

Deterministic BIST Based on a Reconfigurable Interconnection Network^{*}

Lei Li and Krishnendu Chakrabarty

Department of Electrical and Computer Engineering

Duke University, Durham, NC 27708

ll, krish @ee.duke.edu

Abstract—We present a new approach for deterministic BIST in which a reconfigurable interconnection network (RIN) is placed between the outputs of a pseudo-random 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 pseudo-random 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, and fewer control bits compared to the storage required for reseeding techniques or for hybrid BIST. Moreover, as a non-intrusive BIST solution, it does not require any circuit redesign and has minimal impact on circuit performance.

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 for 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 [12, 14], run-length coding [13], Golomb coding [5], FDR coding [4], and VIHC coding [8], have been proposed to reduce test data volume. Test data volume reduction techniques

based on on-chip pattern decompression are also presented in [3, 7, 19, 24, 25, 26, 29].

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 [2] and SmartBIST [16] tools from IBM and the TestKompress tool from Mentor Graphics [23] 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 pseudo-random 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, high pin count. A typical 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 [6, 27], reseeding [10, 20, 22], bit-flipping [31], bit-fixing [28] and weighted random pattern testing [30] 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 pseudo-random test patterns. The other BIST techniques are non-intrusive in that they typically apply a limited number of random patterns, then for the remaining hard to test faults, deterministic test patterns are obtained by either controlling the state of the pattern generator [10, 17, 20, 22] or by altering the output of the pattern generator [28, 30, 31]. A number of studies have also been reported recently on the use of BIST for large industrial circuits [11, 21].

Techniques based on the combination of data compression and BIST have also been developed recently [15, 18]. The hybrid BIST scheme presented in [15] applies weighted pseudo-random 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 [18], 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 used

^{*}This research was supported in part by the National Science Foundation under grants CCR-9875324 and CCR-0204077, and by a graduate fellowship from the Design Automation Conference.

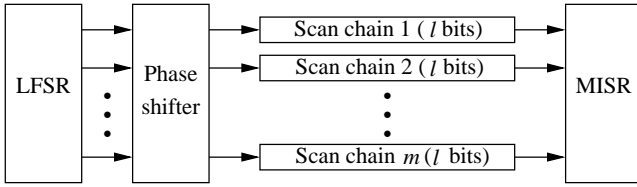


Fig. 1. A generic BIST architecture based on an LFSR, a MISR, and a phase shifter.

in pseudo-random 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, and fewer control bits compared to the storage required for reseeding techniques or for hybrid BIST. Moreover, as a non-intrusive 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 reconfigurable interconnection network and the determination of the control bits. In Section IV, 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 V. Finally, Section VI concludes the paper.

II. RELATED PRIOR WORK

Most BIST techniques rely on the use of a limited number of pseudo-random 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, BIST techniques can be classified into two categories: methods that generate deterministic patterns by controlling the states of the LFSR [10, 17, 20, 22], and techniques that modify the patterns generated by the LFSR [28, 30, 31].

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 [17], 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 [10]. 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 [20], 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).

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 [15], a hybrid BIST method based on weighted pseudo-random 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 [30]. 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 a reconfigurable interconnection network (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, \dots, 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, \dots, 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.

