Improving Test Compression by Retaining Non-Pivot Free Variables in Sequential Linear Decompressors

Sreenivaas S. Muthyala and Nur A. Touba

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712
sreenivaas@utexas.edu, touba@ece.utexas.edu

Abstract

Sequential linear decompressors are inherently efficient and attractive for compressing test cubes with many don't cares. The test cubes are encoded by solving a system of linear equations. In continuous decompression, typically a fixed number of free variables are used to encode each test cube in a "one-size-fits-all" manner. The non-pivot free variables used in Gaussian elimination are wasted when the decompressor is reset before decompressing the next test cube. This paper explores techniques for retaining the non-pivot free variables for a test cube and using them to help encode subsequent test cubes and hence improve encoding efficiency. This approach retains most of the non-pivot free variables with only a minimal increase in runtime for solving the equations and no added control information. Experimental results are presented showing that the encoding efficiency, and hence compression, can be significantly boosted.

1. Introduction

Test data volume continues to increase rapidly as technology scales due to increasing integration density and the need for more types of tests to be used. Test data compression is widely used to compress the amount of data stored on the tester. This helps to reduce tester storage requirements and improve test time as less data has to be transferred over the limited test data bandwidth between the tester and circuit-under-test (CUT) [Touba 06].

One highly efficient approach for compressing test vectors is to use sequential linear decompressors which can be constructed from linear feedback shift registers (LFSRs), ring generators [Mrugalski 04], or any circuit built from flip-flops, XORs, and wires. In a linear circuit, the final value of each scan cell after decompression can be written as a linear equation in terms of the bits coming from the tester which act as *free variables* that can be assigned any value. Test cubes can be encoded by solving a system of linear equations where each equation corresponds to a care bit in the test vector being encoded. This will be described in greater detail in Sec. 2. If the number of free variables is sufficiently larger than the number of care bits in the test cube being encoded, then the linear equations can be solved with a high probability [Könemann 91].

Encoding efficiency is defined as the total number of care bits in the test cubes divided by the total number of free variables used to encode the test cubes (i.e., the number of bits stored on the tester). In order to facilitate continuous (i.e., streaming) decompression with simple control, typically a fixed number of free variables is used to encode each test cube. In this case, the variance in the number of care bits per test cube greatly impacts the encoding efficiency. The number of free variables used to encode each test cube must be sufficiently large enough to handle the test cube with the most care bits. For test cubes with fewer care bits, more free variables end up being used to encode them than necessary

In EDT [Rajski 04], the number of care bits per test cube is balanced out to some degree by performing dynamic compaction during ATPG where additional input assignments are made to target other faults as long as the linear equations remain solvable. In [Könemann 01] and [Krishna 04], the number of free variables used to encode each test cube is dynamically adjusted depending on the number of care bits in the test cube, but this comes at the cost of additional control information that must be stored on the tester. In [Kassab 10] and [Janicki 11], tester channels in a system-on-chip (SOC) environment are dynamically reallocated to other cores-under-test depending on the number of free variables needed to encode test cubes. Regardless of the technique used, there are generally a number of free variables that are not used as pivots when solving the linear equations (i.e., they are extra degrees of freedom that are unused when encoding the test cube). These *non-pivot free variables* are effectively wasted when the sequential decompressor is reset between test cubes.

Some techniques have been proposed to try to utilize the non-pivot free variables. In [Krishna 02], the free variables themselves are compressed using a non-linear code where the non-pivot free variables are treated as don't cares. In [Liu 09], the assignment of values to the non-pivot free variables is selected to try to minimize test power when the test cube is decompressed.

978-1-4673-1595-1/12/$31.00

This paper explores techniques for retaining the non-pivot free variables for one test cube and using them to help encode subsequent test cubes and hence improve encoding efficiency. One approach for this would be to simply not reset the sequential linear decompressor between test cubes. However, this would rapidly increase the size of the system of linear equations making the runtime for solving the equations blow up. A novel approach is presented here for retaining most of the non-pivot free variables with only a small increase in runtime for solving the equations and no added control information. This approach can help to significantly improve encoding efficiency in the following way. Non-pivot free variables for test cubes with fewer care bits can be used to help when encoding the test cubes with more care bits. This helps to minimize the total number of free variables used to encode a test set and hence improve test compression.

