Efficient Seed Utilization for Reseeding based Compression*

Erik H. Volkerink[1,2], Subhasish Mitra[3]

[1]Center for Reliable Computing (CRC)
Stanford University, Stanford, CA

[2]Agilent Laboratories
Palo Alto, CA

[3] Intel Corporation
Sacramento, CA

Abstract

One of the main drawbacks of the conventional reseeding architecture is the limited seed efficiency due to the variance in the number of specified bits per vector. This paper proposes a new LFSR reseeding architecture that essentially solves this problem, resulting in a significant compression ratio. The compression ratio is very close to the entropy in terms of #total bits / #specified bits. The technique is applied on two industrial designs resulting in a compression ratio of 33x (with a seed efficiency of 95%), whereas the conventional reseeding architecture resulted in only 2x (with a seed efficiency of only 50%).

Keywords: Compression, LFSR, Reseeding, BIST

1 Introduction

As the design complexity increases so does the test data volume [ITRS99]. This increase in test data volume results in the following critical problems:

- *Limited vector memory on ATE:* More test data means that the ATE will have to support more vector memory. However, depending on the requirements, such an ATE may not be available or may be very expensive. For this reason, vector sets are often truncated, resulting in a reduced product quality [Hetheringten99].
- *Long upload time:* Large test data volume can result in a long time to load/reload test vectors into the ATE memory, often ranging from several tens of minutes to hours [Hetheringten99] [Ishida98]. Since the ATE remains idle during the loading and reloading of test vectors, the ATE utilization is reduced. This can significantly increase the cost of test.
- *Limited I/O bandwidth:* The I/O bandwidth is defined as the number of input channels multiplied with the frequency capability of these channels. Even if the ATE has sufficient memory to store all the test data bits, the transmission of test data to and from the Device-Under-Test (DUT) can result in an unacceptable long test time due to the limited bandwidth between the ATE and DUT. The bandwidth could be limited due to the limited channel frequency or due to a lack of a sufficient number of scan channels on the ATE or the DUT. In fact, in [Khoche02] it is shown that with current test methods, the test time will increase almost exponentially for the device complexity projections in the ITRS'99 roadmap [ITRS 99]. This significantly impacts the cost of test [Volkerink02a]

This paper focuses on using LFSR reseeding for input test data compression. The paper proposes a new reseeding architecture, based on the conventional reseeding architecture [Koenemann91]. The new architecture enables the use of a seed for a variable number of tester cycles, such that each seed covers the maximum number of specified bits, resulting in a significant compression ratio. The compression technique is very easy to implement and the hardware overhead is less than the conventional LFSR reseeding hardware overhead. The compression technique is evaluated on two real industrial designs resulting in a compression ratio of 33x (and a seed efficiency of 95%), whereas the conventional reseeding architecture resulted in only 2x (and a seed efficiency of only 50%). The results are compared with other techniques.

This paper is organized as follows: Section 2 describes previous work on test data compression techniques. Section 3 presents conventional LFSR reseeding architectures. Section 4 defines the seed efficiency metric. Section 5 describes the proposed LFSR reseeding architecture that essentially solves the limited seed efficiency problem. Section 6 describes experimental results on industrial designs and compares the technique with the conventional reseeding technique. The paper concludes with the conclusions in Section 7.

2 Previous Work

Recently a lot of research has been done in both output test data compression and input test data compression.

The output test data can be compressed by using lossy compression techniques, like for example a MISR [Barnhart01] or an X-compactor [Mitra02].

This paper focuses on input test data compression. For input test data compression, several lossless compression techniques have been proposed. These techniques include using run-length codes [Chandra00],

