Efficient Multiphase Test Set Embedding for Scan-based Testing

E. Kalligeros[1,2], X. Kavousianos[1] and D. Nikolos[2]
[1]*Computer Science Dept., University of Ioannina, 45110 Ioannina, Greece*
[2]*Computer Engineering & Informatics Dept., University of Patras, 26500 Patras, Greece*
kalliger@ceid.upatras.gr, kabousia@cs.uoi.gr, nikolsd@cti.gr

Abstract

In this paper a new test set embedding method with reseeding for scan-based testing is proposed. The bit sequences of multiple cells of an LFSR, which is used as test pattern generator, are exploited for effectively encoding the test set of the core under test (multiphase architecture). A new algorithm which comprises four heuristic criteria is introduced for efficiently selecting the required seeds and LFSR cells. Also, a cost metric for assessing the quality of the algorithm's results is proposed. By using this metric, the process of determining proper values for the algorithm's input parameters is significantly simplified. The proposed method compares favorably with the most recent and effective test set embedding techniques in the literature.

1. Introduction

Due to the very tight time-to-market constraints, contemporary digital circuits (or equivalently, digital systems) embed pre-designed and pre-verified modules, called Intellectual Property (IP) cores. The structure of IP cores is often hidden from the system integrator and only a pre-computed test set *T* is provided by the core vendor. In order for the required testability to be achieved, proper test structures should be incorporated in the system. Various methods have been proposed for testing IP cores of unknown structure. Test set embedding [1]-[7] is a very successful one. According to this approach, the test patterns of *T* are encoded in a longer vector sequence, which is produced by a Test Pattern Generator (TPG) circuit. Test set embedding techniques combine reduced test-data-storage requirements, small hardware overhead and increased unmodeled fault coverage, due to the application to the core under test (CUT) of vectors that do not belong to *T*.

The major problem of test set embedding methods is that they usually require excessively long test sequences (e.g., [2], [3] and [7]). For CUTs with many hard-to-detect faults, it is highly likely that at least two test vectors of *T* will be very far from each other in the TPG's vector sequence. Consequently, many clock cycles will be needed for generating all the patterns of *T*. Reseeding is a technique that can solve this problem at the expense of some extra data storage (the required seeds). For that reason, test set embedding approaches with reseeding have been recently proposed in the literature [4], [6].

This research was co-funded by the European Union in the framework of the program "Pythagoras II" of the "Operational Program for Education and Initial Vocational Training" of the 3rd Community Support Framework of the Hellenic Ministry of Education, funded by 25% from national sources and by 75% from the European Social Fund (ESF).

In this paper, a new test set embedding technique with reseeding, suitable for the testing of sequential cores with scan, is presented. The multiphase TPG architecture proposed in [8] for mixed-mode testing, is used here for test set embedding. According to the multiphase approach, the bit sequences of multiple cells of a Linear Feedback Shift Register (LFSR) are exploited for effectively encoding the test patterns of a CUT. It has been shown in [8] that, compared to the classical reseeding schemes that use a single LFSR cell, the multiphase architecture offers better compression capabilities. Nevertheless, the cornerstone of a test set embedding technique is the embedding algorithm [1], [7]. Thus, for selecting the appropriate seeds and LFSR cells that will be used during testing, a new such algorithm is introduced. The proposed algorithm comprises four heuristic criteria that minimize both the resulting seed volumes and the required execution time. Furthermore, a cost metric for assessing the quality of the algorithm's results, for various values of the input parameters, is presented. By using this metric, proper parameter values can be easily determined. The proposed technique is shown to be more efficient than the most recent test set embedding approaches in the literature.

The rest of the paper is organized as follows: in Section 2 the main characteristics of the utilized multiphase architecture are reviewed, while in Section 3 the proposed seed- and cell-selection algorithm is presented (for convenience we will just call it seed-selection algorithm hereafter). In Section 4 the aforementioned cost metric is discussed and the proposed technique is evaluated with experimental results and comparisons. Section 5 concludes the paper.

2. The multiphase architecture

We will at first present the multiphase TPG scheme for CUTs with a single scan chain, we will explain its main features and afterwards we will discuss its application to multiple-scan-chain architectures.

