Test Point Insertion in Hybrid Test Compression / LBIST Architectures

Elham Moghaddam, Nilanjan Mukherjee, Janusz Rajski, Jerzy Tyszer[*], Justyna Zawada[*]

Mentor Graphics Corporation
Wilsonville, OR 97070, USA

[*]Poznań University of Technology
60-965 Poznań, Poland

Abstract

Logic built-in self-test (LBIST), originally introduced for board, system, and in-field tests, is now being increasingly used with on-chip test compression. This hybrid approach allows LBIST to become a complementary solution for in-system test, where high quality, low power, low silicon area, and most importantly short test application time are key factors affecting ICs that are targeted for safety-critical and automotive systems. Test points are common in BIST-ready designs where they play a key role in reducing both test application time given a test coverage goal and the overall silicon overhead so that one can get a desired coverage with the minimal number of patterns. Unfortunately, these test points are typically dysfunctional when enabled in an ATPG-based test compression mode. Similarly, test points used to reduce ATPG-based test pattern counts cannot guarantee desired random testability. Incompatibility of both types of test points has motivated research presented in this paper. We present a novel hybrid test point technology designed to both reduce deterministic pattern counts and improve fault detection likelihood by means of the same minimal set of test points. Experimental results obtained for large industrial designs illustrate feasibility of the proposed hybrid test points and are reported herein.

1. Introduction

Test data compression – the very successful and industry-proven mainstream DFT methodology of the last decade – is now coping with the rapid pace of technological changes and the resultant test challenges. With semiconductor manufacturing processes scaling down to extremely small feature sizes, using multiple patterning lithography, and moving towards 3D structures, one can observe that the number of ATPG test vectors is exploding due to large combinational depths, staggeringly complex clocking schemes, excessive tail pattern counts, or complex RTL-based control features. Furthermore, in the mass-produced FinFET devices, the gate level abstraction and traditional fault models may not suffice to ensure high-quality and low-DPM metrics for state-of-the-art designs. However, tackling novel timing- and layout-related fault models and patterns based on a post-layout transistor-level netlist (such as cell-aware defects [13]) leads to inflated ATPG test sets that not only require more storage than many testers can afford but also unprecedentedly increase test application time, which is an efficiency-limiting and cost-increasing factor in testing of embedded systems, system-on-chip designs, multi-chip modules, or automotive electronics.

In parallel with test compression, logic built-in self-test (LBIST) has always been a vital research and development area. Classical LBIST applications include detecting infant mortality defects during burn-in test or enabling the use of low-cost and/or low-speed testers that only provide power and clock signals. With a mass market driving automotive, mobile or healthcare systems, LBIST with its ability to run structural tests in-system either during power-up or subsequently during system operation has emerged as an attractive complementary solution. For example, as RF transceivers are smaller than ever and consume less power, the semiconductor test industry is now trying to incorporate wireless communications into on-chip on-line LBIST solutions that ensure highly reliable device operations for the duration of its lifespan. This is of critical importance as the standard ISO 26262, defining functional safety features for automotive equipment applicable throughout the lifecycle of all electronic and electrical safety-related systems, calls for high fault coverage attainable in a very short period of test application time [17].

Since conventional LBIST fault coverage can be unacceptably low for a feasible pattern count, early BIST schemes used weighted random patterns to deal with the test data volume burden [20], [22], [30], [46]. Other techniques obtain desired stimuli by perturbing pseudorandom vectors [12], [14], [28], [29], [40], [41], [47]. But perhaps the most promising direction is to combine both technologies – test compression and LBIST – into a hybrid approach that can result in the shortest test time and very high-test coverage when used synergistically. In fact, there are several common features that test compression and LBIST may offer. For example, ATPG supports a type of plug-and-play capability for blocks that is similar to LBIST reuse with pattern retargeting (using block-level patterns at higher levels). LBIST low-power test remains similar to what one can do with test compression. The same applies to MISR-based diagnosis that has the precision of ATPG but uses the LBIST signatures. Finally, for products that need field system test, a type of hybrid approach will always be preferable because of high test quality of ATPG with the autonomous testing of LBIST.

When coping with random-resistant faults, many LBIST scheme select circuit's internal sites to subsequently add control points (CPs) or observe points (OPs) to activate

978-1-4673-8773-6/16/$31.00 ©2016 IEEE

faults or observe them, respectively [42]. Numerous empirical guidelines and approximate techniques were proposed to identify the most suitable CP and OP locations and improve the overall circuit testability. These methods are based, for example, on fault simulation [4], [19], [38], approximate testability measures [3], [6], [10], [31], or hybrid solutions working with cost functions [8], gradient-based schemes [36], and signal correlation [5]. Although test points may not be necessary in designs tested exclusively through compressed ATPG-produced patterns, it appears that their usage may help to reduce pattern counts [37], [48]. Depending on how a test point is driven or observed, it may require a few extra gates and wires routed to or from additional flip-flops. As it introduces area and performance penalty, the number of test points is usually limited. Furthermore, the identification of test points must be computationally inexpensive despite the complexity of large designs.

Although benefits of using a hybrid (deterministic compression/LBIST) approach are well pronounced, LBIST and ATPG test points remain orthogonal to each other, and therefore their use in hybrid solutions poses nontrivial challenges as they are poorly suited to the other party functionality, they can counteract each other, and, even worse, they can easily double related test logic silicon area. In this paper, we present a novel test point insertion scheme which, while resolving the compatibility problem, minimizes the number of test points needed to enhanced performance of both test compression and LBIST mechanisms at the same time. This technique is accompanied by a new LBIST test coverage estimation method that has been developed to quickly guide test point selection process.

2. Hybrid EDT/LBIST architecture

Earlier attempts to implement a hybrid ATPG/LBIST methodology had a drawback of having separate and expensive logic for BIST and test compression operations despite similarities between their functional objectives. Fortunately, it is the sharing of certain on-chip resources that has created the combination of test compression and logic BIST as a new promising direction in embedded test. There are several requirements, however, that an efficient hybrid approach has to satisfy to address both production and in-system test needs. In particular, a hybrid scheme should remain a valuable low cost manufacturing test solution with high fault coverage, low test data volume, and a short test time, especially for its LBIST mode whose ability to run a power-on self-test must be preserved as well. As scan-related over-testing and power dissipation have been of concern for years, the hybrid scheme should mimic the mission mode during test by reducing toggling to acceptable levels. Finally, the area overhead, primarily determined by a test controller with its low-power features, a source of test data, and a test response evaluator, should be kept at the absolute minimum.

The above observations have laid the foundation for a hybrid ATPG/LBIST scheme whose architecture is shown in Fig. 1. Solutions presented in the remaining parts of this paper will base on the embedded deterministic test (EDT) compression [33]. Here, several on-chip DFT resources are shared, including a single-block PRPG, a test response compactor, and the EDT/LBIST controller. Typically, this type of sharing provides additional 20% – 50% reduction in hardware cost. A single DFT automation flow enables both a flat as well as hierarchical integration of hybrid capabilities. Such a flow incorporates design rule checking, hybrid controller insertion and verification, scan insertion, and fault simulation across both pattern types. A hybrid EDT/LBIST controller can be accessed through a standard IEEE 1687 network, allowing easier access to the embedded test capabilities from anywhere in the system.

