A Highly Regular Multi-Phase Reseeding Technique for Scan-based BIST†

E. Kalligeros
Dept. of Computer Engineering & Informatics, University of Patras, 26500, Patras, Greece
kalliger@ceid.upatras.gr

X. Kavousianos
Dept. of Computer Engineering & Informatics, University of Patras, 26500, Patras, Greece
kabousia@ceid.upatras.gr

D. Nikolos
Computer Technology Institute 61 Riga Feraiou Street, 26221, Patras, Greece
nikolosd@cti.gr

ABSTRACT
In this paper a novel reseeding architecture for scan-based BIST, which uses an LFSR as TPG, is proposed. Multiple cells of the LFSR are utilized as sources for feeding the scan chain in different test phases. The LFSR generates the same state sequence in all phases, keeping that way the implementation cost low. Also, a dynamic reseeding scheme is adopted for further reducing the required hardware overhead. A seed-selection algorithm is moreover presented that, taking advantage of the multi-phase architecture, manages to reduce the number of the required seeds for achieving complete (100 %) fault coverage. Experimental results demonstrate the superiority of the proposed LFSR reseeding approach over the already known reseeding techniques.

Categories and Subject Descriptors
B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance.

General Terms
Algorithms, Design, Reliability, Experimentation.

Keywords
Built-In Self-Test, Scan-based schemes, Linear Feedback Shift Registers, Reseeding.

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 [15]: 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.

Pseudo-random BIST is the most common and widely used BIST approach [1-2]. Although pseudo-random BIST schemes have the advantage of low hardware overhead, for circuits with many random pattern resistant (hard-to-detect) faults, high fault coverage cannot be achieved within acceptable test lengths. To alleviate this problem deterministic patterns should be applied to the CUT.

Several sophisticated deterministic techniques have been recently proposed in the literature [7, 13, 14]. LFSR reseeding [3, 5, 8, 9, 11, 12, 16, 17] is one of the most practical and powerful methods for injecting deterministic patterns in a pseudo-random LFSR sequence. IBM has recently announced test automation tools that support an LFSR reseeding methodology [9]. Advanced test vector encoding techniques have been proposed for reducing the reseeding data volume [9, 11]. However, the effectiveness of these techniques depends on the number of hard-to-detect faults of each circuit. For circuits with many hard-to-detect faults further reduction of the hardware overhead is necessary.

In this paper we present a new LFSR reseeding architecture for scan-based BIST that fully exploits the encoding ability of an LFSR seed by using more than one cells of the LFSR for feeding the scan chain of the CUT, in different test phases. This way the number of seeds required for achieving complete stuck-at fault coverage is significantly reduced. For further reducing the hardware overhead of the proposed architecture, a very regular structure is introduced. This structure can be efficiently combined with a dynamic reseeding scheme for LFSRs, recently proposed in [5]. Along with the proposed architecture, an effective seed-selection algorithm is also presented for selecting the seeds and the LFSR cells that will finally feed the scan chain of the CUT. Experimental results demonstrate the advantages of the proposed reseeding approach.

2. THE PROPOSED ARCHITECTURE
The classical scan-based reseeding approach is shown in Figure 1.

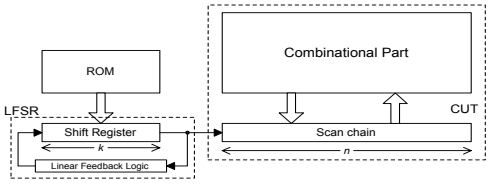

Figure 1. Classical scan-based LFSR reseeding scheme

† This research was financially supported by the Public Benefit Foundation "Alexander S. Onassis" via its scholarships programs, by the Research Committee of Patras University, within the framework of "K. Karatheodoris" scholarships program and by the State Scholarship's Foundation of Greece via its Post-doctoral research scholarships program.

GLSVLSI'03, April 28-29, 2003, Washington, DC, USA.

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.

The overview of the proposed multi-phase scan-loading architecture is shown in Figure 2. Its main feature is that more than one LFSR cells are feeding the scan chain, each one in a different test phase. The LFSR generates the same state sequence in all phases thus keeping the implementation cost low. This regularity in the TPG operation enabled us to effectively adopt in the proposed multi-phase architecture's environment, the dynamic reseeding approach of [5]. This technique eliminates the need for a ROM for storing the seeds and further reduces the required hardware overhead.

