Shrinking the Application Time of Test Set Embedding by Using Variable-State Skip LFSRs

V. Tenentes[1], X. Kavousianos[1] and E. Kalligeros[2]
[1]Computer Science Dept., University of Ioannina, Greece
[2]Information & Communication Systems Engineering Dept., Univ. of the Aegean, Greece

Abstract

It is well-known that the high compression efficiency of test set embedding is compromised by its long test application times. To alleviate this problem we present a sophisticated version of the recently proposed State Skip LFSRs, the Variable-State Skip (VSS) LFSRs. By using VSS LFSR successive jumps of variable lengths can be performed in the state sequence of the LFSRs and thus the useless parts of the test sequences can be effectively skipped. A low-overhead decompression architecture, that overcomes the limitations of simple State Skip LFSRs, is also proposed. The combination of VSS LFSRs with the proposed architecture offers the very small test-data volumes of test set embedding, with drastically shortened test sequences. Also, in a multi-core environment where a common decompressor is used, maximum test-sequence-length reduction can be achieved for every individual IP core that is tested.

1. Introduction

Testing of contemporary Systems on a Chip (SoCs) has become a very challenging task, due to the extensive integration of pre-designed and pre-verified modules (i.e., cores). Previous-generation Automatic Test Equipments (ATEs) are unable to keep up with the increasing memory, speed and channel capacity requirements. On the other hand, the acquisition of new generation ATEs requires large capital investments, which elevate the cost of the final products. Currently, the most effective way for solving this problem is Test Resource Partitioning. According to Test Resource Partitioning, low cost test structures are integrated on-chip for easing the burden of testing on ATEs, without compromising the test quality.

The testing problem becomes even worse when Intellectual Property (IP) cores of unknown structure are embedded in the SoCs. The structure of IP cores is often hidden from the system integrator and thus no design modifications can be made to them. Moreover, testing of such cores cannot take advantage of the Automatic Test Pattern Generation (ATPG) and Fault Simulation tools, which have been traditionally used for reducing the amount of test data that need to be stored in the ATE memory. Consequently, various state of the art methods like [2, 5, 7-9, 13, 18, 19, 22, 27, 28, 30, 31, 37, 40, 41] cannot be directly applied to IP cores. The only test information that is provided by the vendor of each IP core is a pre-computed test set, which has to be directly applied to the core. Due to their great size, these test sets are usually stored compressed in the ATE memory, and, during testing, they are downloaded and decompressed on chip by integrated decompressors.

Most of the test data compression methods proposed in the literature utilize combinational and/or sequential linear decompressors [1, 20, 21, 24, 33, 38, 39]. Recently, various compression methods that are based on compression codes (e.g., Golomb, Run Length, Huffman, etc.) have also emerged [3, 4, 6, 10, 15, 16, 26, 29, 32, 35]. All these methods compress efficiently the pre-computed test sets and have very short test application times. The test data volumes can be further reduced by using test set embedding methods [11, 12, 14, 25, 34]. However, these methods require very long test sequences. To alleviate this problem, State Skip LFSRs (Linear Feedback Shift Registers) were recently proposed [36]. State Skip LFSRs retain the high-compression advantage of LFSR-based test set embedding, shortening at the same time the test sequences dramatically. In this way, they bridge the gap between test data compression and test set embedding, rendering the later a very attractive approach for testing IP cores.

Even though the test sequences of [36] are very short compared to other test set embedding techniques, they are still much longer than those of test data compression methods. Additionally, the test-sequence reduction potential of State Skip LFSRs cannot be fully exploited by the decompression architecture proposed in [36], when multiple IP cores in a SoC should be tested. In this case, the system integrator has to resort to the very expensive solution of using a separate decompressor for each core, so as to minimize the overall test sequence length. On the other hand, if an area efficient solution is required, a single decompressor must be shared among all cores, which however cannot achieve maximum test sequence length reduction.

In this paper, a new test set embedding architecture that is based on Variable-State Skip LFSRs (VSS-LFSRs) is presented. The proposed architecture offers very short test sequences (very close to the test sequences of test data compression methods), with significantly smaller test data volumes than those of the latter methods. VSS LFSRs achieve much greater test-sequence-length reduction compared to State Skip LFSRs, since they can perform jumps of variable lengths in the normal LFSR state sequences. Moreover, the proposed architecture is very flexible and can fully exploit the State Skip property in the case of testing multiple IP cores in a SoC. Finally, the area cost of the decompressor is comparable to that of most state of the art compression schemes.