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,

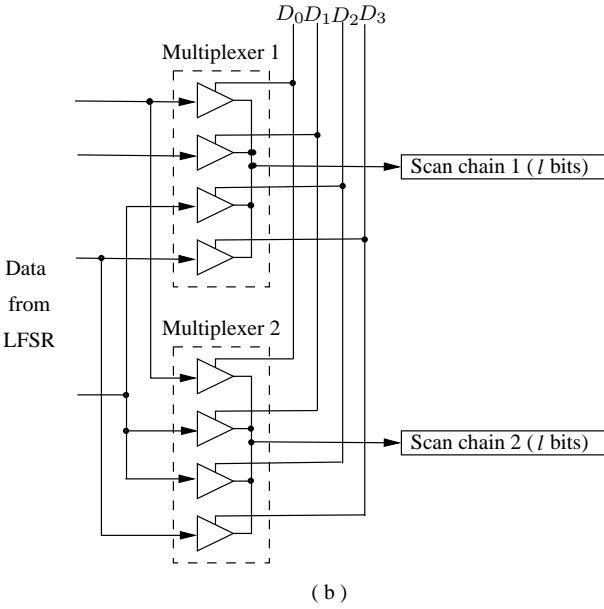
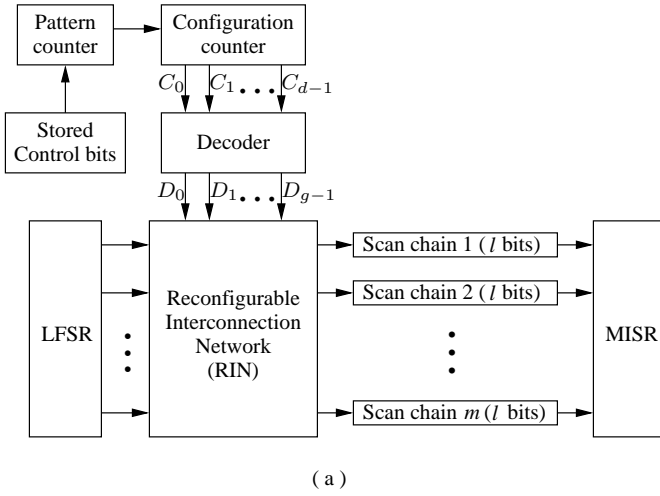


Fig. 2. (a) Proposed BIST architecture (b) Reconfigurable interconnection network for $m = 2$ and $g = 4$.

we use tristate buffers here because of ease of implementation in CMOS. The outputs of the tristate buffers are connected 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 equals the number of scan chains, the number of tristate gates in each multiplexer is equal to the number of configurations.

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

$$c: \text{xxx10 xx01x 1xx1x 0xxx1} \longrightarrow \begin{array}{l} t_1: 0\text{xxx1} \\ t_2: 1\text{xx1x} \\ t_3: \text{xx01x} \\ t_4: \text{xxx10} \end{array}$$

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

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 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.

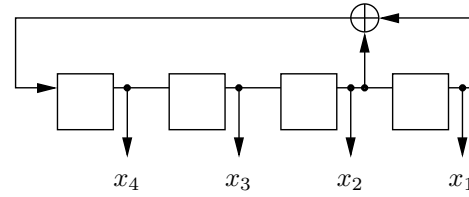
During test application, pseudo-random patterns that do not match any deterministic test cube are also applied to the CUT. These pseudo-random patterns can potentially detect non-modeled faults. However, these patterns increase the testing time. A parameter called *MaxSkipPatterns*, which is defined as the largest number of pseudo-random 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 and the interconnections between the LFSR outputs and the scan chains. We use a simulation procedure to solve this problem. We start with an LFSR of length L , a predetermined seed, and a known characteristic polynomial. Let $T_D = \{c_1, c_2, \dots, c_n\}$ 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 pseudo-random 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 $\{t_1, t_2, \dots, t_m\}$, 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 $Conn_j^{(i)}$ to denote the set of LFSR taps that are connected to the scan chain j in configuration i , where $i = 1, 2, \dots, g$, $j = 1, 2, \dots, m$. The simulation procedure is as follows.

