the embedding of test cubes from [11] targeting all faults, without scan-chain reorganization and with scan-chain reorganization, respectively. In these two sets of experiments, we set the parameter $\mathrm{MaxSkipPatterns}$ to 10 000 and set the number of scan chains in each circuit to 32. As indicated in Table V, we were unable to embed the test cubes of s38417 due to a high degree of clustering of its care bits. This problem was addressed using scan-chain reorganization; the results are shown in Table VI. Scan-chain reorganization for this set of test cubes reduces the number of configurations, and hence the storage and the hardware overhead for all circuits except s5378. Similar results are shown in Tables VII and VIII, with 64 scan chains assumed for each CUT.

Tables IX and X present experimental results obtained using the test cubes from [11] that target random-pattern-resistant faults. We assume that, as in other mixed-mode BIST schemes, the RIN is bypassed using multiplexers for the first 10 000 pseudorandom patterns. We considered scan-chain reorganization for these experiments. As expected, compared with the results for test sets targeting all faults, the total number of patterns here is much smaller. Thus the testing time is also much less than in Table VII. The average storage requirement for the seven circuits is reduced from 900 to 534 bits.

TABLE I
EXPERIMENTAL RESULTS FOR MINTEST TEST SETS TARGETING ALL FAULTS (ASSUMING 32 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations	No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	Encoding efficiency	CPU time
s5378	1458	214	7	9	157184	1257472	6.49%	162	81.44	2m
s9234	1928	247	8	34	363551	3271959	15.31%	646	39.57	3m
s13207	3237	700	22	9	322526	7418098	2.69%	171	157.02	5m
s15850	3920	611	20	24	379377	7966917	5.81%	456	67.02	7m
s35932	10810	1763	56	4	19756	1126092	0.55%	60	771.18	3m
s38417*	10771	1664	52	—	—	—	—	—	—	—
s38584	13468	1464	46	11	377804	17756788	1.19%	209	441.15	16m
Average	—	—	—	15	270033	6466221	5.34%	284	259.56	6m

*Care bits are highly clustered hence test cubes could not be embedded with $MaxSkipPatterns = 5000$.

TABLE II
EXPERIMENTAL RESULTS FOR MINTEST TEST SETS TARGETING ALL FAULTS (ASSUMING 64 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations	No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	Encoding efficiency	CPU time
s13207	3237	700	11	10	114779	1377348	5.31%	170	157.95	5m
s15850	3920	611	10	13	212027	2332297	5.98%	234	130.61	10m
s35932	10810	1763	28	5	7294	211526	1.19%	65	711.86	4m
s38417*	10771	1664	26	—	—	—	—	—	—	—
s38584	13468	1464	23	8	304981	7319544	1.64%	152	606.58	19m
Average	—	—	—	9	159770	2810179	3.53%	155	401.75	10m

*Care bits are highly clustered hence test cubes could not be embedded with $MaxSkipPatterns = 5000$.

TABLE III
EXPERIMENTAL RESULTS FOR MINTEST TEST SETS TARGETING ALL FAULTS AND WITH SCAN-CELL REORGANIZATION (ASSUMING 32 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations	No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	Encoding efficiency	CPU time
s5378	1458	214	7	8	67988	543904	7.68%	136	97.01	2m
s9234	1928	247	8	36	135765	1221885	16.04%	648	39.44	3m
s13207	3237	700	22	9	152596	3509708	2.69%	162	165.75	5m
s15850	3920	611	20	22	222336	4669056	5.38%	396	77.18	5m
s35932	10810	1763	56	5	7079	403503	0.67%	65	711.86	4m
s38417	10771	1664	52	272	625273	33139469	26.39%	5440	26.14	1h 43m
s38584	13468	1464	46	12	383009	18001423	1.29%	228	404.39	14m
Average	—	—	—	52	227721	8784135	8.59%	1011	217.39	19m

In the above set of experiments, we assumed that the information on the different number of patterns for each configuration is stored on-chip. If a fixed number of patterns is applied per configuration, then no storage is required. A tradeoff is that a fixed number of patterns per configuration might increase the number of configurations, and thereby increase the hardware overhead. In the next set of experiments, we limit the number of patterns for each configuration to 1000. As expected, the results in Tables XI and XII show that the hardware overhead increase slightly for each circuit. Nevertheless, an important