2. Previous Work and Motivation

Fig. 1 presents the State Skip LFSR reseeding architecture proposed in [36]. The LFSR is loaded from the ATE with an n-bit seed (n is the LFSR size), which is expanded through a phase shifter into L test vectors of $m \cdot r$ bits each (m is the scan-chain volume and r the scan-chain length). The calculation of every seed is done by solving systems of linear equations [17] that are formed according to the following procedure: let the initial state of the LFSR be equal to the set of binary variables a_0, a_1, ..., 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 (i.e., test vector with x bits) corresponds to exactly 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 a system of linear equations is formed. We distinguish two cases: L=1 (classical LFSR reseeding), where each seed is expanded into exactly one test vector, and L>1 (test set embedding with reseeding), where each seed is expanded into more than one test vectors, or in other words, into a window of L test vectors. When L=1, one system of linear

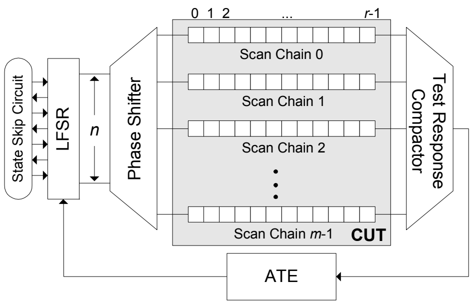

Fig. 1. State Skip LFSR Reseeding Architecture

equations is formed for every test cube, whereas when $L > 1$, we have L systems for each test cube.

Much better compression can be achieved when, during encoding, we can choose among L possible systems instead of one for every test cube. The reason is that the binary variables in the initial state of the LFSR are better exploited due to the greater solution space that vector-windows offer, and thus more test cubes can by encoded by every seed. Indeed, by using the efficient seed selection algorithm of [14], and by increasing L from 1 to 500, a compression improvement of up to 77.3% is achieved. On the other hand, by using windows, the test sequence length increases rapidly (from 100 to 400 times depending on the circuit), compared to the test sequence length for $L=1$. However, the vast majority of the pseudorandom vectors generated by every seed are useless, and only few of them correspond to the required test cubes.

In order to shorten the prohibitively long test sequences without sacrificing the high compression potential of the window-based LFSR reseeding approach, the State Skip LFSRs were introduced in [36]. State Skip LFSRs are normal LFSRs with the addition of a special linear circuit that is called State Skip circuit (see Fig. 1). The State Skip circuit, which consists only of a relatively small number of exclusive-or gates, receives as input the current state of the LFSR and calculates the state of the LFSR k cycles ahead ($k>1$). The State Skip circuit can be easily constructed as follows: the LFSR is initialized at state a_0, $a_1, \dots, a_{n-1}$ and is simulated for k cycles. Then the value of every LFSR cell is a linear expression of the initial variables. The implementation of these linear expressions in hardware constitutes the State Skip circuit. This is why the State Skip circuit can generate, in a single clock cycle, the state of the LFSR k cycles ahead. By using the State Skip circuit instead of the characteristic polynomial (a 2:1 mux is required before every LFSR cell), the LFSR performs successive jumps of constant length k in its state sequence, which is consequently traversed k times faster (k is called speedup factor).

According to [36], the vector-sequence generated by every seed is partitioned into segments of size S. All segments embedding at least one test cube are labeled as useful, whereas the rest are labeled as useless. The useful segments are traversed normally by using the characteristic polynomial of the LFSR, whereas the useless ones are traversed fast (k times faster than useful segments) by using the State Skip linear circuit. Note that test quality is not compromised, because all test cubes are embedded into the useful segments. However, since most of the segments are useless, the major portion of the pseudorandom sequence is bypassed and thus the test sequence length is significantly reduced. The efficiency of this architecture strongly depends on segment size S, as well as on the speedup factor k. As it was shown in [36], the test sequences are considerably reduced when the value of S decreases and/or the value of k increases. In the first case the total size of useful segments decreases whereas the total size of useless segments increases (their sum though remains constant), since fewer useless vectors remain in the useful segments (note that each useful segment may also contain some useless pseudorandom vectors, the volume of which depends on size S). Taking into account that a useless segment is generated faster than a useful one (its major portion is skipped), we deduce that the overall test sequence length decreases. In the second case, when k increases, the number of cycles required for the generation of useless segments decreases, and as a result the test sequence is also reduced. As a conclusion, small segments and large speedup factors are preferable.