- 1) Set $i = 1$.
- 2) Set $Conn_j^{(i)} = \{1, 2, \dots, L\}$ for $j = 1, 2, \dots, m$, 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 $\{O_k | k = 1, 2, \dots, L\}$, where vector O_k is the output stream of the k th 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 $Conn_j^{(i)}$, i.e., for all $j = 1, \dots, m$, there exists $k \in Conn_j^{(i)}$ such that t_j^* is compatible with O_k , where c^* has already been reformatted for m scan chains as a set of vector $\{t_1^*, t_2^*, \dots, t_m^*\}$. (A vector u_1, u_2, \dots, u_r and a vector v_1, v_2, \dots, v_r are mutually compatible if for any i , $1 \leq i \leq r$, one of the following holds: (i) $u_i = v_i$ if they are both care bits; (ii) u_i is a don't-care bit; (iii) 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 $j = 1, 2, \dots, m$, let $\mathbf{U} \subset Conn_j^{(i)}$ such that for any $k \in \mathbf{U}$, O_k is not compatible with t_j^* . Then set $Conn_j^{(i)} = Conn_j^{(i)} - \mathbf{U}$.
- 6) If in the previous *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 pseudo-random 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.

An example of the simulation procedure is illustrated in Fig. 4. A 4-bit autonomous LFSR with characteristic polynomial $x^4 + x + 1$ is used to generate the pseudo-random patterns. There are four scan chains and the length of each scan chain is 4 bits. The parameter *MaxSkipPatterns* is set to 1. The output of the LFSR is divided into patterns p_i , $i = 1, 2, \dots$. Each pattern consists of four 4-bit vectors. The procedure that determines the connections is shown as Step Init) to Step f). Step Init) is the initialization step in which all the connections $Conn_j^{(1)}$, $j = 1, 2, 3, 4$ 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 is shown for each scan chain: Scan chain 1 can be connected to x_1 or x_4 , both Scan chain 2 and Scan chain 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 is p_1, p_2, \dots, 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

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) $p_3 : \text{none}$	d) $p_4 : c_2$	e) $p_5, p_6 : \text{none}$	f) $p_2 : c_4$
(4)	(4)	(4)	(4)
(2)	(2)	(2)	(2)
(2)	(2)	(2)	(2)
(1, 4)	(1)	(1)	(1)

Fig. 4. An illustration of the simulation procedure.

connected to x_4 , both Scan chain 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 .

IV. 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 close 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. The interleaving method does not modify the scan chains, but it requires additional hardware.

The method of reorganization of scan chains is illustrated in Fig. 5. 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 [10]. However, the reorganization used in [10] changes the scan chain structure and makes it unsuitable for response capture—a separate solution is needed in [10] 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 $c_{i,j}$, $i = 0, 1, \dots, m - 1$, $j = 0, 1, \dots, l - 1$, 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 i th scan chain consists of the l scan cells $c_{i,j}$, $j = 0, 1, \dots, l - 1$. We use $c_{i,j}^*$ to denote the reorganized scan cells, in which the i th scan chain consists of the l scan cells $c_{i,j}^*$, $j = 0, 1, \dots, l - 1$. For each $j = 0, 1, \dots, l - 1$, the m cells $c_{0,j}, c_{1,j}, \dots, c_{m-1,j}$ 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 $k = (i + d) \bmod m$.

An alternative method for declustering, based on the interleaving of the inputs to the scan chains, is shown in Fig. 6. 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

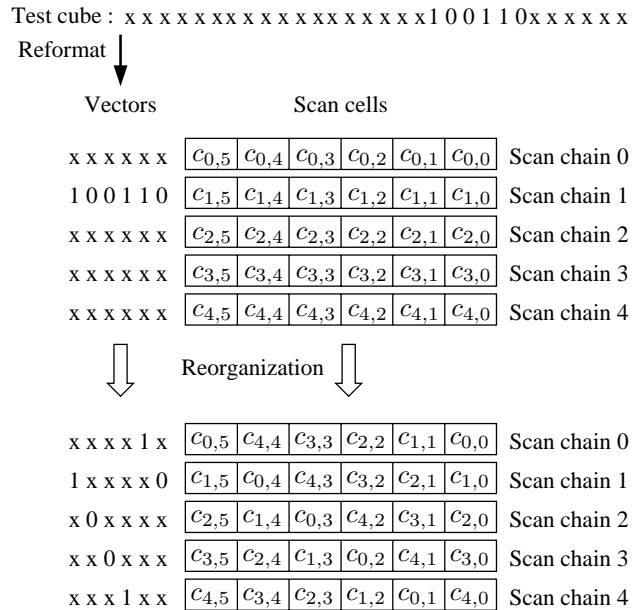


Fig. 5. An illustration of the reorganization of scan chains.