TABLE IV
EXPERIMENTAL RESULTS FOR MINTEST TEST SETS TARGETING ALL FAULTS AND WITH SCAN-CELL REORGANIZATION (ASSUMING 64 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations	No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	Encoding efficiency	CPU time
s13207	3237	700	11	11	75047	900564	5.80%	187	143.59	5m
s15850	3920	611	10	11	179580	1975380	5.12%	198	154.36	9m
s35932	10810	1763	28	6	6080	176320	1.40%	78	593.22	4m
s38417	10771	1664	26	130	616835	16654545	24.40%	2600	54.69	1h 35m
s38584	13468	1464	23	7	291425	6994200	1.41%	133	693.23	17m
Average	—	—	—	33	233793	5340202	7.63%	639	327.82	26m

TABLE V
EXPERIMENTAL RESULTS FOR TEST SETS FROM [11] TARGETING ALL FAULTS (ASSUMING 32 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations	No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	Encoding efficiency	CPU time
s5378	4397	214	7	6	289754	2318032	5.60%	114	313.81	5m
s9234	6475	247	8	57	394496	3550464	24.94%	1083	72.55	16m
s13207	9608	700	22	11	580521	13351983	3.16%	220	315.13	22m
s15850	11330	611	20	32	758497	15928437	7.65%	640	129.96	28m
s38417*	30859	1664	52	—	—	—	—	—	—	—
s38584	34493	1464	46	12	526555	24748085	1.29%	240	822.90	27m
Average	—	—	—	24	509964	11979400	8.53%	459	330.87	20m

*Care bits are highly clustered hence test cubes could not be embedded with $MaxSkipPatterns = 10000$.

TABLE VI
EXPERIMENTAL RESULTS FOR TEST SETS FROM [11] TARGETING ALL FAULTS AND WITH SCAN-CELL REORGANIZATION (ASSUMING 32 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations	No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	Encoding efficiency	CPU time
s5378	4397	214	7	8	250275	2002200	7.68%	144	248.43	2m
s9234	6475	247	8	31	562133	5059197	13.80%	620	126.73	7m
s13207	9608	700	22	11	297713	6847399	3.16%	209	331.72	17m
s15850	11330	611	20	18	634468	13323828	4.43%	360	231.04	17m
s38417	30859	1664	52	184	1707485	90496705	17.94%	3864	75.35	1h 51m
s38584	34493	1464	46	10	736686	34624242	1.10%	200	987.48	30m
Average	—	—	—	44	698126	25392262	8.02%	900	333.46	31m

benefit here is that no storage is necessary for control bits. We also note from Table XII that the number of configurations predicted by the analysis of Section IV is close to the experimental values.

In Section III, we highlighted the fact that the number of multiplexers in the RIN is often smaller than the number of scan chains because not all scan chains need to be driven by different LFSR cells. We also noted that the actual number of tristate gates in each multiplexer is sometimes smaller than its upper limit, which is equal to the smaller of the number of configurations or the number of LFSR cells. Here, we report these numbers for the s5378 benchmark circuit with 32 scan chains. Eleven scan chains are directly connected to the LFSR taps, and only 77 tristate gates are needed for the total of 21 multiplexer

TABLE VII
EXPERIMENTAL RESULTS FOR TEST SETS FROM [11] TARGETING ALL FAULTS (ASSUMING 64 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations	No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	Encoding efficiency	CPU time
s13207	9608	700	11	14	608383	7300596	7.27%	280	247.60	23m
s15850	11330	611	10	27	776581	8542391	12.29%	540	154.03	32m
s38417*	30859	1664	26	—	—	—	—	—	—	—
s38584	34493	1464	23	10	849505	20388120	2.00%	200	987.48	37m
Average	—	—	—	17	744823	12077036	7.19%	340	463.04	31m

*Care bits are highly clustered hence test cubes could not be embedded with $MaxSkipPatterns = 10000$.

TABLE VIII
EXPERIMENTAL RESULTS FOR TEST SETS FROM [11] TARGETING ALL FAULTS AND WITH SCAN-CELL REORGANIZATION (ASSUMING 64 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations	No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	Encoding efficiency	CPU time
s13207	9608	700	11	15	341543	4098516	7.76%	285	243.26	20m
s15850	11330	611	10	14	591651	6508161	6.42%	280	297.05	33m
s38417	30859	1664	26	110	270978	7316406	20.66%	2090	139.30	2h 3m
s38584	34493	1464	23	8	567274	13614576	1.64%	160	1234.35	81m
Average	—	—	—	37	442862	7884415	9.12%	704	478.49	64m