If used synergistically in manufacturing test, the hybrid

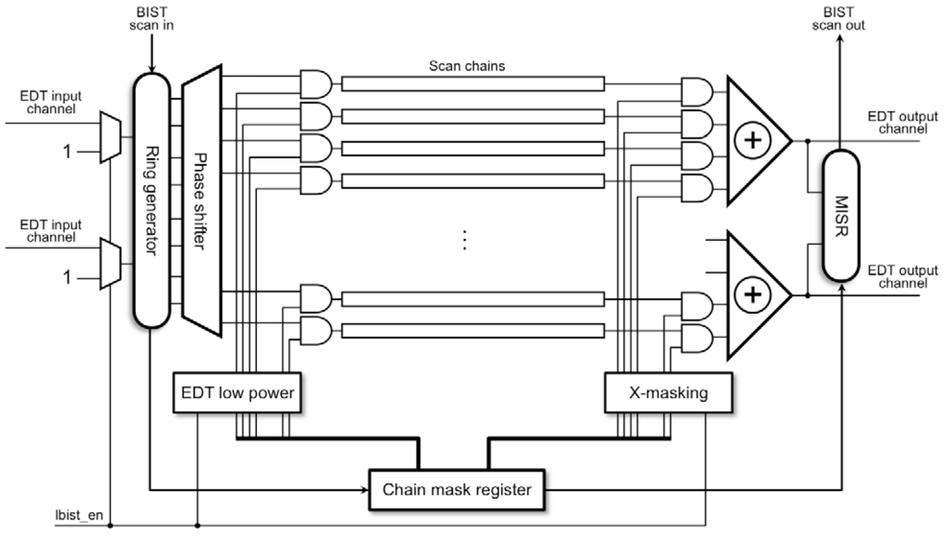

Figure 1 Hybrid test compression / LBIST architecture

scheme saves external test data and reduces the number of pseudorandom patterns. It operates in two steps. First, a pseudorandom pattern generator (PRPG) produces a predetermined number of random stimuli that detect random (or easy) testable faults. To avoid a prohibitively large number of random patterns, the second step targets the random resistant faults by using ATPG patterns whose compressed forms are applied through on-chip decompression logic. Now, the same hardware – a pseudorandom test pattern generator – is reused to decompress test cubes and to feed scan chains. This is only possible provided sequential test compression is deployed. Hence, as can be seen in Fig. 1, an *n*-bit ring generator and a phase shifter make up a sequential test data decompressor feeding scan chains. The decompressor receives compressed test data through input injectors (or EDT input channels). Furthermore, several two-input multiplexers are placed in the front of the ring generator. These multiplexers are controlled by a single *lbist_en* signal. When it is asserted, they feed the ring generator with constant 1s, which facilitates generation of pseudorandom patterns by turning the decompressor into a conventional PRPG.

On the output side, a MISR is used as a test response compactor in the LBIST mode. The very same MISR receives data from XOR trees acting as spatial test response compactors for bundles of scan chains. In addition to serving the MISR, the XOR trees produce compressed test data that are sent directly to an external tester, if a circuit operates in a deterministic test compression mode.

Note that *sharing* BIST logic for the purpose of data decompression has been considered earlier, for example, in [16], [18], [26], [32], [44]. If existing BIST infrastructure is used to handle compressed test data, then encoding schemes typically take advantage of low fill rates, as originally proposed in LFSR coding [23], and then static [9], [11], [15], [25], [27], [35], [43], [45], and dynamic [2], [7], [33] forms of LFSR reseeding. Surveys of these techniques can be found in [21] and [39].

Contrary to components presented earlier, a significant area overhead within a hybrid framework can also be attributed to two groups of test points associated with LBIST and ATPG. Their orthogonality may easily elevate a related silicon real-estate beyond acceptable levels. A proper selection of test points, their sharing and synergistic usage is therefore of primary interest in all hybrid schemes that use test points to boost random testability of a circuit and to reduce its test pattern count. In the next sections, we introduce a more comprehensive characterization of the test point insertion process and propose techniques to reduce the area overhead by inserting bifunctional test points at the most appropriate locations.

3. Two classes of test points

Traditionally, test point insertion (TPI) techniques have been used in support of LBIST to make random resistant logic more testable. Consider an example shown in Fig. 2. In order to propagate faults through gate G1, one needs to set the other input of this gate to 1. With pseudorandom patterns driving gate G2, the probability of having 1 on its output is relatively low. This is why inserting a test point between gates G1 and G2 alleviates this situation.

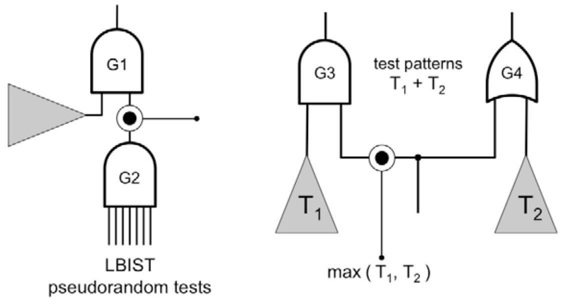

Figure 2 LBIST and EDT test points

A different scenario is presented in Fig. 2 for gates G3 and G4. Suppose T_1 test patterns are needed to detect faults propagating to gate G3. Similarly, T_2 tests are necessary to detect faults reaching gate G4. Clearly, in the former case the other input of gate G3 must be set to 1. It precludes, however, detection of faults propagating through gate G4. On the other hand, setting the other input of gate G4 to 0 blocks propagation of faults through gate G3. Because of mutually opposed fault propagation requirements corresponding to different non-controlling values for gates G3 and G4 driven by the same stem, faults in both groups cannot be detected by the same test patterns. This is because of incompatible decisions made by ATPG on internal lines due to fault excitation, backward justification, or fault propagation. The resultant number of test patterns will therefore be equal to $T_1 + T_2$. However, simultaneous detection of faults propagating to gates G3 and G4 would be possible provided a control point is placed on one of the stem branches to resolve the conflict. For example, one can insert a test point on the left branch to achieve independent 1-controllability of this line. Alternatively, a test point can be placed on the right branch to allow 0-controllability of this particular net. As a result, the number of test patterns becomes equal to $\max\{T_1 , T_2\}$, or, if $T_1 \approx T_2$, then the pattern count is basically reduced by half. As can be seen, test points placed in this type of locations might significantly reduce the overall pattern count.