* This work was done at CRC, Stanford University

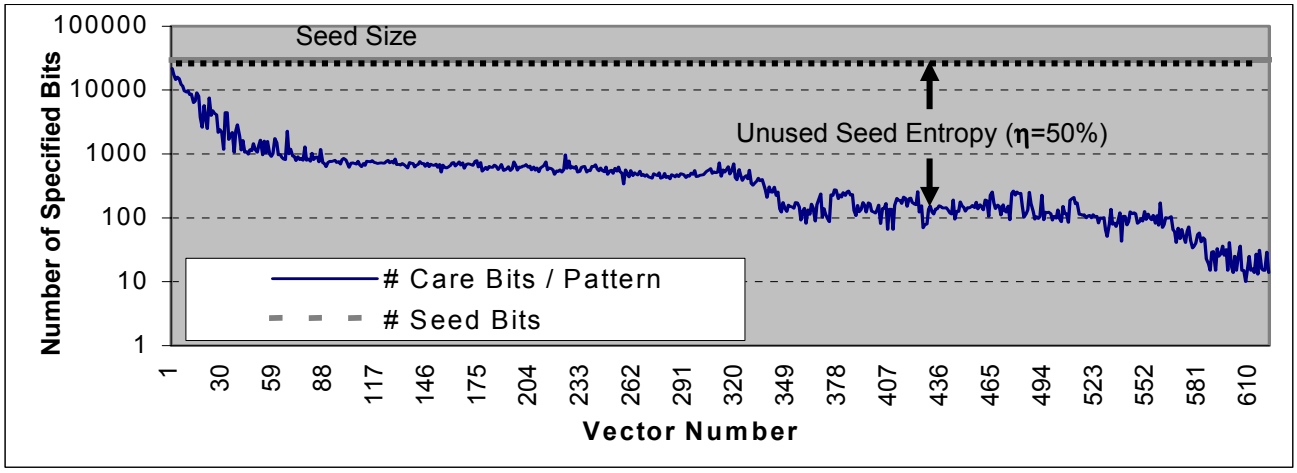

Figure 1: The Number of Specified Bits per Vector as Function of the Vector Number (for ASIC 1)

[Chandra01], and [Hellebrand02], using statistical codes [Jas99], using combinational linear decompressions [Reddy02] and [Bayraktoroglu01], using folding counters [Hellebrand00], and reusing data [Dorsch01]. Another group of compression techniques focus on pseudorandom sequences, like for example logic BIST [Hetheringten99], mapping logic [Touba96], bit flipping [Kiefer00], and mixing pseudo-random bits with ATPG bits [Khoche02] and [Volkerink02b].

This paper focuses on improving the conventional LFSR reseeding techniques [Koenemann91]. The maximum number of specified bits n_{Care_MAX} in a vector across all vectors determines the seed size, n_{Seed_Bits}.

The seed size of the conventional reseeding approach is limited by linear dependencies within the LFSR. If $n_{Seed\_Bits} = n_{Care\_MAX} + 20$ flipflops are used, the probability of not finding a seed due to linear dependencies in the LFSR reduces to 10^{-6}. [Koenemann91]. By using multiple polynomials, only $n_{Seed\_Bits} = n_{Care\_LFSR} + 4$ flipflops are required [Hellebrand95]. Several techniques improve LFSR reseeding, like for example techniques using variable length seeds [Zacharia95] [Rajski98], using two dimensional compression combining LFSR reseeding and folding counters [Liang01], using partial dynamic LFSR reseeding [Krishna01], or using LFSR reseeding with seed compression [Krishna02].

In the conventional LFSR reseeding technique, the number of specified bits per vector vary across all vectors, see Fig. 1, whereas the maximum number of specified bits (n_{Care_MAX}) determine the seed size. Therefore, even if the vector includes only a few specified bits, the vector is still encoded in a $n_{Care\_LFSR}+20$ bits seed. In other words, the information entropy of the seed is not fully utilized.

This paper proposes a new LFSR reseeding architecture that essentially solves this problem. The resulting compression ratio is very close to the entropy in terms of #total bits / #specified bits. The next section will explain the conventional architecture in more details.

3 Conventional LFSR reseeding

3.1 Basic Architecture

In the conventional LFSR based compression technique, using the STUMPS architecture [Bardell87], the total number of flipflops ($n_{FlipFlops}$) are concatenated into a number of (small) scan chains (n_{Scan_Chains}), see Fig 2. A *bit slice* is defined as the bits across all scan chains at a certain tester cycle. During each tester cycle, a bit slice is shifted from the LFSR through the phase shifter into the scan chains. The phase shifter aims at de-correlating the data in the different scan chains and enabling more scan chains. A good overview of phase shifter designs can be found in [Rajski98].

