Short Papers

Single and Variable-State-Skip LFSRs: Bridging the Gap Between Test Data Compression and Test Set Embedding for IP Cores

Vasileios Tenentes, Xrysovalantis Kavousianos, and Emmanouil Kalligeros

***Abstract*—Even though test set embedding (TSE) methods offer very high compression efficiency, their excessively long test application times prohibit their use for testing systems-on-chip (SoC). To alleviate this problem we present two new types of linear feedback shift registers (LFSRs), the Single-State-Skip and the Variable-State-Skip LFSRs. Both are normal LFSRs with the addition of the State-Skip circuit, which is used instead of the characteristic-polynomial feedback structure for performing successive jumps of constant and variable length in their state sequence. By using Single-State-Skip LFSRs for testing single or multiple identical cores and Variable-State-Skip LFSRs for testing multiple non-identical cores we get the well-known high compression efficiency of TSE with substantially reduced test sequences, thus bridging the gap between test data compression and TSE methods.**

***Index Terms*—IP core testing, linear feedback shift registers (LFSRs), test data compression (TDC), test set embedding (TSE).**

I. Introduction

Many efficient test data compression (TDC) techniques have been proposed for testing digital systems. Most of them exploit the capabilities offered by the automatic test pattern generation (ATPG) and fault simulation tools during the encoding process, in order to offer high compression efficiency (see [2], [11], [12], [19]). However, in many cases, digital systems embed IP cores which hide their structure from the system integrator. For such cores the utilization of ATPG and fault simulation tools is not an option. Various compression methods have been proposed for IP cores, which encode pre-computed test sets using linear decompressors [1], [13], [14], [16], [21], [25], [26], or various compression codes [3], [5], [8]. There are also methods that do not belong in the above categories (see [18], [20]).

Test set embedding (TSE) techniques offer another very effective means for compressing the test sets of IP cores. TSE approaches require considerably less test data storage than TDC methods, as they use long pseudorandom sequences generated on-chip, in order to embed the pre-computed test vectors of IP cores [6], [7], [9], [17], [22]. Even though TSE techniques are very attractive in terms of compression ratio, their excessively long test application times make them impractical.

In this paper, two novel linear feedback shift register (LFSR) architectures are presented, which drastically shorten the test sequences of LFSR-reseeding-based TSE techniques: the Single-State-Skip LFSRs (SSS_LFSRs) and the Variable-State-Skip LFSR (VSS_LFSR). A SSS_LFSR is an ordinary LFSR with the addition of a small linear circuit, the State-Skip circuit, which offers the capability of performing successive jumps of constant length in the LFSR state sequence. SSS_LFSRs use this capability to drastically shorten the test sequence of a test set embedding method by traversing very fast its useless parts. A VSS_LFSR is more flexible and achieves greater TSL reduction compared to a SSS_LFSR, since it embeds multiple State-Skip circuits, and thus it can perform jumps of variable lengths in the state sequence. VSS_LFSR fully exploits the State-Skip property for testing multiple IP cores in a system-on-chip (SoC), at the expense of a moderate increase in the hardware overhead, which is though comparable to that of most state of the art compression schemes. Both State-Skip LFSRs offer very short test sequences, very close to the test sequences of test data compression methods, with significantly smaller test data volumes (TDVs). Therefore, State-Skip LFSRs bridge the gap between test data compression and TSE techniques, rendering the latter an attractive testing approach for IP cores.

The concept of "state skipping" has been reported in [24] in a built-in-self-test environment, but it exhibits fundamental differences compared to the proposed scheme. Specifically, [24] presents a method to design "state skipping" logic which causes the circular chains to break out of the limit cycles and correlations, while the proposed State-Skip circuits are systematically designed to perform successive jumps of constant length in the LFSR state sequence, in order to reduce the length of a long deterministic TSE sequence.