TABLE IX
EXPERIMENTAL RESULTS FOR TEST SETS FROM [11] TARGETING RANDOM-PATTERN-RESISTANT FAULTS WITH SCAN-CHAIN REORGANIZATION (ASSUMING 32 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations	No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	Encoding efficiency	CPU time
s5378	39	214	7	5	3269	26152	4.80%	60	10.27	$<$1m
s9234	698	247	8	34	34341	309069	15.31%	544	32.53	2m
s13207	556	700	22	5	54776	1259848	1.53%	80	106.71	3m
s15850	654	611	20	22	26968	566328	5.38%	330	45.15	1m
s38417	2219	1664	52	120	103653	5493609	11.70%	2040	28.66	18m
s38584	441	1464	46	10	21282	1000254	1.10%	150	48.74	2m
Average	—	—	—	33	40714	1442543	6.64%	534	45.34	5m

s for the remaining scan chains. Similarly for s38584, when a 64-bit LFSR is used to drive 64 scan chains, only 59 multiplexers are needed (the remaining five scan chains are directly connected to the LFSR taps) and the number of tristate gates is only 276.

Table XIII compares the storage requirements of the proposed approach with hybrid BIST based on weighted pseudorandom patterns [16], test vector encoding using partial LFSR reseeding [18], the BIST scheme based on reseeding of folding counter [11], and two-dimensional test-data compression [22]. The results for the proposed approach are taken from Table IX. The results presented for these methods in the literature rely on 10 000 initial pseudorandom patterns to eliminate the easy to detect faults, except for [16], which uses 32 000 pseudorandom patterns. The results show that in all but one case, the proposed approach requires less storage than other methods listed in the table. (The hybrid BIST method [16] requires less storage for s9234.) Despite requiring more storage, methods such as [16] and [18] generally require less hardware overhead than the proposed technique. Thus, the choice of an appropriate BIST method for particular CUT must be made by considering a combination of these factors.

TABLE X
EXPERIMENTAL RESULTS FOR TEST SETS FROM [11] TARGETING RANDOM-PATTERN-RESISTANT FAULTS WITH SCAN-CHAIN REORGANIZATION (ASSUMING 64 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations	No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	Encoding efficiency	CPU time
s13207	556	700	11	4	21001	252012	2.23%	60	142.28	1m
s15850	654	611	10	19	21173	232903	8.68%	285	52.27	2m
s38417	2219	1664	26	53	115209	3110643	10.06%	901	64.90	17m
s38584	441	1464	23	8	10236	245664	1.64%	112	65.28	1m
Average	—	—	—	21	41905	960306	5.65%	340	81.18	5m

TABLE XI
EXPERIMENTAL RESULTS FOR TEST SETS FROM [11] TARGETING RANDOM-PATTERN-RESISTANT FAULTS WITH SCAN-CELL REORGANIZATION AND A FIXED NUMBER OF PATTERNS PER CONFIGURATION (ASSUMING 32 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations	No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	CPU time
s5378	39	214	7	5	4003	32024	4.81%	0	<1m
s9234	698	247	8	37	36001	324009	16.49%	0	2m
s13207	556	700	22	8	8000	184000	2.43%	0	<1m
s15850	654	611	20	24	23002	483042	5.35%	0	2m
s38417	2219	1664	52	129	128001	6784053	12.59%	0	21m
s38584	441	1464	46	12	11001	517047	1.29%	0	1m
Average	—	—	—	36	35001	1387363	7.16%	0	5m

In Table XIV, we compare the proposed method with scan-based three-weight weighted random BIST [33]. Since no storage of seeds or control bits is required in [33], we use the results from Table XI for comparison. The third column of Table XIV lists the number of pseudorandom patterns required to achieve 100% coverage of detectable single stuck-at faults, as reported in [1] and [33]. The total number of patterns listed for our approach is obtained by adding 10 000 to the number of patterns listed in Table XI. The testing times listed in the table are obtained by assuming a 20 MHz scan-clock frequency. The testing time for the proposed approach is less, even though [33] requires a smaller number of patterns for some circuits. This is because we use a multiple scan-chain architecture, whereas [33] is based on a single scan-chain architecture. In order to compare the hardware overhead of the proposed method with [33], we calculate the gate equivalent (GE) value for the hardware overhead using the same method as in [33]: $0.5n$ for an n-input NAND or NOR gate, and 0.5 for an inverter. We also use 0.5 as the GE value for a transmission gate, and a GE value of 4 for a flip-flop. The use of a single scan-chain architecture ensures that the hardware overhead in [33] is lower; however, in order to scale three-weight weighted random BIST to multiple scan chains, separate decoding logic is needed for each scan chain, which contributes to increased hardware overhead. The parallel scheme in [33] requires even less hardware, but it relies on explicit control of the set and reset signals of the flip-flops after scan-cell reordering. Slightly lower GE counts are reported in [20] for the BIST hardware; however, the approach in [20] only addresses single-scan chains. Comparable, and sometimes even lower, hardware overhead is reported for seed encoding in [2]; however, the overhead figures in [2] do not include the size of the LFSR, which is determined by the number of specified bits in a test vector, and can be as high as 500 bits in many cases. While our proposed approach is based on the embedding of precomputed test sets without fault simulation, three-valued fault simulation is interleaved with the synthesis of the BIST architecture to obtain low hardware overhead in both [34] and [35]. Thus, a direct comparison between the proposed approach and [34] and [35] is difficult. Finally, the overlap between the experimental results reported in [31] and in this paper is limited to only one common benchmark circuit; hence, it is not meaningful to compare our results with [31].