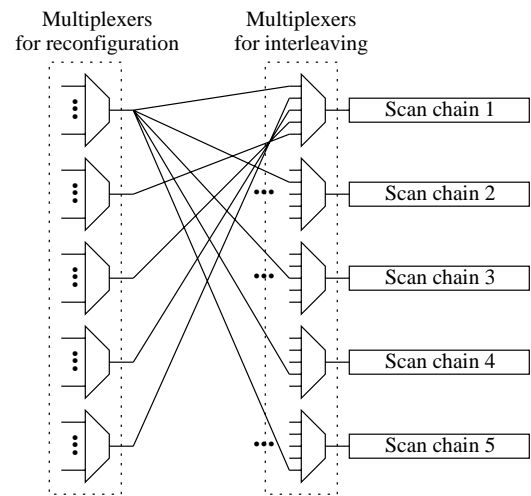


Fig. 6. An illustration of interleaving of the inputs of scan chains.

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 m th input is selected, the procedure is repeated with the first input. We use A_i to denote the output the i th multiplexers for reconfiguration and $B_{i,j}$ to denote the j th input of the i th multiplexers for interleaving, where $i, j = 1, 2, \dots, m$. The interleaving is carried out by connecting the inputs of the multiplexers for interleaving with

the outputs of multiplexers for reconfiguration such that

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

Consider the test cube shown in Fig. 5. 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. 6 (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 procedure yield the same set of test cubes.

V. EXPERIMENTAL RESULTS

In this section, we present experimental results for the seven largest ISCAS-89 circuits and for test cubes for the two production circuit 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 [9] 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 [10]. The second set of the test cubes are obtained without an initial pseudo-random pattern application, and they target all irredundant faults. The third set of the test cubes for the random-pattern-resistant faults is obtained after 10000 pseudo-random patterns are applied to the circuits. In all the following experiments, we used a 64-bit primitive-polynomial LFSR with a fixed seed as the pseudo-random pattern generator, and we assumed that each of the ISCAS-89 circuits contains 32 scan chains. 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.

Table I presents the results on test set embedding where T_D is obtained using the Mintest program. We use a value of 5000 for the *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. The hardware overhead includes the multiplexers for reconfiguration, the decoder and the configuration counter. We calculate the gate equivalent (GE) value for the hardware overhead using the method suggested in [28]: 0.5 n 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 percentage hardware overhead is obtained from the ratio of the amount of BIST hardware in GEs to the GE count 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 last column is the ratio of the number of care bits in the test set to the amount of storage needed.

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 than 8% for five circuits, and only 5.19% on average.

As indicated in Table I, we were unable to embed the test cubes for the s38417 benchmark circuit because the care bits 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. Table II shows the result obtained with scan chain reorganization for the seven largest benchmark circuits with 32 scan chains each.

Tables III and IV present experimental results on the embedding of test cubes from [10] targeting all faults, without scan chain reorganization and with scan chain reorganization, respectively. In these two sets of experiments, we set the parameter *MaxSkipPatterns* to 10000. As indicated in Table III, 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 IV. 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.

Table V presents experimental results obtained using the test cubes from [10] that target random-pattern-resistant faults. We assume that the RIN is bypassed using multiplexers for the first 10000 pseudo-random 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 IV. The average storage requirement for all seven circuits is reduced from 900 bits to 534 bits.

In the above set of experiments, we assume 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 therefore increase the hardware overhead. In the next set of experiments, we set the number of patterns for each configuration to 1000. As expected, the results in Table VI show that the hardware overhead increase slightly for each circuit. Nevertheless and important benefit here is that no storage is necessary for control bits.