II. Motivation

In classical LFSR reseeding [10], every seed encodes a single test cube by solving a system of linear equations. This system is constructed by considering the initial state of the LFSR as a set of variables and by symbolically simulating the LFSR. The solution of each system constitutes a seed of the LFSR. The system with the maximum number of linear equations corresponds to the test cube with the maximum number of specified bits, $s_{\max}$, which in turn determines the minimum required LFSR size. By using one seed for encoding each test cube the achieved compression is moderate, since usually in a test set there are many test cubes with fewer specified bits than $s_{\max}$. As a result, a lot of variables remain unspecified when the corresponding systems are solved, and therefore much of the potential of LFSR encoding is wasted.

Various methods have been proposed for solving this problem (see [9], [12]). A very attractive one is the window-based LFSR encoding which utilizes the same seed for encoding more than one test cube in a sequence of L ($L > 1$) pseudorandom vectors [9]. In other words, each

Manuscript received August 31, 2009; revised December 11, 2009 and March 4, 2010. Date of current version September 22, 2010. This paper was recommended by Associate Editor R. D. (Shawn) Blanton.

V. Tenentes and X. Kavousianos are with the Department of Computer Science, University of Ioannina, Ioannina 45110, Greece (e-mail: tenentes@cs.uoi.gr; kabousia@cs.uoi.gr).

E. Kalligeros is with the Department of Information and Communication Systems Engineering, University of Aegean, Samos 83200, Greece (e-mail: kalliger@aegean.gr).

Digital Object Identifier 10.1109/TCAD.2010.2051096

TABLE I
TDV, TSL: CLASSICAL($L = 1$) AND WINDOW-BASED ($L > 1$) RESEEDING

Circuit	Classical Reseeding		Window-Based Reseeding ($L > 1$)					
			$L = 50$		$L = 200$		$L = 500$	
	TDV	*TSL*	*TDV*	*TSL*	*TDV*	*TSL*	*TDV*	*TSL*
s9234	10 692	243	8008	9100	7128	32 400	6688	76 000
s13207	8856	369	5328	11 100	3816	31 800	2688	56 000
s15850	11 622	298	7410	9500	6669	34 200	6201	79 500
s38417	58 225	685	50 660	29 800	48 110	113 200	47 005	276 500
s38584	22 680	405	10 584	9450	7056	25 200	5152	46 000

seed is expanded into a window of L vectors, instead of one. Table I presents the compression improvement achieved by increasing the window size L for the largest ISCAS'89 benchmark circuits, assuming 32 scan chains. For all the experiments uncompacted test sets generated by Atalanta [15], for 100% coverage of stuck-at faults were used (note that for the ISCAS circuits, the fill rates of uncompacted test sets are in the range of 1%–5%, which resemble the low fill rates of compacted test sets of large designs). As window size L increases, TDV improves a lot, but the test sequence length (TSL) grows rapidly and becomes prohibitively long.

III. ENCODING METHOD

In the sequel, the term "test cube" refers to a test pattern with "0," "1," and "x" (unspecified) logic values, and the term "test vector" refers to a test pattern without "x" logic values.

An efficient method to reduce the TSL of LFSR reseeding with $L > 1$ is the use of State-Skip circuits [23]. A State-Skip circuit is integrated within the LFSR structure and offers the potential to perform jumps of constant length in the LFSR state sequence. Specifically, if we consider an LFSR with n cells, $c_0, c_1, \ldots, c_{n-1}$, and assuming that the contents of cell c_i at clock cycle t_j is $c_i(t_j)$, we can derive n linear expressions $F_0^k, \ldots, F_{n-1}^k$ that satisfy the following relations for every value of i, k:

$$\begin{aligned} c_0(t_{i+k}) &= F_0^k(c_0(t_i), c_1(t_i), \ldots, c_{n-1}(t_i)) \\ &\vdots \qquad\qquad \forall i \\ c_{n-1}(t_{i+k}) &= F_{n-1}^k(c_0(t_i), c_1(t_i), \ldots, c_{n-1}(t_i)). \end{aligned}$$

When $k = 1$, the above expressions represent the operation of the LFSR according to its characteristic polynomial. The linear expressions $F_0^k, \ldots, F_{n-1}^k$ are easily calculated by setting $i = 0$ and simulating the LFSR symbolically for k successive clock cycles. After the kth clock cycle, the contents of the LFSR cells $c_0(t_k)$, ..., $c_{n-1}(t_k)$ are linear expressions of the initial contents $c_0(t_0)$, ..., $c_{n-1}(t_0)$ of the LFSR cells, and they constitute the expressions $F_0^k, \ldots, F_{n-1}^k$ (more details can be found in [23]).