We also implemented a phase shifter using the synthesis procedure presented in [25] and compared the hardware overhead of the phase shifter to that of the proposed BIST architecture as shown in Table XV. The hardware overhead for the phase shifter is obtained in the same way as we obtain the hardware overhead for the proposed architecture. The hardware overheads for the RIN are the minimum, average, and maximum values from Tables XI and XII with the same number of scan chains. The results

TABLE XII
EXPERIMENTAL RESULTS FOR TEST SETS FROM [11] TARGETING RANDOM-PATTERN-RESISTANT FAULTS WITH SCAN-CELL REORGANIZATION AND A FIXED NUMBER OF PATTERNS PER CONFIGURATION (ASSUMING 64 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	No. of test cubes	No. of scan cells	Length of scan chain	No. of configurations		No. of BIST patterns	Testing time (clock cycles)	Hardware overhead (percentage)	Storage requirement (bits)	CPU time
				Actual results	Analysis results					
s13207	556	700	11	7	8.7	6001	72012	3.74%	0	1m
s15850	654	611	10	18	16.0	17649	194139	8.24%	0	2m
s38417	2219	1664	26	61	60.0	60098	1622646	11.58%	0	19m
s38584	441	1464	23	7	7.5	6004	144096	1.41%	0	1m
Average	—	—	—	23	23.1	22438	508223	6.24%	0	6m

TABLE XIII
COMPARISON OF STORAGE (IN BITS) REQUIRED FOR VARIOUS BIST METHODS

Circuit	Hybrid BIST [16]	Partial Reseeding [18]	Reseeding of folding counter [11]	Two-dimensional compression [22]	Proposed approach
s5378	N/A	502	132	196	60
s9234	371	5013	2310	3800	544
s13207	110	3008	247	1044	80
s15850	535	5204	2403	3360	330
s38417	2663	24513	6802	11214	2040
s38584	615	2942	660	2891	150

in the table show that the average hardware overhead of the proposed RIN architecture is sometimes slightly larger than that of the phase shifter. An advantage of the phase shifter is that it is CUT-independent; however, the test patterns generated by the LFSR and the phase shifter do not guarantee complete test set embedding and fault coverage. The proposed RIN architecture can embed all the deterministic test patterns and provide the same coverage as the deterministic test set. This is highlighted in Table XVI, which shows the number of patterns embedded and fault efficiency obtained using RIN and a phase shifter with shift distance 1024, respectively. The test patterns are taken from [11], and they target hard faults that remain undetected after 10 000 pseudorandom patterns are applied. For the RIN, we fix the number of patterns per configuration to 1000 as in Table XI.

In all of the above experiments, we use the proposed synthesis procedure with a random seed for the LFSR. In order to investigate the influence of the initial seed on the results, we carried out the experiments with 20 randomly selected initial seeds for the test set from [11], targeting all faults with scan-cell reorganization and scan chains. The statistics on the number of configurations are listed in Table XVII(A). We also carried out the same experiments for the test set from [11], targeting random-pattern-resistant faults and listed results in Table XVII(B). The results show that the number of configurations depends on the initial seed. However, the dependency is not very significant due in part to the reconfigurability of the interconnection network. A detailed study of the seed selection problem is left for future work.