One limitation of the architecture proposed in [36] is that k must divide exactly the product $S\cdot(r+1)$, which is the number of clock cycles required for the normal generation (i.e., using the characteristic polynomial) of each segment (each vector requires r clock cycles for loading the scan chains plus one capture cycle). If this condition is satisfied then a whole segment is traversed by using the State Skip circuit for exactly $S\cdot(r+1)/k$ successive clock cycles. However, since small segments are preferable (usually in the range [2, 5]), the maximum value of k that divides exactly the product $S\cdot(r+1)$ is bounded by the value of r and thus a large speedup factor may not be possible. This limitation has an even more serious effect when multiple IP cores should be tested in a SoC. In this case, it is almost certain that the value of r will be different for every core. Then the only way to develop a single State Skip LFSR for testing all cores is to adjust the values of S and k, so as to satisfy the above condition for every core, which means that k must divide S exactly. As a result, it is impossible to select a small value for S and, at the same time, a large value of k. On the contrary, the values of both k and S will be either large or small. In the first case, every useful segment will contain many useless pseudorandom vectors and thus the total number of cycles required for its generation will increase. In the second case, the small speedup factor k will not be able to drastically shorten the time required for the generation of the useless segments. Both scenarios negatively affect the performance of the proposed method.

In this paper we propose a very efficient decompression architecture which utilizes Variable-State Skip LFSRs, i.e. State Skip LFSRs which can perform jumps of variable lengths. Specifically, the State Skip circuit implements two speedup factors, one small (k) and one large (K). The large factor is used for minimizing the test time required for traversing large useless parts of the pseudorandom sequences, whereas the small factor is used for smaller useless parts. Even though two speedup factors are implemented, the proposed method does not need much more hardware than [36], since the speedup factors are properly selected so as to keep the overall overhead low. On the other hand, the test sequences are reduced considerably compared to [36]. Another significant advantage of the proposed method is that the values of k and K are independent of segment size S. This enables the designer to fully exploit the "test time-hardware overhead" trade-off for single cores, as well as to achieve maximum test-sequence-length reduction for every core that is tested by a common decompressor in a SoC.

3. Proposed Scheme

After seed-calculation (i.e., test cube encoding in pseudo-random-vector-windows), which is performed according to [14], every L vector-window of a seed is partitioned into L/S segments of S vectors each. The seed calculation algorithm guarantees that every test cube will be embedded in at least one segment of one of the seeds. However, many test cubes consist of a small number of specified bits and thus they are fortuitously embedded in more than one segment of one or more seeds. For that reason, a greedy segment-selection process is applied for selecting a small number of segments that embed all test cubes. The selected segments are labeled as useful and the rest as useless. Then, the seeds are grouped according to the number of useful segments that their windows contain, and they 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. By applying this grouping, the generation of the vector-window of each seed terminates right after the generation of the last useful segment.

According to the proposed method, the test vectors of all useful segments are generated in Normal Mode (i.e. the LFSR generates the vectors of the segment by using its characteristic polynomial) and they are applied to the core under test. On the other hand, all useless segments are traversed in the Variable State Skip mode, i.e., the LFSR operates using the Variable State Skip circuit, and performs successive jumps of length k (k-mode) or K (K-mode).