Table VII compares the storage requirements of the proposed approach with hybrid BIST based on weighted pseudo-random patterns [15], test vector encoding using partial LFSR reseeding [17], the BIST scheme based on reseeding of folding counter [10], and two-dimensional test data compression [20]. The results for the proposed approach are taken from Table V. All these methods rely on 10000 initial pseudo-random patterns to eliminate the easy to detect faults, except for [15],

TABLE I
EXPERIMENTAL RESULTS FOR MINTEST TEST SETS TARGETING ALL FAULTS.

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 in GEs, and as a percentage	Storage requirement (bits)	Encoding efficiency
s5378	1458	214	7	9	157184	1257472	184.5 (7.08%)	162	81.44
s9234	1928	247	8	34	363551	3271959	690.0 (14.82%)	646	39.57
s13207	3237	700	22	9	322526	7418098	184.5 (2.36%)	171	157.02
s15850	3920	611	20	24	379377	7966917	478.5 (5.47%)	456	67.02
s35932	10810	1763	56	4	19756	1126092	79.0 (0.38%)	60	771.18
s38417*	10771	1664	52	—	—	—	—	—	—
s38584	13468	1464	46	11	377804	17756788	221.5 (1.05%)	209	441.15
Average	—	—	—	15	270033	6466221	306.3 (5.19%)	284	259.56

*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 AND WITH SCAN CELL REORGANIZATION.

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 in GEs, and as a percentage	Storage requirement (bits)	Encoding efficiency
s5378	1458	214	7	8	67988	543904	157.5 (6.04%)	136	97.01
s9234	1928	247	8	36	135765	1221885	729.0 (15.66%)	648	39.44
s13207	3237	700	22	9	152596	3509708	184.5 (2.36%)	162	165.75
s15850	3920	611	20	22	222336	4669056	440.5 (5.04%)	396	77.18
s35932	10810	1763	56	5	7079	403503	103.5 (0.49%)	65	711.86
s38417	10771	1664	52	272	625273	33139469	5752.5 (26.16%)	5440	26.14
s38584	13468	1464	46	12	383009	18001423	240.0 (1.14%)	228	404.39
Average	—	—	—	52	227721	8784135	1086.8 (8.13%)	1011	217.39

TABLE III
EXPERIMENTAL RESULTS FOR TEST SETS FROM [10] TARGETING ALL FAULTS.

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 in GEs, and as a percentage	Storage requirement (bits)	Encoding efficiency
s5378	4397	214	7	6	289754	2318032	121.5 (4.66%)	114	313.81
s9234	6475	247	8	57	394496	3550464	1138.5 (24.45%)	1083	72.55
s13207	9608	700	22	11	580521	13351983	221.5 (2.83%)	220	315.13
s15850	11330	611	20	32	758497	15928437	630.5 (7.21%)	640	129.96
s38417*	30859	1664	52	—	—	—	—	—	—
s38584	34493	1464	46	12	526555	24748085	240.0 (1.14%)	240	822.90
Average	—	—	—	24	509964	11979400	470.4 (8.06%)	459	330.87

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

TABLE IV
EXPERIMENTAL RESULTS FOR TEST SETS FROM [10] TARGETING ALL FAULTS AND WITH SCAN CELL REORGANIZATION.

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 in GEs, and as a percentage	Storage requirement (bits)	Encoding efficiency
s5378	4397	214	7	8	250275	2002200	157.5 (6.04%)	144	248.43
s9234	6475	247	8	31	562133	5059197	611.5 (13.13%)	620	126.73
s13207	9608	700	22	11	297713	6847399	221.5 (2.83%)	209	331.72
s15850	11330	611	20	18	634468	13323828	364.5 (4.17%)	360	231.04
s38417	30859	1664	52	184	1707485	90496705	3808.0 (17.32%)	3864	75.35
s38584	34493	1464	46	10	736686	34624242	203.0 (0.96%)	200	987.48
Average	—	—	—	44	698126	25392262	894.3 (7.41%)	900	333.46

which uses 32000 pseudo-random patterns. The results show that in all but one case, the proposed approach requires less storage than other methods. (The hybrid BIST method [15] requires less storage for s9234.)