The paper is organized as follows: Sec. 2 reviews the process of encoding test vectors using sequential linear decompressors. Sec. 3 describes the proposed approach and its computational complexity. Sec. 4 discusses the required hardware. Sec. 5 explains the process for encoding test cubes using the proposed approach. Sec. 6 shows experimental results, and Sec. 7 is a conclusion.

2. Sequential Linear Decompression

Before describing the proposed ideas, it is helpful to review the process used for encoding test cubes for sequential linear decompression.

The example in Fig. 1 shows how symbolic simulation of free variables coming from the tester is performed to obtain linear equations corresponding for the value of each bit in the scan cells after decompression. These linear equations can be represented as a Boolean matrix as shown in Fig. 2. To encode a particular test cube, there is one equation for each care bit which can be formed into a system of linear equations as shown in Fig. 3. A solution is found using Gauss-Jordan reduction which creates one pivot for each care bit in the test cube. The non-pivots represent free-variables that can be assigned any value. Depending on the values assigned to the non-pivots, the pivots can always be assigned appropriate values to solve the system of linear equations. A more detailed explanation of the encoding process can be found in [Könemann 91], [Krisha 01], and [Wang 06].

Existing techniques reset the sequential linear decompressor before decompressing the next test cube to decouple the linear equations. However, suppose the decompressor is not reset when decompressing the next test cube. Then the system of linear equations after decompressing the second test cube would be for solving for two test cubes and would contain twice as many columns because it corresponds to twice as many free variables coming from the tester. The non-pivots for the first test cube are effectively retained in the decompressor and can be used to help solve for the second test cube. The number of non-pivot free-variables that can be retained in this manner cannot exceed the number of flip-flops in the sequential linear decompressor because otherwise they start becoming linearly dependent with each other. The drawback of retaining free-variables in this manner is that the size of the Boolean matrix doubles in both the number of rows and columns. If we assume the number of rows and columns are roughly equal, which is a reasonable first-order approximation (i.e., the number of bits used for encoding is on the same order as the number of care bits), then the complexity of solving the linear equations is $O(n^3)$ where n is the number of columns/rows. So doubling the size of the Boolean matrix increases the time for solving the equations by a factor of $2^3=8$. If three test vectors were solved together, then the size of the Boolean matrix would triple which increase the time for solving the equations by a factor of $3^3=27$. So clearly this approach is not scalable.

3. Proposed Approach

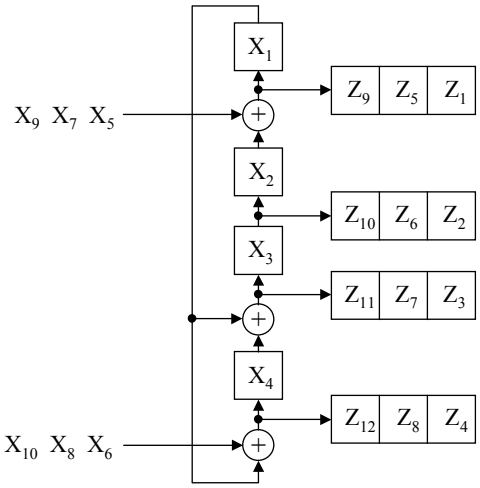
The proposed approach for retaining the non-pivot free variables from one test vector to help with encoding the next test vector involves using a FIFO (first-in first-out) buffer to achieve nearly the same benefit as encoding the test vectors together, but without blowing up the runtime for solving the linear equations.

In the proposed approach, the free-variables coming from the tester to the sequential linear decompressor are also stored in the FIFO buffer as illustrated in Fig. 4. The number of clock cycles worth of tester data (i.e., the number of tester slices), q, that the FIFO can store is equal to:

$$q = \frac{FIFO_{size}}{Tester\,Channels}$$

So the FIFO is used to capture (i.e., sink) the free variables coming from the tester in the last q clock cycles of one test cube, and then it is used to output (i.e., source) those free variables in the first q clock cycles of next test cube. Note that when the FIFO is capturing the free variables in the last q clock cycles, they are still going to the decompressor as well, so those free variables are available to help encode the first test cube if needed, but those that end up as non-pivots can be used to help encode the next test cube. The hardware details for how this is implemented will be described in detail in Sec. 4 including some special cases where the hardware becomes very simple.