The number of LFSR inputs (n_{LFSR_Inputs}) are chosen such that the number of tester cycles required to shift in the seed is **equal** to the number of tester cycles required to shift in the vector (i.e. using the seed) using all small scan chains, see equation 1.

$$m_{\#ShiftSeed} = m_{\#UseSeed} \Leftrightarrow \frac{n_{Seed\_Bits}}{n_{LFSR\_Inputs}} = \frac{n_{FlipFlops}}{n_{Scan\_chains}} \quad [1]$$

Therefore, it is possible to get a continuous flow, in which a seed is decoded in a vector and shifted into the scan chains, while the next seed is shifted into a seed buffer. If all of the next seed bits are shifted into the seed buffer, then its content is copied to the LFSR flipflops (also called *reseeding*) and directly the next vector can be decoded and shifted into the scan chains.

For example, imagine a core with 10,000 flipflops with a maximum specified bit percentage of 5%. Therefore, the required LFSR size is about 500 flipflops (≈5% * 10k + 20). If 5 input channels are available to shift the 500 bit seed into the seed buffer, then shifting in the seed will take 100 tester cycles. Hence, the scan chain length needs to be 100 bits. Since the total number of flipflops

is equal to 10,000, there will be 100 scan chains of each 100 bits (see equation 2). This architecture is visualized in Fig. 2.

3.2 Serial Shift Mode

Introducing a serial shift mode can reduce the variance in the number of specified bits in vectors that need to be generated by the LFSR. In the serial shift mode all scan chains are reconfigured as one (or five) long scan chains (using multiplexers). Moreover, deterministic vectors can be directly shifted into the scan chains, i.e. the LFSR and the phaseshifter can be bypassed. The seed size can now be reduced, because vectors that can't be generated using the LFSR, can be applied to the CUT using the serial shift mode. Note that the reduced seed size also reduces the number of LFSR flipflops, i.e. the area overhead of the LFSR.

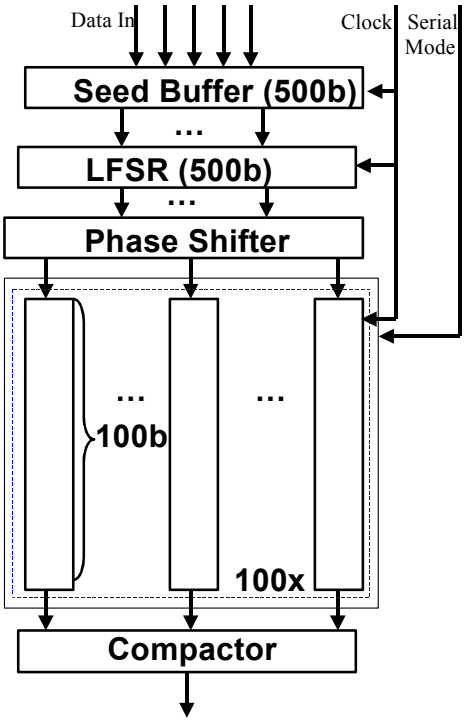

Figure 2: Conventional Reseeding Architecture

The next section defines two metrics that will be used to evaluate the conventional and the proposed compression architectures.

4 Evaluation Metrics

4.1 Seed Efficiency

The seed efficiency for a certain seed is defined in equation 2.

$$\eta = \frac{1}{q \bullet n_{Care\_LFSR}} \sum_{i=1}^{q} (n_{Care,i}) \qquad [2]$$

In which, $n_{care,i}$ represents the number of specified bits covered by seed number i. The parameter q represents the number of vectors and n_{Care_LFSR} represents the maximum number of specified bits that can be covered by one seed.

4.2 Compression Ratio

The overall specified bit percentage α is defined as being the total number of specified bits divided by the total number of bits. Therefore, the overall compression ratio can be calculated as follows:

$$\gamma_{Overall} = \frac{\#UncompressedBits}{\#CompressedBits} = \frac{\eta_{Overall}}{\alpha_{Overall}} \qquad [3]$$

The maximum possible (entropy) compression ratio, without including accidental coverage and potential regularity in the seeds, equals $1/\alpha$ *(i.e.* γ*=100%)*.

The next section introduces the new LFSR reseeding architecture. The architecture increases the seed efficiency to almost 100% and therefore enables a compression ratio that almost reaches the entropy compression ratio.

5 Decoupling Seeds with Vectors

The problem of the limited seed efficiency can essentially be *solved* by calculating the seed such that it covers the **maximum number of specified bits** instead of calculating the seed such that it covers **one vector** (i.e. 100 cycles for 100 scan chains). The main challenge here is dealing with the resulting varying number of tester cycles that a seed is used, while maintaining a continuous flow of new seeds (see equation 2). Fig. 3 shows the proposed architecture that tackles this challenge.