Let A be the number of useless segments between two useful segments S_i, S_j ($j=i+A+1$). Then the LFSR requires $C=A\cdot S\cdot(r+1)$ cycles for traversing these useless segments in Normal mode. By using the Variable State Skip circuit, the proposed method traverses these A segments much faster: at first K-mode is used (the LFSR performs jumps of length K) for $C_1=\lfloor(A\cdot S\cdot(r+1))/K\rfloor$ successive cycles. We then switch to k-mode (the LFSR performs jumps of length k) for $C_2=\lfloor(A\cdot S\cdot(r+1)-C_1\cdot K)/k\rfloor$ cycles, and finally Normal mode is used for $C_3=A\cdot S\cdot(r+1)-C_1\cdot K-C_2\cdot k$ cycles (note that each one of the C_1, C_2 and C_3 cycles corresponds to a jump of length K, k and 1 respectively, and thus $C= C_1\cdot K+C_2\cdot k+C_3$). In other words, K–mode is used for traversing the major part of the useless segments, k-mode is used for the remaining part which cannot be traversed in K–mode (its length is smaller than K), and finally Normal mode is used for the last part that cannot be traversed in k–mode (its length is smaller than k). Therefore, instead of C cycles, only $C_1+C_2+C_3$ cycles, are required for traversing the useless segments. For example if K=100, k=15, A=20, S=6 and r=20, we have C=2520, C_1=25, C_2=1, and C_3=5. Consequently 31 cycles are required instead of 2520 cycles. The calculation of C is done during the generation of the test vectors of useful segment S_i that precedes the A useless segments. 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 until the next useful one is found. For every useless segment found, its size in cycles [i.e., $S\cdot(r+1)$] is added to a counter. Consequently, when the next useful segment is found, this counter contains the number of cycles C for traversing the A useless segments. Then the reverse process begins and this counter is decreased initially by K, then by k and finally by 1, until it reaches 0. Specifically, while the value of the counter is greater than or equal to K, the K-mode is used and the counter is decreased by K. This is repeated until the value of the counter drops below K. Then, while the value of the counter is greater than or equal to k, the k-mode is used and the counter is decreased by k. This is repeated until the value of the counter drops below k. Finally, if the value of the counter is not zero, the Normal mode is used and the counter decreases by one. This is repeated until the counter reaches 0, which indicates that all useless segments have been traversed.

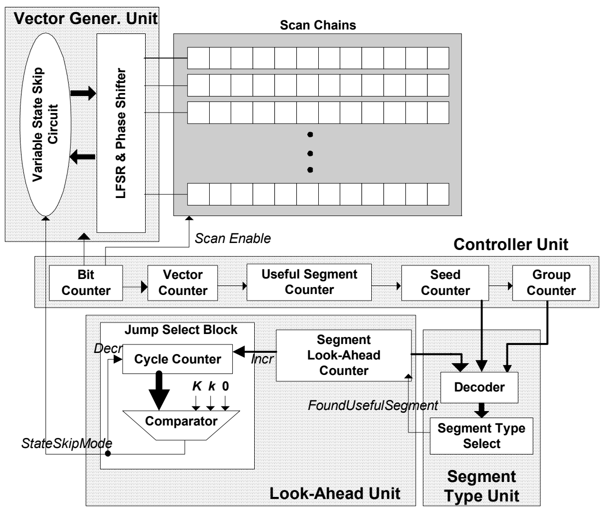

Fig. 2. Proposed Architecture

The proposed architecture is shown in Fig. 2. It consists of four main units:

a) The Vector Generation unit, which comprises the LFSR, the Phase Shifter and the Variable State Skip circuit. This unit is responsible for loading the scan chains with the test vectors, and for traversing the useless pseudorandom segments by performing successive jumps of length K or k. The operation of this unit is controlled by the signal *StateSkipMode*.
b) The Segment Type unit, which consists of a decoder and a combinational logic block, the Segment Type Select block, which determines whether a segment is useful or not. The decoder is used for reducing the hardware overhead of the common decompressor in a multi-core environment.
c) The Look-Ahead unit, which consists of the Segment Look-Ahead counter and the Jump Select block. During the generation of a useful segment, it locates the next useful segment and when the generation of the current useful segment is complete, it controls the Vector Generation unit so as to traverse the intermediate useless segments in State Skip mode.
d) The Controller, which comprises various counters that control the operation of the whole decompressor.