978-1-4673-1595-1/12/$31.00 ©2012 IEEE



$Z_9 = X_1 \oplus X_4 \oplus X_9$	$Z_5 = X_3 \oplus X_7$	$Z_1 = X_2 \oplus X_5$
$Z_{10} = X_1 \oplus X_2 \oplus X_5 \oplus X_6$	$Z_6 = X_1 \oplus X_4$	$Z_2 = X_3$
$Z_{11} = X_2 \oplus X_3 \oplus X_5 \oplus X_7 \oplus X_8$	$Z_7 = X_1 \oplus X_2 \oplus X_5 \oplus X_6$	$Z_3 = X_1 \oplus X_4$
$Z_{12} = X_3 \oplus X_7 \oplus X_{10}$	$Z_8 = X_2 \oplus X_5 \oplus X_8$	$Z_4 = X_1 \oplus X_6$

Figure 1. Example of Symbolic Simulation of Linear Decompressor

$$\begin{pmatrix} 0&1&0&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&1&0&0&0 \\ 1&0&0&1&0&0&0&0&0&0 \\ 1&1&0&0&1&1&0&0&0&0 \\ 0&1&0&0&1&0&0&1&0&0 \\ 1&0&0&1&0&0&0&0&1&0 \\ 1&1&0&0&1&1&0&0&0&0 \\ 0&1&1&0&1&0&1&1&0&0 \\ 0&0&1&0&0&0&1&0&0&1 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \\ X_8 \\ X_9 \\ X_{10} \end{pmatrix} = \begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \\ Z_7 \\ Z_8 \\ Z_9 \\ Z_{10} \\ Z_{11} \\ Z_{12} \end{pmatrix}$$

Figure 2. System of Linear Equations for the Decompressor in Figure 1

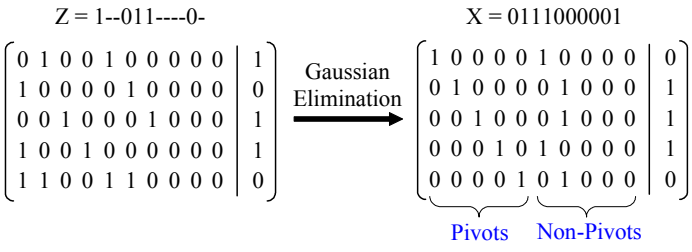


Figure 3. Example of solving system of linear equations for a particular test cube

978-1-4673-1595-1/12/$31.00 ©2012 IEEE

The advantage of using the FIFO in this manner is that the Gaussian elimination can be ordered to first try to use the early free-variables as pivots as much as possible, and only create pivots in the free variables coming in the last q clock cycles when necessary. This approach maximizes the number of non-pivot free-variables that get retained in the FIFO for encoding the next test cube and achieves nearly the same benefit as encoding the test vectors together without resetting the sequential linear decompressor. It only misses out on non-pivots that occur before the last q clock cycles which would generally be few.

To understand the additional computational complexity in solving the linear equations using this method, consider what the linear equations will look like. Fig. 5 shows the structure of the linear equations assuming only 2 test cubes are encoded. Let n be the number of free variables shifted in from the tester when decompressing each test cube, then the fraction of the free variables, F, that are retained by the FIFO to help in encoding the next test cube is equal to:

$$F = \frac{FIFO_{size}}{n}$$

As labeled in Fig. 5, the Boolean matrix corresponding to the linear equations can be divided into four non-zero regions labeled as submatrices A-D. As a first-order approximation to simplify the analysis, assume the number of care bits in the test cubes is on the order of n and the pivots are evenly distributed among the submatrices in proportion to their width (i.e., number of columns). For submatrices B and C which share the same columns, assume half the pivots are in B and half in C. Using these approximations, the computational complexity for Gaussian elimination for each of the submatrices can be represented in terms of the number of XOR operations which is equal to: (number of pivots) x (length of rows) x (height of pivot column). This is calculated assuming the Gaussian elimination for the submatrices is done in the order shown below:

Submatrix $A = [(1\text{-}F)(n)]\ [n]\ [n] = (1\text{-}F)n^3$
Submatrix $D = [n]\ [(1+F)(n)]\ [n] = (1+F)n^3$
Submatrix $B = [(0.5)(F)(n)]\ [n]\ [2n] = (F)n^3$
Submatrix $C = [(0.5)(F)(n)]\ [2n]\ [2n] = (2F)n^3$
Total $= (2+3F)n^3$

In the conventional case where no FIFO is used and each of the two test cubes are solved independently, then the number of operations is $2n^3$. If we use F=10%, then the number of operations would be $(2.3)n^3$. So the complexity increases by a factor of 1.15 which is very reasonable.