As before, it is assumed that maximally 5% of the bits are specified.

Instead of using a seed size of 500 bits, as described in Fig. 2, a seed size of 100 bits is used, i.e. reducing the area impact of the LFSR. A *stall clock input* (and AND gate) replaces the *serial shift mode input* (and the required multiplexers to reconfigure the scan chains), reducing the area overhead further.

The stall clock signal makes it possible to shift in seed bits into the seed buffer, without clocking the LFSR and scan chains. To keep the same number of tester cycles per seed, the number of seed input channels is reduced from 5 input channels to only 1 input channel.

Note that the reduction in the number of input channels can be used to get an additional test time reduction by replicating the decompression architecture multiple times, i.e. splitting the number of flipflops in smaller scan chains.

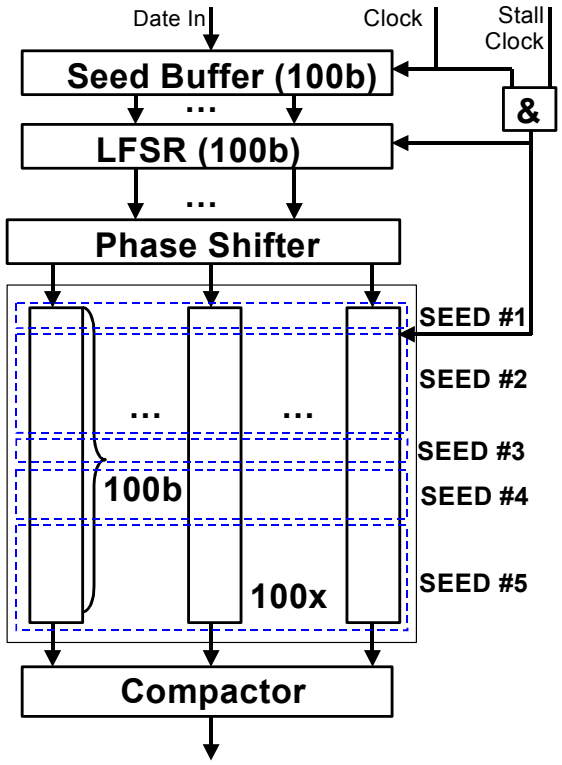

Figure 3: Architecture for Input Channel Reduction

The proposed architecture will be explained in the following examples, by comparing it with the conventional architecture.

Example 5.2.1 – Number of specified bits in the vector equals the conventional seed size

For example, the architecture needs to generate a vector that includes the maximum number of specified bits, i.e. 500 bits. Table 1 gives the different seeds that are required to cover the 500 specified bits of this specific vector.

Table 1: Seed vs. Bit Slice Example 5.2.1

Seed number	Number of covered bit slices	Total number of bits	Total number of specified bits
1	9 slices	1.9kb	100b
2	30 slices	900b	100b
3	10 slices	1kb	100b
4	22 slices	2.2kb	100b
5	29 slices	2.5kb	100b
Total	100 slices	**10kb**	**500b**

The table shows for example that the first seed covers 9 bit slices of the vector, i.e. 900 vector bits. Of these 900 vector bits, 100 bits are specified. To cover the whole vector, 5 different seeds are required (see Fig. 3).

The first seed is shifted into the LFSR seed buffer and the LFSR will decode the seed into the 9 bit slices by shifting its outputs through the phase shifter into all scan chains. This will require 9 clock cycles. During these clock cycles the first 9 bits of the next seed is shifted into the seed buffer. After this, the remaining 100-9=91 bits of the next seed needs to be shifted into the seed buffer. By using the stall clock signal, the next seed can be shifted into the seed buffer, without starting the LFSR decoding process. The total number of input tester cycles required to decode the vector equals 500. Therefore, there is *no advantage* compared to the conventional approach, i.e. which required 100 input test cycles through 5 channels (100*5 = 500 bits). However, the bulk of the vectors includes only a fraction of the maximum number of specified bits (see Fig. 1). The next example will show that for these vectors a significant additional compression ratio can be achieved.

Example 5.2.2 – Number of specified bits equals the new smaller seed size

For example, the architecture needs to generate a vector that includes 100 specified bits, see table 2.

