A ROMless LFSR Reseeding Scheme for Scan-based BIST[†]

E. Kalligeros, X. Kavousianos and D. Nikolos

Dept. of Computer Engineering & Informatics, Univ. of Patras, 26500, Patras, Greece
Computer Technology Institute, 61 Riga Feraiou Str., 26221, Patras, Greece
e-mail: kalliger@ceid.upatras.gr, kabousia@ceid.upatras.gr, nikolosd@cti.gr

Abstract

In this paper we present a new LFSR reseeding scheme for scan-based BIST suitable for circuits with random-pattern-resistant faults. The proposed scheme eliminates the need of a ROM for storing the seeds since the reseedings are performed dynamically by inverting some selected bits of the LFSR register. A time-to-market efficient algorithm is also presented for selecting the reseeding points in the test sequence, as well as a proper seed at each point. This algorithm targets complete fault coverage and minimization of the resulting test length and hardware overhead. Experimental results on ISCAS '85 and ISCAS '89 benchmark circuits demonstrate the advantages of this new LFSR reseeding approach in terms of area overhead and test application time.

1. Introduction

Built-In Self-Test (BIST) is an effective approach for testing large and complex circuits [1, 2]. Minimal test application time, area overhead, and test data storage as well as minimal performance degradation are essential in many BIST applications. Also, complete (100%) fault coverage is often desirable.

BIST schemes can be classified into two general categories [3]: test-per-scan and test-per-clock. In a test-per-scan scheme a complete or partial scan is serially filled by the *Test Pattern Generator* (TPG), while in a test-per-clock scheme a new test vector is applied to the Circuit Under Test (CUT) at each clock cycle. In this paper we consider only test-per-scan BIST schemes.

The most common and widely used BIST approach is pseudo-random BIST [1, 2]. It is usually performed by a *Linear Feedback Shift Register* (LFSR) used as TPG, which applies pseudo-random patterns to the CUT. Although pseudo-random BIST schemes have the advantage of low hardware overhead, for circuits with random pattern resistant faults, high fault coverage cannot be achieved within an acceptable test length. Test point insertion, weighted pseudo-random patterns and mixed-mode BIST [4-16] are the main techniques, which have been proposed to solve this problem. Mixed-mode BIST techniques have the advantage of achieving high fault coverage without altering the mission logic and with relatively short test sequences and small hardware overhead.

When a mixed-mode BIST technique is adopted, an LFSR is usually used for generating pseudo-random patterns that detect the random-pattern-testable faults. The remaining hard-to-detect faults are tested by deterministic patterns. Some techniques [4, 5] alter the sequence generated by the LFSR by means of some control logic in order to produce such patterns.

Reseeding techniques [6-16] on the other hand change the state of the TPG register with a new one (seed), starting from which, the TPG produces a pre-calculated deterministic pattern. The original idea of encoding test patterns as LFSR seeds by solving systems of linear equations, was proposed in [6]. This technique needs s_{max}+20 bits to encode each test cube, where s_{max} is the maximum number of care bits in any test cube testing a hard-to-detect fault. In [7] a method for improving the encoding efficiency of LFSR reseeding by using multiple-polynomial LFSRs was proposed. By using 16 polynomials instead of 1, the width of the seeds was reduced to s_{max}. However, in both [6] and [7] the encoding efficiency is limited, since there are many test cubes that contain fewer than s_{max} bits, which should be encoded using s_{max}+20 or s_{max} bits respectively. Two approaches for addressing this problem have been proposed. The first one [8] tries to encode more than one test cubes in just one seed for achieving vertical compression (reduction in the number of the required seeds) and the other uses variable-length seeds [9-11].