In order to evaluate the effectiveness of the proposed approach for large circuits, we applied the method to test sets for two production circuits from IBM, namely CKT1 and CKT2. CKT1 is a logic core consisting of 51 082 gates and its test set provides 99.80% fault coverage. CKT2 is a logic core consisting of 94 340 gates and its test set provides 99.76% fault coverage. The number of scan chains is fixed to 64 and 128 for each of these two circuits. We modified the simulation procedure such that the configuration of the interconnection network can be changed during the shifting in of a test cube, and we set the parameter $\mathrm{MaxSkipPatterns}$ to 0. Accordingly, in the proposed BIST architecture shown in Fig. 2(a), the stored control bits are the number of bits per configuration instead of the number of patterns per configuration, and the pattern counter is replaced by a bit counter that counts the number of bits that have been shifted into the scan chains. Table XVIII lists the results for these two industrial circuits. Since we do not have the gate-level netlists for these two circuits, we compute the hardware overhead for the proposed BIST architecture for these two circuits in GEs, and the percentage hardware overhead is obtained from the ratio of the amount of BIST hardware in GEs to the GE count of the CUT. The hardware overhead is less than 10%, and very high encoding efficiency (up to 77.71) is achieved for both circuits. As mentioned above, we allow the configuration of the interconnection network to be changed during the shifting in of a test cube. Table XIX, Figs. 10 and 11 present the statistics on the number of reconfigurations per test cube. The number of intrapattern configurations is small for both circuits.

TABLE XIV
COMPARISON WITH THREE-WEIGHT WEIGHTED RANDOM BIST

Circuit	No. of scan cells	Required random patterns	3-weight weighted random BIST [33]						Proposed approach		
			No. of random patterns	No. of weighted patterns	Total no. of patterns	Testing time (ms)	Hardware overhead in GEs (parallel)	Hardware overhead in GEs (serial)	Total no. of patterns	Testing time (ms)	Hardware overhead (GEs)
s5378	214	>10M	4000	5120	9120	98.04	16	86.5	14003	5.60	103.5
s9234	247	11M	32000	11264	43264	536.47	91	146.5	46001	20.70	748.5
s13207	700	264K	64000	6144	70144	2458.55	16.5	120.0	18000	20.70	157.5
s15850	611	>100M	64000	21504	85504	2616.42	82	264.5	33002	34.65	478.5
s38417	1664	>100M	32000	53248	85248	7096.90	169.5	626.6	138001	365.70	2680.5
s38584	1464	>100M	2000	16384	18384	1346.63	30	197.0	21001	49.35	240

TABLE XV
COMPARISON OF THE HARDWARE OVERHEAD OF THE PROPOSED METHOD WITH THAT OF A PHASE SHIFTER

LFSR length (bits)	No. of scan chains	Minimum shifts between scan chains (two values of shift distance)	Maximum XOR inputs for any scan chain	Hardware overhead of the phase shifter	Hardware overhead of the RIN		
					Max	Min	Avg
64	32	1024 (16)	18 (9)	1273.9 (443.1)	2680.5	103.5	734.8
64	64	1024 (16)	19 (12)	2482.8 (1199.7)	2192.9	251.5	837.0
32	32	1024 (16)	7 (7)	476.0 (462.5)	2680.5	103.5	734.8
32	64	1024 (16)	7 (7)	857.8 (845.2)	2192.9	251.5	837.0

TABLE XVI
COMPARISON OF TEST-SET EMBEDDING BETWEEN RIN AND A PHASE SHIFTER

Circuit	No. of deterministic test patterns	Results for RIN			Results for phase shifter		
		No. of patterns required for embedding	No. of deterministic patterns embedded	Fault efficiency (percentage)	No. of patterns considered for embedding	No. of deterministic patterns embedded	Fault efficiency (percentage)
s5378	39	4003	39	100%	14003	8	98.9%
s9234	698	36001	698	100%	46001	117	94.0%
s13207	556	8000	556	100%	18000	191	96.5%
s15850	654	23002	654	100%	33002	51	95.9%
s38417	2219	128001	2219	100%	138001	140	97.9%
s38584	441	11001	441	100%	21001	207	99.2%

VII. CONCLUSION

We have presented a new approach for deterministic BIST based on the use of an RIN. The RIN is placed between the outputs of pseudorandom pattern generator, e.g., an LFSR, and the scan inputs of the CUT. It consists only of multiplexer switches and is designed using a synthesis procedure that takes as inputs the pseudorandom sequence from the LFSR and the deterministic test cubes for the CUT. The connections between the LFSR and the scan chains can be changed dynamically (reconfigured) during a test session. In this way, the RIN is used to match the LFSR outputs to the set of test cubes T_D. The control data bits used for reconfiguration guarantee that T_D is embedded in the test patterns applied to the CUT. We have shown through several sets of experiments that the proposed approach requires very little hardware overhead and only a modest amount of CPU time. We have also shown that the fewer control bits are required compared to the storage required for reseeding methods or for hybrid BIST. Finally, as a nonintrusive BIST solution, the proposed approach does not require any circuit redesign and it has minimal impact on circuit performance.