Table 2: Seed vs. Bit Slice Example 5.2.2

Seed number	Number of covered bit slices	Total number of bits	Total number of specified bits
1	100 slices	10kb	100b
Total	100 slices	**10kb**	**100b**

One seed is able to cover these 100 specified bits. The LFSR will decode this seed into the 100 bit slices in 100 tester cycles, during which the next seed is shifted in. In the conventional approach, decoding this pattern would require a 500 bit seed. Therefore, an *additional compression of 5x* on top of the compression ratio of the conventional reseeding architecture is achieved.

Example 5.2.3 – Number of specified bits in a vector exceeds the conventional seed size

For example, the architecture needs to generate a vector that includes 2,000 specified bits. This will require 20 reseedings of 100 bits each. Hence, the total number of tester cycles equals 20*100 = 2,000 bits.

In the conventional scheme this vector could not be generated. Therefore, the vector would be shifted into the flipflops serially using a serial shift in mode (see section 3.2). This would take 10,000 tester cycles (and memory bits). Therefore, the proposed architecture achieves an additional *compression of 5x.*

Example 5.2.4 – Number of specified bits in a vector is smaller than the new seed size

In this case, the seed can be calculated such that bit slices are covered that belong to two or more vectors. For example, consider 5 consecutive vectors with 20 specified bits each. Shifting the seed into the seed buffer will require 100 tester cycles, whereas shifting the 5

consecutive vectors into the scan chains will require 500 tester cycles. Therefore, the input channels will first supply 400 dummy bits (=unutilized input bandwidth) after which the 100 bits for the next seed will be supplied. Hence, there is *no advantage* compared to using 5 different seeds to cover the 5 different patterns (except for the fact that the 400 dummy bits can be generated using an ATE repeat count instruction, potentially resulting in an additional reduction in memory requirements).

However, note that in the conventional architecture the maximum number of specified bits in *one vector (i.e. 100 bit slices)* determines the seed size. In the proposed architecture the seed size is determined by the maximum number of specified bits in only *one bit slice*. Therefore, the number of phase shifter outputs may potentially be increased. For example, if 5% of the bits are specified, then a bit slice of 2,000 bits (2,000 * 5% = 100 specified bits) may be supported by a LFSR of 100 bits. Note, however, that the maximum specified bit percentage of vectors that can be generated (see example 5.2.3) is limited by the maximum specified bit percentage of a bit slice, i.e. *the seed size / the number of scan chains [%] (boundary).*

Table 3: Overview of the Different Vector Types

Vector Type	Condition (n = specified bits / vector)	Description
I	*n < new seed size*	Multiple vectors per seed. Compression. Requires an increased number of scan chains.
II	*new seed size < n < conventional seed size*	Multiple seeds per vector. Compression.
III	*conventional seed size < n < boundary*	Multiple seeds per vector. Compression. (in conventional scheme: serial shift mode)
IV	*n >boundary*	Serial shift mode. No compression. These vectors should by prevented by designing the architecture correctly

For example, the number of scan chains in Fig. 3 is increased from 100 scan chains (of 100 bits each) to 500 scan chains (of 20 bits each). All previous examples are still valid, only the LFSR and the scan chains may be stalled more often, because complete vectors are shifted into the flipflops in only 20 cycles, whereas shifting in the seed into the seed buffer still requires 100 cycles.

Shifting in the 5 consecutive vectors will require only 20*5=100 tester cycles. Therefore, during the 100 tester cycles required to shift in the next seed into the seed buffer, the 5 consecutive vectors are applied on the CUT.

In the conventional approach applying 5 vectors would require 5*500 = 2,500 bits of memory (and 500 tester cycles). In the proposed approach only one reseeding of 100 bits will be required, i.e. 100 bits of memory (and 100 tester cycles). Therefore, the test *memory requirements* are reduced by *25x*, the *test time* is reduced by *5x*, and the number of *required input channels* was already reduced by *5x.*

Fig. 1 shows that the bulk of the vectors consist of only a few specified bits. Therefore, the situation described in this example, happens frequently.

Note that the *stall clock* data consists of very long sequences of 1s or 0s. These sequences can be easily created using ATE repeat counts. Therefore, the required memory for storing the *stall clock* data is very limited.

Table 3 gives an overview of the different categories of vectors together with how the decompression architecture is able to achieve compression. The table assumes that the seed starts at the beginning of the vector. Note that in practice the seeds cover bit slices that may belong to multiple vectors.