In [12, 13] the reseeding technique is applied to BIST schemes based not on LFSRs but on twisted-ring counters. This approach is as simple to implement as an LFSR-based one and features a very small control logic for controlling the reseeding operation. Its main disadvantage is that a twisted-ring counter cannot offer high encoding efficiency and as a result many seeds and thus many ROM bits are required for fully testing the CUT. In [14] a reseeding scheme based on folding counters (twisted-ring counters with programmable feedback) is presented. Although it

[†] This research was financially supported by the Public Benefit Foundation "Alexander S. Onassis" via its scholarships programs and by the Research Committee of Patras University, within the framework of "K. Karatheodoris" scholarships programs.

manages to significantly reduce the storage requirements, this is done at the expense of a rearrangement in the scan path of the CUT, which may be very costly. This problem is solved in [15] where the properties of folding counters are exploited in order to reduce the number of the required seeds (vertical compression), while an LFSR and a small control logic are used to decompress the seeds into folding counter states (horizontal compression). The method of [15] (2-D Compression) needs slightly more ROM bits than that of [14], with the advantage that no rearrangements of the scan chain are required. Finally the technique proposed in [16] reseeds the LFSR dynamically and partially (not all the LFSR register is changed) reducing this way the storage requirements significantly. We should mention that all the above reseeding techniques make use of a ROM in order to store the necessary seeds.

In this paper we present a new LFSR reseeding scheme for scan-based BIST that does not need a ROM for its implementation. The reseeding operation, that is the alteration of the state sequence A -> B of the LFSR to the sequence A -> B', with $B' \neq B$, is performed dynamically, by inverting the logic value of the bits of state B which are complementary to those of B'. The idea behind this is that, when the LFSR is reseeded, not all its bits need to be changed and, as a result, storing a whole seed in a ROM is not necessary. With the proper selection of the seeds and the reseeding points, only a few bits of B have to be inverted at each reseeding. By inverting only these bits, the proposed approach induces a further compression on the seeds' length. Experimental results demonstrate the advantages of the new LFSR reseeding approach.

2. The proposed architecture

The classical scan-based reseeding approach is shown in Figure 1.

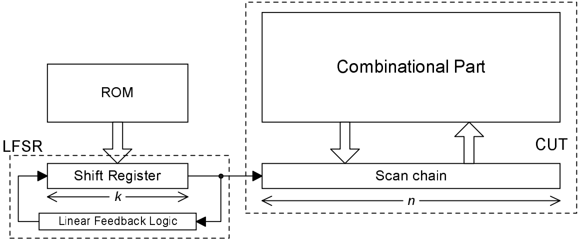

Figure 1. Classical LFSR reseeding scheme for scan-based BIST

As CUT we consider a sequential circuit consisting of a combinational part and of a scan chain of length n. The TPG circuit consists of an LFSR with k flip-flop cells ($k<n$) and a ROM for storing the seeds, which are periodically loaded into the LFSR. Unlike the classical reseeding approaches, the method proposed in this paper eliminates the need of a ROM. The seeds are produced dynamically and loaded into the LFSR in a single clock cycle, according to a combinational control module.

The architecture of the proposed TPG scheme is shown in Figure 2. The reseeding operation is performed by inverting, at certain clock cycles, the outputs of some of the LFSR cells before being stored to their adjacent cells. This is achieved by means of additional exclusive-OR (XOR) gates, which are placed between the cells of the LFSR, as shown in Figure 2 (these XOR gates are drawn using dashed lines). We observe that one of the inputs of each of the additional XORs is driven by the output of the previous LFSR cell, while the other one is driven by an output C_i of the Inversion Control Module. C_i is responsible for controlling all the inversion operations of cell i, with $1 \leq i \leq k$.