The SSS_LFSR can be constructed by integrating $F_0^k, \ldots, F_{n-1}^k$ in the LFSR structure. The modified LFSR operates in two different modes, Normal and State-Skip. In Normal mode, the sequence of the LFSR states is generated according to the characteristic polynomial, whereas in State-Skip mode, the state sequence is generated by the integrated linear circuit implementing the $F_0^k, \ldots, F_{n-1}^k$ functions. When the LFSR operates in State-Skip mode, it performs a jump of k states ahead at every cycle, skipping in this way the $k - 1$ intermediate states which would have been generated if the LFSR had operated in the Normal mode. Therefore, in State-Skip mode, the generated vector sequence is shortened by a factor k, which is called hereafter *speedup factor*.

SSS_LFSRs can be combined with efficient encoding methods, like [9], in order to provide a unified solution with small test data volume and short TSL. To this end, we propose a unified seed computation and test sequence reduction process consisting of four steps. During step 1, the test cubes are encoded using the window-based LFSR reseeding proposed in [9], which is very efficient in terms of compression ratio (other encoding methods can be also used). According to this method every seed is computed so as to encode as many test cubes as possible in the sequence of L successive test vectors (L is the vector-window size). Even though [9] achieves high compression results, usually, most of the test vectors of a seed do not encode any test cubes. These pseudorandom test vectors can be omitted by using the State-Skip mode. To this end, during step 2 we partition the test sequence generated by each seed into useful and useless parts as follows: we first partition the window of test vectors into L/S segments of size S, where S is a designer-defined parameter in the range $[1, L]$. During step 3 we determine if each segment is useful or useless. A segment is useful if it embeds at least one test cube not embedded in any other useful segment; otherwise it is a useless one. This reduces the number of useful segments when sparsely specified test cubes fortuitously appear in multiple segments.

The test sequence of every seed can be further shortened if the generation of the test vectors of the seed is terminated immediately after the generation of the last useful segment. To this end, during step 4 the groups are sorted in ascending order of their useful segment (i.e., group 1 contains all the seeds with one useful segment, group 2 contains all the seeds with two useful segments, etc.). The seeds are applied in this order and thus the decompressor uses only a counter to indicate the current group (this counter is incremented every time the first seed of each group is loaded in the LFSR). At the same time, every seed belonging to group i comprises exactly i useful segments. The decompressor keeps track of the number of useful segments applied to the core under test (CUT) for every seed, and when the last (i.e., the ith useful segment) is applied, it immediately terminates the generation of the test vectors of the current seed and initiates the loading of the next seed from the automatic test equipment (ATE).

IV. SSS_LFSRS: ARCHITECTURE AND LIMITATIONS

The SSS_LFSR architecture [Fig. 1(a)] consists of three units: the Vector Generation unit, the Controller unit, and the Segment Type unit.

Vector Generation unit consists of the LFSR, the phase shifter and the State-Skip circuit. Controller unit controls the operation of decompression and specifically the generation of all segments. The Group Counter is initialized to the value "1" and is incremented by one at the beginning of every new group of seeds. For each group, the seed counter is initialized to "0" and increases whenever a new seed of the group is loaded