6 Experimental results

6.1 Experiment Setup

The proposed reseeding architecture is applied on two large industrial designs with the characteristics, described in table 4.

Table 4: Design Characteristics of the ASICs

Parameters	ASIC 1	ASIC 2
Design Size	~300k gates	~7M gates
Total Number of F/F	~30k	~105K
Total Number of Scan Chains	10	23
Total number of Collapsed Faults	~300k	~6M
Test Data Volume	~18M	~1.6G

The test vectors for the considered designs were generated using a commercial ATPG tool. The ATPG is performed with maximal dynamic and static compaction, without any constrains on the number of specified bits per vector.

After STIL generation, a PERL script was used to extract the scan input vectors from the STIL output file. A software package has been developed to perform the described compression techniques on the extracted scan input vectors using the idealized linear dependency statistics described in [Koenemann91] and using an idealized phase shifter.

		Single Polynomial [Koenemann91]		Multiple Polynomial [Hellebrand95]	
		Memory Compression Ratio	Seed Efficiency	Memory Compression Ratio	Seed Efficiency
ASIC 1	Conventional reseeding [Koenemann91]	2x	50%	2x	50%
	Reseeding with serial shift mode	8x	70%	9x	71%
	Proposed compression technique	**32x**	**94%**	**33x**	**95%**
ASIC 2	Conventional reseeding [Koenemann91]	-	-	-	-
	Reseeding with serial shift mode				
	Proposed compression technique	-	-	-	-

Table 5: Experimental Results Proposed Compression Techniques Compared to Previous Work

The compression software was executed on a HP 9000/785/J5600 machine with 2 550MHz PA-RISC processors and 2G of physical RAM.

The compression ratio is calculated by dividing the volume of the conventional (uncompressed) random filled vectors with the volume of the total number of required seeds (see equation 3).

6.2 Reseeding using a Seed Size based on the Maximum # Specified Bits per Vector

Fig. 1 gives an overview of the number of specified bits per vector as function of the vector number for ASIC 1. It is clear to see that the number of specified bits in the initial vectors (12,000 specified bits) is significantly higher than the number of specified bits in the remaining vectors (the average is around 200 specified bits).

If the seed size is chosen to be equal to the maximum number of specified bits, then the compression ratio is only about 2x compared to the fully compacted set. The seed efficiency is only about 50%.

6.3 Serial Input Mode

It is possible to reduce the seed size, by adding a mode that enables us to directly shift in a vector in case the LFSR is not able to generate the vector (i.e. due to too many specified bits). Note that the vectors that are directly serially shifted into the flipflops will not be compressed.

If the seed size is too small, too many vectors need to be shifted in directly. Therefore, the compression ratio reduces (see example 5.2.3). If the seed size is too big, the seed efficiency reduces. Therefore, the compression ratio reduces. The optimal seed size is a tradeoff between both effects. This is illustrated in Fig. 6 for ASIC 1. It is clear to see that in this case the maximum possible compression ratio equals 8.6x. To achieve this compression ratio, a seed size of 1,200 bits was used.

6.4 Proposed LFSR Reseeding Architecture

The new LFSR reseeding architecture is applied on both ASIC 1 and ASIC 2 using single polynomial LFSR reseeding [Koenemann91] and multiple polynomial LFSR reseeding [Hellebrand95]. Note that due to the reduced seed size, the advantages of going to a multiple polynomial technique increases. Table 5 shows the experimental results together with comparisons with previous work.

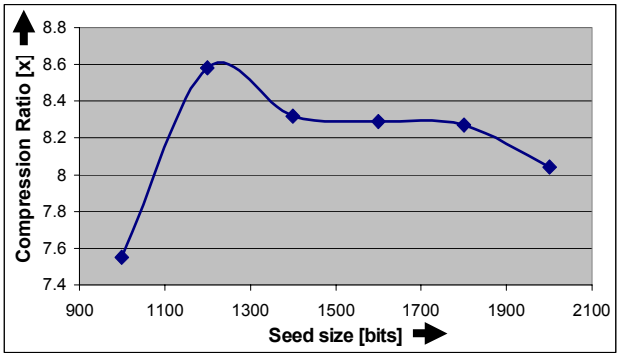

Figure 6: Serial Input Mode Compression Ratio (for ASIC 1)

The table shows that a compression ratio of 33x can be achieved by using the new approach, whereas the conventional approach only resulted in 2x (or 9x with a serial shift mode). Also the seed efficiency is increased from only 50% to 95%. In other words, the limited seed efficiency problem, one of the main drawbacks of the conventional reseeding architecture, is essentially

solved. Note that the entropy compression equals 37x. Therefore, the compression ratio almost reaches the entropy bound.