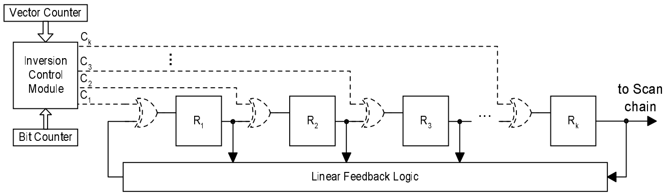

Figure 2. The proposed reseeding scheme

We assume that the LFSR changes state at clock times t_1, t_2, …, t_j,… The time interval T, with $t_{j-1} < T \leq t_j$, actually represents the clock cycle j, where, between t_{j-1} and t_j, a new state of the LFSR is generated. This new state is stored in the LFSR flip-flops exactly at clock time t_j. For C_i=0, no inversions occur and the LFSR changes state at each clock time according to its feedback structure. If a reseeding must take place at clock time t_j, some of the control lines C_i must be set to the logic value 1 in the time interval between t_{j-1} and t_j. During this interval (i.e. clock cycle j) the states of the bit and the vector counter (bc_{j-1} and vc_{j-1} respectively) are captured by the Inversion Control Module and a proper subset S_j of the lines C_1, C_2, …, C_k is activated, while the rest control lines are left to 0. In that way, at clock time t_j, the flip-flops R_1, R_2, … R_k receive the new state which is inverted at the bit positions that correspond to the lines of S_j and, therefore, a new seed is produced and stored in the LFSR register.

Note that, in some cases, m XORs ($m < k$) may be sufficient for producing all the necessary seeds for fully testing the CUT. However, since usually a small LFSR is used to produce fairly longer test vectors, in most cases, nearly all k XORs must be used.

We observe that the proposed reseeding scheme is more flexible compared to reseeding schemes that use a ROM. This is due to its inherent property that the reseedings can take place any time during the Test Pattern Generation, even in the middle of the scan-in operation, by altering any bit positions in the LFSR register. Furthermore, different bit positions can be altered at each reseeding. This flexibility gives the opportunity to the designer to fine-tune the reseeding algorithm, so as to take advantage of any particular testing characteristics of the circuit under test.

Let us now demonstrate the versatility of the proposed scheme through a very simple example. Suppose that we have to test a sequential circuit with a scan path of length 7. For controlling the loading of the test patterns in the scan chain, we need a bit counter of length 3 that cycles through its first seven states (0 to 6). Consider also that we use as TPG the 4-bit LFSR of Figure 3, which implements the characteristic polynomial $x^4 + x + 1$ and that the required vector counter is of length 4. Now assume that after the application of 8 test vectors to the CUT the LFSR contains the state 0101, the vector counter the state 1000 (8) and a hard-to-detect fault needs test cube 1x10xx1 in order to be tested (x denotes a don't care value). In order to produce one of the test vectors represented by test cube 1x10xx1, we can choose among many alternatives.

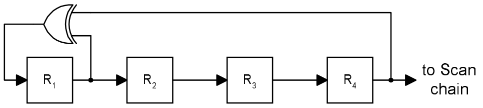

Figure 3. External-XOR LFSR (char. polynomial: $x^4 + x + 1$)

A first alternative is to invert some of the bits of state 0101, before it is stored in the register cells, so that the LFSR, starting from that new state (new seed), to generate one of the required test vectors. To calculate this seed we solve a system of linear equations based on the feedback structure of the LFSR [6]. By solving the corresponding system of equations, we conclude that the seed that will produce such a vector is of the form 0*YY*1 (*Y* is a binary variable), which means that it is either 0001 or 0111. Indeed, using as seed the state 0001, the LFSR generates vector 1110001, while using as seed the state 0111, the LFSR generates vector 1010111 (the rightmost bit is produced first by the LFSR). If for example we choose to load the seed 0111 in place of state 0101, we should invert the third bit of the LFSR, before it is stored to cell R_3. Therefore, we place a XOR gate between flip-flops R_2 and R_3 and we activate control line C_3 when states 6 and 7 of the bit and vector counter respectively are captured, as shown in Figure 4. At the next clock time (bit counter = 0, vector counter = 8) state 0111 instead of 0101 is stored in the LFSR register.