Conflict-aware test points similar to those discussed above (and further dubbed EDT test points) have been introduced recently in [1], [24]. Contrary to LBIST test points, this new technology aims specifically at reducing ATPG test pattern counts and test data volume. A key feature of the scheme is its ability to identify and resolve conflicts between signals assigned to design's internal nodes by ATPG. This ability allows one to increase the number of faults detected by a single pattern, and thus to reduce both the number of deterministic tests and test data volume in a test compression environment, leading eventually to visibly shorter ATPG and test application times.

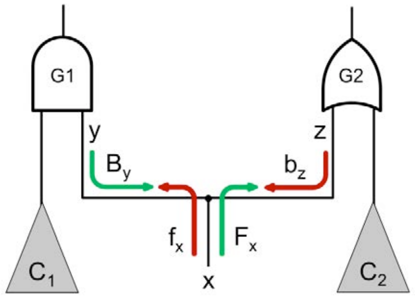

Figure 3 Conflict on internal lines

Interestingly, both EDT and LBIST test points use exactly the same logic structures as far as their implementation is concerned. Procedures used to determine the most prominent conflicts and to place test points that resolve them are comprehensively detailed in [1] and [24]. Here, for the sake of completeness, we recall their key concepts. Consider a circuit of Fig. 3. Let cones of logic C_1 and C_2 host D_1 and D_2 faults, respectively. If the only propagation paths from C_1 and C_2 go through gates G1 and G2, then the conflict-aware TPI will assess the degree of inconsistency between signals needed to propagate faults (backward justifications) and values forward implied (on stems) due to these decisions. The corresponding metrics used in this process are named b, B, and f, F, respectively [1]. For example, to propagate D_1 faults through gate G1, input y must be set to 1 at least $B_y = D_1$ times. Similarly, we will get $b_z = D_2$ (the number of 0s enabling fault propagation from C_2). It is worth noting that metrics b and B could also be regarded as the amount of blocked faults, should a given line was set to 1 and 0, respectively.

The values of b and B are deployed to estimate the numbers f and F of forward-implied 0s and 1s, due to backward justifications. For instance (compare Fig. 3), $f_y = b_z + f_x$, i.e., 0s at y are implied by 0s on branch z plus 0s occurring on stem x (as a result of earlier computations). Next, given a gate and its input values of f and F, simple formulas [1] can be used to determine the output values of f and F. As a result, one can measure the degree of conflicts occurring at a given fan-out branch by comparing the associated values of B and f (likewise branch y in Fig. 3) or values of b and F (branch z in Fig. 3). It is the result of this comparison that eventually determines the type and the location of a control test point helping to resolve ATPG-induced assignment conflicts.

4. Hybrid test points

Having two complementary classes of test points raises questions of practical importance. For example: can EDT test points improve LBIST-based test coverage? Or can LBIST test points reduce ATPG-based pattern counts? Tables 1 and 2 report industrial experimental results that address these concerns. The first two columns of both tables give the number of gates and the number of scan cells. The third column of Table 1 reports the baseline test coverage obtained for 10K pseudorandom patterns, while the third column of Table 2 provides the baseline ATPG pattern count. Now, given the number of inserted EDT test points (column TPs in Table 1), the last column of the table provides the resultant LBIST test coverage increase. In a similar manner, the last column of Table 2 reports test pattern count reduction relative to the number of tests needed to yield test coverage of the baseline case, i.e., with all LBIST test points inserted, as shown in column TPs. In all experiments the number of test points varies between 1% and 2.7% of memory elements deployed in the designs.

As can be seen in Table 1, EDT test points, by resolving internal conflicts, may also improve to some extent – as a side effect – designs' random testability. Clearly, it does not guarantee complete fault coverage yet. On the contrary, conventional LBIST test points [36] produce much fewer predictable results. Table 2 reveals that these test points may affect test pattern counts in both directions (cases of undesired PC grow are printed in bold). Consequently, one may conclude that neither EDT nor LBIST test points are exclusively suitable for hybrid test solutions. Furthermore, having both types of test point on chip may in fact compromise the resultant data compression, random testability, and a silicon real estate area.

Table 1. EDT TPs vs. LBIST test coverage

Gates	Scan cells	Baseline	EDT TPs	LBIST TC
1.5M	75K	79.31%	2K	84.08%
2.6M	154K	76.46%	1.5K	83.91%
3.6M	41K	81.43%	1.1K	84.63%
2.6M	150K	83.29%	3K	85.79%

Table 2. LBIST TPs vs. ATPG pattern count (PC)

Gates	Scan cells	Baseline	LBIST TPs	ATPG PC
2.1M	148K	10,175	3K	6,335
2.2M	143K	23,089	2.3K	**23,777**
4.4M	308K	69,606	4.5K	48,905
1.2M	63K	8,539	1.2K	**10,759**

In the light of the above findings, we propose a new methodology based on insertion of *hybrid test points*. Consider a circuit of Fig. 4. As far as EDT test points are concerned, we get $B_x = D_1 + D_3$ (for fan-out branch x). Clearly, setting x to 0 blocks propagation of faults from C_1, and then from C_3. To facilitate propagation of faults from C_2, we need to assert the branch y (here represented by $B_y = D_2$). In the worst case, therefore, stem s must be set to 1 for each fault within cones C_1, C_2, and C_3, that is, $B_s = B_x + B_y$. To estimate a degree of conflict it may cause, one needs to know values of f and F observed at stem s that primarily depend on the network driving that stem. Nevertheless, it is evident that having a conflict here may easily lead to undesired inflation of ATPG pattern counts. It is also evident that having 1 at the output gate G1 is a very unlikely (1 out of 2^{32}) random event because of its 32 inputs. To resolve this type of conflict, the analysis of [36] would identify the output of gate G1 as a suitable candidate for the OR type LBIST control test point allowing to significantly increase the probability of faults within cones C_1, C_2, and C_3 to move forward and possibly get detected.

Attractive as the output of gate G1 seems for both EDT and LBIST test points, it will need careful assistance in identification of such locations in a design to become mutually beneficial. This is a *hybrid conflict* that allows one to determine internal nets whose very high demands for

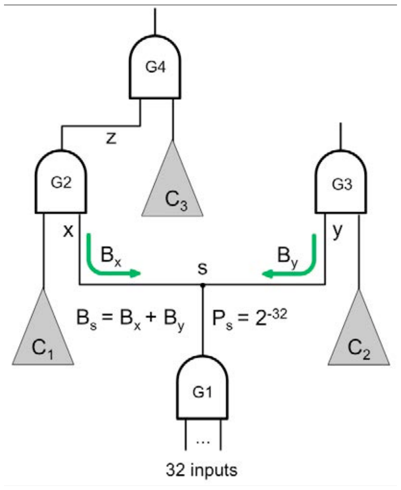

Figure 4 Hybrid conflict

signals of a given polarity come face to face with very low probabilities of actually getting them at those particular sites. To identify a hybrid conflict, we first assign EDT metrics *b* and *B* to each relevant line of a design. This is achieved as follows. Each fan-out stem in a circuit is assigned, in a levelized-order, the logic values of 0 and 1. These values propagate then forward until they hit inputs(s) of a gate as the non-dominant logic value for that particular type of gate (this propagation might continue provided all inputs of the gate are set to the non-dominant value). Clearly, the forward propagation may imply values on other stems and their branches. In this case, these nets will not be evaluated separately. Note that the dominant logic value at the input of a gate blocks fault propagation through that gate. In other words, one can determine metrics *b* and *B* of how many times the non-dominant values of 0 or 1 have to be assigned to this input in order to make the gate output sensitive to the incoming faults. In addition to these quantities, every *b* or *B* value accumulates, depending on whether the gate has inversion or not, the corresponding values of *b* or *B* obtained earlier for the gate output. By going backwards, the starting fan-out stem *x* receives eventually the following values:

$$b_x = b_{x1} + b_{x2} + \ldots + b_{xn} \qquad (1)$$

$$B_x = B_{x1} + B_{x2} + \ldots + B_{xn} \qquad (2)$$

where *n* is the number of fan-out branches. The stem values are now justified, depending on the gate type, in the process of back tracing, within a fan-out free region (FFR) it is driven by, thus producing values of *b* and *B* for relevant nets of the FFR. For example, let the gate G1 output (see Fig. 4) has $b_s = 0$ and $B_s = 100$. Since 1 is a non-controlling value for the AND gate, each input *k* of G1 gets $B_k = 100$, and the process continues backwards in a similar manner. On the other hand, we cannot justify b_s on the inputs of the same gate, as 0 is a dominant value here.

In addition to metrics *b* and *B*, the analysis of hybrid conflicts employs COP-based [3] line controllability measures. Let p_x and P_x be the probabilities of producing values of 0 and 1 at line *x*, respectively. Given a net *x*, we propose to measure the degree of conflict between a logic value enabling fault propagation and the probability of occurrence of this particular value at the same location by means of the following formulas:

$$c_x = -\, b_x \log_{10} p_x \qquad (3)$$

$$C_x = -\, B_x \log_{10} P_x \qquad (4)$$

Since $p_x \leq 1$ and $P_x \leq 1$, the larger the values of c_x or C_x, the smaller the chance that desired fault propagation conditions can be satisfactorily matched by randomly produced signals. For example, if $b_x = 15$, and $p_x = 0.0001$, then $c_x = 60$. It remains in a vivid contrast to $b_x = 15$, and $p_x = 0.5$, that yields $c_x = 4.52$. The former case clearly indicates that insertion of a test point at *x* is highly desired and should alleviate the backward justification problem, not to mention enhanced random controllability of *x*. Note that selection of a particular control test point type depends on which metric actually determines the dominating type of conflict. The AND control point is used if $c_x > C_x$, and the OR control point is deployed otherwise.

The following greedy steps are carried out in order to identify hybrid conflicts. A circuit is processed in a gate-level order. Starting from the first level, COP-based controllabilities are computed for each gate. Next, metrics *b* and *B* are determined for relevant lines by processing stems in a levelized-order. Note that initially all metrics *b* and *B* are set to 0. Since metrics b_s and B_s are equal to the number of blocked faults when stem *s* is set to 1 and 0, respectively, this contrapositive rule allows a circuit-tracing-based technique (essentially it counts faults that can reach stem *s*) to obtain estimations of b_s and B_s, and subsequently logic values assigned to nets within an FFR driving stem *s*. Eventually, formulas (3) and (4) assess the degree of potential conflicts associated with all relevant lines. Hence, the first iteration results in a sorted list of conflicts.

During the next iterations, we repeatedly remove the top of the list (occupied by the largest degree conflict represented by either c_x or C_x), and insert the corresponding test point into a design at a designated net *x*. Clearly, inserting a new control point affects metrics *b*, *B*, as well as signal controllabilities at several locations. It rearranges the list of conflicts. Consider the circuit of Fig. 5. Recall that the number B_x of 1s needed at stem *x* (or, alternatively, the number of blocked faults when line *x* is set to 0) is equal to $D_1 + D_2$.

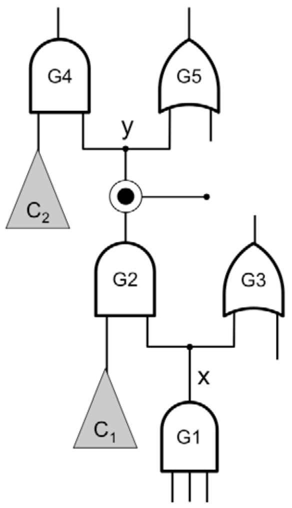

Figure 5 Example of hybrid test point

Suppose that the OR control point was added at the output of gate G2 after the first iteration. Now, B_x reduces to D_1, as the OR control point can stop propagation of the value of 0 further down the road. Moreover, the probabilities of having 0 and 1 at stem *y* change as a result of inserting of the same control point. These COP metrics are subsequently propagated forward, hence also changing conflicts in surroundings of gate G1.

In order to update the list of conflicts in the most efficient way, values of *b* and *B* are recomputed first, beginning with the line hosting a newly inserted control point, and following with each input gate as long as there is a difference between the former values and the updated ones. Furthermore, starting from the newly inserted control point, the signal probabilities are recalculated until the difference between the previous values and the new probabilities becomes smaller than a certain threshold. Finally, the conflicts are updated in all areas affected by the new control point. A next test point candidate is then taken from the updated list of conflicts.

5. Probabilistic test coverage estimation

The procedure presented in the last section iterates until the number of inserted control test points matches the desired and user-defined threshold, commonly defined as a small fraction of all memory elements used in a design. However, the same process can also be guided by monitoring the resultant test coverage as the algorithm keeps adding new test points into the design. Typically, probabilistic techniques based on various testability measures are more feasible here than a CPU-intensive fault simulation.

Conventional techniques, such as test coverage estimation based on COP [3] or SCOAP [10] measures, view the fault *f* detection probability T_f as the simple product formulas:

$$T_{x/0} = P_x \cdot O_x \tag{5}$$

$$T_{x/1} = p_x \cdot O_x \tag{6}$$

where *x*/0 and *x*/1 are stuck-at-0 and stuck-at-1 faults at line *x*, while P_x and p_x denote 1- and 0-controllabilities that measure difficulty of setting line *x* to 1 and 0, or exciting faults *x*/0 and *x*/1, respectively. The line *x* observability O_x is the probability of sensitizing a fault propagation path from *x* to a (pseudo) primary output. Consequently, test coverage for T test patterns can be estimated as the average over the entire list of testable faults by using the probabilities of detecting successive faults by at least one out of T test patterns. Unfortunately, the inherent inaccuracies of such computations are due to a simplifying assumption that signals at reconvergent fan-out stems are independent.

Consider a circuit in Fig. 6. Let $P_a = 0.1$, $P_b = 0.5$, and $P_c = 0.7$. As primary outputs, lines *g* and *h* are fully observable, i.e., $O_g = O_h = 1$. The COP-based observabilities of branches *e* and *f* are $O_e = O_g P_a = 0.1$ and $O_f = O_h p_c = 0.3$, whereas $O_d = 1 - (1 - O_e)(1 - O_f) = 0.37$. Now, we can determine $O_b = O_d P_c = 0.259$. If line *b* is stuck-at-0, then from (5) we get testability $T_{b/0} = P_b O_b = 0.1295$ of this fault.