Fig. 6 shows the structure of the matrix if 3 test cubes are encoded together. The matrix has 7 non-zero regions labeled submatrices A-G. Using the same approximations as before, the number of XOR operations is calculated assuming the Gaussian elimination for the submatrices is done in the order shown below:

Submatrix $A = [(1\text{-}F)(n)]\ [n]\ [n] = (1\text{-}F)n^3$
Submatrix $D = [(1\text{-}F)(n)]\ [(1+F)(n)]\ [n] = (1\text{-}F^2)n^3$
Submatrix $G = [n]\ [(1+F)(n)]\ [n] = (1+F)n^3$
Submatrix $B = [(0.5)(F)(n)]\ [n]\ [2n] = (F)n^3$
Submatrix $F = [(0.5)(F)(n)]\ [2n]\ [2n] = (2F)n^3$
Submatrix $C = [(0.5)(F)(n)]\ [2n]\ [3n] = (3F)n^3$
Submatrix $E = [(0.5)(F)(n)]\ [3n]\ [3n] = (4.5F)n^3$
Total $= (3+10.5F\text{-}F^2)n^3$

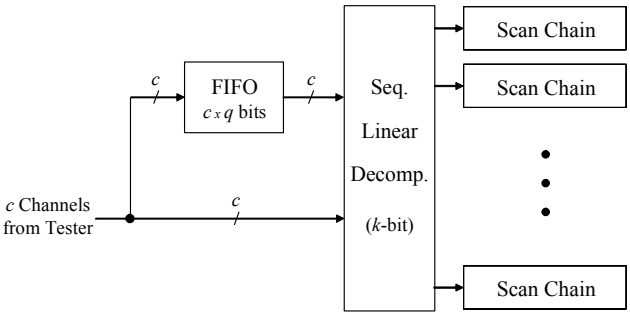


Figure 4. Block Diagram of Proposed Scheme

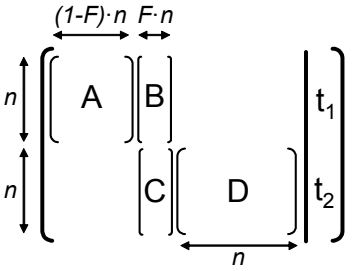


Figure 5. Structure of Boolean Matrix using FIFO with Size Equal to $F \cdot n$ to Solve for Two Testcubes

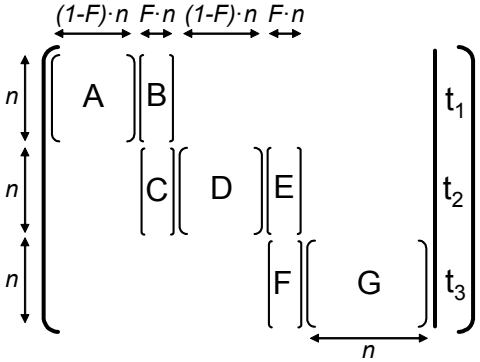


Figure 6. Structure of Boolean Matrix using FIFO with Size Equal to $F \cdot n$ to Solve for Three Testcubes

978-1-4673-1595-1/12/$31.00 ©2012 IEEE

In the conventional case where no FIFO is used and each of the three test cubes are solved independently, then the number of operations is $3n^3$. If we use $F=10\%$, then the number of operations would be $(4.04)n^3$. So the complexity increases by a factor of *1.35* which is still quite reasonable.

