State Skip LFSRs: Bridging the Gap between Test Data Compression and Test Set Embedding for IP Cores*

V. Tenentes[1], X. Kavousianos[1] and E. Kalligeros[2]
[1]Computer Science Department, University of Ioannina, Greece
[2] Information & Communication Systems Engineering Dept., University of the Aegean, Greece
tenentes@uoi.gr, kabousia@cs.uoi.gr, kalliger@aegean.gr

Abstract

We present a new type of Linear Feedback Shift Registers, State Skip LFSRs. State Skip LFSRs are normal LFSRs with the addition of a small linear circuit, the State Skip circuit, which can be used, instead of the characteristic-polynomial feedback structure, for advancing the state of the LFSR. In such a case, the LFSR performs successive jumps of constant length in its state sequence, since the State Skip circuit omits a predetermined number of states by calculating directly the state after them. By using State Skip LFSRs we get the well-known high compression efficiency of test set embedding with substantially reduced test sequences, since the useless parts of the test sequences are dramatically shortened by traversing them in State Skip mode. The length of the shortened test sequences approaches that of test data compression methods. A systematic method for minimizing the test sequences of reseeding-based test set embedding methods, and a low overhead decompression architecture are also presented.

1. Introduction

The extensive use of pre-designed and pre-verified cores in contemporary Systems-on-a-Chip (SoCs) and the limited channel capacity, memory and speed of Automatic Test Equipments (ATEs) render testing the bottleneck of SoCs production cycle. Embedded test eases the burden of testing on ATEs by combining the ATE capabilities with on-chip integrated structures. The test set is stored compressed in the ATE memory, and, during testing, it is downloaded on-chip where it is decompressed by an embedded decoder and then applied to the core under test (CUT).

Many embedded testing techniques maximize compression by utilizing structural information of the CUT and by exploiting the advantages offered by the use of Automatic Test Pattern Generation (ATPG) and/or fault simulation tools. Most of these methods utilize linear decompressors due to their high efficiency and simplicity [3], [6], [10], [16], [19], [24], [25], [28], [35], [36]. Commercial tools have been also developed [2], [15], [27]. Often, the structure of the cores is unavailable to the system integrator and a pre-computed test set is the only test information provided by the core vendors. For such cores, which are called Intellectual Property (IP) cores, neither ATPG, nor fault simulation can be performed and thus the only option is to directly compress the pre-computed test set. Linear decompressors have been extensively used in this case as well [1], [17], [18], [21], [30], [33], [34]. Other techniques utilize various compression codes [4], [5], [7], [12], [13], [26], [32], and are suitable for cores with a single scan chain. Of course, there are also methods that do not belong in either of the above categories, e.g., [23], [29].

Test set embedding is another option for cores of unknown structure. Test set embedding techniques require less test data storage than test data compression methods, since they encode the pre-computed test vectors in long pseudo-random sequences generated on-chip. In [8] and [31] the pseudorandom sequences are generated by counters. In [22] an area-demanding reconfigurable interconnection network is presented that achieves a vast reduction of the test data stored on ATE. The main drawback of these techniques is their prohibitively long test application time. The multiphase method proposed in [9] has small hardware overhead and generates shorter test sequences than [8], [22] and [31]. An even higher reduction of the test sequence length is achieved in [11] at the expense of a slight increase in test data volume. However, [9] and [11] still require long test sequences.

In this paper a new type of Linear Feedback Shift Registers (LFSRs), called State Skip LFSRs, is presented. Apart from their linear feedback structure that corresponds to their characteristic polynomial, State Skip LFSRs also incorporate a small linear circuit called State Skip circuit. The state of a State Skip LFSR can be advanced by using either the polynomial feedback structure (Normal mode) or the State Skip Circuit (State Skip Mode). In State Skip mode, the LFSR performs successive jumps of constant length in its state sequence, since the State Skip circuit omits a predetermined number of states by calculating directly the state after them. State Skip LFSRs drastically shorten the test sequences of LFSR-reseeding-based test set embedding methods, since they can operate in State Skip mode in the useless parts of the test sequence. By offering the well-known high compression efficiency of test set embedding with substantially reduced test sequences, State Skip LFSRs bridge the test-sequence-length gap between test data compression and test set embedding techniques, and render the latter a very attractive testing approach for IP cores.