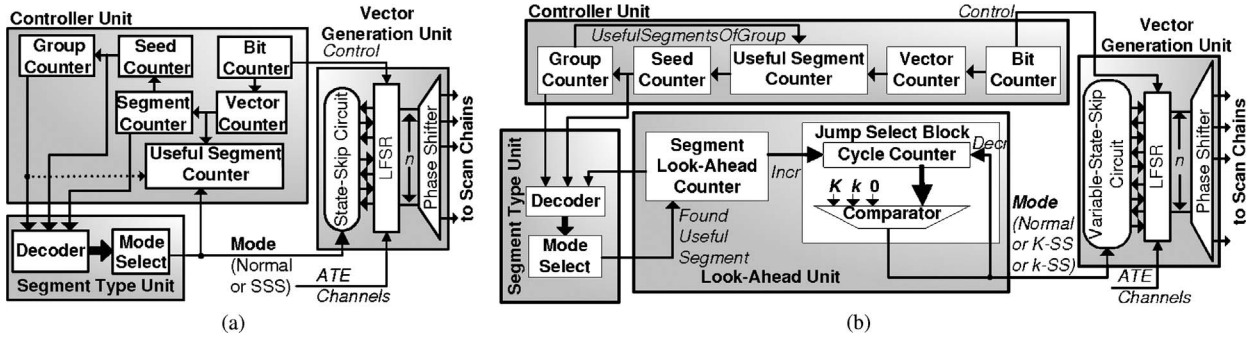

Fig. 1. (a) SSS_LFSR decompression architecture. (b) VSS_LFSR decompression architecture.

in the LFSR. Every time a new seed is loaded in the LFSR, Useful Segment Counter is loaded with Group Counter's value and thus it is set equal to the number of useful segments of the seed (note that every seed of group i consists of exactly i useful segments). At the same time, the Segment Counter is initialized to "0." For every new generated segment, the Segment Counter is incremented by one and the Segment Type unit determines if this segment is useful or not. For every generated useful segment, Useful Segment Counter decreases by one. When the Useful Segment Counter reaches "0," Seed Counter is incremented and the next seed is loaded in the LFSR from the ATE.

The Segment Type unit consists of the Mode Select block and the Decoder block. The Mode Select block receives the decoded outputs of the Segment, Seed and Group Counter and determines if the next segment is a useful or a useless one. It generates the Mode signal that puts the Vector Generation unit in Normal Mode (the segment is useful and is generated using the characteristic polynomial of the LFSR), or in SSS mode (the segment is useless and it is skipped using the State-Skip circuit). The overhead of this combinational circuit depends mainly on the total number of useful segments, which are only a very small portion of the total segments.

When multiple non-identical IP cores exist in a SoC, the TSL reduction potential of State-Skip circuits cannot be fully exploited by the SSS_LFSR architecture. This is attributed to the design of an SSS_LFSR, which has to be based on a single set of values for S, L, k. It is highly unlikely that the TSL of every core will be drastically shortened using the same values of S, L and k. Therefore, for minimizing the overall TSL, the system integrator has either to use a separate decompressor for each core, or to share a single decompressor among all cores, limiting though the maximum TSL reduction than can be achieved.

V. Variable-State-Skip LFSRs

VSS_LFSRs consist of multiple State-Skip circuits that implement different speedup factors, and thus they are able to perform jumps of variable lengths. We confine our study to the case of VSS_LFSRs that incorporate two State-Skip circuits, one with a small (k) and one with a large (K) speedup factor. The reason for this choice is that we observed that two speedup factors are sufficient to achieve very high TSL reduction.

A VSS_LFSR with two State-Skip circuits operates in two State-Skip modes: 1) K-mode which enables the VSS_LFSR to perform a long jump of K cycles ahead; and 2) k-mode which enables the VSS_LFSR to perform a short jump of k cycles ahead. Let A be the number of useless segments between two useful segments S_i, S_j ($j = i + A + 1$). The total length (in clock cycles) of these A useless segments is $C = A{\cdot}S{\cdot}r$(S is the segment size, and r the length of the longest scan chain of the CUT). Then an ordinary LFSR requires C cycles for traversing these useless segments. By using the VSS_LFSR, these segments can be traversed much faster. At first K-mode is used (the LFSR performs long jumps of length K) for $C_1 = \lfloor C/K \rfloor$ successive cycles. Then the VSS_LFSR switches to k-mode (the LFSR performs short jumps of length k) for $C_2 = \lfloor (C - C_1{\cdot}K)/k \rfloor$ cycles. Finally, Normal mode is used for $C_3 = C - C_1 \cdot K - C_2 \cdot k$ cycles. Thus, instead of C cycles, only $C_1 + C_2 + C_3 << C$ cycles, are required for traversing the useless segments.