7 Conclusions

One of the main drawbacks of the conventional reseeding architecture is the limited seed efficiency problem. This paper proposed a new reseeding architecture that essentially solves the limited seed efficiency problem, by enabling a seed efficiency of almost 100%.

The architecture enables fine-grained use of multiple seeds per vector (or multiple seeds per partial vector), or multiple (partial) vectors per seed, resulting in a significant increase in the compression ratio.

The seed size is no longer limited by the maximum number of specified bits in a vector (*Smax+20*), but only by the maximum number of specified bits in a bit slice. This enables a reduction in the seed size (and LFSR area overhead) and an increase in the number of scan chains.

The proposed architecture reduces the hardware overhead compared to conventional LFSR reseeding architectures. Moreover, there is no performance impact and the techniques are very easy to implement.

The techniques are applied on two industrial designs resulting in compression ratios of 33x (with a seed efficiency of 95%) whereas conventional reseeding only resulted in about 2x (with a seed efficiency of only 50%). If decompression with an additional serial shift mode is used, then a compression ratio of about 9x could be achieved (with a seed efficiency of only 71%).

8 Acknowledgements

This work was sponsored by Agilent Technologies, Inc.

9 References

[ITRS99] International Technology Roadmap for Semiconductors (ITRS), 1999.

[Chandra00] A. Chandra, and K. Chakravarty, "Test Data Compression and Decompression for System-on-a-chip using Golomb codes", VLSI Test Symposium, pp. 113-120, 2000.

[Chandra01] A. Chandra and K. Chakravarty, "Frequency Directed Run-length Codes with applications to System-on-a-chip", VLSI Test Symposium, pp. 42-27, 2001.

[Bayraktaroglu01] I. Bayraktaroglu and A. Orailoglu, "Test Volume and Application Time Reduction Through Scan Chain Concealment", Design Automation Conference, pp. 151-155, 2001.

[Bardell87] P. H. Bardell, W. H. McAnney and J. Savir, "*Built-in Test For VLSI: Pseudo-random Techniques*", Wiley Inter-Science, 1987.

[Barnhart01] K. Barnhart, B. Keller, B. Konenmann and R. Walther, "OPMISR: The Foundation for Compressed ATPG Vectors", ITC 2001.

[Bayraktaroglu01] I. Bayraktaroglu, and A. Ogailoglu, "Test Volume and Application Time Reduction Through Scan Chain Concealment," IEEE Proc. of Design Automation Conference, pp. 151-155, 2001.

[Dorsch01] R. Dorsch, H.J. Wunderlich, "Tailoring ATPG for Embedded Testing", IEEE Proc. of International Test Conference, p. 530-537,2001.

[Golomb66] S.W. Golomb, "Run-length encoding," IEEE Trans. Inform. Theory, vol.. 11-12, pp.399-401, Dec. 1966.

[Hamzaoglu98] I. Hamzaoglu and J. Patel, "Test Set Compaction Algorithms for Combinational Circuits" IEEE Proc. of International Conference on Computer-Aided Design (ICCAD), pp. 283-289, 1998.

[Hamzaoglu99] I. Hamzaoglu, J. Patel, "Reducing Test Application Time for Full Scan Embedded Cores," Proc. of Int. Symposium on Fault Tolerant Computing, pp. 260-267, 1999.

[Hellebrand00] Hellebrand, S., H.-G. Liang, and H.-J. Wunderlich, "A Mixed Mode BIST Scheme Based on Reseeding of Folding Counters," Proc. of IEEE International Test Conference (ITC), 2000.

[Hellebrand02] S. Hellebrand, et. al, "Alternating Run-length Encoding", IEEE Test Resource Partitioning Workshop, 2002.

[Hellebrand95] S. Hellebrand, J. Rajski, S Tarnick, S. Venkatraman and B. Courtois, "Built-in Test for Circuits with Scan Based Reseeding of Multiple Polynomial Linear Feedback Shift Registers", IEEE Transaction on Computers, vol. 44, No. 2, pp.223-233, 1995.