This work was co-funded by the European Union in the framework of the project "Support of Computer Science Studies in the Univ. of Ioannina" of the "Operational Program for Education and Initial Vocational Training" of the 3rd Community Support Framework of the Hellenic Ministry of Education, funded by national sources and by the European Social Fund (ESF).

978-3-9810801-3-1/DATE08 © 2008 EDAA

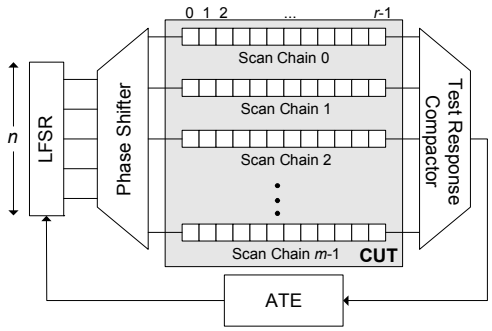

Fig. 1. Classical LFSR Reseeding Architecture

2. Motivation

Fig. 1 shows the classical LFSR reseeding architecture. Every n-bit seed (n is the LFSR size) is transferred from the ATE to the LFSR, where it is expanded into a test vector of $m \cdot r$ bits (m is the scan-chain volume and r the scan-chain length). A phase shifter is also needed for reducing the linear dependencies of the LFSR-generated bit sequences. Each n-bit seed is the compressed version of an $m \cdot r$-bit test cube (test vectors with x bits are called test cubes) and is calculated by solving a system of linear equations, which is formed according to the specified bits of the test cube [14] (the x bits are filled with pseudorandom data during decompression). Specifically, the initial state of the LFSR is considered as a set of binary variables a_0, ..., a_{n-1}. At every clock cycle, m linear expressions of these variables are generated at the m outputs of the phase shifter. Thus, each bit of a test cube corresponds to one linear expression. Every linear expression corresponding to a specified bit of a test cube is set equal to that bit, and in this way the system of linear equations is formed. The solution of this system is the 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}, and determines the minimum required LFSR size.

If each seed is used for encoding a single 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 tackle this problem [6], [16], [27], [33]. A very attractive one is to utilize the same seed for encoding more than one test cube in a sequence of L pseudorandom vectors. In other words, each seed is expanded into a window of L vectors, instead of one. The number of test cubes encoded in the window is usually much smaller than L, which means that useless vectors are also applied to the CUT. This approach is very effective since for every test cube, L (and not just one) systems of equations are constructed, and among the solvable systems, the one resulting in the highest compression is selected. In other words, each test cube is encoded in such a way so as to maximize the overall encoding efficiency.

There are many ways to encode multiple test cubes in an L-vector window. One very effective algorithm for minimizing the number of seeds is the following [11]: initially, the test cube with the highest number of specified bits is selected and the system corresponding to the first vector of the window is solved. The rest test cubes are selected iteratively according to the following criteria: Among the solvable systems that correspond to the test cubes containing the maximum number of specified bits, we identify those that their solution leads to the replacement of the fewest variables in the L-vector window. Among them, we find those corresponding to the cube that can be encoded the fewest times in the window, and we finally select the system nearest to the first vector of the window. After solving the selected system, some of the variables are replaced by logic values, whereas the rest remain unspecified and they are utilized for encoding additional test cubes. The construction of a seed is completed when no system for any of the unencoded test cubes can be solved in the L-vector window.

Table 1. Classical vs. Window-based LFSR Reseeding

		Classical Reseeding (L=1)		Window Based Reseeding (L>1)					
	LFSR			L=50		L=200		L=500	
Circuit	Size	TDV	TSL	TDV	TSL	TDV	TSL	TDV	TSL
s9234	44	10692	243	8008	9100	7128	32400	6688	76000
s13207	24	8856	369	5328	11100	3816	31800	2688	56000
s15850	39	11622	298	7410	9500	6669	34200	6201	79500
s38417	85	58225	685	50660	29800	48110	113200	47005	276500
s38584	56	22680	405	10584	9450	7056	25200	5152	46000

In order to illustrate the compression superiority of the method which expands seeds into windows of L >1 vectors, over the classical encoding where each seed is expanded into a single vector (L=1), we conducted the following experiment: Uncompacted test sets generated by Atalanta [20] for the largest ISCAS 89 benchmark circuits were compressed using the classical LFSR encoding (L=1) as well as the window-based encoding with window sizes L=50, 200 and 500. 32 scan chains were assumed for each circuit. For providing a fair comparison, the algorithm of the previous paragraph was applied for all examined window sizes. Hence, even for L=1, each seed was let encode as many test cubes as possible (i.e., all compatible cubes that can be compressed into a single seed, which will be then expanded to just one test vector). Table 1 presents the size of the LFSR, the test data volume-TDV (# bits) and test sequence length-TSL (# test vectors applied) for each core. It is obvious that as window size L increases, the encoding improves a lot, but, on the other hand, the test sequences grow rapidly and become prohibitively long.