The VSS_LFSR architecture is shown in Fig. 1(b). Except for the Look-Ahead unit, which will be described in detail, the remaining units are similar to the units of SSS_LFSR.

The Look-Ahead unit consists of the Segment Look-Ahead counter and the Jump Select Block. The Look-Ahead unit has two different modes of operation: 1) the C-calculation mode for locating the next useful segment; and 2) the C-skipping mode for controlling the Vector Generation unit so as to traverse the intermediate useless segments in K-mode, k-mode, or Normal mode. The Look-Ahead unit enters the first mode at the beginning of the generation of every useful segment, say S_i. Then, during the generation of useful segment S_i, it calculates the value of C, i.e., the number of cycles than must be skipped after the generation of useful segment S_i, in order to reach the next useful segment, let say S_j. This calculation is done on the fly, concurrently with the generation of the test vectors of S_i. Specifically, while the test vectors of segment S_i are applied to the core, the next segments (S_{i+1}, S_{i+2}, ...) are examined one by one by increasing the value of Segment Look-Ahead counter ($i+1, i+2, \ldots$) until the next useful segment S_j is found (the *Mode Select* unit monitors the Segment Look-Ahead counter and activates signal FoundUsefulSegment when S_j is found). For every useless segment found, the value $S{\cdot}r$ (i.e., its size in cycles) is added to Cycle counter. Consequently, when the next useful segment is found, this counter contains the number C of intermediate clock cycles between S_i and S_j. After the calculation of C, and upon completion of the generation of the test vectors of useful segment S_i, the Look-Ahead unit enters the second mode (the C-skipping mode).

TABLE II
COMPARISONS WITH BOTH TDC AND TSE

Circuit	[1]		[13] [16]	[13]	[16]	[18] [20]	[18]	[20]	[14]		[21]		Dynamic Reseed.		[9]		[17]		Proposed ($L = 200$)		
																			SSS	*VSS*	*TDV*
	TSL	*TDV*	*TSL*	*TDV*		*TSL*	*TDV*		*TSL*	*TDV*	*TSL*	*TDV*	*TSL*	*TDV*	*TSL*	*TDV*	*TSL*	*TDV*	*TSL*	*TSL*	
s9234	170	15.1	205	12.4	10.3	159	30	-	-	-	161	17.2	477	10.6	24 592	6.7	135 765	0.65	1784	1465	7.1
s13207	229	12.8	266	11.9	10.5	236	21	74	266	14.3	242	26	536	8.2	24 724	2.7	152 596	0.16	1756	1180	3.8
s15850	244	15.5	269	12.7	11.4	126	25	26	226	15.1	306	32.2	524	10.8	27 630	6.2	222 336	0.4	1740	1091	6.7
s38417	376	37	376	36.4	32. 2	99	85	45	376	49	854	89.1	920	55.2	85 885	47	625 273	5.5	13 113	3026	48.1
s38584	296	31.6	296	30.4	31.2	136	57	74	296	29	599	63.2	639	21.2	29 358	5.2	383 009	0.23	6639	1935	7.1

Then, the value of Cycle counter is compared against K, and while it is greater than or equal to K, the K-mode is used and the counter is decremented by K (i.e., at every clock cycle, K states of the LFSR sequence are skipped). When the value of Cycle counter drops below K, the above process continues with comparisons against k and the counter is decremented by k. When Cycle counter drops below k, then the Normal mode is used and the counter is decremented by 1 until it reaches 0. At this point the LFSR is already at the first state of useful segment S_j, and thus its generation begins.