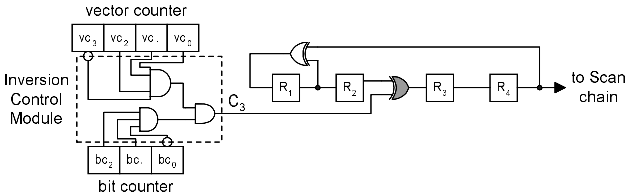

Figure 4. Example TPG (first alternative)

Another alternative is to let the LFSR evolve from state 0101 and invert some bits when necessary, in order to finally generate one of the desired test vectors. There are more than one ways to implement the TPG if we choose this alternative. One of them is the following: the test vector generated by the LFSR when it starts from state 0101 is 0110101. It differs from test cube 1x10xx1 only in the leftmost bit (the last bit produced by the LFSR). So, an easy way to produce one of the required test vectors is to invert the leftmost bit of 0110101, that is the bit that would be stored in cell R_4 at the clock time when the bit counter changes state from 5 to 6 (the vector counter contains state 8). To do this we activate control line C_4 in the clock cycle when the vector and the bit counter contain states 8 and 5 respectively, as shown in Figure 5:

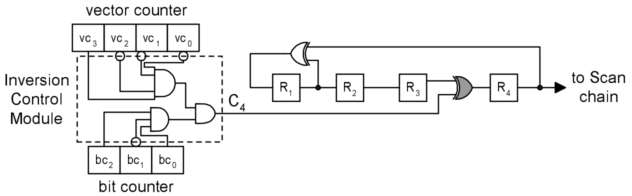

Figure 5. Example TPG (second alternative)

In both the above cases, instead of having to store a full 4-bit seed, we invert just one bit of an LFSR state to produce one of the required test vectors. Although, except for the additional XOR gate, some extra control logic is needed, we expect that some parts of this control logic would be shared among different reseedings and thus a synthesis tool would be able to reduce this control logic significantly. In any case, however, the proposed architecture leaves the scan chain of the CUT untouched.

We presented the proposed architecture using LFSRs with their feedback logic implemented with external XOR gates. It is obvious that the same architecture can also be used when the XOR gates of the feedback logic are located between the cells of the LFSR.

3. The reseeding algorithm

In this section we present a time-to-market efficient algorithm for selecting the seeds and the exact times for performing the reseedings. The efficiency of this algorithm is due to its simple, straightforward and easy implementation along with its short running time. The main goals are complete fault coverage, low hardware overhead and short test sequence length.

The algorithm implements the first of the reseeding alternatives presented in Section 2, that is to calculate for each test cube one seed, starting from which, the LFSR generates an appropriate test vector without any other reseeding operation. This way, taking advantage of the fact that the Inversion Control Module performs inversions only at clock cycles that correspond to a single state of the bit counter (specifically state *n* -1), we reduce the required control logic.

In the beginning, the initial state of the LFSR is set to a random value and 10,000 random patterns are applied to the CUT for testing the easy-to-detect faults. This pseudo-random part of the test sequence detects the vast majority of the faults of the CUT and the remaining faults are considered as hard-to-detect. We have to note that all these pseudo-random patterns are usually not necessary, since many of the random-pattern-testable faults are also detected by the patterns generated between successive reseedings of the LFSR. We will explain later how we can throw out these useless random patterns.

For the remaining hard-to-detect faults we extract test cubes using a home made, deterministic ATPG tool. For each one of the resulting test cubes c_i, we solve a system of linear equations created according to the feedback structure of the LFSR, so as to determine a seed starting from which, the LFSR will generate a test vector represented by c_i. Since these systems contain equations consisting of binary variables and constants that are related together with the modulo-2 (XOR) operation only, solving such a system is much faster than solving a conventional system of linear equations [16]. Moreover, taking into account that we solve such a system only once for each of the extracted test cubes, it is obvious that very short running time is required for calculating the seeds.