[Hetheringten99] G. Hetheringten, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan and J. Rajski, "LBIST for Large Industrial Designs", Proc. Of International Test Conference", pp. 358-367, 1999.

[Hsu01] F. F. Hsu, K. M. Butler, J. H. Patel, "A Case Study on the Implementation of the Illinois Scan Architecture," Proc. of the IEEE International Test Conference (ITC), pp. 538-547, 2001.

[Huffman52] D.A. Huffman, "A Method for the Construction of Minimum Redundancy Codes", Proc. of IRE, Vol.40, No.9, pp. 1098-1101, 1952.

[Ishida98] M. Ishida, D.S Ha and T. Yamaguchi, "COMPACT: A Hybrid Method for Compressing Test Data", VLSI Test Symposium, pp. 62-69, 1998.

[Jas99] A. Jas, J. Ghosh Dastidar and N. A. Touba, "Scan Vector Compression Using Statistical Coding", VLSI Test Symposium, pp. 114-120, 1999.

[Khoche00] A. Khoche and J. Rivoir, "I/O Bandwidth Bottleneck for Test:Is it Real?" IEEE Test Resource Partitioning Workshop, pp. 2.3-1 - 2.3-6, 2000.

[Khoche02] A. Khoche, E.Volkerink, J. Rivoir and S. Mitra, "Vector Compression using EDA/ATE Synergies", IEEE VLSI Test Symposium, pp. 97-102, 2002.

[Kiefer00] G. Kiefer and H-J. Wunderlich, "Application of Deterministic Logic BIST on Industrial Circuits", International Test Conference (ITC). pp. 105-114, 2000.

[Koenemann01] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, D. Wheater, "A SmartBIST Variant with Guaranteed Encoding", IEEE Test Resource Partitioning Workshop, pp.2.4.1-6, 2001

[Koenemann91] B. Koenemann, "LFSR-Coded Test Patterns for scan Designs", Proceedings of European Test Conference", pp. 237-242, 1991.

[Krishna01] C.V. Krishna, A. Jas and N.A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding", Proc. of IEEE International Test Conference (ITC), pp. 885-893, 2001.

[Krishna02] C.V. Krishna and N. A. Touba, "Reducing Test Data Volume Using LFSR Reseeding with Seed Compression", Proc. of the IEEE International Test Conference (ITC), 2002.

[Liang01] H. Liang, S. Hellebrand, H. Wunderlich, "Two dimensional test data compression for scab based deterministic BIST", Proceedings of the IEEE International Test Converence (ITC), 2001.

[McCluskey00] E. J. McCluskey and C.W. Tseng, "Stuck-Fault Tests vs. Actual Defects," IEEE Proceedings of the International Test Conference, 2000.

[Mitra02] S. Mitra and K. S. Kim, "X-compact: An Efficient Response Compaction for Test Cost Reduction", To appear in Proc. of the IEEE International Test Conference (ITC), 2002.

[Mukherjee91] A. Mukherjee, N. Ranganathan, and M. Bassiouni, "Efficent VLSI Designs for Data Transformation of Tree-based Codes," IEEE Transactions on Circuits and Systems, pp. 306-314, March 1991.

[Koenemann00] B. Koenemann, "SMARTBIST", Presentation by IBM at ITC 2000.

[Pennebakerand93] W.B. Pennebakerand J. L. Mitchell, "*JPEG Still Image Data Compression Standard*", Van Nostrand Reinhold, 1993.

[Rajski98] J. Rajski, J. Tyszer, "Design of Phase Shifters for BIST applications", Proc. of IEEE International Test Conference (ITC), 1998.

[Touba96] N.A. Touba and E.J. McCluskey, "Altering a Pseudo-Random Sequence of Bits for Scan-Based BIST", Proc. of IEEE International Test Conference (ITC), pp. 167-175, 1996.

[Volkerink02a] E. Volkerink, A. Khoche, J. Rivoir, **"**Test Economics for Multi-site Test with Modern Cost Reduction Techniques", IEEE VLSI Test Symposium (VTS), 2002.

[Volkerink02b] E. Volkerink, A. Khoche, S.Mitra, "Packet-based test vector compression", Proc. Of the IEEE International Test Conference (ITC), 2002