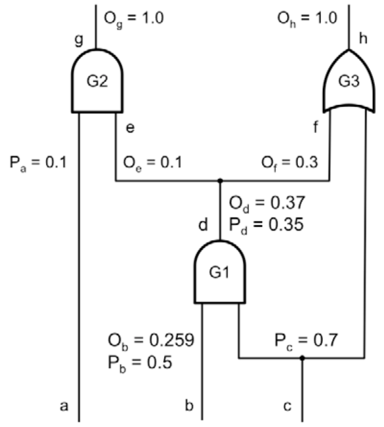

Figure 6 Basic test coverage estimation

A closer examination of Fig. 6 reveals that setting stem *c* to 1 enables propagation of *b*/0 through gate G1, but blocks the same at gate G3, leaving gate G2 as the only propagation path for *b*/0. Let us reexamine, therefore, how the *b*/0 detection probability can be computed. Assume that stem *c* is assigned the values of 0 (Fig. 7a), and then 1 (Fig. 7b). In the first case, straightforward computations indicate that observability of line *b* is 0. Let's denote this as $O_b | c_0 = 0$. When stem *c* is set to 1, we get $O_f | c_1 = 0$, though line *e* remains observable, i.e., $O_e | c_1 = 0.1$. Hence, fault *b*/0 is observed with probability $O_b | c_1 = O_d P_c = 0.1$. Since the observability of line *b* depends on a value assigned to stem *c*, we introduce a notion of *weighted observability* as follows (weighted metrics are underlined):

$$\underline{O_b} = O_b|c_0 \cdot p_c + O_b|c_1 \cdot P_c \tag{7}$$

In a similar manner one can determine the *weighted controllabilities* of line *b*:

$$\underline{p_b} = p_b|c_0 \cdot p_c + p_b|c_1 \cdot P_c \tag{8}$$

$$\underline{P_b} = P_b|c_0 \cdot p_c + P_b|c_1 \cdot P_c \tag{9}$$

Example: Let stem c in Fig. 7a be set to 0. As a result, we have $P_d|c_0 = 0$ and $p_d|c_0 = 1$. On the other hand, setting c to 1 implies $P_d|c_1 = 0.5$ and $p_d|c_1 = 0.5$. Thus, the weighted 0-controllability of line d is

$$\underline{p}_d = p_d|c_0 \cdot p_c + p_d|c_1 \cdot P_c = 1 \cdot 0.3 + 0.5 \cdot 0.7 = 0.65.$$

Clearly, the weighted 1-controllability of the same line is equal to $1 - 0.65 = 0.35$. Moreover, the weighted observability of line b equals

$$\underline{O}_b = O_b|c_0 \cdot p_c + O_b|c_1 \cdot P_c = 0 \cdot 0.3 + 0.1 \cdot 0.7 = 0.07.$$

As can be seen, the logic value at stem c has no impact on the controllability of b, and thus $\underline{p}_b = p_b$ and $\underline{P}_b = P_b$. The weighted probabilities allow now computing the *weighted detection probability* of faults at b as follows:

$$\underline{T}_{b/0} = \underline{P}_b \cdot \underline{O}_b \qquad (10)$$

$$\underline{T}_{b/1} = \underline{p}_b \cdot \underline{O}_b \qquad (11)$$

In particular, the weighted detection probability of fault b/0 is $\underline{T}_{b/0} = 0.5 \cdot 0.07 = 0.035$. Recall that testability of fault b/0 produced by the method proposed in this section is almost four times smaller and much more realistic than that obtained by using conventional metrics (0.13, see Fig. 6). This is because traditional testability measures assume that fault b/0 may propagate either through gate G2 or gate G3 (compare Fig. 6), whereas one of these propagation paths is in fact disabled. This wrong assumption leads to incorrect overestimation of b/0 testability.

In general, several stems may affect a single line x. So, having individual impact of every stem on line x controllability, observability, and their weighted counterparts (determined by propagating values of 0 and 1, assigned to stems, as long as they remain the dominant values for gates they pass), we propose to measure their *impact* on controllability (Δ_x) and observability (δ_x) of a given line x by computing these two metrics as follows:

$$\Delta_x = \max\{\max\{p_x, \underline{p}_x\} / \min\{p_x, \underline{p}_x\},$$
$$\max\{P_x, \underline{P}_x\} / \min\{P_x, \underline{P}_x\}\} \qquad (12)$$

$$\delta_x = \max\{O_x, \underline{O}_x\} / \min\{O_x, \underline{O}_x\} \qquad (13)$$

The values of Δ_x and δ_x indicate how sensitive the testability measures of a given node x are relative to changes occurring on a stem driving that site. Eventually, the actual impact is determined as a maximum over such metrics obtained independently for all involved stems.

Example: Assume that testability measures for line b are: $p_b = 0.01$ and $P_b = 0.99$. Let x, y, and z be stems whose impact on weighted controllabilities of b is as follows: $\underline{p}_b = 0.2$, $\underline{P}_b = 0.8$ (stem x), $\underline{p}_b = 0.0001$, $\underline{P}_b = 0.9999$ (stem y), and $\underline{p}_b = 0.1$, $\underline{P}_b = 0.9$ (stem z). Consequently, the degree of influence of stems x, y, and z on controllability of line b is $\Delta_b = \max\{20, 1.125\} = 20$, $\Delta_b = \max\{100, 1.01\} = 100$, and $\Delta_b = \max\{10, 1.1\} = 10$, respectively. Clearly, stem y causes the most significant change of line b 0-controllability that appears to be two orders of magnitude smaller than that of the conventional approach.

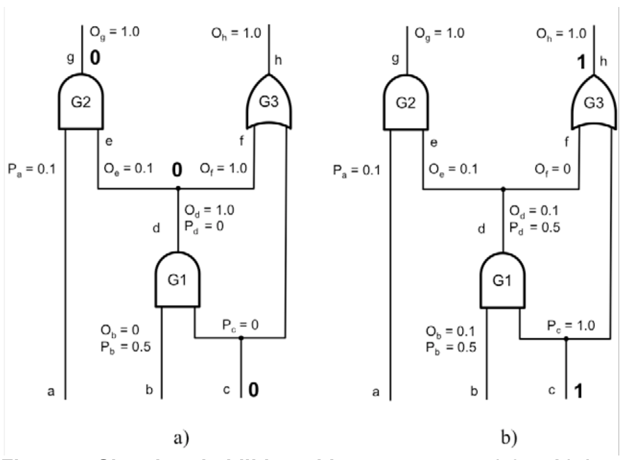

Figure 7 Signal probabilities with stem c set to a) 0 or b) 1

To find the weighted detection probabilities for successive faults, we process a circuit in the gate-level order in two major steps. First, compute COP-based controllabilities and observability for each line, followed by the weighted controllability calculus starting from the primary inputs. Note that Δ of each line is initially equal to 1. Furthermore, each stem is processed as follows:

1) Inject values of 0 and 1, propagate them forward while they are the dominant values for a particular type of gate (or all inputs of the gate are set to a non-dominant value), and compute controllabilities for affected nets.
2) Determine the weighted controllabilities of each affected line x, and find the corresponding Δ_x based on these values.
3) If new Δ_x is greater than its current value, save the weighted controllabilities and change Δ_x; otherwise proceed to the next stem.