Next we select the exact times that the calculated seeds will be loaded in the LFSR. A new reseeding is performed every n cycles, that is when the previous test vector has been completely loaded in the scan chain (n is the scan chain length). In other words, after the i-th seed has been loaded, the LFSR is let produce the corresponding test vector in n clock cycles and, during the n-th clock cycle, some new inversions are stimulated by the Inversion Control Module and seed $i + 1$ is loaded. The benefit from this choice is that the test sequence is shortened, since the useless vectors between successive reseedings are eliminated. We must note that since we do not attempt to encode more than one test cubes in a single seed, letting the LFSR produce more than one test vectors after each reseeding is, in most cases, useless.

For the selection of the less hardware intensive seed to be loaded in the LFSR at a reseeding time t_i, we have adopted a simple greedy approach. We compare all the available seeds that we have calculated, with the LFSR state that would have occurred in clock time t_i if no reseeding had taken place and we select the seed that differs in the fewer bits from that state. This way the smallest possible number of bit inversions is required in cycle i in order for a seed to be loaded in the LFSR. If two or more seeds require the same minimum number of bit inversions, we choose the one that inserts the fewer additional XORs in the LFSR structure.

After having decided about the new reseeding operation we run fault simulation for the produced test vector. More than one faults are often detected by applying this test vector to the CUT. The seeds that correspond to test cubes of these faults are thrown out from the set of the available seeds for reseeding.

When all the faults are detected, we perform reverse simulation [17]. As we mentioned above, some of the 10,000 pseudo-random patterns that were initially applied to the CUT may be redundant. In order to minimize the cardinality of the test set, we fault simulate all the test vectors in reverse order until we reach 100% fault coverage. Then we set the last reverse simulated vector as the initial one (and its corresponding state as the initial state of the LFSR), and we exclude the rest test vectors from the test sequence.

Before proceeding to the experimental results let us note the following: in the reseeding algorithm we just presented we did not attempt to apply any compression technique to the extracted seeds since our purpose was to provide a very time-to-market efficient reseeding method. This method, as we will see in Section 4, exhibits major advantages in terms of the resulting test sequence lengths and hardware overhead. However any other, more sophisticated reseeding algorithm featuring seed compression can be used for reducing the hardware overhead, at the expense of extra computational time. Such algorithms have been reported in the open literature [8, 16].

4. Experimental results

To evaluate the effectiveness of the proposed BIST scheme, we implemented the algorithm described in Section 3 in C programming language and we performed a series of experiments using the ISCAS '85 and the ISCAS '89 benchmark circuits. Only circuits with undetected faults after the application of 10,000 random patterns have been considered. Before applying the algorithm of Section 3, for each one of these circuits we tried to spot an LFSR with an appropriate feedback polynomial. Such a polynomial allows the above mentioned systems of linear equations to be solved for at least one test cube for each fault. Several primitive, feedback polynomials were examined. These polynomials were generated with the tools that can be found in [18]. The degrees of the examined polynomials were varied from s_{max} - 2 to s_{max} + 5 [15], where s_{max} is the maximum number of defined bits that a test cube, detecting a hard-to-detect fault, contained. After having determined a suitable primitive polynomial, we performed 20 experiments for each benchmark circuit, starting from a different random initial seed each time. The running time of such an experiment varied from few seconds for the smaller circuits, to one hour for the larger, excluding ATPG. The larger portion of this time (over 95%) was expended to fault simulation (initial and reverse). The best results are shown in Table 1.

The names of the benchmark circuits as well as the length of the corresponding scan paths are given in the first two columns of Table 1. The subsequent columns present the length of the LFSR used as TPG, the number of the required XORs for performing the inversions, the number of the seeds used, the total number of vectors for achieving 100% fault coverage and the hardware overhead in gate equivalents. For the calculation of the hardware overhead of the proposed scheme, we have used a commercial synthesis tool for synthesizing the Inversion Control Module and, to the synthesis results, we have added the hardware overhead of the additional XOR gates. The bit as well as the vector counter have not been taken into account in the derivation of the hardware overhead results, since such counters are needed nearly by any mixed-mode, scan-based BIST scheme. We note that 1 gate equivalent corresponds to a 2-input NAND gate.