In Table VIII, we compare the proposed method with scan-

based 3-weight weighted random BIST [30]. Since no storage of seeds or control bits is required in [30], we use the results from Table VI for comparison. The third column of Table VIII lists the number of pseudo-random patterns required to achieve 100% coverage of detectable single stuck-at faults, as reported

TABLE V

EXPERIMENTAL RESULTS FOR TEST SETS FROM [10] TARGETING RANDOM-PATTERN-RESISTANT FAULTS WITH SCAN CHAIN REORGANIZATION.

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 in GEs, and as a percentage	Storage requirement (bits)	Encoding efficiency
s5378	39	214	7	5	3269	26152	103.5 (3.97%)	60	10.27
s9234	698	247	8	34	34341	309069	690.0 (14.82%)	544	32.53
s13207	556	700	22	5	54776	1259848	103.5 (1.32%)	80	106.71
s15850	654	611	20	22	26968	566328	440.5 (5.04%)	330	45.15
s38417	2219	1664	52	120	103653	5493609	2431.5 (11.06%)	2040	28.66
s38584	441	1464	46	10	21282	1000254	203.0 (0.96%)	150	48.74
Average	—	—	—	33	40714	1442543	662.0 (6.20%)	534	45.34

TABLE VI

EXPERIMENTAL RESULTS FOR TEST SETS FROM [10] TARGETING RANDOM-PATTERN-RESISTANT FAULTS WITH SCAN CELL REORGANIZATION AND A FIXED NUMBER OF PATTERNS PER CONFIGURATION.

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 in GEs, and as a percentage	Storage requirement (bits)
s5378	39	214	7	5	4003	32024	103.5 (3.97%)	0
s9234	698	247	8	37	36001	324009	748.5 (16.08%)	0
s13207	556	700	22	8	8000	184000	157.5 (2.02%)	0
s15850	654	611	20	24	23002	483042	478.5 (5.47%)	0
s38417	2219	1664	52	129	128001	6784053	2680.5 (12.19%)	0
s38584	441	1464	46	12	11001	517047	240 (1.14%)	0
Average	—	—	—	36	35001	1387363	734.8 (6.81%)	0

TABLE VII

COMPARISON OF STORAGE REQUIRED FOR VARIOUS BIST METHODS.

Circuit	Hybrid BIST [15]	Partial Reseeding [17]	Reseeding of folding counter [10]	Two-dimensional compression [20]	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

TABLE VIII

COMPARISON WITH 3-WEIGHT WEIGHTED RANDOM BIST.

Circuit	No. of scan cells	Required random patterns	3-weight weighted random BIST [30]						Proposed approach		
			No. of random patterns	No. of weighted patterns	Total no. of patterns	Testing time (ms)	Hardware overhead (parallel)	Hardware overhead (serial)	Total no. of patterns	Testing time (ms)	Hardware overhead
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

in [1, 30]. The total number of patterns listed for our approach is obtained by adding 10000 to the number of patterns listed in Table VI. 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 [30] requires a smaller number of patterns for some circuits. This is because we use a multiple scan chain architecture, whereas [30] is based on a single scan chain architecture. The use of a sin-

gle scan chain architecture ensures that the hardware overhead in [30] is lower; however, in order to scale 3-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 [30] requires even less hardware, but it relies on explicit control of the set and reset signal of the flip-flops after scan cell reordering.