2.1. The single-scan-chain case

The multiphase TPG architecture for the single-scan-chain case is shown in Figure 1. The shaded areas correspond to the hardware required by the classical LFSR-based reseeding schemes.

The main characteristic of the multiphase scheme is that, instead of generating a long vector sequence from a single LFSR cell for each seed, it feeds the scan chain of the CUT with shorter vector sequences from multiple LFSR cells. This feature is demonstrated by a simple example in Figure 2.

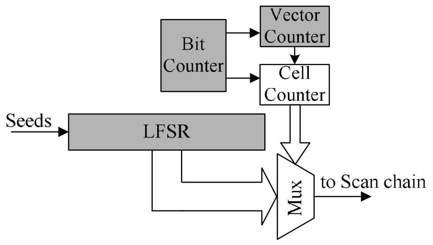

Figure 1. Multiphase architecture for CUTs with a single scan chain

Let us assume that we have to encode the test cubes (test vectors with 'x' values) of test set *T* in the pseudorandom sequence generated by an LFSR and that each LFSR seed is expanded to six test vectors (i.e., six test vectors are generated starting from each LFSR seed). In the classical LFSR reseeding schemes, only a single cell of the LFSR (usually the last one) is utilized and the test vectors are generated as shown in Figure 2.a. According to the multiphase technique, more than one LFSR cells are selected for feeding the scan chain of the CUT. In the example of Figure 2.b three cells (namely 1, *i* and *l*) have been chosen by the seed-selection algorithm and, as a result, two test vectors are generated by each one of them, starting from the **same** initial seed. That is, each seed is expanded to the predetermined number of test vectors (which is equal to 6 in our example) in different test phases (hence the name multiphase). For the same seed, the only thing that differentiates two test phases is the LFSR cell that feeds the scan chain of the CUT.

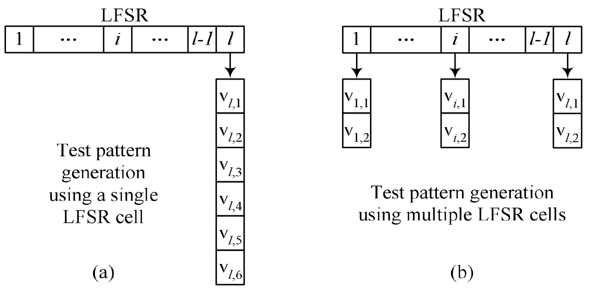
LFSR
1 ··· i ··· l-1 l
$v_{l,1}$ $v_{l,2}$ $v_{l,3}$ $v_{l,4}$ $v_{l,5}$ $v_{l,6}$
Test pattern generation using a single LFSR cell
(a)
LFSR
1 ··· i ··· l-1 l
$v_{1,1}$ $v_{1,2}$ $v_{i,1}$ $v_{i,2}$ $v_{l,1}$ $v_{l,2}$
Test pattern generation using multiple LFSR cells
(b)

Figure 2. Classical vs. proposed TPG scheme

More formally, let us consider that *p* LFSR cells have been chosen by the seed-selection algorithm and that *w* test vectors are generated by each selected cell for each seed ($p = 3$ and $w=2$ in the example of Figure 2). The LFSR is loaded with the first seed and is let produce *w* test vectors from the first of the *p* selected cells. Then the value of the Cell Counter is increased by one and thus a different LFSR cell is ready to feed the scan chain through the multiplexer. The LFSR is loaded **again with the first seed** (this is possible even when an external tester is used [9]) and *w* test vectors are generated, this time from the second selected cell. The same procedure is repeated until each of the *p* cells has been used for generating *w* test vectors starting from the first seed. The Cell Counter is then reset and the LFSR is loaded with the second seed. Eventually, all *p* cells will be used for generating *w* test vectors for all seeds. Assuming *R* seeds, the overall test-sequence length will be equal to:

$$\textit{Test-sequence length} = w \cdot p \cdot R$$

As shown in [8], the encoding ability of an LFSR seed is enhanced when multiple LFSR cells are used instead of one, since the potential of various shifted versions of the LFSR's *m*-sequence is exploited. That is, instead of having to use new seeds for test cubes that cannot be encoded together in the same cell's vector sequence, we may embed them in different cells' sequences, which are though generated by the same seed. This way, fewer seeds are required for embedding a CUT's test set in a pseudorandom LFSR sequence and, consequently, the test data storage requirements are reduced.