We are currently extending this work to ensure that undesirable input patterns that cause problems such as bus contention are forwarded to the scan chains by the RIN. It appears that this problem, which is typical of most logic BIST techniques, can be handled by suitably modifying the RIN synthesis procedure.

TABLE XVII
STATISTICS ON THE NUMBER OF CONFIGURATIONS WITH RANDOM SEEDS FOR TEST SETS FROM [11] TARGETING (A) ALL FAULTS AND (B) RANDOM-PATTERN-RESISTANT FAULTS, WITH SCAN-CHAIN REORGANIZATION (ASSUMING 32 SCAN CHAINS FOR EACH CIRCUIT)

Circuit	Minimum	Maximum	Mean	Standard deviation
s5378	6	9	7.2	0.83
s9234	29	33	30.5	1.10
s13207	10	14	11.85	1.18
s15850	16	20	17.5	1.24
s38417	180	192	185.9	3.23
s38584	9	12	9.8	0.89

(A)

Circuit	Minimum	Maximum	Mean	Standard deviation
s5378	3	5	3.55	0.60
s9234	32	36	33.7	1.38
s13207	5	8	5.95	0.89
s15850	19	25	21.8	1.51
s38417	118	129	121.45	3.07
s38584	9	12	10.8	0.83

(B)

TABLE XVIII
RESULTS FOR TEST CUBES FOR CIRCUITS FROM IBM

Circuit	No. of test cubes	No. of scan cells	No. of scan chains	Length of scan chain (bits)	No. of configurations	Testing time (clock cycles)	Hardware overhead in GEs, and as a percentage	Storage requirement (bits)	Encoding efficiency	CPU time
CKT1	17176	12256	64	192	1792	1351104	68145.5 (8.52%)	21504	46.79	1h 37m
			128	96	1079	566496	75579.5 (9.45%)	12948	77.71	1h 26m
CKT2	43079	22216	64	348	3221	4051764	124062.5 (7.26%)	38652	26.03	6h 35m
			128	174	1828	2338005	128009.5 (7.49%)	21936	45.87	6h 06m

TABLE XIX
NUMBER OF RECONFIGURATIONS PER PATTERN FOR TEST SETS FROM IBM

Circuit	No. of scan chains	Length of a test cube per scan chain (bits)	No. of reconfigurations per pattern			
			Minimum	Maximum	Mean	Standard deviation
CKT1	64	192	0	3	0.11	0.0210
	128	96	0	3	0.07	0.0354
CKT2	64	348	0	3	0.11	0.0204
	128	174	0	15	0.06	0.3018

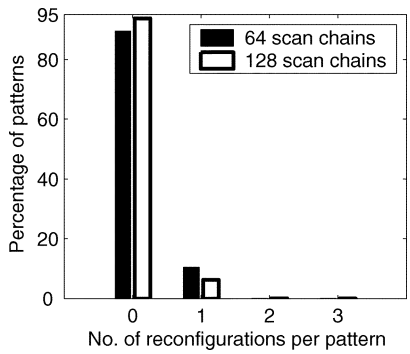


Fig. 10. Number of patterns versus the number of reconfigurations needed for CKT1.

ACKNOWLEDGMENT

The authors thank Prof. S. Hellebrand of the University of Innsbruck, Innsbruck, Austria, for providing the test cubes used in [11]. They also thank S. Swaminathan of the IBM Corporation for providing the test cubes for the production circuits from IBM, and the anonymous reviewers for many constructive comments.

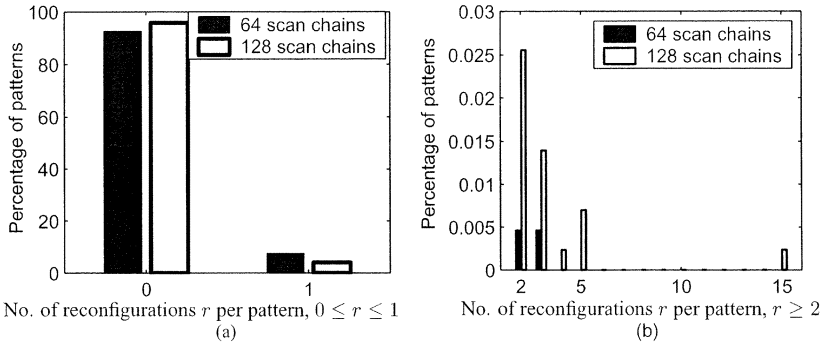


Fig. 11. Number of patterns versus the number of reconfigurations needed for CKT2.