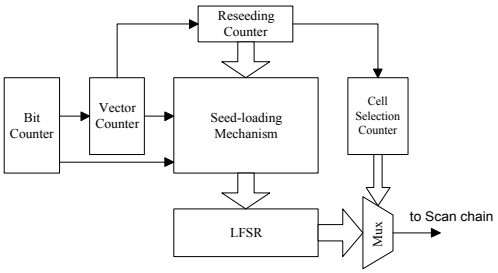

Figure 2. The Multi-Phase scan-loading architecture

In the following, we will present the multi-phase scan-loading architecture and its functionality in detail, while in subsection 2.2 a brief description of the adopted dynamic reseeding scheme will be given.

2.1 The Multi-Phase Architecture

Suppose that a subset of p LFSR cells has been chosen to feed the scan chain. Then the test session consists of p phases, and in each phase one of the p selected cells is used to feed the scan chain. All phases are identical considering the operation of the reseeding scheme, with the difference that in each phase a different cell of the LFSR is used to feed the scan-chain of the CUT and therefore a different vector sequence is produced. Specifically in each phase:

i. the same number of vectors are loaded in the scan chain,
ii. the same seeds in the same order and at the same clock times are loaded in the LFSR and, as a result,
iii. the LFSR passes through the same sequence of states.

Additionally, between every two successive reseedings, the same constant number of vectors, *VectorsPerSeed*, is loaded into the scan chain. The above properties make the structure of the proposed architecture very regular and easy to implement.

Let us now describe the operation of the proposed architecture more thoroughly. In the beginning of every phase all counters, except for the Cell Selection Counter, are initialized to zero. The Cell Selection Counter is reset only once, when testing starts. Its value is increased by one at each new phase resulting in a new LFSR cell to feed the scan chain through the Mux. The Bit Counter controls the scan-in operation of each produced vector, and signals the Vector Counter to increase. The Vector Counter checks when exactly a number of vectors equal to *VectorsPerSeed* have been loaded in the scan chain. Then it signals the Reseeding Counter to increase its value by one and as a result the next reseeding is performed by the Seed-loading Mechanism. That is, the Seed-loading Mechanism synchronizes the reseedings according to the values of the Bit and the Vector Counter and loads the appropriate seed according to the value of the Reseeding Counter. The Seed-loading Mechanism can be a ROM as in the classical reseeding approach or a combinational logic (Inversion Control Module) as will be described in the following subsection. When all the reseedings of a phase have been performed, the Reseeding Counter signals the Cell Selection Counter to increase and the next phase is initiated. Assuming that in each test phase R reseedings are performed, then the total number of clock cycles of the test session is: $TotalClockCycles = p * R * VectorsPerSeed * n$.

An important feature of the multi-phase architecture is that its hardware overhead, which is mainly the hardware overhead of the Seed-loading Mechanism does not depend on the number of test phases, since its operation as well as that of the LFSR is the same in all phases.

We should also note that the proposed architecture does not require any modifications of the scan chain of the CUT, being that way fully compatible with standard scan design.

2.2 The Dynamic Reseeding Scheme

The dynamic reseeding scheme that the proposed architecture incorporates is shown in Figure 3. The reseedings are 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, as shown in Figure 3 (these XOR gates are drawn using dashed lines). A detailed description of the dynamic reseeding scheme can be found in [5].

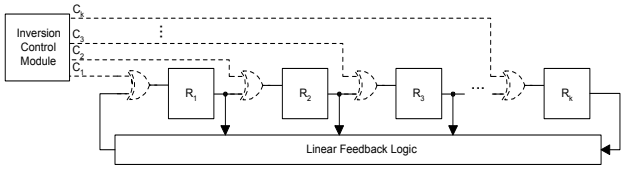

Figure 3. The adopted reseeding scheme

The main advantage of the dynamic reseeding scheme is that it controls just a few LFSR cells at each reseeding, reducing that way the overall hardware overhead required for the reseeding control logic.

3. THE RESEEDING ALGORITHM

In this section we present an efficient algorithm for selecting the seeds and the LFSR cells, which will feed the scan chain throughout the test pattern generation procedure. The main goals of this algorithm are complete fault coverage and minimization of the necessary seeds. The algorithm consists of two parts: (1) the selection of a subset of the LFSR cells for testing the easy-to-detect faults and (2) the selection of the seeds and some additional LFSR cells for detecting the hard faults. The second part also contains a test sequence reduction procedure.