3. Proposed Method

3.1. State Skip LFSRs

Consider the LFSR shown in Fig. 2 without the State Skip circuit (i.e., assume that Input 1 of each multiplexer is selected, which means that the LFSR operates normally according to its feedback polynomial). The symbolic contents of the LFSR during cycles $t_0 ... t_3$, assuming that the initial state is $(c_0, c_1, c_2, c_3) = (a_0, a_1, a_2, a_3)$, are shown in the table of Fig. 2. Let us focus on the contents of the LFSR cells during clock cycles t_0 and t_2. Observe that the value of cell c_0 at cycle t_2 is equal to the XOR of the values of cells c_2, c_3 at cycle t_0, i.e., $c_0(t_2) = c_2(t_0) \oplus c_3(t_0)$, where $c_i(t_j)$ is the value of cell

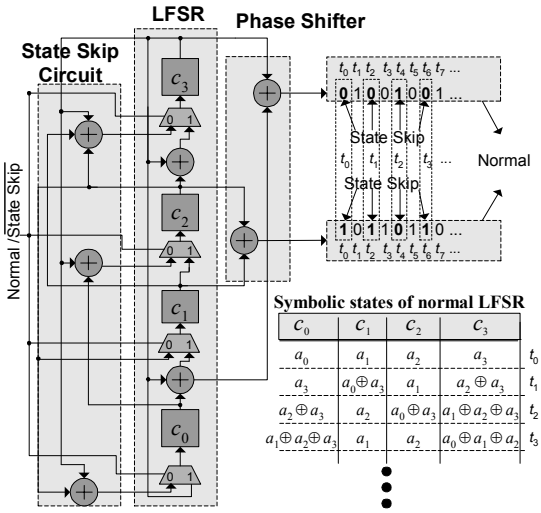

Fig. 2. Example of State Skip LFSR

c_i during cycle t_j. For the rest cells we derive similar relations: $c_1(t_2)=c_2(t_0)$, $c_2(t_2)=c_0(t_0)\oplus c_3(t_0)$, $c_3(t_2)=c_1(t_0)\oplus c_2(t_0)\oplus c_3(t_0)$. These relations depend solely on the characteristic polynomial and the distance between the clock cycles of interest (2 cycles in the above example) and not on the LFSR state. Hence, they are satisfied for every pair of cycles t_{i+2}, t_i, i.e.: $c_0(t_{i+2})=c_2(t_i)\oplus c_3(t_i)$, $c_1(t_{i+2})=c_2(t_i)$, $c_2(t_{i+2})=c_0(t_i)\oplus c_3(t_i)$, and $c_3(t_{i+2})=c_1(t_i)\oplus c_2(t_i)\oplus c_3(t_i)$. Generally, for an LFSR of size n and for every $k\geq 1$, n linear expressions $F_0^k,...,F_{n-1}^k$ exist that satisfy the following relations, for every value of i:

$$c_0(t_{i+k})=F_0^k\left(c_0(t_i),...,c_{n-1}(t_i)\right),\ \ \ldots,\ \ c_{n-1}(t_{i+k})=F_{n-1}^k\left(c_0(t_i),...,c_{n-1}(t_i)\right) \quad \textbf{(1)}$$

When k=1, the above expressions represent the LFSR operation according to the characteristic polynomial. $F_0^k,...,F_{n-1}^k$ are easily calculated by setting i=0 and simulating the LFSR symbolically [equations (1) are satisfied for every value of i]. Specifically, the LFSR is initialized with symbolic state $(c_0(t_0), ..., c_{n-1}(t_0))=(a_0, ..., a_{n-1})$ and is clocked k times. After the k-th clock cycle, the LFSR contents $c_0(t_k)$, ..., $c_{n-1}(t_k)$, which are linear expressions of the initial contents $c_0(t_0)$, ..., $c_{n-1}(t_0)$, constitute the required linear expressions $F_0^k,...,F_{n-1}^k$.