We should note that both normal and inverted outputs of the LFSR cells may be used for feeding the scan chain of the CUT, as it will be discussed in Section 3.

If an external tester is used for providing the seeds, the number of clock cycles required for loading them serially in the LFSR may be significant compared to the actual test application time. This problem can be solved by using a shadow register, as described in [10]. Contrary to the LFSR, the same seed should be loaded only once in the shadow register.

As for the additional hardware overhead of the multiphase architecture, it is confined to that of the extra multiplexer. The overhead of the Cell Counter is counterbalanced by the reduction of Vector Counter's length. Since Vector and Cell Counters are combined together so as to control the generation of $w \cdot p$ vectors, it can be easily proven that their aggregate length is at most one bit greater than that of a single counter controlling the generation of the same number of vectors (classical schemes). Therefore, splitting Vector Counter into Vector and Cell Counters imposes minimal additional hardware overhead (at most one flip-flop).

2.2. The multiple-scan-chain case

When the CUT contains multiple scan chains, the multiphase architecture is modified in such a way that both, fulfils the requirement of low linear dependency of the bit sequences shifted in the scan chains of the CUT, as well as takes advantage of the various shifted versions of the LFSR's *m*-sequence generated by different LFSR cells. In Figure 3 the block diagram of the multiphase TPG, for that case is given.

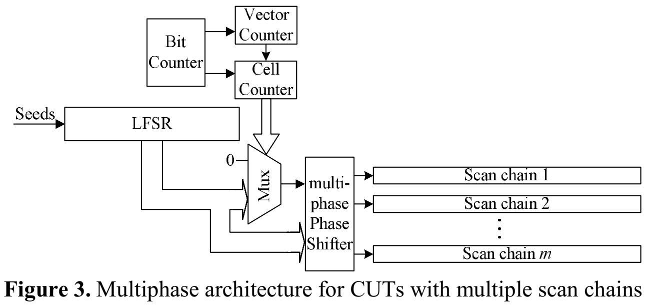

Figure 3. Multiphase architecture for CUTs with multiple scan chains

The main difference from the architecture of the previous subsection is the multiphase Phase Shifter module, which is inserted between the multiplexer and the scan chains. The purpose of this module is twofold: it minimizes the linear dependencies among the bit sequences shifted in the scan chains, as well as, by receiving the output of the multiplexer, it feeds the scan chains of the CUT with different shifted versions of the LFSR's m-sequence. A more detailed diagram of the multiphase Phase Shifter is shown in Figure 4.

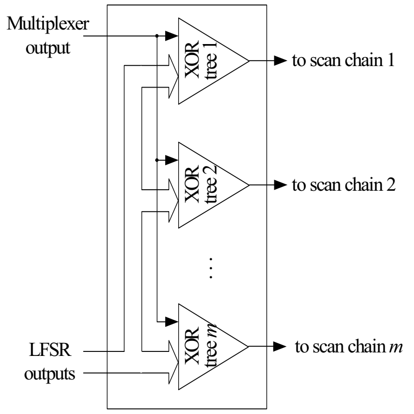

Figure 4. The multiphase Phase Shifter

The multiphase Phase Shifter is initially designed as a normal phase shifter, as proposed by the authors of [11]. Then, an extra input is added to each XOR tree, which is driven by the multiplexer of the multiphase architecture. At each test phase a different LFSR cell is selected to drive, through the multiplexer, that extra input. This way the shifting function of each of the Phase Shifter's outputs is altered and a different part of the LFSR's m-sequence is generated. This is essentially equivalent to the use of different LFSR cells in the single scan chain case. In the case of multiple scan chains, the multiplexer can also drive the XOR trees' extra input with the constant 0 value. When this happens, the Phase Shifter operates as if the additional input was not present and therefore, as it was originally designed to shift the bit sequences. This is done for exploiting the potential of the original phase shifter. We should note that when the output of an LFSR cell (let assume cell i) is fed to the XOR trees, there is some probability that the resulting inter-chain separation, for some of the phase shifter's output sequences, will be smaller than the maximum scan-chain length. This may reduce the encoding ability of the scheme when cell i drives the XOR trees, but it is highly unlikely that the same will be true for many LFSR cells. Therefore, the encoding efficiency of the scheme remains unaffected, since such cells will not be chosen by the seed-selection algorithm.