3.1 Selection of an Initial Set of LFSR Cells for Testing the Easy-To-Detect Faults

Let *VectorsForEasyFaults* and *NumberOfInitialCells* be user-defined parameters which denote the maximum number of vectors for detecting the easy faults and the maximum number of LFSR cells that will be selected to feed the scan chain for detecting the easy faults respectively. Each of the selected LFSR cells will feed

the scan chain with the same number of vectors, therefore each selected cell will produce
T = *VectorsForEasyFaults* / *NumberOfInitialCells* vectors. We fault simulate T vectors produced by each one of the cells of the LFSR and we select the *NumberOfInitialCells* cells that maximize the coverage of the faults. The initial seed is selected randomly.

According to the proposed architecture the number of successive vectors shifted in the scan chain of the CUT from an LFSR cell between two successive reseedings is constant and equal to the user-defined parameter *VectorsPerSeed*. If $T >$ *VectorsPerSeed* we divide the sequence of the T vectors in $\lceil T/VectorsPerSeed \rceil$ successive subsequences, during which the LFSR is let evolve based only on its feedback structure, i.e. no inversions occur. The faults that have not been detected by this procedure are identified as hard-to-detect and test cubes are extracted for them using the ATALANTA Test Pattern Generator tool [10].

3.2 Selection of Seeds and Additional LFSR Cells for Testing the Hard-To-Detect Faults

The procedure that will be described in this subsection determines a seed and some additional LFSR cells for feeding the scan chain of the CUT in order to detect as many hard faults as possible, starting from that seed. The LFSR cells that will finally feed the scan chain of the CUT, are mainly defined in this part of the algorithm by trying to encode as many test cubes as possible to just one seed. This is primarily achieved by exploiting the bit-sequences produced by more than one cells of the LFSR.

The selection of a new seed and of the appropriate LFSR cells is done by solving systems of linear equations based on the feedback structure of the LFSR [8]. Initially, the logic value stored in cell q of the LFSR is represented by the binary variable a_q. Therefore, the initial state $\{E_1(1), E_2(1), \ldots, E_k(1)\}$ of the LFSR consists of k variables, $\{a_1, a_2, \ldots, a_k\}$, where k is the LFSR length and $E_1(1)=a_1$, $E_2(1)=a_2$, …, $E_k(1)=a_k$. Then, we let the LFSR evolve for n*VectorsPerSeed states (i.e. as if it was generating *VectorsPerSeed* vectors), where the i-th LFSR state is equal to $\{E_1(i), E_2(i), \ldots, E_k(i)\}$ and each one of the $E_1(i), E_2(i), \ldots, E_k(i)$ is a binary expression containing one or more variables from the set $\{a_1, a_2, \ldots, a_k\}$ (the variables in each binary expression are related together with the modulo-2, i.e. XOR, operation only). We define as $EV_i(j)$ (Expression Vector) the set of binary expressions produced by the i-th cell of the LFSR during the generation of vector j. $EV_i(j)$ is the j-th vector produced by the i-th cell of the LFSR, if its initial state is equal to $\{a_1, a_2, \ldots, a_k\}$. Let $t = \{t_1\ t_2 \ldots t_n\}$, $t_r \in \{0, 1, x\}$ with $1 \le r \le n$, be a test cube detecting fault f (x denotes a don't care value). If the system of linear equations $EV_i(j) = t$, which is $\{E_i((j-1)*n + r) = t_r$, if $t_r \neq x\}$, $1 \le r \le n$, can be solved, then a test vector detecting fault f can be produced by cell i of the LFSR during the j-th n-tuplet of clock cycles after its reseeding. If this system has a solution, then some of the variables $\{a_1, a_2, \ldots, a_k\}$ can be replaced by expressions containing other binary variables and/or constants (0 or 1). If we replace these variables in the initial state of the LFSR $\{E_1(1), E_2(1), \ldots, E_k(1)\}$ we get a seed that will produce a test vector for detecting fault f from LFSR cell i, after j n-tuplets of clock cycles.

The first step of the seed-selection procedure is to construct all sets of binary expressions $EV_i(j)$ with $i \in [1, k]$ and $j \in [1, VectorsPerSeed]$. Then, a weight is assigned to each hard fault, equal to the average number of defined bits of its test cubes. The larger the weight is, the more "difficult" is for the algorithm to find a seed for detecting this fault. The seed-selection algorithm tries to encode the test cubes of the hard-to-detect faults in LFSR seeds according to the following two rules [6]: a) at each step as few variables as possible are replaced and b) the more "difficult" faults, according to their weight, have to be covered first.