The basic idea proposed in this paper is to integrate $F_0^k,...,F_{n-1}^k$ in the LFSR structure. The modified LFSR, which is called hereafter State Skip 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, while, in State Skip mode, the state sequence is generated by the integrated linear circuit implementing $F_0^k,...,F_{n-1}^k$. 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. We will see that the hardware overhead of the linear logic implementing the expressions $F_0^k,...,F_{n-1}^k$ is small when k is not very high, and that a value of k up to 24 is sufficient for a vast reduction of the test sequence length.

Example. Fig. 2 presents the State Skip version of the previously mentioned LFSR, for k=2. At the input of every LFSR cell, a 2:1 multiplexer selects either the logic value generated by the characteristic polynomial (Normal mode) or the value generated by the State Skip circuit (State Skip mode). Assuming that the initial state of the LFSR is (c_0, c_1, c_2, c_3) = 1011, the logic values generated at the outputs of the phase shifter are shown in the upper right part of Fig. 2, for operation either in Normal mode (all logic values inside the grey horizontal bars) or in State Skip mode (boldfaced and highlighted by the vertical bars). As we can see, in State Skip mode, only half of the logic values are generated and thus the test sequence is reduced by a factor 2 (= k). ■

3.2. Test Sequence Reduction Method

By using State Skip LFSRs we can minimize the length of the test sequence generated when each LFSR seed is expanded into a window of L vectors. At first, every window is partitioned into L/S segments of S vectors (S is a designer-defined parameter in the range [1, L]). Every segment is labeled either as useful, if it embeds at least one test cube, or as useless, if it does not embed any test cubes. Useful segments are generated using Normal mode, whereas useless segments are shortened by a factor k using State Skip mode.

Many test cubes consist of a small number of specified bits and thus they are fortuitously embedded in more than one segment. We exploit this property in order to minimize the number of useful segments and consequently the test sequence length. Specifically, we partition the test cubes into two sets, A and B. Set A consists of the test cubes that are embedded in only one segment of all windows, whereas set B consists of the test cubes that are embedded in more than one segments. All segments embedding test cubes of set A are selected and labeled as useful. All test cubes of set B embedded in those segments are removed from set B. For the remaining test cubes in set B, we apply the following greedy useful-segment-selection procedure:

a. Select the segment embedding most of the remaining test cubes. If more than one such segment exists, select the one that is closest to the beginning of the window.
b. Drop the test cubes embedded in the selected segment.
c. If there are any remaining test cubes go to step a.

After useful-segment selection, the seeds are grouped according to the number of useful segments that their windows contain, and the groups are sorted in ascending order: group 1 contains all seeds with 1 useful segment, group 2 contains all seeds with 2 useful segments and so on. This grouping enable us to terminate the generation of the vector-window of each seed right after the generation of the last useful segment, shortening in this way the test sequences even more.

The efficiency of the described test-sequence-reduction process strongly depends on the segment size (S). As it will be shown in Section 4, small segments lead to higher test-sequence-length reductions compared to large segments, but impose a little higher hardware overhead than the large ones.

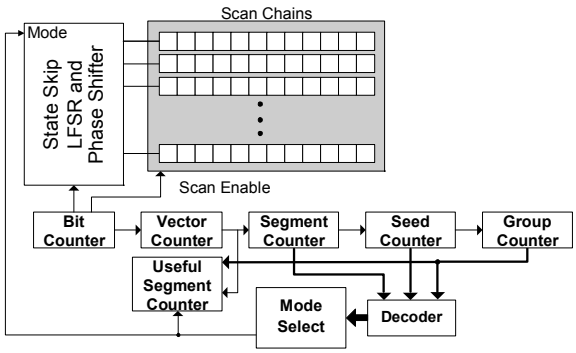

Fig. 3. Proposed Decompression Architecture

3.3. Decompression Architecture

The proposed decompression architecture is shown in Fig. 3. The Bit and Vector Counters control the loading of the test vectors in the scan chains, while the Segment and Useful Segment Counters count respectively the total number of segments and the number of useful segments generated for each seed. Seed Counter counts the seeds of every seed-group, and Group Counter counts the seed-groups. Every time a new seed is loaded in the LFSR, Useful Segment Counter is also loaded with Group Counter's value, which is equal to the number of useful segments of every seed belonging in a seed-group. Then, after the generation of a useful segment, Useful Segment Counter decreases by one and when it reaches 0, Seed Counter increases and the next seed is loaded in the LFSR. When all seeds of a group have been generated, Group Counter increases by one in order to continue with the next group.