Initially the LFSR is loaded from the ATE with the first seed of the seed-group 1 (note that every seed of group 1 includes only one useful segment). Group counter is initialized to 1, Useful Segment counter is loaded with the value of Group counter and the rest counters are reset to 0. The generation of the test vectors of the first segment (which is always useful according to the seed selection algorithm) begins. The loading of the test vectors into the scan chains is controlled by the Bit and Vector counters. During the generation of every useful segment, Normal Mode is used. When all test vectors of the segment have been applied, Useful Segment counter decreases by one and, for the case at hand, reaches 0. Every time Useful Segment counter reaches 0, all useful segments of the current seed have been generated and thus the decompressor can proceed with the next seed. Thus, Seed counter increases by one in order to enable the loading of the next seed from the ATE, and all counters except for the Seed and Group counters are initialized again in the same way as before. This is repeated for every seed of the same group.

When all seeds of the current group have been processed, Group counter increases by one so as to enable the loading of the next group of seeds (group 2 - it includes seeds with 2 use-

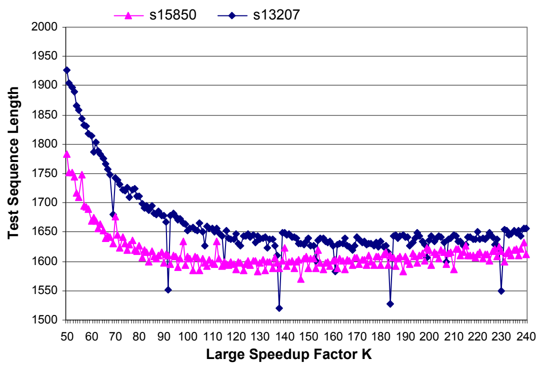

Fig. 3. Test Sequence Length Using Speedup Factor *K*

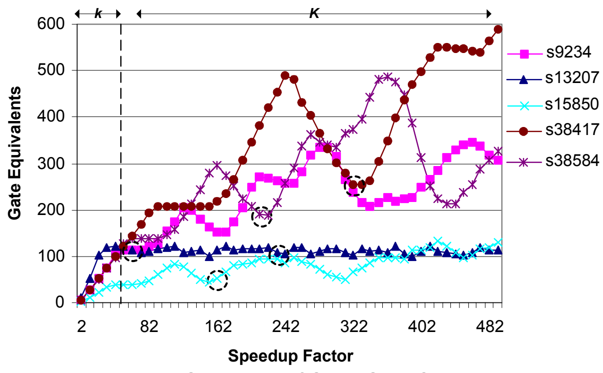

Fig. 4. Overhead of State Skip Circuits

ful segments). Note that the value of Group counter is equal to the number of useful segments of every seed in the group (this is why Useful Segment counter is loaded with the value of group counter every time a new seed is loaded in the LFSR).

The Look-Ahead unit is also activated at the beginning of seed-group 2. This unit operates concurrently with the generation of every useful segment of each seed and determines the number of cycles between the last state of the current useful segment and the first state of the next useful one (i.e., it calculates number *C*). The Segment Look-Ahead counter is reset at the beginning of every seed. The values of this counter are in the range [0, *L*/*S*) and correspond to the *L*/*S* segments of every window. During the generation of any useful segment, this counter increases by one at every clock cycle and checks the segments that follow the useful segment that is currently generated, until the next useful one is found. This is indicated by the *FoundUsefulSegment* signal of the Segment Type Select unit, which determines if a segment is useful or not by examining the values of the Segment Look-Ahead, Seed and Group counters. Every time the Segment Look-Ahead counter is increased by one, the Cycle counter inside the Jump Select unit is increased by $S{\cdot}(r{+}1)$. Thus, when the Segment Look-Ahead counter stops after having found the next useful segment, Cycle counter contains the number *C*.

When the application of the test vectors of the current useful segment completes, and if the next segment is not a useful one, the operation mode of the LFSR is switched to the Variable State Skip mode. 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*. When the value of Cycle counter drops below *K*, the above process continues with comparisons against *k*. While the Cycle counter value is greater than or equal to *k*, the *k*-mode is used and the counter is decremented by *k*. When Cycle counter drops below *k*, the above process continues with comparisons against 0. While its value is larger than 0, the Normal mode is used and the counter is decremented by 1 (note that in this case, the LFSR simply passes through the states, i.e., no vector is loaded in the scan chains).