Initially, from the set of test cubes that detect the fault with the greater weight, the algorithm selects the one with the fewest defined bits and attempts to solve one of the systems $EV_i(j)=t$, for all i, j. The first system that can be solved is selected. Such a solution always exist according to [11], given that the LFSR length k is slightly greater than the maximum number of defined bits of the test cubes. The selected solution leads to the replacement of some variables as explained above. These variables are replaced in all $EV_i(j)$ sets and, in that way, we get new reduced sets $EV'_i(j)$. Then for each test cube t of the remaining hard-to-detect faults, the algorithm attempts to solve the systems $EV'_i(j)=t$, for all i, j. All the systems that can be solved are inserted in the set *ValidSolutions* and, from those, a system that corresponds to the fault with the greatest weight is selected first. The system is solved and some more variables are replaced. This time the replacement of these variables is done only in the systems of the set *ValidSolutions*. Many of the systems of this set will no longer be solvable, due to the replacement of those variables. Such systems are dropped from the set and the selection procedure is repeated, until the set *ValidSolutions* becomes empty. During the selection of a system $EV_i(j)=t$, $EV'_i(j)=t$ etc., along with the replacement of the appropriate variables, the algorithm also selects cell i for feeding the scan chain.

The successive replacements of the variables of the initial state of the LFSR, $a_1, a_2, \ldots, a_k$, with binary expressions, leads gradually to their replacement with constant values (0 or 1). The resulting state is the required seed. Any variables not replaced by constant values are set to a random value. Starting from that seed, we fault simulate all the *VectorsPerSeed* vectors from each selected cell and we drop any additionally detected faults. The whole seed-selection procedure is then repeated targeting a new seed, until complete fault coverage is achieved.

Finally, after having determined all the necessary seeds for achieving complete fault coverage, a test sequence reduction procedure is performed. This procedure attempts to reduce the number of the derived seeds, the *VectorsPerSeed* and the number of selected LFSR cells by fault simulating the vectors that correspond to each seed in various permutations.

4. EXPERIMENTAL RESULTS

The results of the proposed method for the ISCAS '85 and the ISCAS '89 benchmark circuits that contain a large number of hard-to-detect faults are shown in Table 1. The size of the LFSRs used was determined by the maximum number of defined bits (s_{max}) that a test cube, detecting a hard-to-detect fault, contained. According to [11], an LFSR of size $s \in [s_{max}-5, s_{max}+2]$ suffices for generating test cubes with s_{max} defined bits. For boosting the encoding procedure, we used LFSRs, the size of which ranged from $s_{max}+5$ to $s_{max}+30$. The corresponding primitive polynomials were generated with the tools that can be found in [18]. We note that in our experiments we used internal-XOR LFSRs, while the value of parameter *VectorsForEasyFaults* was set to 5000, *NumberOfInitialCells* was set to 5 and *VectorsPerSeed* varied from 10 to 20.

Table 1. Experimental results for the ISCAS circuits

Circuit	Scan Elements	LFSR length	# Additional XORs	# Source cells	# Seeds	# Vectors
c2670	233	66	66	17	31	10880
c7552	207	130	130	16	29	11680
s838	66	45	45	27	13	8775
s9234	247	55	55	19	84	26600
s13207	700	35	35	19	42	33212
s15850	611	50	50	27	69	58860
s38417	1664	110	110	32	138	116640
s38584	1464	70	70	21	35	22680

We compare the proposed reseeding architecture with the 2-D Compression approach of [11], which is the scan-based reseeding technique with the best hardware overhead results for the benchmark circuits which contain many random pattern resistant faults, in the open literature. It features a relatively small control module and requires test sequences of acceptable length. Also, it does not require any rearrangements of the scan chain of the CUT.

Table 2. Comparisons

Circuit	# Seeds		# Vectors			Hardware Overhead (gate equivalents)		
	[11]	Pro-posed	[11]	Pro-posed	Reduct. (%)	[11]	Pro-posed	Reduct. (%)
c2670	28	31	16552	10880	34.27	393	401	-1.95
c7552	36	29	17488	11680	33.21	1451	598	59.04
s838	26	13	11742	8775	25.27	338	202	40.82
s9234	95	84	33560	26600	20.74	1097	675	38.76
s13207	58	42	50658	33212	34.44	393	298	24.66
s15850	112	69	78544	58860	25.06	989	568	42.83
s38417	267	138	454555	116640	74.34	2976	2180	26.74
s38584	59	35	96435	22680	76.48	893	473	47.09

As can be seen from Table 2, the proposed technique leads to fairly better results in terms of test vectors compared to the approach of [11]. This is mainly due to the fact that the proposed seed-selection algorithm achieves excellent test cube encoding, i.e. it manages to reduce significantly the number of the required seeds, while using just a few *VectorsPerSeed*.