We have to note that the *Mode Select* unit depends on the test set and it should be re-implemented for every core in a multi-core environment, whereas the rest of the units are common for all cores. Additionally, the ATE-SoC synchronization problem can be avoided by inserting small first-in first-out (FIFO) memory between the LFSR and the ATE channels (see [4]).

VI. COMPARISONS

We have conducted extended experiments (which are omitted due to lack of space) and we verified that for SSS_LFSRs, the improvement increases when speedup factor k increases and/or segment size S decreases. The TSL reduction is high for relatively small values of k and improves as k increases, but the improvement saturates for large values of k. At the same time, the hardware overhead of the State-Skip circuit increases almost linearly with k. Based on these observations we used values of k in the range [12, 24], which offer high TSL reduction and small hardware overhead of State-Skip circuit (between 60 and 100 gate equivalents on average—one gate equivalent or g.e. corresponds to a 2-input NAND gate). Similar observations were made for VSS_LFSRs, with the addition that the overhead of the State-Skip circuit exhibits significant fluctuations (i.e., ups and downs) for large values of K. Therefore, we chose high speedup factors (between 54 and 318) near to local minimums. Such speedup factors achieve high performance and low State-Skip circuit area overhead at the same time (between 50 and 250 g.e.). In all cases, segment sizes in the range [2, 10] were used.

In Table II we compare the proposed methods against the most efficient TSE and TDC methods, which are suitable for IP cores of unknown structure. The TSL (reported in vectors) and TDV (reported in Kbits, with $1\,\text{Kbit} = 10^3$ bits) comparisons of the proposed SSS_LFSRs (labeled as SSS) and VSS_LFSRs (labeled as VSS), for $L = 200$, against the TSE approaches of [9] and [17] are presented in the shaded columns. Both SSS_LFSRs and VSS_LFSRs exhibit very short test sequences compared to both [9] and [17]. The approach of [17] has very small ATE-memory requirements, but extremely long TSL. Moreover, in [9] it is shown that the hardware overhead required for implementing this method is prohibitively large (between 1300 and 9800 g.e. for 32 scan chains, and 4500–12 500 g.e. for 64 scan chains, for the larger ISCAS'89 circuits).

Table II compares also the proposed SSS_LFSRs and VSS_LFSRs for $L = 200$, against the most efficient TDC methods that are suitable for IP cores of unknown structure and report results for the ISCAS benchmarks (not shaded columns). Furthermore, comparisons are provided against a version of dynamic reseeding that we implemented using ring generators [19] instead of LFSRs. Contrary to the dynamic reseeding of [19], the fault simulation step was omitted in this implementation, in order to comply with the testing requirements of IP cores (note that, as expected, the omission of the fault simulation step adversely affects the TDV). In all but one case (s38417) the proposed method performs better than the compared TDC methods, in terms of TDV. We have to note though that in the case of s38417, the specified-bits volume of the utilized test set is very high (93 123 specified bits) and this negatively affects the achieved compression. With respect to the TSL results, we have to note that the proposed method offers much shorter test sequences than the rest TSE techniques, but longer test sequences than the TDC methods. Much shorter test sequences can be achieved by using smaller window sizes (e.g., 100, 50, etc.).

Next, we present the hardware overhead of the proposed decompressors. We focus on s13207, since the results for all circuits are similar, as, apart from the LFSR and the Mode Select unit, the hardware overhead of the remaining decompressor units does not depend on the test set. For SSS_LFSRs, the overhead of the State-Skip circuit is between 52 and 119 g.e. for $k \leq 24$. The average hardware overhead of the remaining decompressor units, for various values of L and S, excluding the Mode Select unit, is 320 g.e. The overhead of the Mode Select unit, is between 44 and 262 g.e., for $50 \leq L \leq 500$ and $2 \leq S \leq 50$. Note that only the Mode Select unit has to be implemented separately for each core under test (the rest of the logic is shared among all cores). In the case of VSS_LFSR for the same circuit, the overhead of the Variable-State-Skip circuit for $k = 46$ and $K = 230$ is equal to 203 g.e., and the total overhead of the LFSR, Phase Shifter, Controller unit, Look-Ahead unit, and the Decoder of the Segment Type unit, for $L = 200$ and $S = 5$, is 627 g.e. All the above units need to be implemented only once in a SoC. The only unit that has to be implemented separately for every core is the