Note that if the next useful segment has been found before the completion of the generation of the current useful segment, the Look-Ahead unit stops and waits until generation is complete. In the case that the generation of the current useful segment has finished and the next useful segment has not yet been determined, the above described state skipping process begins, provided that the value of Cycle counter is greater than or equal to *K*. Every time the value of Cycle counter is greater than or equal to *K*, the *StateSkipMode* signal is set properly so as the LFSR to operate in *K*-mode. If the value of Cycle counter is smaller than *K* but equal or greater than *k*, then a *k*-jump is temporarily avoided (the Vector Generation unit stalls), since there is the possibility that Cycle counter will increase above *K* in the next cycles (the Segment Look-Ahead counter continues to increase), and thus a large jump will be performed. Note that Cycle counter has to be increased by $S{\cdot}(r{+}1)$ (due to the increase of the Segment Look-Ahead counter), and, at the same time, it has to be decreased by *K* (due to a jump of length *K*). For that reason, Cycle counter is either decreased by $K{-}S{\cdot}(r{+}1)$ [if $K{\geq}S{\cdot}(r{+}1)$] or it is increased by $S{\cdot}(r{+}1){-}K$ [if $K{<}S{\cdot}(r{+}1)$]. In this way, the two operations are overlapped and the generation of the test vectors terminates faster.

4. Evaluation and Comparisons

The proposed method was implemented in C programming language and experiments were conducted on a Pentium PC for the larger ISCAS '89 benchmark circuits, assuming 32 scan chains for each one of them. Uncompacted test sets generated by Atalanta [23] that offer complete stuck-at fault coverage (100%) were used. The running time of the proposed method is only a few minutes. In the sequel, the test sequence length results are reported as number of vectors and the test data volume results are reported as number of bits.

In our first set of experiments we study the effect of the large speedup factor *K* on the test sequence length. We provide results for the s13207 and s15850 benchmark circuits (the rest circuits exhibit similar behavior). Fig 3. presents the test sequence length (TSL) results obtained by varying *K* in the range [50, 240], with k=15. The segment size used in all cases was S=2 and the window size was L=200. It is obvious that when *K* increases, TSL reduces. The achieved reduction saturates when *K* increases above a value, which depends on the core under test. We observe that for some values of *K*, TSL exhibits large instant drops. The reason is that the corresponding values of *K* happen to divide exactly the number of cycles required for generating normally a number of, let say *m*, segments. Consequently, every time the number of useless segments between two useful ones is a multiple of *m*, these segments are traversed by using only the large speedup factor *K* (i.e., without using the small speedup factor and the LFSR polynomial).

In the next set of experiments, we study the area overhead of the State Skip circuits for various speedup factors. Fig. 4 presents the results for the examined benchmark circuits (the LFSR sizes can be found in Table 1). The area overhead of the State Skip circuits is reported in gate equivalents (one gate equivalent corresponds to a 2-input nand gate). Note that in Fig. 4 every examined State Skip circuit implements one speedup factor. The results in Fig. 4 are partitioned into two regions separated by the dashed line: the left region corresponds to the

Table 1. Comparisons among the Various LFSR Reseeding Methods

	LFSR Size	*Clas. Reseeed. (L=1)* TDV	TSL	*L*=200 *K/k*	TDV	TSL Norm.	SS	VSS	Impr. (%) over Norm.	SS	*L*=500 *K/k*	TDV	TSL Norm.	SS	VSS	Impr. (%) over Norm.	SS
s9234	44	10692	243	54/18	7128	32400	1784	1465	95.5%	17.9%	165/15	6688	76000	3055	2455	96.8%	19.6%
s13207	24	8856	369	230/46	3816	31800	1756	1180	96.3%	32.8%	230/46	2688	56000	2701	1440	97.4%	46.7%
s15850	39	11622	298	168/42	6669	34200	1740	1091	96.8%	37.3%	168/42	6201	79500	2791	1470	98.2%	47.3%
s38417	85	58225	685	318/53	48110	113200	13113	3026	97.3%	76.9%	318/53	47005	276500	21865	3230	98.8%	85.2%
s38584	56	22680	405	235/47	7056	25200	6639	1935	92.3%	70.9%	235/10	5152	46000	9054	1958	95.7%	78.4%

Table 2. Comparisons of the Proposed Approach against Test Data Compression Methods