Compared to the classical LFSR-reseeding schemes, the multiple-scan-chain multiphase architecture requires only one additional multiplexer (as in the single-scan-chain case) and an extra XOR gate per scan chain. We should mention that the additional XOR trees' input does not affect their propagation delay when the number of the rest inputs is not equal to a power of 2, since the number of tree-levels does not increase. In the opposite case, the phase shifter's delay is increased by one XOR-gate delay.

3. Selection of seeds and LFSR cells

The seed-selection algorithm receives as inputs a test-cube set T and the user-defined parameters p (maximum number of cells to be selected) and w (number of vectors generated by each selected cell for each seed, i.e. window size). Its goal is to calculate R seeds and select at most p cells so that each test cube of T is compatible with at least one of the w vectors generated by each of the p LFSR cells, for all R seeds. The set of chosen seeds should be as small as possible.

The search space of the seed-selection algorithm is shown in Figure 5. It is initially comprised of w rows of $2{\cdot}l$ symbolic vectors (sv), where l is the LFSR length. Symbolic vector $sv_{i,j}$ is the jth vector that would have been shifted in the scan chain(s) of the CUT, when using the ith LFSR cell, if each bit of the initial state of the LFSR has been equal to a binary variable. Since we also consider the inverted outputs of the LFSR cells, for each $sv_{i,j}$ in the search space there is also its complementary $\overline{sv_{i,j}}$. We should note that for the multiple-scan-chain case there is an additional column of symbolic vectors on the right side of the search space, which corresponds to the constant zero value of the multiplexer.

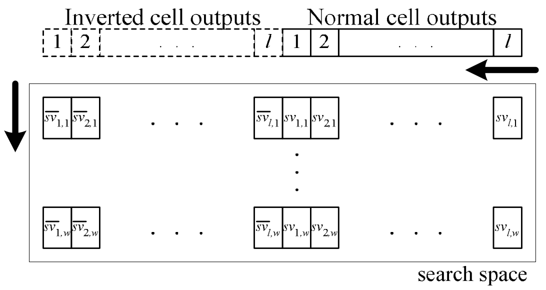

Figure 5. The search space of the seed-selection algorithm

At first, the seed-selection algorithm generates the above described search space by simulating the LFSR symbolically. Then, for each test cube t of set T, it traverses the search space from the right to the left, row by row and, by solving the linear systems $sv_{i,j} = t$ and $\overline{sv_{i,j}} = t$, tries to verify if a vector compatible to t can be generated at each search space position. If the corresponding system is solvable, then such a vector exists. In order to be generated, the initial LFSR state should be updated according to the solution of the system (i.e., assuming Gauss-Jordan elimination, all the variables belonging in the pivot columns of the system should be replaced by linear expressions of the free variables). In other words, if system $sv_{i,j} = t$ is solvable, then test cube t can be encoded at the corresponding position of the search space.

After selecting a solvable system for a test cube t of T, the binary variables in the initial LFSR state are updated as described above. Test cube t is removed from T and the algorithm's search space is generated again. After that, the symbolic vectors contain only the free variables of the selected system. Another solvable system is then chosen and the same procedure is repeated until no system is solvable for any of the cubes of T. At this point, a new seed has been determined. The seed-selection algorithm continues to generate seeds this way, until all the test cubes have been removed from T.