The Mode Select unit is a combinational circuit that determines if the next segment is a useful one or not. It receives the decoded outputs of the Segment, Seed and Group counters and generates the Mode signal that is driven to the State Skip LFSR (the decoding of the outputs of the aforementioned counters leads to significantly smaller Mode Select units when testing multiple cores of a SoC). Mode signal is equal to 1 only if the segment is a useful one. 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. Moreover, according to the seed-selection process, the first segment of every seed is always a useful one, since the first vector generated by each seed embeds at least one test cube (see section 2). Consequently, the first segment of each seed needs minimum decoding logic and therefore the implementation overhead of Mode Select unit is significantly reduced. Additionally, in a multi-core environment, only the Mode Select unit has to be re-implemented for every core, whereas the rest of the units are common for all cores.

4. Evaluation and Comparisons

The proposed method was implemented in C programming language and experiments were conducted on a Pentium PC for the largest ISCAS 89 benchmark circuits, assuming 32 scan chains for each one of them. We used uncompacted test sets for stuck-at faults (offering 100% non-

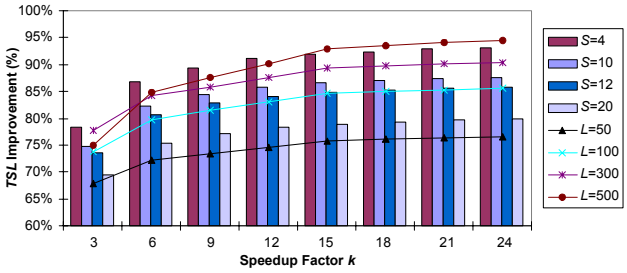

Fig. 4. TSL Impr. for Various Values of *k*, *S* and *L*

redundant fault coverage) generated by Atalanta [20]. The run-time of the proposed method is only a few minutes.

Initially, we study the influence of speedup factor k, segment size S and window size L on the test sequence length (TSL) improvement achieved by the proposed method. The TSL improvement is calculated by the following formula:

$$TSL\ \text{Improvement}\ (\%) = \left(1 - \frac{TSL\ \text{of prop. method}}{TSL\ \text{of orig. window-based method}}\right) \cdot 100 \qquad (2)$$

Due to the high volume of the experiments, we focus on s13207 (the rest circuits exhibit similar behavior). In the sequel, the test sequence length is reported as the number of test vectors applied to the CUT and the test data volume as the number of bits stored in the tester.

The first set of experiments (the bars in Fig. 4) demonstrates the influence of speedup factor k on the TSL improvement for various segment sizes (S). We present results for $3 \le k \le 24$, and S = 4, 10, 12 and 20, assuming windows of L=300 vectors. It is obvious that the TSL improvement is significant (from 69-78% for k=3, to 80-93% for k=24) for all segment sizes. The improvement increases when speedup factor k increases and/or segment size S decreases. When k increases, the number of cycles required for the generation of useless segments reduces, and thus TSL reduces too. When S decreases, the segmentation becomes finer, i.e. the total size of useful segments decreases while the total size of useless segments increases (their sum though remains constant). This is explained by the fact that each useful segment may also contain some useless pseudorandom vectors, the number of which depends on size S. By decreasing S, fewer useless vectors remain in the useful segments, and since a useless segment is generated faster than a useful one (its major portion is skipped), the overall test sequence length decreases.

We next study the influence of speedup factor k on the TSL improvement for various window sizes (L). The curves in Fig. 4 present the TSL improvement for $3 \le k \le 24$ and L=50, 100, 300 and 500 (S was equal to 5 in all experiments). We observe that as L increases, the TSL improvement increases too. This is explained by the fact that large windows contain more useless segments than the small ones, and the length of useless segments is drastically shortened by the proposed technique.