REFERENCES

[1] M. F. AlShaibi and C. R. Kime, "MFBIST: A BIST method for random pattern resistant circuits," in *Proc. Int. Test Conf.*, 1996, pp. 176–185.

[2] A. A. Al-Yamani and E. J. McCluskey, "Seed encoding with LFSRs and cellular automata," in *Proc. ACM/IEEE Design Automation Conf.*, 2003, pp. 560–565.

[3] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller, and B. Koenemann, "OPMISR: The foundation for compressed ATPG vectors," in *Proc. Int. Test Conf.*, 2001, pp. 748–757.

[4] I. Bayraktaroglu and A. Orailoglu, "Test volume and application time reduction through scan-chain concealment," in *Proc. ACM/IEEE Design Automation Conf.*, 2001, pp. 151–155.

[5] A. Chandra and K. Chakrabarty, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes," *IEEE Trans. Comput.*, vol. 52, pp. 1076–1088, Aug. 2003.

[6] ——, "System-on-a-chip test data compression and decompression architectures based on Golomb codes," *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 355–368, Mar. 2001.

[7] K.-T. Cheng and C.-J. Lin, "Timing driven test point insertion for full-scan and partial-scan BIST," in *Proc. Int. Test Conf.*, 1995, pp. 506–514.

[8] A. El-Maleh, S. al Zahir, and E. Khan, "A geometric-primitives-based compression scheme for testing systems-on-chip," in *Proc. VLSI Test Symp.*, 2001, pp. 54–59.

[9] P. T. Gonciari, B. Al-Hashimi, and N. Nicolici, "Improving compression ratio, area overhead, and test application time for system-on-a-chip test data compression/decompression," in *Proc. Design, Automation Test Eur. Conf.*, 2002, pp. 604–611.

[10] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithms for combinational circuits," in *Proc. Int. Conf. Computer-Aided Design*, 1998, pp. 283–289.

[11] S. Hellebrand, H.-G. Liang, and H.-J. Wunderlich, "A mixed-mode BIST scheme based on reseeding of folding counters," in *Proc. Int. Test Conf.*, 2000, pp. 778–784.

[12] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic BIST for large industrial designs: Real issues and case studies," in *Proc. Int. Test Conf.*, 1999, pp. 358–367.

[13] V. Iyengar, K. Chakrabarty, and B. T. Murray, "Deterministic built-in pattern generation for sequential circuits," *J. Electron. Testing: Theory Applicat.*, vol. 15, pp. 97–115, 1999.

[14] A. Jas and N. A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based design," in *Proc. Int. Test Conf.*, 1998, pp. 458–464.

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

[16] A. Jas, C. V. Krishna, and N. A. Touba, "Hybrid BIST based on weighted pseudo-random testing: A new test resource partitioning scheme," in *Proc. VLSI Test Symp.*, 2001, pp. 2–8.

[17] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, "A SmartBIST variant with guaranteed encoding," in *Proc. Asian Test Symp.*, 2001, pp. 325–330.

[18] C. V. Krishna, A. Jas, and N. A. Touba, "Test vector encoding using partial LFSR reseeding," in *Proc. Int. Test Conf.*, 2001, pp. 885–893.

[19] C. V. Krishna and N. A. Touba, "Reducing test data volume using LFSR reseeding with seed compression," in *Proc. Int. Test Conf.*, 2002, pp. 321–330.

[20] W. Li, C. Yu, S. M. Reddy, and I. Pomeranz, "A scan BIST generation method using a Markov source and partial bit-fixing," in *Proc. ACM/IEEE Design Automation Conf.*, 2003, pp. 554–559.

[21] L. Li and K. Chakrabarty, "Test data compression using dictionaries with fixed-length indices," in *Proc. VLSI Test Symp.*, 2003, pp. 219–224.

[22] H.-G. Liang, S. Hellebrand, and H.-J. Wunderlich, "Two-dimensional test data compression for scan-based deterministic BIST," in *Proc. Int. Test Conf.*, 2001, pp. 894–902.

[23] M. Nakao, S. Kobayashi, K. Hatayama, K. Iijima, and S. Terada, "Low overhead test point insertion for scan-based BIST," in *Proc. Int. Test Conf.*, 1999, pp. 348–357.