As the seed-selection algorithm traverses the search space for each test cube, it selects a solvable system $sv_{i,j}$=t or $\overline{sv_{i,j}} = t$ (and consequently cube t to be encoded), according to a set of heuristic criteria. Two basic seed-volume-minimization rules are followed by the application of the aforementioned criteria: I) the "hardest" cubes have to be encoded first (cube t_i is considered harder than t_j if it is more difficult for t_i to be encoded, along with other cubes, in a seed's vector sequences - we will explain later how we recognize a hard cube), and II) the system, the solution of which requires the replacement of the fewest variables, should be selected at each step of the algorithm. The reason for using the second rule is obvious: fewer replaced variables when encoding a test cube, mean more variables remaining in the linear systems in the following algorithm iterations and thus increased probability of solving some of those systems (or equivalently, increased probability of encoding another cube in the same seed's sequences). As for the first rule, its thorough justification is provided in [12]. Its essence is that by trying to handle the hardest cubes in the early steps of the algorithm, we avoid having seeds that encode very few (or just one) test cubes in its final steps. Such seeds increase the seed count considerably. As an example, let us consider a test set with four cubes, t_1, t_2, t_3 and t_4, from which t_1 and t_2 are the hardest. If we encode t_3 and t_4 in the first seed's sequences, then t_1 and t_2 cannot be encoded in the same seed since they are hard, and thus 3 seeds would be totally needed. On the contrary, if we started with t_1, then one of the non-hard test cubes (let say t_3) could be also encoded in the same seed. The same holds for t_2 and t_4, and therefore, only 2 seeds are required in this second scenario.

However, the above two rules lead to conflicting decisions. A very good metric for identifying a hard cube is the number of defined bits it includes. The more defined bits in a cube, the more variables will be replaced when solving a linear system and hence the fewer variables will remain for encoding the rest test cubes. It is obvious though that this metric contradicts with the min-replaced-variables rule. Therefore, the order in which these two rules will be applied is critical and depends on the testing conditions. We have verified through extensive simulations that when a lot of variables are available (i.e., a long LFSR is used), better compression results are achieved by primarily replacing as few of them as possible (rule II). Since for each seed there are a lot of available variables, we can always encode many cubes in one seed, even hard ones. Hence, in this case rule II has higher priority than rule I. On the contrary, when a smaller LFSR is used, as in our case (i.e., scan-based testing), the application order of the two rules should be inverted. The early encoding of the hardest cubes (rule I) becomes the primary target.

After the above analysis, it is easy to determine heuristic criteria for selecting a solvable system at each iteration of the algorithm. Four such criteria are proposed in this paper. The first two have been previously described. Particularly, among the solvable systems, those corresponding to the test cubes with the maximum number of defined bits are first selected (first criterion). From them, the systems that their solution eliminates the fewest variables in the initial LFSR state are chosen (second criterion). However, a third criterion is required for further distinguishing the selected systems. Its goal is to refine the hardest-cube-first rule, since the min-replaced-variables one was fully covered by the second criterion. Specifically, from the systems chosen by the first two criteria, those that correspond to the test cube which can be encoded in the fewest search space positions are preferred (third criterion). This is another indication of how hard a test cube is. Finally, if there are more than one systems selected by the third criterion (for the same, hardest cube of course), we choose the one that is higher in the search space. This last criterion is used for shortening the resulting test sequence.

As far as the cell selection is concerned, it is driven by the above described system-selection procedure. That is, when a linear system $sv_{i,j} = t$ or $\overline{sv_{i,j}} = t$ is chosen by the algorithm, cell i is also selected. When the number of selected LFSR cells reaches value p, then the search space is confined to p columns (those corresponding to the p cells chosen).

Various optimizations can be adopted for reducing the execution time of the seed-selection algorithm. The "select the hardest cube first" feature of the algorithm allows the categorization of the test cubes of T in cube-groups. Each cube-group contains only test cubes with the same number of defined bits. The groups are sorted in descending order starting from the one that contains the cubes with the maximum number of defined bits. In the process of selecting a new seed, the algorithm considers solely the test cubes of the first group and only when no cube of that group can be further encoded, proceeds to the next group. This reduces significantly the number of traverses of the search space and consequently, the number of linear systems solved at each step of the algorithm. Also, when a test cube is considered, the positions in the search space that a linear system can be solved for that cube are marked, so that if the same cube has to be re-examined (after the selection of another cube in the same group), only these positions are checked.

We should also mention that the selection of the hardest cube at each step of the algorithm speeds up the execution time considerably, since more variables are replaced in the initial LFSR state and thus, fewer systems are solvable and are examined in the following iterations. The search time of the algorithm is furthermore reduced when, after the first cubes have been chosen, the number of selected cells becomes equal to the value of parameter p, which causes the confinement of the search space from $2 \cdot l$ to p columns. Since also, a binary linear system can be solved much faster than a conventional one due to the fast modulo-2 operations performed during its solution [8], short running times of the described seed-selection algorithm have been achieved.