Depending on the computation time that can be spent on solving the linear equations, the multiplicity of test cubes that are encoded together can be selected appropriately. If m test cubes are encoded together at a time, then after the m-th test cube is decompressed, the contents of the FIFO are reset before the next group of m test cube are decompressed. This decouples the linear equations and keeps the complexity of solving the equations reasonable. Most of the benefit can be obtained even if only encoding two test cubes together at a time (i.e., m=2) by ordering the test cubes such that a test cube with fewer care bits is paired with one with more care bits. This is because the variance in the number of care bits per encoded test cube pair will be less than the variance in the number of care bits per individual test cube. By smoothing out the variance, the encoding efficiency can be improved. Moreover, fewer non-pivot free variables are being wasted.

4. Hardware Implementation

The proposed approach involves storing the free variables coming from the tester in the last q clock cycles when decompressing each test cube in a FIFO. The design of the FIFO can be simplified if the FIFO is the same size or smaller than the number of flip-flops in the sequential linear decompressor itself. In this case, the FIFO can be replaced by a shadow register of size $q \cdot c$ that captures during the last q clock cycles as illustrated in Fig. 7. Normally the sequential linear decompressor is reset before decompressing the next test cube. However, in this case, instead of resetting it, the contents of the shadow register are transferred to overwrite the state of the sequential linear decompresor, and then the shadow register is reset. This operation effectively initializes the sequential linear decompressor with the non-pivot free variables from the previous test cube which can then be used to help in encoding the next test cube.

If the size of the FIFO is larger than the size of the sequential linear decompressor, then it needs to be implemented as a conventional FIFO as illustrated in Fig. 4. In this case, the sequential linear decompressor is reset between test cubes, and the contents of the FIFO are transferred incrementally to the sequential linear decompressor as it decompresses the next test cube through additional injectors that are designed as part of the sequential linear decompressor. Normally the sequential linear decompressor is designed with a number of injectors equal to the number of channels feeding it from the tester. In this case, additional injectors are added to allow free variables to come from the output of the FIFO at the same time. The FIFO injects its contents into the sequential linear decompressor during the first q clock cycles when decompressing the next test cube concurrently with the free variables being injected from the tester. The design of ring generators with an arbitrary number of injectors is described in [Mrugalski 04].

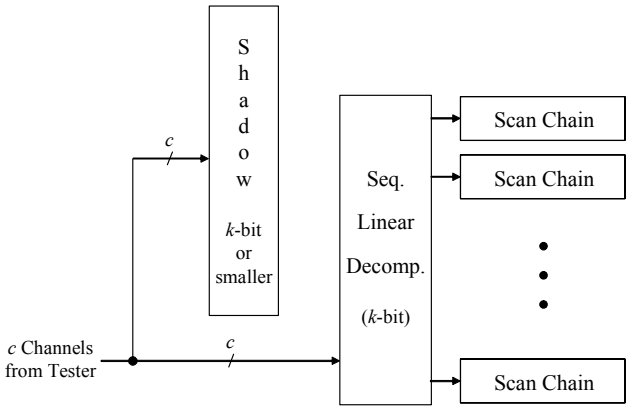


Figure 7. Proposed Scheme when Retaining k-bits or Less for a k-bit Decompressor

Note that it may be possible to implement the FIFO for the proposed method using an existing memory/buffer that is already present in the design for functional purposes. In that case, the hardware overhead for implementing the proposed method would be minimal.

The experimental results in Sec. 6 indicate that good results can be obtained when the size of the FIFO is on the order of the size of the decompressor. So the shadow register implementation is generally sufficient.

5. Procedure for Encoding Test Cubes

There are two scenarios for encoding test cubes using a sequential linear decompressor: static and dynamic. In static encoding, the test cubes are generated a priori with automatic test pattern generation (ATPG) leaving unassigned inputs a *X*'s, and are then encoded by solving the linear equations. The other is dynamic encoding where the equation solver is run iteratively during the ATPG process to regulate dynamic compaction and make sure that each generated test cube is solvable with a given decompressor. The procedure for using the proposed approach in these two scenarios is described in the following two subsections.

5.1 Static Encoding

Given the number of test cubes that will be solved together, m, the set of test cubes are partitioned into groups of m such that the maximum number of care bits in any group of m is minimized. This helps to reduce, n, the number free variables shifted in from the tester when performing the decompression for each test cube.

978-1-4673-1595-1/12/$31.00 ©2012 IEEE

Furthermore, the test cubes within each group are then ordered starting with the test cube with fewest care bits and finishing with the test cube with the most care bits. Since the non-pivot free-variables from earlier test cubes get passed on for encoding later test cubes, this allows the non-pivot free variables collected from the earlier test cubes with fewer care bits to be pooled up to help with solving the test cubes with more care bits.