In order to evaluate the effectiveness of the proposed ap-

TABLE IX
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	No. of configurations	Testing time (clock cycles)	Hardware overhead in GEs, and as a percentage	Storage requirement (bits)	Encoding efficiency
CKT1	32	362921	64	5671	258	181504	9586.5 (0.15%)	3096	23.89
			128	2836	177	90784	12160.5 (0.19%)	2124	34.82
			256	1418	437	45408	58161.5 (0.89%)	4807	15.39
CKT2	4	1031072	64	16111	277	64448	10289.5 (0.77%)	3047	28.52
			128	8056	166	32228	11407.0 (0.86%)	1660	52.35
			256	4028	516	16116	68931.0 (5.19%)	4644	18.71

proach for large circuits, we applied the method to two real-life test sets from industry. The test set provided to us for the first circuit (CKT1) from IBM consists of 32 statically-compacted scan vectors (a total of 362921 bits of test data per vector). This microprocessor design consists of 3.6 million gates and 726000 latches. The test set for a second microprocessor circuit (CKT2) from IBM consists of a set of 4 scan vectors (a total of 1031072 bits of test data per vector); this design contains 1.2 million gates and 32200 latches. Since we do not have access to the gate-level models for these circuits, we are unable to report fault coverage values for these test sets. The number of scan chains is varied from 64 to 256 for these two circuits. We modified the simulation procedure such that the configuration of the interconnection network can be changed during the application of a test cube, and we set the parameter *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 which counts the number of bits that have been shifted into the scan chains. Table IX lists the results for these two industrial circuits. Since no additional pseudo-random patterns are applied to these circuits, the testing time is simply equal to the number of the clock cycles that are required to shift the deterministic patterns into the scan chains, and capture and scan out the responses. The hardware overhead is negligible and very high encoding efficiency is achieved for both circuits. Using a Sun-Blade-1000 workstation with one 750MHz UltraSPARC-III CPU and 1GB memory, the results for each row of Table IX are obtained in less than two minutes.

VI. CONCLUSION

We have presented a new approach for deterministic BIST based on the use of a reconfigurable interconnection network (RIN). The RIN is placed between the outputs of pseudo-random pattern generator, e.g., an LFSR, and the scan inputs of the circuit under test (CUT). It consists only of multiplexer switches and it is designed using a synthesis procedure that takes as inputs the pseudo-random 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. 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 non-intrusive 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.

ACKNOWLEDGMENT

We thank Prof. Sybille Hellebrand of University of Innsbruck in Austria for providing us with the test cubes used in [10]. We also thank Brion Keller of Cadence Design Systems (formerly with IBM Corporation) for providing us with the test cubes for the production circuits from IBM.

REFERENCES

- [1] M. F. AlShaibi and C. R. Kime, "MFBIST: a BIST method for random pattern resistant circuits," *Proc. Int. Test Conf.*, pp. 176–185, 1996.
- [2] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller and B. Koenemann, "OPMISR: the foundation for compressed ATPG vectors," *Proc. Int. Test Conf.*, pp. 748–757, 2001.
- [3] I. Bayraktaroglu and A. Orailoglu, "Test volume and application time reduction through scan chain concealment," *Proc. ACM/IEEE Design Automation Conf.*, pp. 151–155, 2001.
- [4] A. Chandra and K. Chakrabarty, "Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression," *Proc. VLSI Test Symp.*, pp. 42–47, 2001.
- [5] A. Chandra and K. Chakrabarty, "System-on-a-chip test data compression and decompression architectures based on Golomb codes," *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 355–368, March 2001.
- [6] K.-T. Cheng and C.-J. Lin, "Timing driven test point insertion for full-scan and partial-scan BIST," *Proc. Intl. Test Conf.*, pp. 506–514, 1995.