When this phase comes to an end, each net receives the weighted controllabilities corresponding to maximal Δ's. They allow one to update the line observabilities, starting from the primary outputs, based on these newly assigned metrics. Next, we begin the second phase to determine the weighted observabilities. Again, δ_x for each line x assumes initially the value of 1. Starting from the primary outputs, we process each stem as follows:

1) Repeat step 1 of the first phase; this time, however, propagate logic values forward and backward. In the latter case, stop propagation once a controlling value reaches a gate output. Moreover, redirect a logic value that hits another stem through one of its branches to the remaining branches, and continue forward propagation from there. Once both forms of propagation die out, update the controllabilities, and then the observability of each affected line.
2) Determine the weighted observability of each affected line x and the corresponding δ_x, which is saved, if its new value is greater than the current one assigned to the line.

When the second phase is finished, each line x assumes the weighted observability based on the recorded the maximal value of δ_x. As a result, one can estimate the test coverage through computing of weighted detection probabilities, as defined by formulas (10), (11). Note that all (pseudo) primary inputs are fully controllable ($P_i = p_i = 0.5$) and the (pseudo) primary outputs are fully observable ($O_o = 1$).

Table 3. Circuit characteristics

	Gates	Scan cells	Scan chains	EDT In : Out	Chain / channels	CPs	OPs
D1	1.2M	72K	400 × 181	4 ; 4	100x	500	1.5K
D2	2.3M	252K	490 × 515	4 ; 4	123x	1.2K	3.8K
D3	1.5M	162K	330 × 491	3 ; 3	110x	680	2.52K
D4	3.3M	326K	400 × 814	4 ; 4	100x	2K	4K
D5	1.2M	85K	428 × 200	6 ; 6	72x	600	1.1K
D6	2.1M	143K	400 × 359	4 ; 4	100x	800	2.2K
D7	1M	57K	400 × 143	4 ; 4	100x	600	1.4K
D8	1.6M	144K	400 × 366	4 ; 4	100x	1K	1.8K

6. Experimental results

The novel DFT technology and its probabilistic test coverage estimation have been verified by conducting experiments on 8 industrial designs. The basic data regarding these circuits as baseline test cases are listed in Table 3. It includes the number of gates, the number of scan cells, scan architecture, the EDT interface, and the target input compression. Moreover, for the reference purpose, the last two columns provide the number of control points (CPs) and observation points (OPs) whose breakdown was obtained through the analysis presented in [36]. The actual number of test points is a design-dependent factor and varies between 1.9% and 3.3% of the total number of memory elements in the design.

First, we examine how different types of test points affect both the ATPG pattern count and LBIST test coverage. For each test case, four different configurations are used: EDT test points [1] only, LBIST test points only based on traditional COP-based testability measures [36], and hybrid test points of Section 4 combined with EDT test points in two different scenarios, denoted as EDT/H and H/EDT, respectively. In the EDT/H scheme, EDT test points are inserted first (half of the total number of test points), and then come hybrid test points. H/EDT indicates that hybrid test points are inserted first, followed by EDT test points. All configurations assume the number of control and observation points as shown in Table 3. In the EDT/H and H/EDT setups, however, these numbers are evenly split up between both components. EDT and hybrid test points have been merged because of EDT test points ability to identify crucial ATPG conflicts, which may escape the hybrid approach. Furthermore, as shown in Table 1, EDT test points can also improve LBIST test coverage. In all experiments reported in this section, the observation points in both EDT/H and H/EDT schemes are inserted by deploying the state-of-the-art method of [34].

Table 4 presents ATPG pattern counts. Column PC reports the baseline (no test points) pattern count with the corresponding test coverage in column TC. The next columns list pattern counts necessary to get the same test coverage as that of the baseline case once different TPI configurations are employed. Note that in all test cases but D2, the combination of EDT and hybrid test points results in either lower pattern counts or similar to those when EDT test

Table 4. ATPG pattern count

	PC	TC [%]	EDT	EDT / H	H / EDT
D1	8,633	96.99	5,178	4,781	4,757
D2	4,397	99.06	1,805	2,048	1,966
D3	4,946	99.05	2,432	2,432	2,581
D4	3,515	96.49	2,150	2,066	2,117
D5	4,377	98.27	3,072	2,964	2,778
D6	24,027	99.51	8,429	7,817	7,478
D7	18,450	95.75	5,512	4,672	4,298
D8	2,874	97.60	1,803	1,471	1,551

Table 5. LBIST test coverage

	TC [%]	LBIST [%]	EDT/H [%]	H/EDT [%]
D1	76.70	82.75	87.70	87.88
D2	85.50	93.16	95.54	96.26
D3	82.82	86.24	93.03	92.26
D4	83.76	86.36	90.60	90.08
D5	85.08	90.82	90.67	90.65
D6	78.65	79.84	86.30	85.95
D7	74.53	82.32	89.69	89.43
D8	87.82	91.95	94.53	94.27

Table 6. Test coverage estimation

Patterns × 10^3		EDT / H			H / EDT		
		TC [%]	New scheme	DR	TC [%]	New scheme	DR
D1	10	87.70	-0.27	8.67x	87.88	-0.24	9.08x
	100	91.58	0.09	14.00x	91.66	0.08	15.00x
D2	10	95.54	-5.07	0.77x	96.26	-5.59	0.75x
	100	96.94	-4.77	0.92x	97.41	-4.76	0.91x
D3	10	93.03	-3.91	0.34x	92.26	-3.00	0.11x
	100	96.11	-2.60	0.39x	96.00	-2.55	0.43x
D4	10	90.60	0.41	4.51x	90.08	0.42	4.52x
	100	93.07	0.48	2.23x	92.59	0.47	2.36x
D5	10	90.67	1.18	2.21x	90.65	0.50	4.42x
	100	92.35	1.55	1.32x	92.42	0.55	3.33x
D6	10	86.30	1.43	2.22x	85.95	1.51	2.18x
	100	89.74	1.08	1.81x	89.21	1.18	1.89x
D7	10	89.69	-0.10	18.20x	89.43	-0.20	9.20x
	100	92.44	0.05	23.80x	92.33	-0.10	11.10x
D8	10	94.53	-0.97	0.65x	94.27	-0.97	0.76x
	100	96.01	-0.16	3.44x	95.87	-0.10	6.50x

points are used exclusively. It also appears that it might be preferable to use EDT/H configuration rather than H/EDT, or vice versa. As can be seen from the table, the combined test point solution provides, on the average, 2.3 times pattern count reduction for both EDT/H and H/EDT.