5.2 Dynamic Encoding

In dynamic encoding, dynamic compaction is performed until it is no longer possible to solve the system of linear equations. With the proposed approach, the system of linear equations is first constructed assuming the test cube is the last test cube in the group of *m*, and all previous test cubes in the group of *m* are don't cares. Dynamic compaction is performed until it is no longer possible to solve the system of linear equations under these conditions. For the next test cube, the system of linear equations is constructed assuming that test cube is second to last in the group of *m*, and the previous test cubes are all don't cares and last test cube is the one previously generated. This process continues in this manner until all *m* test cubes in a group have been generated. At that point, the system of linear equations is solved to obtain a final solution which can then be used to perform fault simulation to drop additional faults. This process is then repeated to obtain the next set of *m* test cubes.

6. Experimental Results

Experiments were performed using a 64-bit LFSR with a phase shifter as a sequential linear decompressor using static encoding of test cubes which provide 100% fault coverage of detectable faults. The results are shown in Table 1. The first four columns show information about each circuit: number of test cubes, number of scan cells, and number of tester channels. Results are then shown for the conventional case where each test cube is encoded individually where the decompressor is reset between each test cube. The number of scan chains was increased until it was no longer possible to encode all the test cubes with the given number of tester channels. The number of scan chains and the resulting amount of data that needs to be stored on the tester is shown for the conventional case.

Next, results were generated using the proposed approach to retain non-pivot free variables. The results under the major heading *m=2* show the case where two test cubes were encoded at a time. The number of scan chains was increased until it was no longer possible to encode all the test cubes with the given number of tester channels. As the number of scan chains goes up, the scan length goes down. Consequently, less data is shifted in from the tester to the decompressor, so the amount of data stored on the tester is reduced (i.e., the amount of test compression increases). The percentage reduction in test data is computed as:

$$\frac{(Original\ Test\ Data) - (NewTest\ Data)}{(Original\ Test\ Data)}$$

To measure how large the FIFO needs to be, the maximum reduction in tester data was first determined by not resetting the LFSR between the two test cubes. Then the size of the FIFO was increased until the proposed method was able to achieve the same reduction in tester data as the case where the LFSR is not reset. This FIFO size is reported is the overhead column. It corresponds to the minimum size FIFO that would achieve the results reported in the table. Note that the FIFO size was always less than 64 bits (which is the size of the LFSR). This indicates that the simpler hardware implementation for the proposed method shown in Fig. 7 could be utilized.

Lastly, results were generated using the proposed approach with *m=3* where three test cubes were encoded at a time. Some improvement was obtained compared with *m=2*, but there is a diminishing marginal return as *m* is increased as is to be expected.

Table 1. Results for Using Proposed Scheme to Encode Test Data

Circuit	Num. Vect.	Scan Cells	Tester Channels	Conventional m = 1		Proposed m = 2				Proposed m = 3			
				Scan Chains	Tester Data	Scan Chains	Tester Data	Percent Reduction	Overhead (Min. FFs)	Scan Chains	Tester Data	Percent Reduction	Overhead (Min. FFs)
Ckt-A	205	2,522	6	76	46,740	86	41820	10.5%	14	94	38130	18.4%	14
Ckt-B	212	2,764	6	87	43,248	107	35,616	17.6%	36	111	34,344	20.6%	36
Ckt-C	426	3,644	8	68	190,848	81	160,176	16.1%	48	81	160,176	16.1%	48
Ckt-D	253	4,376	4	88	53,636	116	41,492	22.6%	44	122	39,468	26.4%	44
Ckt-E	308	6,338	4	127	65,296	166	51,744	20.8%	40	173	49,280	24.5%	40
Ckt-F	402	10,184	4	131	128,640	155	109344	15.0%	44	162	104,520	18.8%	44

978-1-4673-1595-1/12/$31.00 ©2012 IEEE

The results show a 10%-26% improvement in compression can be obtained with the proposed approach. While the test set was fixed in these experiments, an alternate experiment to measure the benefit for the proposed method would be fix the compression ratio and see how much the number of test patterns can be reduced due to the relaxed constraints on solving the linear equations that the proposed method provides. More static and dynamic compaction can be performed which should reduce the number of test patterns and provide a similar benefit in terms of overall compression. Due to tool limitations, we were not able to perform these experiments.