	[1]			[20]	[24]	[39]		[26]	[32]	[21]		[33]		Clas. LFSR Reseed. (*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	1465	**7128**
s13207	229	12798	266	11859	10484	10810	236	20988	74423	266	14307	242	26004	369	8856	1180	**3816**
s15850	244	15480	269	12663	11411	12405	126	25140	26021	226	15067	306	32226	298	11622	1091	**6669**
s38417	376	37020	376	36430	**32152**	32154	99	85225	45003	376	49001	854	89132	685	58225	3026	48110
s38584	296	31574	296	30355	31152	31000	136	57120	73464	296	28994	599	63232	405	22680	1935	**7056**

State Skip circuits implementing the small speedup factors *k*, whereas the right region corresponds to the State Skip circuits implementing the large speedup factors *K*. By looking at the left region (small speedup factors *k*) we can see that the overhead of the State Skip circuit is low (below 120 gate equivalents in all cases) and increases almost linearly with *k*. By looking at the right region (large speedup factors *K*) we see that the overhead is higher compared to the results of the left region, but interestingly it exhibits significant fluctuations (i.e., ups and downs). According to our experiments, this behavior is caused by exactly the same fluctuations in the mean number of binary variables per LFSR cell, during symbolic LFSR simulation. The designer can take advantage of this property and choose a high speedup factor that is in the neighborhood of a local minimum so as to achieve high performance and low area overhead at the same time. The large speedup factors that were chosen for each benchmark circuit are circled in Fig. 4. Even though the selected values of *K* are very high (between 54 and 318) the hardware overhead of the corresponding State Skip circuit is between 50 and 250 gate equivalents, which is rather small. Note that the overall area overhead of the Variable State Skip circuits will be higher, due to the addition of the State Skip circuit implementing the small speedup factor *k*.

Table 1 presents the test data volumes (TDV) and test sequence lengths (TSL) achieved by: a) the classical LFSR reseeding method, which does not use vector-windows (*L*=1), and b) the window-based test set embedding approaches with reseeding that utilize: i) normal LFSRs (labeled "Norm."), ii) State Skip LFSRs (labeled "SS") [36], and iii) Variable State Skip LFSRs (labeled "VSS"). The window sizes for the last three approaches were *L*=200 and 500. The first two columns present the circuit name and the size of the utilized LFSR. Columns 3 and 4 present the TDVs and TSLs of the classical reseeding method. Columns 5-11 and 12-18 present the TDV-TSL results of the rest three methods for *L*=200 and *L*=500 (note that the test data volumes of these methods are the same and thus they are reported under the same column). Columns labeled "*K/k*" present the values of the large/small speedup factors of the VSS LFSRs that were used in each case [the large speedup factors are near to local area-minimums (see Fig. 4) in order to keep the hardware overhead low]. Columns labeled "Impr. (%) over Norm." and "Impr. (%) over SS" report the TSL reduction percentage achieved by VSS LFSRs over normal and SS LFSRs respectively. It is obvious that when vector-windows are used, the TDV drops significantly compared to the classical LFSR resseding. However, at the same time the test sequence length increases rapidly. The test sequence length is greatly improved by using SS-LFSRs but it still remains high, especially in the case of large benchmarks. On the other hand, VSS LFSRs offer short test sequences in all cases. Moreover, their test-sequence-reduction ability is only slightly affected by the utilized window size, giving the designer the opportunity to increase the compression as much as possible with negligible test-sequence overhead.

Table 2 compares the proposed method against the most efficient test data compression methods which are suitable for IP cores of unknown structure. Comparisons against the test set embedding techniques of [12, 14, 25] are omitted, since the method proposed in [36] (which is included in Table 1) performs better than these techniques. In all but one case (s38417) the proposed method performs better than the compared test data compression methods, in terms of test data volume. The test sequences of the proposed method are longer but close to those of the test data compression methods. We note that for the s38417 circuit, the test set generated by Atalanta consists of a very high volume of specified bits (93123 specified bits), which leads to the relatively reduced TDV performance in this case.