For the calculation of the hardware overhead of the proposed architecture, we have used a commercial synthesis tool for synthesizing the Inversion Control Module and the required Multiplexer for selecting among the LFSR cells and, to the synthesis results, we have added the hardware overhead of the additional XOR gates. We should note that 1 gate equivalent corresponds to a 2-input NAND gate. For the 2-D Compression approach, we have described the required control modules presented in [11] in Verilog HDL and we have synthesized them using the same tool as for the synthesis of the Inversion Control Modules of the proposed architecture. For translating the ROM bits to gate equivalents, we have taken into account the estimation of [4] that, on average, 0.25 gates are required for each memory cell of a ROM. For both approaches we have not considered the Bit Counter, whereas to the hardware overhead of the proposed one we have added any extra gate equivalents that may result from any difference in the registers' length (counters and LFSRs).

From Table 2 we observe that in all circuits, the superiority of the proposed method over [11] is obvious. The reseeding algorithm manages to reduce the required number of seeds and, as a result, the hardware overhead of the proposed architecture is significantly lowered. We finally note that the proposed technique, for the smaller benchmark circuits with hard-to-detect faults (s420, s641, s713, s953, s1196, s1238, s5378), requires on average 21.4 % less hardware overhead than the approach of [11].

5. CONCLUSIONS

We have described a highly regular LFSR-based reseeding architecture for scan-based BIST. The scan-chain of the CUT is fed by more than one cells of the LFSR, in different test phases while the reseedings are performed dynamically without using a ROM. These features combined with a very effective seed-selection algorithm, lead to significantly better results in terms of hardware overhead and test sequence length, compared to already published reseeding techniques.

6. REFERENCES

[1] M. Abramovici, M. A. Breuer & A. D. Friedman, Digital Systems Testing and Testable Design, Computer Sc. Press, NY, 1990.

[2] P. H. Bardell, W. H. McAnney & J. Savir, Built-In Test for VLSI: PseudoRandom Techniques, John Wiley & Sons, 1987.

[3] S. Hellebrand et al., "Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers", IEEE Trans. Comp., Feb. 1995, pp. 223-233.

[4] L. R. Huang et al., "Gauss-Elimination-Based Generation of Multiple Seed-Polynomial Pairs for LFSR", IEEE Trans. on CAD, vol. 16, no. 9, Sept. 1997, pp. 1015-1024.

[5] E. Kalligeros et al., "A ROMless LFSR Reseeding Scheme for Scan-based BIST", Proc. of ATS, Nov. 2002, pp. 206-211.

[6] E. Kalligeros et al., "An Efficient Seeds Selection Method for LFSR-based Test-per-clock BIST", Proc. 3rd IEEE ISQED, San Jose, CA, USA, March 2002, pp. 261-266.

[7] G. Kiefer et al., "Application of Deterministic Logic BIST on Industrial Circuits", Proc. of ITC, Oct. 2000, pp. 105-114.

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

[9] C. V. Krishna et al., "Test Vector Encoding Using Partial LFSR Reseeding", Proc. of ITC, Oct.-Nov. 2001, pp. 885-893.

[10] H. K. Lee & D. S. Ha, "ATALANTA: An efficient ATPG for compbinational circuits", Dept. of Elect. Eng., Virginia Polytechnic Inst. and State Univ., Blacksburg, VA, USA, Tech. Rep. 93-12, 1993.

[11] H.-G. Liang et al., "Two-Dimensional Test Data Compression for Scan-Based Deterministic BIST", Proc. of ITC, Oct.-Nov. 2001, pp. 894-902.

[12] J. Rajski et al., "Test Data Decompression for Multiple Scan Designs with Boundary Scan", IEEE Trans. on Computers, vol. 47, no. 11, Nov. 1998, pp. 1188-1200.

[13] J. Rajski et al., "Embedded Deterministic Test for Low Cost Manufacuring Test", Proc. of ITC, Oct. 2002, pp. 301-310.

[14] N. A. Touba & E. J. McCluskey, " Bit-Fixing in Pseudo-random Sequences for Scan BIST", IEEE Trans. on CAD, vol. 20, no. 4, April 2001, pp. 545-555.

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

[16] N. Zacharia et al., "Decompression of Test Data Using Variable-Length Seed LFSRs", Proc. 13th VTS, Apr.-May 1995, pp. 426-433.

[17] N. Zacharia et al., "Two Dimensional Test Data Decompressor for Multiple Scan Designs", Proc. of ITC, Oct. 1996, pp. 186-194.

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