In Table 2 we present the test sequence length reduction achieved by the proposed method for L=50, 200, 500, S=2, 5, 10, and $5 \le k \le 24$ (the best results for the various values of S, k are reported). Columns labeled "Orig." present the test sequence

Table 4. *TSL* and *TDV* Results of LFSR-Reseeding-based Methods for IP Cores with Multiple Scan Chains

	[1]		[17]	[21]	[34]	[23]	[29]		[18]		[30]		Classical LFSR Reseeding L=1		Prop. L=200	
Circuit	*TSL*	*TDV*	*TSL*	*TDV*			*TSL*	*TDV*		*TSL*	*TDV*	*TSL*	*TDV*	*TSL*	*TDV*	*TSL*	*TDV*
s9234	170	15092	205	12445	10302	-	159	30144	-	-	-	161	17198	243	10692	1784	**7128**
s13207	229	12798	266	11859	10484	10810	236	20988	74423	266	14307	242	26004	369	8856	1756	**3816**
s15850	244	15480	269	12663	11411	12405	126	25140	26021	226	15067	306	32226	298	11622	1740	**6669**
s38417	376	37020	376	36430	**32152**	32154	99	85225	45003	376	49001	854	89132	685	58225	13113	48110
s38584	296	31574	296	30355	31152	31000	136	57120	73464	296	28994	599	63232	405	22680	6639	**7056**

Table 2. Test Sequence Length Improvements

	L=50			L=200			L=500		
Circuit	Orig.	Prop.	Impr.	Orig.	Prop.	Impr.	Orig.	Prop.	Impr.
s9234	9100	1082	88%	32400	1784	94%	76000	3055	96%
s13207	11100	1309	88%	31800	1756	94%	56000	2701	95%
s15850	9500	1129	88%	34200	1740	95%	79500	2791	96%
s38417	29800	7626	74%	113200	13113	88%	276500	21865	92%
s38584	9450	3805	60%	25200	6639	74%	46000	9054	80%

Table 3. Comparisons against Test Set Emb. Methods

	Test Data Volume			Test Sequence Length			*TSL* Impr.	
Circuit	[11]	[22]	Prop.	[11]	[22]	Prop.	[11]	[22]
s9234	7020	648	6864	24592	135765	2163	91.2%	98.4%
s13207	3475	162	3336	24724	152596	2072	91.6%	98.6%
s15850	6520	396	6357	27630	222336	2138	92.3%	99.0%
s38417	48418	5440	47855	85885	625273	18512	78.4%	97.0%
s38584	6384	228	6272	29358	383009	7489	74.5%	98.0%

length (# vectors) of the window-based approach with normal LFSRs, whereas the columns with label "Prop." present the test sequence length of the window-based approach with State Skip LFSRs. Columns labeled "Impr." present the reduction percentage for each case. Note that both approaches (the original and the proposed one) have the same test data volumes (the TDVs in Table 1). It can be seen that the reduction achieved by the proposed method is very high (60%-96%).

We will now compare the proposed method against the most efficient test set embedding and test data compression methods, which are suitable for IP cores of unknown structure with multiple scan chains. No comparisons are provided against approaches that need structural information of the CUT or require ATPG synergy. Such methods target cores of known structure and thus employ fault simulation, and, most of the times, specially constrained ATPG processes, which reduce significantly and tailor to the encoding method the data that need to be compressed. Note that for cores of unknown structure neither ATPG nor fault simulation can be performed. The *TSL* improvements are calculated according to relation (2), by replacing the "*TSL* of the orig. window-based method" with the "*TSL* of the compared method".

In Table 3, the *TDV*-*TSL* comparisons of the test set embedding approaches of [11] and [22] with the proposed one, for L=300, are presented (comparisons against [9] are omitted, since [11] reports much shorter test sequences than [9] with comparable *TDV*s). As can be seen from Table 3, the proposed approach exhibits very short test sequences as compared to both [11] and [22]. The approach of [22] has very small *TDV* requirements, but its test sequences are extremely long. Moreover, as shown in [11], the hardware overhead required for implementing this method is prohibitively large (estimated between 1300-9800 gate equivalents for 32 scan chains - a gate equivalent corresponds to a 2-input nand gate).

In Table 4 we compare the proposed approach against various test data compression methods which are suitable for IP cores with multiple scan chains ([1], [17], [18], [21], [23], [29], [30] and [34]), as well as with the classical LFSR reseeding approach (L=1). Note that [17], [21] and [34], as well as [23] and [29] have the same *TSL*s, and for that reason they have been reported under one common column. We can see that in all but one case (s38417) the proposed method outperforms the other ones in terms of test data volume. The reduced performance in the case of s38417 is due to the high volume of specified bits in the test set used in our experiments (93123 specified bits). The test sequence length of the proposed method is higher than that of the rest methods. However, the speedup factor k in the presented experiments is relatively small ($k \leq 24$), and therefore by increasing k much shorter sequences can be achieved.