4. Evaluation and comparisons

For evaluating the effectiveness of the proposed method, we implemented the seed-selection algorithm of the previous section in C programming language and we conducted a series of experiments for the largest ISCAS'89 benchmark circuits with many hard-to-detect faults, assuming 64 scan chains. The required test sets were obtained by using the Atalanta ATPG tool [13]. The characteristic polynomials of the

LFSRs were selected to be primitive and the multiphase Phase Shifters were designed according to the work of [11], adding an extra input to each XOR tree (total cost: 3 XOR gates per tree).

The selection of the size of the algorithm's search space ($w \cdot p$) is critical, since it affects both the resulting seed volumes and test-sequence lengths. For that reason a cost metric is introduced for identifying the proper search-space size for each circuit. Specifically, a series of experiments are performed for every benchmark circuit, keeping the value of parameter p (max number of selected cells) constant, while modifying the value of parameter w (window size). This is done for simplifying the experimental procedure, since similar results are derived when both parameters p and w are modified. Let *max_seeds* and *max_sequence* be respectively the maximum number of seeds and the maximum test-sequence length among the results of all the experiments, for the same circuit. Usually, the experiment with the smallest search space provides *max_seeds*, while the one with the largest search space provides *max_sequence*. For every experiment, we calculate the ratios $\frac{seeds}{max\_seeds}$ and $\frac{test\_sequence\_length}{max\_sequence}$, where *seeds* and *test_sequence_length* are respectively the seed-volume and test-sequence-length results of each experiment. The smaller these ratios are, the better are considered the corresponding results (i.e., the smaller hardware and test-sequence costs are paid). Consequently, the sum of the above ratios provides a good indication of an experiment's total cost. Depending on the implementation, hardware overhead may be more important than test-sequence length or vice versa. Therefore, the two ratios must be properly weighted. Specifically, we multiply $\frac{seeds}{max\_seeds}$ with factor k. Values of $k > 1$ mean that a small seed volume is more important than a short test sequence (i.e., seed reduction is of higher priority), while values of $k < 1$ mean the opposite. Thus, the proposed cost metric is calculated by the following relation:

$$Cost\ metric = k \cdot \frac{seeds}{max\_seeds} + \frac{test\_sequence\_length}{max\_sequence} \qquad (1)$$

Since the seed-volume reduction is our primary target, in our experiments we set $k = 2$. A graphical representation of the cost metric with $k = 2$, for various search-space sizes for s13207 is shown in Figure 6.

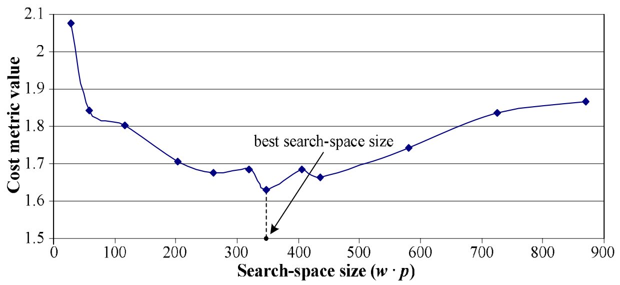

Figure 6. The proposed cost metric with k=2, for various search-space sizes for s13207

As can be seen, for very small search spaces the increased number of seeds leads to increased cost-metric values. The same holds for very large search-space sizes due to the resulting lengthy test sequences. However, the proposed metric's values are higher in the former than in the latter case, as a result of the greater weight of the $\frac{seeds}{max\_seeds}$ ratio in relation (1) [$k = 2$]. The smallest and hence best metric values occur for moderate-sized search spaces. As we move from small to greater search-space sizes, decreasing metric values mean that by allowing longer test sequences we get considerably smaller seed-volume results, while increasing metric values indicate that longer test sequences do not offer significant seed-volume reductions. After the best search-space size has been determined, various combinations of parameters w and p are examined (keeping $w \cdot p$=constant=best search-space size) and the one leading to the smallest seed count is finally selected.

In Table 1 we provide the results of the proposed technique for 64 scan chains, having determined the values of parameters w and p as explained above. In columns 6 and 7 the resulting seed volumes and test-sequence lengths are presented.