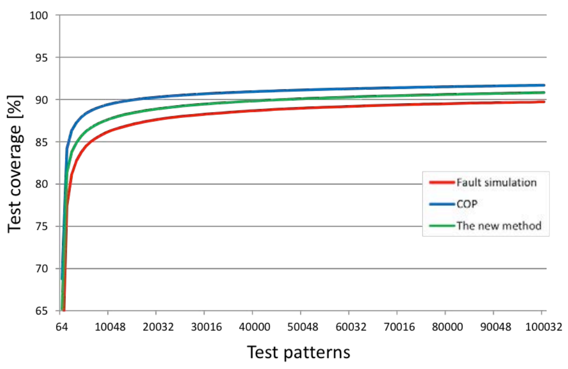

Figure 8 Test coverage (design D6, EDT/H)

Table 5 reports what happens in terms of test coverage when exactly the same test points as above are used within the framework of logic BIST applying 10K pseudorandom test patterns. Column TC corresponds to the baseline test case. Once again, the combined approach yields higher test coverage than that of conventional LBIST test points (column LBIST). Design D5 is an exception here with its minor (0.15%) coverage drop. The hybrid test points not only resolve ATPG-induced conflicts but also improve design random testability. We also observe that EDT test points, whose primary role is to reduce ATPG pattern counts, enable – due to their sites – detection of random-resistant faults.

Finally, we evaluate an accuracy of the probabilistic test coverage estimation introduced in Section 5 for 10K and 100K pseudorandom patterns. In particular, a departure from the reference fault simulation-based test coverage (column TC in Table 6) is used as a primary figure of merit. The results due to the new method for EDT/H and H/EDT scenarios are shown in Table 6 as a difference between TC and the proposed estimator (negative numbers represent estimations lower than TC). We also compare the new algorithm with conventional COP-based test coverage estimation [3] by using a difference reduction (DR) factor as follows:

$$DR = | TC - TC_{COP} | / | TC - TC_{NEW} | \qquad (14)$$

where TC_{COP} and TC_{NEW} are test coverage estimations obtained by means of COP and the new scheme presented in this work, respectively. As can be seen, DR indicates how much the new estimator can refine results over the conventional technique. Column DR of Table 6 clearly demonstrates superiority of the new method as it offers more realistic test coverage estimation than COP-based algorithm in most cases. The average DR (with the exception of D2 and D3 – see below) for 10K random patterns equals 6.08 and 5.03 for EDT/H and H/EDT, respectively. Similarly, for 100K vectors, the same statistics amounts to 7.76 (EDT/H) and 6.7 (H/EDT). It is worth noting that our approach assumes a single capture per clock. In this scheme, all scan cells have the same probability of getting 0 and 1 before a clock pulse. This assumption does not hold for customized procedures such as a multi-clock capture used in designs D2 and D3, where a capture cycle follows a number of additional clock cycles.

Fig. 8 illustrates, for design D6 with a setup EDT/H, how test coverage changes with the increasing number of random patterns. The red curve corresponds to test coverage produced by a fault simulator. The blue and green lines represent the coverage estimations computed by a COP-based algorithm and the new method of Section 5, respectively. Clearly, the proposed probabilistic analysis provides more accurate results than the conventional testability measures-based estimation. Similar results, not reported here, have been obtained for the remaining designs.

7. Conclusion

In this paper, we introduce the new DFT technology that aims at reducing deterministic test pattern counts, while at the same time it increases circuits' random testability. This is accomplished by deploying EDT and hybrid test points that work synergistically in designs where on-chip sequential test compression is integrated with logic BIST infrastructure. It offers not only a comprehensive test solution for manufacturing tests, but also reduces test application time – a key factor when running in-system tests for safety-critical and automotive applications. The new scheme identifies internal conflicts precluding efficient ATPG-based test compaction and detection of random resistant faults. Locations corresponding to such hybrid conflicts are modified by inserting test points, which increase the number of faults targeted by a single pattern, significantly reduce ATPG test pattern counts and the resultant test data volume, and finally allow one to run high-quality tests during BIST sessions. Experimental results obtained for large industrial designs with on-chip hybrid test logic confirm feasibility of the proposed solution.

8. References

[1] C. Acero, F. Hapke, D. Feltham, E. Moghaddam, N. Mukherjee, V. Neerkundar, M. Patyra, J. Rajski, J. Tyszer, and J. Zawada, "Embedded deterministic test points for compact cell-aware tests," *Proc. ITC*, 2015, paper 2.2.

[2] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, A. Ferko, B. Keller, D. Scott, B. Koenemann, and T. Onodera, "Extending OPMISR beyond 10x scan test efficiency," *IEEE Design & Test*, vol. 19, No. 5, pp. 65-73, 2002.

[3] F. Brglez, P. Pownall, and R. Hum, "Applications of testability analysis: from ATPG to critical delay path tracing," *Proc. ITC*, 1984, pp. 705-712.

[4] A.J. Briers and K.A.E. Totton, "Random pattern testability by fault simulation," *Proc. ITC*, 1986, pp. 274-281.

[5] S.-C. Chang, S.-S. Chang, W.-B. Jone, and C.-C. Tsai, "A novel combinational testability analysis by considering signal correlation," *Proc. ITC*, 1998, pp. 658-667.

[6] K.-T. Cheng and C.-J. Lin, “Timing-driven test point insertion for full-scan and partial-scan BIST,” *Proc. ITC*, 1995, pp. 506-514.

[7] D. Czysz, G. Mrugalski, N. Mukherjee, J. Rajski, P. Szczerbicki, and J. Tyszer, “Deterministic clustering of incompatible test cubes for higher power-aware EDT compression”, *IEEE Trans. CAD*, vol. 30, pp. 1225-1238, 2011.

[8] M.J. Geuzebroek, J.T. van der Linden, and A.J. van de Goor, “Test point insertion that facilitates ATPG in reducing test time and data volume,” *Proc. ITC*, 2002, pp. 138-147.

[9] V. Gherman, H.-J. Wunderlich, H. Vranken, F. Hapke, M. Wittke, and M. Garbers, “Efficient pattern mapping for deterministic logic BIST,” *Proc. ITC*, 2004, pp. 48–56.

[10] L.H. Goldstein and E.L. Thigpen, “SCOAP: Sandia controllability/observability analysis program,” *Proc. DAC,* 1980, pp. 190-196.

[11] A.-W. Hakmi, S. Holst, H.-J. Wunderlich, J. Schloffel, F. Hapke, and A. Glowatz, “Restrict encoding for mixed-mode BIST,” *Proc. VTS*, 2009, pp. 179-184.