Table 4 demonstrates the two options for testing IP cores of unknown structure: test data compression (many data, small test sequences) and test set embedding (few data, greater test sequences). Until now, the test sequences of the latter category of techniques were prohibitively long. State Skip LFSRs bridge this gap by offering the well-known high compression efficiency of test set embedding with very small test sequences. Taking also into account the high volume of scan chains (a few, fast, internal-clock cycles are required for loading each vector) and the fact that, compared to test data compression, significantly fewer data need to be transferred through the slow ATE-SoC connections in test set embedding, we conclude that the actual test application time of State-Skip-LFSR-based test set embedding, renders the latter a very attractive testing approach.

Finally, we present hardware overhead results of the proposed method. We will again focus on s13207 (the results for the rest circuits are similar, since apart from the LFSR and the Mode Select unit, the hardware overhead of the rest decompressor does not depend on the test set). The overhead of the State Skip circuit is very low for the speedup factors of interest ($k \leq 24$). For example, in the case of s13207, as k increases from 12 to 32, the overhead of the State Skip circuit increases from 52 to 119 gate equivalents. For the same circuit and for various values of L and S, the average total overhead of the rest of the decompressor (LFSR, phase shifter, counters, control and decoding logic), excluding the Mode Select unit, was around 320 gate equivalents. This

overhead is very small and similar to that of most test data compression and test set embedding techniques in the literature. Moreover, the aforementioned decompressor units, as well as the State Skip circuit have to be implemented only once in a SoC and reused for all cores. On the other hand, the hardware overhead of the Mode Select unit, which has to be implemented for every core separately, was between 44 and 262 gate equivalents, for $50 \leq L \leq 500$ and $2 \leq S \leq 50$.

In order to assess the overall cost of the proposed scheme, we synthesized the decompressor of a hypothetical multi-core SoC comprising all five examined ISCAS'89 circuits. For each circuit we set L=200, S=10 and k=10. The Mode Select unit was implemented separately for every core and its overhead was between 107 and 373 gate equivalents. The rest parts were shared among all cores. The overall area of the decompressor was only 6.6% of the area occupied by the SoC.

5. Conclusions

A new type of LFSR which drastically shortens the test sequence of LFSR-reseeding-based test set embedding methods was introduced. State Skip LFSRs incorporate a small linear circuit, which calculates at each clock cycle the LFSR state k cycles after the current state, shortening in this way the useless parts of the test sequence by a factor k. State Skip LFSRs bridge the gap between test data compression and test set embedding by offering the high compression efficiency of test set embedding with test sequences reduced to such an amount (up to 96%) that approach the length of the sequences of test data compression methods. In this way, test set embedding becomes an attractive approach for testing IP cores.

References

[1] K. Balakrishnan et al,"PIDISC:Pattern Independent Design Independent Seed Compression Technique",*VLSID* 2006, pp.811-817.

[2] C. Barnhart et al., "OPMISR: the foundation for compressed ATPG vectors", in *Proc. ITC*, 2001, pp. 748 – 757.

[3] I. Bayraktaroglu, A. Orailoglu "Concurrent application of compaction and compression for test time and data volume reduction in scan designs" *IEEE Trans. Comp*, vol 52, pp.1480-1489, Nov 2003

[4] 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. on Comp.*, vol. 52, pp. 1076 – 1088, Aug. 2003.

[5] P. T. Gonciari, B. Al-Hashimi, and N. Nicolici, "Variable-length input Huffman coding for system-on-a-chip test", *IEEE Trans. on CAD,* vol. 22, pp. 783 – 796, June 2003.

[6] S. Hellebrand et al., "Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers", *IEEE Trans. on Comp.*, vol. 44, pp.223–233, Feb. 1995.

[7] A. Jas, J. Ghosh-Dastidar, M. Ng., and N. Touba, "An efficient test vector compression scheme using selective Huffman coding", *IEEE Trans. on CAD*, vol. 22, pp. 797-806, June 2003.

[8] D. Kagaris, and S. Tragoudas, "On the design of optimal counter based schemes for test set embedding", *IEEE Trans. on CAD*, pp. 219-230, Feb. 1999.

[9] E. Kalligeros et al., "Efficient Multiphase Test Set Embedding for Scan-based Testing", in Proc. ISQED, 2006, pp. 433-438.