Table 1. Experimental results

Circuit	Scan Elements	LFSR length	# Invert. XORs	# Seeds	# Vectors	Hardware overhead (gate equiv.)
c2670	233	65	65	55	2514	462
c7552	207	154	153	99	9292	902
s420	34	27	24	21	7266	119
s641	54	22	21	8	2220	60
s713	54	22	22	7	2600	62
s838	66	52	52	116	6815	652
s953	45	15	11	3	8426	18
s1196	32	17	14	7	8992	46
s1238	32	17	14	7	9726	45
s5378	214	24	24	22	9104	116
s9234	247	46	46	217	9787	955
s13207	700	50	49	160	10084	456
s15850	611	47	47	233	9658	955

As we can see from Table 1, in many cases the number of inverting XORs is equal to the length of the LFSR, as we had expected. Even if fewer additional XORs are required, their number is very close to the LFSR length.

We compare the proposed reseeding scheme with the 2-D Compression approach of [15], which is the most recent scan-based reseeding technique in the open literature, it features a relatively small control module and requires less storage than any other scan-based reseeding technique. Also, it does not require any rearrangements of the scan chain of the CUT.

Table 2. Test vector comparisons

Circuit	Proposed scheme	2-D Compression [15]	Reduction percentage
c2670	2514	16552	84.81 %
c7552	9292	17488	46.87 %
s420	7266	10350	29.80 %
s641	2220	10220	78.28 %
s713	2600	10220	74.56 %
s838	6815	11742	41.96 %
s953	8426	10092	16.51 %
s1196	8992	10099	10.96 %
s1238	9726	10099	3.69 %
s5378	9104	13010	30.02 %
s9234	9787	33560	70.84 %
s13207	10084	50658	80.09 %
s15850	9658	78544	87.70 %

As we can see from Table 2, the combination of the proposed scheme with the algorithm described in Section 3, leads to significantly better results in terms of the number of test vectors needed for fully testing the CUT. This is due to two reasons. The first one is that we avoid, with the consecutive reseedings, the existence of dummy patterns in the part of the test sequence that tests the hard-to-detect faults. We must note that, by using the proposed TPG scheme, it is possible to perform consecutive reseedings without any significant increase in the hardware overhead, even if the number of required seeds is increased. The reason for this is that we don't need to store a whole new seed as in the case of ROM-based reseeding schemes, but we just have to perform a few inversions in order to load it in the LFSR register. The second reason that explains the short test sequences of the proposed approach is the reverse simulation process we perform at the end of each experiment. As we can see, in all cases except for one (s13207), we need less than 10,000 patterns for testing the CUT. This means that the initial sequence of 10,000 vectors has been reduced by the reverse simulation.

In Table 3 we present the hardware overhead comparisons between the proposed scheme and the approach of [15]. We have described the control modules of the 2-D Compression approach in Verilog HDL (excluding the same counters as for the proposed method) and we have synthesized them using the same tool as for the synthesis of the Inversion Control Modules of the proposed scheme. For translating the ROM bits to gate equivalents, we have taken into account the assumption made in [19] that, on average, 0.25 gates are required for each memory cell of a ROM. To the hardware overhead of the proposed scheme, we have added the extra gates required for implementing the bigger LFSRs it uses.