Table 1. The results of the proposed technique

Circuit	Test-set size (# vectors)	LFSR length	Window size (w)	Selected cells (p)	#Seeds (R)	Test-sequence length
s13207	2217	24	12	29	112	38976
s15850	2391	39	12	26	144	44928
s38417	6322	85	10	30	516	154800
s38584	8317	56	20	25	77	38500

We compare the proposed technique against the multiplexer-based, Reconfigurable Interconnection Network (RIN) approach of [5]. This method has been shown to be the most successful scan-based test set embedding technique in the literature, in terms of both the required test-data storage and test-sequence length. In [5], a Mux network (RIN) is used for directly connecting to the scan chains of the CUT, the outputs of various cells of an LFSR in different test configurations. The Muxes of the RIN are constructed using tristate gates. Two strategies are proposed in [5] for tackling the problem of highly clustered care bits in test cubes: scan cell reorganization and the insertion of an extra level of multiplexers between the outputs of the RIN and the inputs of the scan chains of the CUT (Interleaving Muxes). Due to the fact that scan cell reorganization is usually an unacceptable approach, in the comparisons we considered only the strategy of the extra interleaving level.

According to [5], the number of tristate gates of each Mux of the RIN is equal to the number of either the test configurations, or the cells of the LFSR used, whichever is smaller. By doing so, the authors ensure that the minimum number of tristate gates will be used. However, if the number of configurations is greater than the size of the LFSR, some extra gates will be needed in order to actually connect each configuration control line D_i to a tristate gate that is used in more than one configurations. That is, for controlling a tristate gate in multiple configurations, an extra gate, which receives as inputs the respective configurations' control lines, is required. If we assume that the overhead of each extra gate is at least 4 transistors, then the overall hardware overhead of the RIN will be equal to or greater than that of a RIN in which a single tristate gate (4 transistors) is used in each Mux for every configuration. Consequently, the easiest and fairest way to compare the two techniques is to assume that the number of tristate buffers of each Mux of the RIN is equal to the number of configurations' control lines (D_i), as shown in Figure 2.b of [5].

Taking into consideration all the above, we have estimated the hardware overhead of [5] by summing the transistors required for the implementation of the Muxes of the RIN and the interleaving level, as well as of the ROM for storing some necessary control bits. We assumed that for each tristate buffer 4 transistors are needed as mentioned in [5], while the area that each ROM bit occupies is considered to be equal to that of 1 transistor [14]. Moreover, for every configuration control line (D_i) we have calculated the overhead of the inverter required for generating the signal D_i', which, combined with the D_i, controls the inputs *enable* and *enable'* of the tristate buffers, respectively. The hardware overhead of the proposed approach was calculated as the sum of the transistors required for the implementation of the multiphase phase shifter, the Mux that drives the extra inputs of the phase shifter's XOR trees, plus the transistors that correspond to the ROM bits that should be stored (as above, we assumed that 1 ROM bit occupies 1 transistor area). The hardware overhead of the multiphase phase shifter is equal to 24·*Number of scan chains*, since 3 standard transmission-gate-based XORs were used for the realization of each XOR tree. Such a XOR gate requires 8 transistors for its implementation. We should note that, for the calculation of the hardware overhead of the approach of [5] we did not take into account the excessive wiring of the RIN and the interleaving level. As for the rest of the control logic of the two compared schemes, it is very small and imposes similar area overhead and, for that reason, it has not been considered in the comparisons.

Table 2. Hardware overhead and test-sequence length comparisons

	Hardware Overhead			**Test-sequence length**		
Circuit	**[5] (#trans.)**	**Proposed (#trans.)**	**Reduction (%)**	**[5] (#vec.)**	**Proposed (#vec.)**	**Reduction (%)**
s13207	19409	4398	77.34	75047	38976	48.06
s15850	19420	7308	62.37	179580	44928	74.98
s38417	52524	45576	13.23	616835	154800	74.90
s38584	18323	5998	67.27	291425	38500	86.79

In Table 2, the hardware overhead (columns 2-4) and the test-sequence length (columns 5-7) comparisons between the proposed technique and that of [5] are presented. It is obvious that the multiphase approach outperforms the RIN-based one in both cases. As far as the hardware overhead is concerned, although the technique of [5] has minimal test-data storage requirements, its total implementation cost is much greater than that of the multiphase scheme. This is due to the small seed-counts achieved by the presented seed-selection algorithm. As for the test-sequence length comparisons, the superiority of the proposed approach, which performs reseedings, is clear, as can be seen in columns 5-7 of Table 2.