TABLE III
VARIABLE VERSUS SINGLE STATE SKIPPING FOR MULTIPLE CORES

S	SSS_LFSR			VSS_LFSR				TSL
	k	TSL	HO	K	k	TSL	HO	Impr. (%)
2	2	53 471	9%	318	21	8511	10,5%	84.10%
5	5	31 358	7.7%	159	5	15 682	8.8%	50.00%
10	10	33 736	6.6%	18	10	26 731	7.8%	20.80%

Mode Select unit, whose hardware overhead lies between 44 and 262 g.e., for $50 \leq L \leq 500$ and $2 \leq S \leq 50$.

In our next set of experiments, we used the SSS_LFSRs as well as the VSS_LFSRs on a hypothetical multi-core SoC consisting of the largest ISCAS'89 benchmarks. In both cases a common decompressor was used and only the Mode Select unit was implemented separately for each core. Table III presents the TSL and area overhead (HO) results for three segment sizes, 2, 5, 10, and for LFSR size = 85. The HO is reported as the percentage of the HO of the decompresssor to the total HO of the five cores. It is obvious that the TSL gain offered by VSS_LFSRs is very high compared to SSS_LFSR and reaches 84.1%, at the expense of small additional hardware overhead.

In order to demonstrate the effectiveness of the proposed technique on large compacted test sets, we applied the classical, the dynamic and the window-based LFSR reseeding techniques to the IWLS'05 *Ethernet* benchmark circuit [27], which consists of 10.6K scan cells and 136.2×10^3 gates (11.71 Mbits test data). The TDVs of the classical and the dynamic LFSR reseeding were equal to 1222 Kbits and 689 kbits, respectively, whereas the TDV of the window-based LFSR reseeding, for L = 50, was equal to 211 Kbits. The TSL of the window-based reseeding was equal to 9600 vectors, while the TSLs of the other two methods were both equal to 1111 vectors. By using an SSS_LFSR with k = 4 and a VSS_LFSR with K = 221 and k = 4, the TSL of window-based reseeding was reduced to 2254 vectors and 1449 vectors, respectively. Thus, we conclude that the proposed method achieves considerable TDV reduction compared to the classical and dynamic reseeding methods with similar TSL.

VII. CONCLUSION

Two new types of LFSRs, the SSS and VSS LFSRs were introduced, which drastically shorten the test sequences of LFSR-reseeding-based test set embedding methods. Both types of LFSRs bridge the gap between TDC and TSE, by offering the high compression efficiency of TSE with test sequences reduced to such an amount (up to 98.8%) that their length approaches that of TDC methods. In this way, test set embedding becomes an attractive approach for testing IP cores.

REFERENCES

[1] G. K. Balakrishnan, S. Wang, and S. T. Chakradhar, "PIDISC: Pattern independent design independent seed compression technique," in *Proc. VLSID*, 2006, pp. 811–817.

[2] B. Keller and B. Koenemann, "OPMISR: The foundation for compressed ATPG vectors," in *Proc. ITC*, 2001, pp. 748–757.

[3] A. Chandra and K. Chakrabarty, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes," *IEEE Trans. Comp.*, vol. 52, no. 8, pp. 1076–1088, Aug. 2003.

[4] P. Gonciari, B. Al-Hashimi, and N. Nicolici, "Synchronization overhead in SoC compressed test," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 13, no. 1, pp. 140–152, Jan. 2005.

[5] A. Jas, J. Ghosh-Dastidar, M. Ng, and N. Touba, "An efficient test vector compression scheme using selective Huffman coding," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 6, pp. 797–806, Jun. 2003.