We next present the hardware overhead results of the proposed method for the case of s13207 (the results for the rest circuits are similar, since excluding the LFSR and the Segment Type Select unit, the hardware overhead of the remaining decompressor units does not depend on the test set). The overhead of the Variable State Skip circuit for *k*=46 and *K*=230 is equal to 203 gate equivalents. For the same circuit, the total overhead of the LFSR, Phase Shifter, counters, control and decoding logic, for *L*=200 and *S*=5, is 627 gate equivalents. All the above units need to be implemented only once in a SoC, where a common decompressor is used for testing different cores. This makes their overall cost much smaller. The only unit that has to be implemented separately for every core is the Segment Type Select unit, whose hardware overhead for s13207 is between 44 and 262 gate equivalents, for $50 \leq L \leq 500$ and $2 \leq S \leq 50$. Note that, excluding the Variable State Skip circuit, which is larger than the State Skip circuit of [36], the area overhead of the rest of the proposed decompressor is close to that of [36].

In our last experiment we applied the proposed approach and that of [36] to a hypothetical multi-core SoC consisting of the 5 larger ISCAS '89 benchmarks. In both cases a common decompressor was used and only the Segment Type Select unit was implemented separately for each core. Table 3 presents the TSL

Table 3. VSS vs. SS LFSRs for multiple cores

	SS-LFSR			VSS-LFSR				TSL
S	***k***	**TSL**	**Area**	***K***	***k***	**TSL**	**Area**	**Impr. (%)**
2	2	53471	9%	318	21	8511	10,5%	84.1%
5	5	31358	7,7%	159	5	15682	8,8%	50.0%
10	10	33736	6,6%	18	10	26731	7,8%	20.8%

and area overhead results for three segment sizes, 2, 5 and 10, and for LFSR size=85. The area overhead is reported as the percentage of the area of the decompresssor to the total area occupied by the 5 cores. It is obvious that the TSL gain is very high compared to [36] and reaches 84.1%. On the other hand, the hardware overhead increase is very small (between 1-1.5% of the total area of the 5 cores). Therefore, we conclude that the proposed scheme exploits better than [36] the State Skip property, especially when it is applied to multiple cores in a SoC.

5. Conclusions

An advanced version of the State Skip LFSRs, which were recently proposed in [36], was introduced. Variable State Skip LFSRs reduce considerably the test application time by utilizing two speedup factors, a large and a small one. With proper selection of these factors the area of the proposed scheme is only a little higher than that of [36]. In the same time great reduction of the test application time is achieved (up to 85.2%) compared to [36]. Moreover, the proposed architecture does not suffer from the limitations of [36] and thus it can be efficiently used for testing multiple IP cores in a SoC.

References

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

[2] 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

[3] A. Chandra and K. Chakrabarty, "System-on-a-Chip Test-Data Compression and Decompression Architectures Based on Golomb Codes", IEEE Trans. CAD, vol. 20, pp. 355-368, March 2001.

[4] A. Chandra, K. Chakrabarty, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length codes" *IEEE Trans. on Comp*, vol. 52, pp 1076–1088, Aug 2003.

[5] R. Dorsch and H. -J. Wunderlich, "Tailoring ATPG for Embedded Testing", in Proc. ITC, 2001, pp. 530-537.

[6] 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.

[7] 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.

[8] S. Hellebrand, H. Liang and H. -J. Wunderlich, "A mixed mode BIST scheme based on reseeding of folding counters", J. of Electronic Testing: Theory and Applications, vol. 17, pp. 341 - 349, June 2001.

[9] A. Jas and N. A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based designs", in Proc. ITC, 1998, pp. 458-464.

[10] 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.

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

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

[13] 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.

[14] 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. Dig. ETS, 2005, pp. 147-150.

[15] 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.

[16] 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.

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

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

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

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

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

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

[23] 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.

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

[25] 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.

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

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

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

[29] 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.

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

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

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

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

[34] 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.

[35] 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.

[36] V. Tenentes, et. al. "State Skip LFSRs: Bridging the Gap between Test Data Compression and Test Set Embedding for IP Cores", to appear in DATE 2008 (URL: http://www.cs.uoi.gr/~kabousia/Paper.htm)

[37] E. Volkerink, A. Khoche, S. Mitra, "Packet-Based Input Test Data Compression Techniques", Proc. ITC, Oct. 2002, pp. 154 – 163.

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

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

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

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