- [7] A. El-Maleh, S. al Zahir and E. Khan, "A geometric-primitives-based compression scheme for testing systems-on-chip," *Proc. VLSI Test Symp.*, pp. 54–59, 2001.
- [8] 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," *Proc. Design, Automation and Test in Europe Conf.*, pp. 604–611, 2002.
- [9] I. Hamzaoglu and J. H. Patel, "Test set compaction algorithms for combinational circuits," *Proc. Int. Conf. CAD*, pp. 283–289, 1998.
- [10] S. Hellebrand, H.-G. Liang and H.-J. Wunderlich, "A mixed-mode BIST scheme based on reseeding of folding counters," *Proc. Int. Test Conf.*, pp. 778–784, 2000.
- [11] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan and J. Rajski, "Logic BIST for large industrial designs: real issues and case studies," *Proc. Int. Test Conf.*, pp. 358–367, 1999.
- [12] V. Iyengar, K. Chakrabarty and B. T. Murray, "Deterministic built-in pattern generation for sequential circuits," *Journal of Electronic Testing: Theory and Applications*, vol. 15, pp. 97–115, 1999.
- [13] A. Jas and N. A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based design," *Proc. Int. Test Conf.*, pp. 458–464, 1998.
- [14] A. Jas, J. Ghosh-Dastidar and N. A. Touba, "Scan vector compression/decompression using statistical coding," *Proc. VLSI Test Symp.*, pp. 114–120, 1999.
- [15] A. Jas, C. V. Krishna and N. A. Touba, "Hybrid BIST based on weighted pseudo-random testing: a new test resource partitioning scheme," *Proc. VLSI Test Symp.*, pp. 2–8, 2001.
- [16] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth and D. Wheeler, "A SmartBIST variant with guaranteed encoding," *Proc. Asian Test Symp.*, pp. 325–330, 2001.
- [17] C. V. Krishna, A. Jas and N. A. Touba, "Test vector encoding using partial LFSR reseeding," *Proc. Int. Test Conf.*, pp. 885–893, 2001.
- [18] C. V. Krishna and N. A. Touba, "Reducing test data volume using LFSR reseeding with seed compression," *Proc. Int. Test Conf.*, pp. 321–330, 2002.
- [19] L. Li and K. Chakrabarty, "Test data compression using dictionaries with fixed-length indices," *Proc. VLSI Test Symp.*, pp. 219–224, 2003.
- [20] H.-G. Liang, S. Hellebrand and H.-J. Wunderlich, "Two-dimensional test data compression for scan-based deterministic BIST," *Proc. Int. Test Conf.*, pp. 894–902, 2001.
- [21] M. Nakao, S. Kobayashi, K. Hatayama, K. Iijima and S. Terada, "Low overhead test point insertion for scan-based BIST," *Proc. Int. Test Conf.*, pp. 348–357, 1999.
- [22] J. Rajski, J. Tyszer and N. Zacharia, "Test data decompression for multiple scan designs with boundary scan," *IEEE Trans. Computers*, vol. 47, pp. 1188–1200, November 1998.
- [23] 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," *Proc. Int. Test Conf.*, pp. 301–310, 2002.
- [24] S. Reda and A. Orailoglu, "Reducing test application time through test data mutation encoding," *Proc. Design, Automation and Test in Europe Conf.*, pp. 387–393, 2002.
- [25] S. M. Reddy, K. Miyase, S. Kajihara and I. Pomeranz, "On test data volume reduction for multiple scan chain design," *Proc. VLSI Test Symp.*, pp. 103–108, 2002.
- [26] L. Schafer, R. Dorsch and H.-J. Wunderlich, "RESPIN++ – deterministic embedded test," *Proc. European Test Workshop*, pp. 37–44, 2002.
- [27] C. Schotten and H. Meyr, "Test point insertion for an area efficient BIST," *Proc. Intl. Test Conf.*, pp. 515–523, 1995.
- [28] N. A. Touba and E. J. McCluskey, "Altering a pseudo-random bit sequence for scan based BIST," *Proc. Int. Test Conf.*, pp. 167–175, 1996.
- [29] E. H. Volkerink, A. Khoche and S. Mitra, "Packet-based input test data compression techniques," *Proc. Int. Test Conf.*, pp. 154–163, 2002.
- [30] S. Wang, "Low hardware overhead scan based 3-weight weighted random BIST," *Proc. Int. Test Conf.*, pp. 868–877, 2001.
- [31] H.-J. Wunderlich and G. Kiefer, "Bit-flipping BIST," *Proc. Int. Conf. Computer-Aided Design*, pp. 337–343, 1996.