[10] E. Kalligeros, X. Kavousianos, and D. Nikolos, "Multiphase BIST: a new reseeding technique for high test-data compression", IEEE Trans. on CAD, vol. 23, pp. 1429-1446, Oct. 2004.

[11] D. Kaseridis et al., "An efficient test set embedding scheme with reduced test data storage and test sequence length requirements for scan-based testing", Inf. Pap. Dig. IEEE ETS, 2005, pp. 147-150.

[12] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Multilevel Huffman coding: an efficient test-data compression method for IP cores", IEEE Trans. on CAD, vol. 26, pp. 1070-1083, June 2007.

[13] X. Kavousianos, E. Kalligeros, D. Nikolos, "Optimal Selective Huffman Coding for Test-Data Compression", IEEE Trans. on Computers, Vol. 56, No 8, Aug. 2007, pp. 1146-1152.

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

[15] B. Koenemann, et al., "A SmartBIST variant with guaranteed encoding", in Proc. ATS, 2001, pp. 325-330.

[16] C. Krishna, A. Jas, and N. Touba, "Test Vector Encoding Using Partial LFSR Reseeding", in Proc. ITC, 2001, pp. 885-893.

[17] C. Krishna, N. Touba, "Reducing test data volume using LFSR reseeding with seed compression", ITC, 2002, pp. 321-330.

[18] C. Krishna, N. Touba, "Adjustable width linear combinational scan vector decompression", Proc. ICCAD, 2003, pp. 863- 866.

[19] C. Krishna, N. Touba, "3-Stage variable length continuous-flow scan vector decompression scheme", VTS, 2004, pp. 79- 86.

[20] 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.

[21] J. Lee, and N. Touba, "Low Power Test Data Compression Based on LFSR Reseeding", in Proc. ICCD, 2004, pp. 180-185.

[22] L. Li, and K. Chakrabarty, "Test set embedding for deterministic BIST using A reconfigurable interconnection network", *IEEE Trans. on CAD*, vol.23, pp. 1289-1305, Sept. 2004.

[23] L. Li et al., "Efficient space/time compression to reduce test data volume and testing time for IP cores", in Proc. 18th Int. Conf. on VLSI Des., 2005, pp. 53-58.

[24] H. Liang et al., "Two-Dimensional Test Data Compression for Scan-Based Deterministic BIST", ITC 2001, pp. 894-902.

[25] S. Mitra, K. Kim, "XPAND: An Efficient Test Stimulus Compression Technique", *IEEE Trans. on Comp.*, vol. 55, pp. 163-173, Feb. 2006.

[26] M. Nourani, M. Tehranipour, "RL-Huffman encoding for test compression and power reduction in scan applications", *ACM Trans. on Des. Aut. of Electr. Syst.*, vol. 10, pp. 91–115, Jan. 2005.

[27] J. Rajski et al., "Embedded deterministic test", *IEEE Trans. on CAD*, vol. 23, pp. 776-792, May 2004.

[28] W. Rao et al., "Test Application Time and Volume Compression through Seed Overlapping", in Proc. DAC, 2003, pp. 732-737.

[29] S. Reda, and A. Orailoglu, "Reducing Test Application Time Through Test Data Mutation Encoding", DATE, 2002, pp. 1-5.

[30] L. Schäfer, R. Dorsch, and H.-J. Wunderlich, "RESPIN++ – Deterministic Embedded Test", Proc. ETW, 2002, pp. 37-44.

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

[32] M. Tehranipour, M. Nourani, and K. Chakrabarty, "Nine-coded compression technique for testing embedded cores in SoCs", *IEEE Trans. on VLSI Syst.*, vol. 13, pp. 719-731, June 2005.

[33] E. Volkerink, and S. Mitra, "Efficient Seed Utilization for Reseeding based Compression", Proc. VTS, 2003, pp. 232-237.

[34] S. Ward et al., "Using Statistical Transformations to Improve Compression for Linear Decompressors", DFT, 2005, pp. 42-50.

[35] P. Wohl, et al.,"Efficient Compression of Deterministic Patterns into Multiple PRPG Seeds", Proc. ITC, 2005, pp. 1-10.

[36] P. Wohl et al,"X-tolerant Compression and Application of Scan ATPG Patterns in a BIST Archirtecture", ITC, 2003, pp 727-736.