[24] J. Rajski, J. Tyszer, and N. Zacharia, "Test data decompression for multiple scan designs with boundary scan," *IEEE Trans. Comput.*, vol. 47, pp. 1188–1200, Nov. 1998.

[25] J. Rajski and J. Tyszer, "Design of phase shifters for BIST applications," in *Proc. VLSI Test Symp.*, 1998, pp. 218–224.

[26] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and J. Qian, "Embedded deterministic test for low-cost manufacturing test," in *Proc. Int. Test Conf.*, 2002, pp. 301–310.

[27] S. Reda and A. Orailoglu, "Reducing test application time through test data mutation encoding," in *Proc. Design, Automation Test Eur. Conf.*, 2002, pp. 387–393.

[28] S. M. Reddy, K. Miyase, S. Kajihara, and I. Pomeranz, "On test data volume reduction for multiple scan chain design," in *Proc. VLSI Test Symp.*, 2002, pp. 103–108.

[29] L. Schafer, R. Dorsch, and H.-J. Wunderlich, "RESPIN++ – deterministic embedded test," in *Proc. Eur. Test Workshop*, 2002, pp. 37–44.

[30] C. Schotten and H. Meyr, "Test point insertion for an area efficient BIST," in *Proc. Int. Test Conf.*, 1995, pp. 515–523.

[31] N. A. Touba and E. J. McCluskey, "Altering a pseudo-random bit sequence for scan based BIST," in *Proc. Int. Test Conf.*, 1996, pp. 167–175.

[32] E. H. Volkerink, A. Khoche, and S. Mitra, "Packet-based input test data compression techniques," in *Proc. Int. Test Conf.*, 2002, pp. 154–163.

[33] S. Wang, "Low hardware overhead scan based 3-weight weighted random BIST," in *Proc. Int. Test Conf.*, 2001, pp. 868–877.

[34] H.-J. Wunderlich and G. Kiefer, "Bit-flipping BIST," in *Proc. Int. Conf. Computer-Aided Design*, 1996, pp. 337–343.

[35] G. Kiefer and H.-J. Wunderlich, "Deterministic BIST with multiple scan chains," in *Proc. Int. Test Conf.*, 1998, pp. 1057–1064.

Lei Li received the B.S. and M.S. degrees in electrical engineering from Peking University, Beijing, China, in 1996 and 1999, respectively, and the M.S. degree in electrical and computer engineering in 2002 from Duke University, Durham, NC, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

He is a currently a Research Assistant at Duke University. His research interests are in the field of VLSI testing, with an emphasis on the techniques reducing test data volume and test application time.

Krishnendu Chakrabarty (S'92–M'96–SM'00) received the B.Tech. degree from the Indian Institute of Technology, Kharagpur, India, 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 currently an Associate Professor of Electrical and Computer Engineering at Duke University, Durham, NC. From 2000 to 2002, he was also a Mercator Visiting Professor at the University of Potsdam, Potsdam, Germany. His current research projects include: design and testing of system-on-a-chip integrated circuits; embedded real-time systems; distributed sensor networks; modeling, simulation, and optimization of microelectrofluidic systems; microfluidics-based chip cooling. He is a coauthor of two books: *Microelectrofluidic Systems: Modeling and Simulation* (Boca Raton, FL: CRC Press, 2002) and *Test Resource Partitioning for System-on-a-Chip* (Norwell, MA: Kluwer, 2002), and an editor of *SOC (System-on-a-Chip) Testing for Plug and Play Test Automation* (Norwell, MA: Kluwer, 2002). He has published over 160 papers in journals and refereed conference proceedings, and holds a US patent in BIST.

Dr. Chakrabarty is a Member of ACM, ACM SIGDA, and Sigma Xi. He is a recipient of the National Science Foundation Early Faculty (CAREER) Award and the Office of Naval Research Young Investigator Award. He received a best paper award at the 2001 Design, Automation, and Test in Europe (DATE) Conference. He is also the recipient of the Humboldt Research Fellowship, awarded by the Alexander von Humboldt Foundation, Germany. He is an Associate Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, an Editor of the *Journal of Electronic Testing: Theory and Applications (JETTA)*, and a member of the editorial board for *Sensor Letters* and the *Journal of Embedded Computing*. He has also served as an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: ANALOG AND DIGITAL SIGNAL PROCESSING. He serves as Vice Chair of Technical Activities of the IEEE's Test Technology Technical Council and is a Member of the program committees of several IEEE/ACM conferences and workshops.