To fit in the normal design flow, the FIFO needs to be sized before knowing the test data. The results in Table 1 suggest a good size for the FIFO is to simply make it the same size as the decompressor. In this case, a shadow register can be used in place of a FIFO as explained in Sec. 4. Consequently, the control is very simple. The shadow register is activated to capture the last q clock cycles of free variables for one test cube and then used to initialize the LFSR between test cubes. For the experiments in Table 1, this process was able to achieve the same results as not resetting the LFSR between test cubes, but with much less computational complexity as explained in Sec. 3.

7. Conclusions

The proposed method provides a nice boost in the amount of compression achieved by a sequential linear decompressor at very minimal cost in terms of hardware, computational complexity, and control complexity. It is a simple scheme for allowing multiple test cubes to be encoded together without significantly increasing the runtime for solving the linear equations.

Acknowledgements

This research was supported in part by the National Science Foundation under Grant No. CCF-0916837.

References

[Balakrishnan 04] K.J. Balakrishnan and N.A. Touba, "Relating Entropy Theory to Test Data Compression", *Proc. of European Test Symposium*, pp. 94-99, 2004.

[Dutta 06] A. Dutta and N.A. Touba, "Using Limited Dependence Sequential Expansion for Decompressing Test Vectors," *Proc. of International Test Conference*, Paper 23.1, 2006.

[Janicki 11] J. Janicki, J. Tyszer, A. Dutta, M. Kassab, G. Mrugulski, N. Mukherjee, and J. Rajski, "EDT Channel Bandwidth Management in SoC Designs with Pattern-Independent Test Access Mechanism", *Proc. of International Test Conference,* Paper 14.1, 2011.

[Kassab 10] M. Kassab, G. Mrugalski, N. Mukherjee, J. Rajski, J. Janicki, and J. Tyszer, "Dynamic Channel Allocation for Higher EDT Compression for SoC Designs," *Proc. of International Test Conference*, Paper 9.2, 2010.

[Könemann 91] B. Könemann, "LFSR-Coded Test Patterns for Scan Designs", *Proc. of European Test Conference*, pp. 237-242, 1991.

[Könemann 01] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, "A SmartBIST Variant with Guaranteed Encoding," *Proc. Asian Test Symposium*, pp. 325-330, 2001.

[Krishna 01] C.V. Krishna and N.A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding", *Proc. of International Test Conference*, pp. 885-893, 2001.

[Krishna 02] C.V. Krishna and N.A. Touba, "Reducing Test Data Volume Using LFSR Reseeding with Seed Compression", *Proc. of International Test Conference,* pp. 321-330, 2002.

[Krishna 04] C.V. Krishna and N.A. Touba, "3-Stage Variable Length Continuous-Flow Scan Vector Decompression Scheme", *Proc. of VLSI Test Symposium,* pp. 79-86, 2004.

[Lee 04] J. Lee and N.A. Touba, "Low Power Test Data Compression based on LFSR Reseeding", *Proc. of Int. Conf. on Computer Design*, pp. 180-185, 2004.

[Liu 09] X. Liu and Q. Xu, "On Simultaneous Shift- and Capture-Power Reduction in Linear Decompressor-Based Test Compression Environment," *Proc. of International Test Conference*, Paper 9.3, 2009.

[Mrugalski 04] G. Mrugalski, J. Rajski, and J. Tyszer, "Ring Generators – New Devices for Embedded Test Applications", *IEEE Trans. on Computer-Aided Design*, Vol. 23, Issue 9, pp. 1306-1320, Sept. 2004.

[Rajski 04] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded Deterministic Test", *IEEE Trans. on Computer-Aided Design*, Vol. 23, Issue 5, pp. 1306-1320, May 2004.

[Touba 06] N.A. Touba, "Survey of Test Vector Compression Techniques", *IEEE Design & Test Magazine,* Vol. 23, Issue 4, pp. 294-303, Jul. 2006.

[Wang 06] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*, Morgan Kaufmann, 2006.

978-1-4673-1595-1/12/$31.00 ©2012 IEEE