5. Conclusions

A new test set embedding approach with reseeding for scan-based testing has been proposed in this paper. The presented approach combines the multiphase TPG architecture of [8] with a new, very effective seed- and cell-selection algorithm, which consists of four heuristic criteria. The values of the input parameters of the algorithm can be easily determined by calculating a simple cost metric that is also introduced. Experimental results and comparisons demonstrate the advantages of the proposed technique.

References

[1] M. Lempel, S. Gupta and A. Breuer, "Test Embedding with Discrete Logarithms", *IEEE Trans. on CAD*, vol. 14, May 1995, pp. 554-566.

[2] D. Kagaris, S. Tragoudas and A. Majumdar, "On the Use of Counters for Reproducing Deterministic Test Sets", *IEEE Trans. on Comp.*, vol. 45, Dec. 1996, pp. 1405-1419.

[3] D. Kagaris and S. Tragoudas, "On the Design of Optimal Counter-based Schemes for Test Set Embedding", *IEEE Trans. on CAD*, vol. 18, Feb. 1999, pp. 219-230.

[4] S. Swaminathan and K. Chakrabarty, "On Using Twisted-Ring Counters for Test Set Embedding in BIST", *J. of El. Testing, Th. and Appl.*, Kluwer Academic Publishers, vol. 17, no. 6, Dec. 2001, pp. 529-542.

[5] L. Li and K. Chakrabarty, "Test Set Embedding for Deterministic BIST Using a Reconfigurable Interconnection Network", *IEEE Trans. on CAD*, vol. 23, Sept. 2004, pp. 1289-1305.

[6] E. Kalligeros, D. Kaseridis, X. Kavousianos and D. Nikolos, "Reseeding-based Test Set Embedding with Reduced Test Sequences", in Proc. of Int. Symp. on Quality El. Des., March 2005, pp. 226-231.

[7] I. Voyiatzis, "Test Vector Embedding into Accumulator-Generated Sequences: A Linear-Time Solution", *IEEE Trans. on Comp.*, vol. 54, Apr. 2005, pp. 476-484.

[8] E. Kalligeros, X. Kavousianos and D. Nikolos, "Multiphase BIST: A New Reseeding Technique for High Test Data Compression", *IEEE Trans. on CAD*, vol. 23, Oct. 2004, pp. 1429-1446.

[9] X. Liu, M. S. Hsiao, S. Chakravarty and P. J. Thadikaran, "Efficient Techniques for Transition Testing", ACM *Trans. on Des. Autom. of El. Syst.*, vol. 10, no. 2, Apr. 2005, pp. 258-278.

[10] P. Wohl, J. A. Waicukauski, S. Patel and M. B. Amin, "Efficient Compression and Application of Deterministic Patterns in a Logic BIST Architecture", in Proc. of Des. Autom. Conf., June 2003, pp. 566-569.

[11] J. Rajski, N. Tamarapalli and J. Tyszer, "Automated Synthesis of Phase Shifters for Built-In Self-Test Applications", *IEEE Trans. on CAD*, vol. 19, Oct. 2000, pp. 1175-1188.

[12] E. Kalligeros, X. Kavousianos, D. Bakalis and D. Nikolos, "An Efficient Seeds Selection Method for LFSR-based Test-Per-Clock BIST", in Proc. of Int. Symp. on Quality El. Des., March 2002, pp. 261-266.

[13] H. K. Lee and D. S. Ha, "Atalanta: An Efficient ATPG for Combinational Circuits", TR, 93-12, Dep't of Electrical Eng., Virginia Polytechnic Institute and State University, 1993.

[14] L. R. Huang, J. -Y. Jou and S. -Y. Kuo, "Gauss-Elimination-Based Generation of Multiple Seed-Polynomial Pairs for LFSR", *IEEE Trans. on CAD*, vol. 16, Sept. 1997, pp. 1015-1024.