Table 3. Hardware overhead comparisons

Circuit	LFSR length		# Seeds		Hardware Overhead		
	Proposed scheme	[15]	Proposed scheme	[15]	Proposed scheme (gate equiv.)	[15] (gate equiv.)	Reduct.
c2670	65	37	55	28	493	393	-20.28 %
c7552	154	133	99	36	943	1451	35.01 %
s420	27	16	21	10	138	132	-4.35 %
s641	22	16	8	4	70	100	30.00 %
s713	22	16	7	4	72	100	28.00 %
s838	52	33	116	26	687	338	-50.80 %
s953	15	10	3	2	18	79	77.22 %
s1196	17	10	7	3	46	85	45.88 %
s1238	17	14	7	3	50	92	45.65 %
s5378	24	14	22	14	133	151	11.92 %
s9234	46	40	217	95	964	1097	12.12 %
s13207	50	18	160	58	508	393	-22.64 %
s15850	47	30	233	112	987	989	0.20 %

From Table 3, we can infer the following: firstly, the lengths of the LFSRs used in this paper are quite larger than those of [15]. Since in both cases their choice has been made according to the value s_{max}, we conclude that the test vectors produced by our home made ATPG tool are poor compared to the test vectors used in [15] because they have more defined bits. Additionally, the number of seeds required by our algorithm is much larger than the number of seeds in [15]. This is due to the fact that our algorithm does not attempt to compress the seeds in any way. However, we observe that the proposed scheme requires less hardware in most cases, even if it needs significantly more reseedings. This is not true only for the small circuits, but for the larger ones as well, like c7552, s5378, s9234 and s15850. The 2-D Compression technique is better than the proposed one in 4 out of 13 circuits (c2670, s420, s838 and s13207). These results can be explained by observing Figure 6. For each circuit we have calculated the required ROM bits for storing the seeds of [15], $R_{2\text{-}D}$, as well as for the proposed method, $R_{ROMless}$, assuming that a ROM had been used. Both $R_{2\text{-}D}$ and $R_{Romless}$ depend on the LFSR length (quality of test vectors) and the number of seeds (quality of seed compression technique). The ratio $R_{ROMless}$ / $R_{2\text{-}D}$ is an indication of the difference in the quality of these two parameters between the two methods. We observe that, according to the above ratio, our method is significantly worse than that of [15] ($R_{ROMless}/R_{2\text{-}D} > 2$ for all circuits). However, only when this ratio is too high (3.45 for c2670, 3.54 for s420, 7.03 for s838 and 7.66 for s13207), the proposed scheme requires more hardware overhead than the technique of [15]. In these cases the advantage of performing the reseedings with inversions instead of using a ROM, cannot balance the increase in the LFSR length and in the number of the seeds.

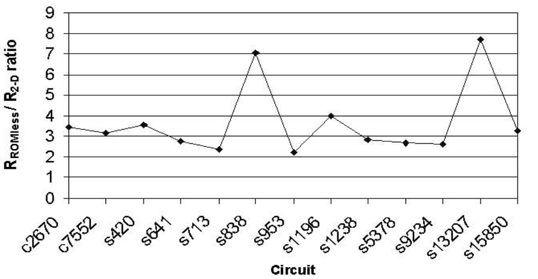

Figure 6. Graphical representation of the $R_{ROMless}$ /$R_{2\text{-}D}$ ratio

The LFSR length as well as the number of the required seeds can be reduced if a commercial ATPG tool and a more sophisticated compression algorithm are employed respectively. This way, the above ratio is expected to decrease and as a result the hardware overhead of the proposed method will improve significantly. We are currently working towards both these directions.

5. Conclusions

In this paper a new LFSR reseeding scheme for scan-based BIST is proposed. The proposed scheme does not use a ROM, but performs the reseedings dynamically by inverting some selected bits of the LFSR register. Thus, it avoids reseeding all the LFSR cells, which is inevitable when the seeds are stored in a ROM, gaining this way in hardware overhead. The proposed scheme is very flexible and can be used for implementing the reseedings in many different ways. Experimental results on ISCAS '85 and ISCAS '89 benchmark circuits show its potential, although it is combined with a simple ATPG tool and a non-sophisticated, but very fast reseeding algorithm. We expect that the experimental results will improve even more when a sophisticated algorithm along with a more effective ATPG tool will be adopted.