[12] A.-W. Hakmi, H.-J. Wunderlich, C.G. Zoellin, A. Glowatz, F. Hapke, J. Schloeffel, and L. Souef, “Programmable deterministic built-in self-test,” *Proc. ITC*, 2007, paper 18.1.

[13] F. Hapke, W. Redemund, A. Glowatz, J. Rajski, M. Reese, M. Hustava, M. Keim, J. Schloeffel, and A. Fast, “Cell-aware test,” *IEEE Trans. CAD,* vol. 33, pp. 396-1409, 2014.

[14] S. Hellebrand, H.-G. Liang, and H.-J. Wunderlich, “A mixed mode BIST scheme based on reseeding of folding counters,” *Proc. ITC*, 2000, pp. 778-784.

[15] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, “Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers,” *IEEE Trans. Comput*., vol. 44, pp. 223-233, 1995.

[16] S. Hellebrand, B. Reeb, S. Tarnick, and H.-J. Wunderlich, “Pattern generation for a deterministic BIST scheme,” *Proc. ICCAD*, 1995, pp. 88—941.

[17] C. Hobbs and P. Lee, “Understanding ISO 26262 ASILs,” *Electronic Design*, July 9, 2013.

[18] K. Ichino, T. Asakawa, S. Fukumoto, K. Iwasaki, and S. Kajihara, “Hybrid BIST using partially rotational scan,” *Proc. ATS*, 2001, pp. 379-384.

[19] V.S. Iyengar and D. Brand, “Synthesis of pseudorandom pattern testable designs,” *Proc. ITC*, 1989, pp. 501-508.

[20] A. Jas, C.V. Krishna, and N.A. Touba, “Weighted pseudorandom hybrid BIST,” *IEEE Trans. VLSI*, vol. 12, pp. 1277-1283, 2004.

[21] R. Kapur, S. Mitra, and T.W. Williams, “Historical perspective on scan compression,” *IEEE Design & Test*, vol. 25, No. 2, pp. 114-120, 2008.

[22] R. Kapur, S. Patil, T.J. Snethen, and T.W. Williams, “A weighted random pattern test generation system,” *IEEE Trans. CAD*, vol. 15, pp. 1020-1025, 1996.

[23] B. Koenemann, “LFSR-coded test patterns for scan designs,” *Proc. ETC*, 1991, pp. 237-242.

[24] H. Konuk, E. Moghaddam, N. Mukherjee, J. Rajski, D. Solanki, J. Tyszer, and J. Zawada, “Design for low test pattern counts,” *Proc. DAC*, 2015, paper 58.4.

[25] C.V. Krishna and N.A. Touba, “Reducing test data volume using LFSR reseeding with seed compression,” *Proc. ITC*, 2002, pp. 321-330.

[26] C.V. Krishna and N.A. Touba, “Hybrid BIST using an incrementally guided LFSR,” *Proc. Symp. Defect and Fault Tolerance*, 2003, pp. 217-224.

[27] J. Lee and N. A. Touba, “LFSR-reseeding scheme achieving low-power dissipation during test,” *IEEE Trans. CAD*, vol. 26, pp. 396-401, 2007.

[28] L. Lei and K. Chakrabarty, “Test set embedding for deterministic BIST using a reconfigurable interconnection network,” *IEEE Trans. CAD*, vol. 23, pp. 1289-1305, 2004.

[29] H.-G. Liang, S. Hellebrand, and H.-J. Wunderlich, “Two-dimensional test data compression for scan-based deterministic BIST,” *Proc. ITC*, 2001, pp. 894-902.

[30] F. Muradali, V. K. Agarwal, and B. Nadeau-Dostie, “A new procedure for weighted random built-in self-test,” *Proc. ITC*, 1990, pp. 660-669.

[31] M. Nakao, K. Hatayama, and I. Highasi, “Accelerated test points selection method for scan-based BIST,” *Proc. ATS*, 1997, pp. 359-364.

[32] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, “Decompressor/PRPG for applying pseudo-random and deterministic test patterns,” US patent 6,684,358, 2004.

[33] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, “Embedded deterministic test,” *IEEE Trans. CAD*, vol. 23, pp. 776-792, 2004.

[34] S. Romersaro, J. Rajski, T. Rinderknecht, S.M. Reddy, and I. Pomeranz, “ATPG heuristics dependent observation point insertion for enhanced compaction and data volume reduction,” *Proc. DFTVS*, 2008, pp. 385-393.

[35] P. M. Rosinger, B. M. Al-Hashimi, and N. Nicolici, “Low power mixed-mode BIST based on mask pattern generation using dual LFSR re-seeding,” *Proc. ICCD*, 2002, pp. 474-479.

[36] B.H. Seiss, P. Trouborst, and M. Schulz, “Test point insertion for scan-based BIST,” *Proc. ETC*, 1991, pp. 253-262.

[37] R. Sethuram, S. Wang, S.T. Chakradhar, and M.L. Bushnell, “Zero cost test point insertion technique to reduce test set size and test generation time for structured ASICs,” *Proc. ATS*, 2006, pp. 339-348.

[38] N. Tamarapalli and J. Rajski, “Constructive multi-phase test point insertion for scan-based BIST,” *Proc. ITC*, 1996, pp. 649-658.

[39] N.A. Touba, “Survey of test vector compression techniques,” *IEEE Design & Test*, vol. 23, pp. 294-303, 2006.

[40] N.A. Touba and E.J. McCluskey, “Transformed pseudo-random patterns for BIST,” *Proc. VTS*, 1995, pp. 2-8.

[41] N.A. Touba and E.J. McCluskey, “Bit-fixing in pseudo-random sequences for scan BIST,” *IEEE Trans. CAD*, vol. 20, pp. 545-555, 2001.

[42] H.-C. Tsai, K.-T. Cheng, Ch.-J. Lin, and S. Bhawmik, “A hybrid algorithm for test point selection for scan-based BIST,” *Proc. DAC*, 1997, pp. 478-483.

[43] Z. Wang and K. Chakrabarty, “Test data compression for IP embedded cores using selective encoding of scan slices,” *Proc. ITC*, 2005, pp. 581-590.

[44] P. Wohl, J.A. Waicukauski, S. Patel, and M. Amin, “Efficient compression and application of deterministic patterns in a logic BIST architecture,” *Proc. DAC*, 2003, pp. 566-569.

[45] P. Wohl, J.A. Waicukauski, S. Patel, F. DaSilva, T.W. Williams, and R. Kapur, “Efficient compression of deterministic patterns into multiple PRPG seeds,” *Proc. ITC*, 2005, pp. 916-925.

[46] H.-J. Wunderlich, “Multiple distributions for biased random test patterns,” *IEEE Trans. CAD*, vol. 9, pp. 584-593, 1990.

[47] H.-J. Wunderlich and G. Kiefer, “Bit-flipping BIST,” *Proc. ICCAD*, 1996, pp. 337-343.

[48] M. Yoshimura, T. Hosokawa, and M. Ohta, “A test point insertion method to reduce the number of test patterns,” *Proc. ATS*, 2002, pp. 298-304.