[6] D. Kagaris and S. Tragoudas, "On the design of optimal counter based schemes for test set embedding," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, no. 2, pp. 219–230, Feb. 1999.

[7] E. Kalligeros, X. Kavousianos, and D. Nikolos, "Efficient multiphase test set embedding for scan-based testing," in *Proc. ISQED*, 2006, pp. 433–438.

[8] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Test data compression based on variable-to-variable Huffman encoding with codeword reusability," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 7, pp. 1333–1338, Jul. 2008.

[9] D. Kaseridis, E. Kalligeros, X. Kavousianos, and D. Nikolos, "An efficient test set embedding scheme with reduced test data storage and test sequence length requirements for scan-based testing," in *Inf. Pap. Dig. IEEE ETS*, 2005, pp. 147–150 [Online]. Available: http://www.icsd.aegean.gr/lecturers/kalliger/Papers/ETS05_published.pdf

[10] B. Koenemann, "LFSR-coded Test Patterns for Scan Design," in *Proc. ETC*, 1991, pp. 237–242.

[11] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, "A SmartBIST variant with guaranteed encoding," in *Proc. ATS*, 2001, pp. 325–330.

[12] C. Krishna, A. Jas, and N. Touba, "Test vector encoding using partial LFSR reseeding," in *Proc. ITC*, 2001, pp. 885–893.

[13] C. Krishna and N. Touba, "Reducing test data volume using LFSR reseeding with seed compression," in *Proc. ITC*, 2002, pp. 321–330.

[14] C. V. Krishna and N. A. Touba, "Adjustable width linear combinational scan vector decompression," in *Proc. ICCAD*, 2003, pp. 863–866.

[15] H. K. Lee and D. S. Ha, "Atalanta: An efficient ATPG for combinational circuits," Dept. Electr. Eng., Virginia Polytech. Inst. and State Univ., Blacksburg, Tech. Rep. 93-12, 1993.

[16] J. Lee and N. Touba, "Low power test data compression based on LFSR reseeding," in *Proc. ICCD*, 2004, pp. 180–185.

[17] L. Li and K. Chakrabarty, "Test set embedding for deterministic BIST using a reconfigurable interconnection network," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 9, pp. 1289–1305, Sep. 2004.

[18] L. Li, K. Chakrabarty, S. Kajihara, and S. Swaminathan, "Efficient space/time compression to reduce test data volume and testing time for IP cores," in *Proc. 18th Int. Conf. VLSI Des.*, 2005, pp. 53–58.

[19] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 5, pp. 776–792, May 2004.

[20] S. Reda and A. Orailoglu, "Reducing test application time through test data mutation encoding," in *Proc. DATE*, 2002, pp. 1–5.

[21] L. Schäfer, R. Dorsch, and H.-J. Wunderlich, "RESPIN++: Deterministic embedded test," in *Proc. ETW*, 2002, pp. 37–44.

[22] S. Swaminathan and K. Chakrabarty, "On using twisted-ring counters for test set embedding in BIST," *JETTA*, vol. 17, no. 6, pp. 529–542, Dec. 2001.

[23] V. Tenentes, X. Kavousianos, and E. Kalligeros, "State skip LFSRs: Bridging the gap between test data compression and test set embedding for IP cores," in *Proc. DATE*, 2008, pp. 474–479.

[24] N. A. Touba, "Circular BIST with state skipping," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 10, no. 5, pp. 668–672, Oct. 2002.

[25] Z. Wang, K. Chakrabarty, and S. Wang, "Integrated LFSR reseeding, test-access optimization, and test scheduling for core–based system-on-chip," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 28, no. 8, pp. 1251–1264, Aug. 2009.

[26] S. Ward, C. Schattauer, and N. A. Touba, "Using statistical transformations to improve compression for linear decompressors," in *Proc. DFT*, 2005, pp. 42–50.

[27] *IWLS'05 Benchmark Circuits* [Online]. Available: http://www.iwls.org/iwls2005/benchmarks.html