Acknowledgements

The authors would like to thank Professor P. Girard for providing the motivation for this work.

References

[1] P. H. Bardell, W. H. McAnney & J. Savir, *Built-In Test for VLSI: Pseudo-Random Techniques*, John Wiley & Sons, New York, NY, 1987.

[2] M. Abramovici, M. A. Breuer & A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, New York, NY, 1990.

[3] H.-J. Wunderlich, "BIST for Systems-on-a-chip", *Integration, the VLSI Journal*, vol. 26, no. 1-2, December 1998, pp. 55-78.

[4] G. Kiefer & H.-J. Wunderlich, "Using BIST Control for Pattern Generation", Proc. of ITC, Nov. 1997, pp. 347-355.

[5] N. A. Touba & E. J. McCluskey, "Bit-Fixing in Pseudorandom Sequences for Scan BIST", *IEEE Trans. on CAD of Int. Circuits & Systems*, vol. 20, no. 4, April 2001, pp. 545-555.

[6] B. Koenemann, "LFSR-Coded Test Patterns for Scan Design", Proc. of ETC, April 1991, pp. 237-242.

[7] S. Hellebrand, S. Tarnick, B. Courtois & J. Rajski, "Generation of Vector Patterns through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers", Proc. of ITC, Sept. 1992, pp. 120-129.

[8] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman & B. Courtois, "Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers", *IEEE Trans. on Computers*, vol. 44, no. 2, Feb. 1995, pp. 223-233.

[9] N. Zacharia, J. Rajski & J. Tyszer, "Decompression of Test Data Using Variable-Length Seed LFSRs", Proc. of 13th VTS, April-May 1995, pp. 426-433.

[10] N. Zacharia, J. Rajski, J. Tyszer & J. Waicukauski, "Two Dimensional Test Data Decompressor for Multiple Scan Designs", Proc. of ITC, Oct. 1996, pp. 186-194.

[11] J. Rajski, J. Tyszer & N. Zacharia, "Test Data Decompression for Multiple Scan Designs with Boundary Scan", *IEEE Trans. on Computers*, vol. 47, no. 11, Nov. 1998, pp. 1188-1200.

[12] K. Chakrabarty, B. T. Murray & V. Iyengar, "Built-in Test Pattern Generation for High Performance Circuits Using Twisted-Ring Counters", Proc. of 17th VTS, April 1999, pp. 22-27.

[13] K. Chakrabarty & S. Swaminathan, "Built-in Testing of High-Performance Circuits Using Twisted-Ring Counters", Proc. of ISCAS, May 2000, pp. 72-75.

[14] S. Hellebrand, H.-G. Liang & H.-J. Wunderlich, "A Mixed Mode BIST Scheme Based on Reseeding of Folding Counters", *Journal of Electronic Testing: Theory and Applications*, vol. 17, no. 3-4, June-August 2001, pp. 341-349.

[15] H.-G. Liang, S. Hellebrand & H.-J. Wunderlich, "Two-Dimensional Test Data Compression for Scan-Based Deterministic BIST", *Journal of Electronic Testing: Theory and Applications*, vol. 18, no. 2, March 2002, pp. 159-170.

[16] C. V. Krishna, A. Jas & N. A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding", Proc. of ITC, October-November 2001, pp. 885-893.

[17] A. P. Stroele & F. Mayer, "Methods to Reduce Test Application Time for Accumulator-based Self-Test", Proc. of 15th VTS, April-May 1997, pp. 48-53.

[18] "A Primitive Polynomial Search Program", available from http://users2.ev1.net/~sduplichan/primitivepolynomials/primitivepolynomials.htm.

[19] L. R. Huang, J. Y. Jou & S. Y. Kuo, "Gauss-Elimination-Based Generation of Multiple Seed-Polynomial Pairs for LFSR", *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 16, no. 9, September 1997, pp. 1015-1024.