Reconfigurable Linear Decompressors Using Symbolic Gaussian Elimination

Kedarnath J. Balakrishnan, Nur A. Touba

Reconfigurable Linear Decompressors Using Symbolic Gaussian Elimination

Kedarnath J. Balakrishnan* and Nur A. Touba
Computer Engineering Research Center
University of Texas at Austin
{kjbala,touba}@ece.utexas.edu

Abstract

A methodology for designing a reconfigurable linear decompressor is presented. A symbolic Gaussian elimination method to solve a constrained Boolean matrix is proposed and utilized for designing the reconfigurable network. The proposed scheme can be implemented in conjunction with any decompressor that has a combinational linear network. Using the given linear decompressor as a starting point, the proposed method improves the compression further. A nice feature of the proposed method is that it can be implemented with very little hardware overhead. Experimental results indicate that significant improvements can be achieved.

1. Introduction

As circuits become increasingly complex with each generation, the amount of test data required to test each chip is also becoming humongous. Hence test data storage requirements on the tester and test data bandwidth requirements between the tester and chip are growing rapidly [9]. Test data compression techniques provide a way to reduce these requirements thereby allowing less expensive testers to be used. Test time can also be simultaneously reduced with most test data compression techniques. The output response is relatively easy to compress since lossy compression techniques can be employed, e.g., using a multiple input signature register (MISR). However, compressing test stimuli is much more difficult because lossless compression techniques must be used. A lot of research has been done on lossless compression techniques for test vectors since reducing test data volume has become such an important problem.

Test vector compression schemes that use only linear operations to decompress the test vectors are called linear decompression schemes. Linear decompression techniques exploit the unspecified (don't care) bit positions in scan test cubes (i.e., deterministic scan test vectors where the unassigned bit positions are left as don't cares) to achieve large amounts of compression. A number of different techniques for designing linear decompressors have been proposed in the literature. These include both techniques based on linear feedback shift register (LFSR) reseeding and combinational linear expansion circuits consisting of XOR gates. The original idea of using an LFSR as a linear decompressor and solving for test cubes using linear algebra was described in [10].

A decompressor based on XOR tree network was described in [1]. This is an example of a continuous-flow decompressor. Continuous-flow linear decompressors are those that receive data from the tester in a continuous-flow manner *i.e. every cycle*. These operate very efficiently since they can be directly connected to the tester and they simply receive the data as fast as the tester can transfer it. This simulates scan chain concealment since for the tester it mimics the normal behavior of scan chains though the number of scan chains visible to the tester is much less than that in the design. Combinational continuous-flow linear decompressors are described in [6], [7], [1, 2], [16], and [13]. Continuous-flow techniques that have sequential elements like LFSRs *i.e.* sequential continuous-flow linear decompressors are described in [8], [11], [17], [18], and [14]. Most of the commercial tools for compressing test vectors are also based on linear decompressors. TestKompress from Mentor Graphics [17], SmartBIST from IBM/Cadence [11], and DBIST [4] from Synopsys are some examples.

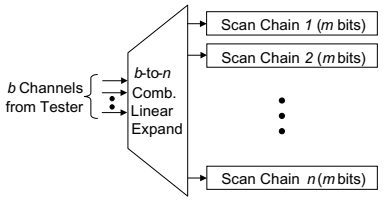

Figure 1. Combinational Linear Decompressor

In order to be able to compress a test set, the output space of the linear decompressor must contain all the test cubes in the test set. The output space of a linear decompressor is a linear subspace spanned by a Boolean matrix, $\boldsymbol{A}_{c\times n}$. Each row in $\boldsymbol{A}$ corresponds to a scan cell and each column corresponds to a free-variable in the input sequence. To determine whether a particular test cube is contained in the out-

*Kedarnath J. Balakrishnan is now with NEC Labs. in Princeton, NJ

Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)
1530-1591/05 $ 20.00 IEEE

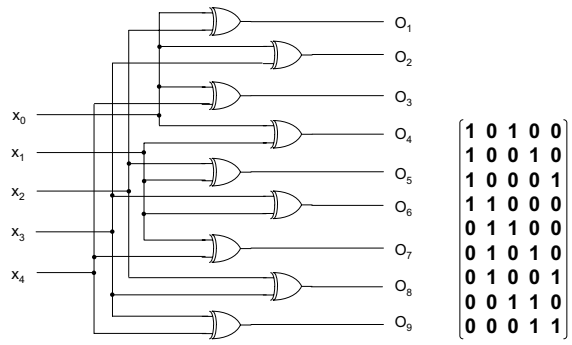

Figure 2. XOR Network and Matrix for Figure 1

put space and the corresponding input sequence to generate it can be quickly done through Gaussian elimination.

Figure 1 shows a combinational linear decompressor that receives b bits from the tester and expands it to n scan chains. The Boolean matrix for this decompressor can be constructed simply from the XOR network. Each row corresponding to a scan chain will have the inputs that are *xor*-ed together to get the scan chain value as 1 and all the others as 0. An example of a XOR network with $b = 5$ and $n = 9$ and the corresponding matrix is shown in Figure 2. The columns of the matrix correspond to inputs with the first column representing x_0 and the last column representing x_4. Output O_1 is the XOR of inputs x_0 and x_2 and hence in the first row, the values corresponding to these two inputs are 1 while the rest are 0. Obtaining the Boolean matrix A by symbolic simulation of the linear decompressor is described in detail in [12].

When a LFSR is used as the decompressor, it has been proved that if the number of free variables used to generate a test cube is 20 more than the number of specified bits in test cube, then the probability of the test cube not being solvable is less than one in a million [10]. However, for a given test set, the number of free variables can be further reduced provided the corresponding equations for each test cube are solvable. Hence, traditionally linear decompressors are designed on the basis of the worst-case scenario. In LFSR reseeding the size of LFSR is usually proportional to s_{max} - the maximum specified bits in any test cube of the test set.

Currently test compression methods are deployed in either of two formats. The first one is to decide on a compression scheme and the decompressor beforehand, during the circuit design stage itself and then generate the test patterns for the design based on that scheme. In this method, the decompressor circuit is also integrated into the design flow. Hence, fault coverage during test pattern generation may be affected due to the limitations of the decompressor. Also, last minute design changes may need test pattern regeneration and there may be some patterns that are not compressible using the given scheme. Techniques that can "reconfigure" the decompressor to take into account these changes will be very helpful. The other method is to design the decompressor after the circuit has been finalized and the test patterns available. Legacy circuits or intellectual property (IP) cores in a system-on-a-chip (SoC) environment are good examples of this. Each core in a SoC may require test compression scheme. Using a single decompressor that can be "reconfigured" for several cores instead of having several decompressors can result in a lot of area savings.

The idea of adding a reconfigurable part for test data reduction has been proposed earlier. [15] employs a reconfigurable interconnect network (RIN) within BIST to embed deterministic test patterns. The RIN has been proposed as an alternative to traditional phase shifters consisting of XOR gates. In [19], the original Illinois scan architecture [6] is altered by using a reconfigurable switch to control the connection between external pins and the scan chains. In [13], control bits are used to determine the number of scan chains that are fed by the combinational network.

This paper describes a methodology for "reconfiguring" any linear decompressor. It can be implemented in conjunction with any decompressor that has a combinational linear network and significantly improve the compression obtained. The rest of the paper is organized as follows. Section 2 describes the idea of a reconfigurable linear decompressor. In Section 3, we present a symbolic Gaussian elimination technique that is used to design the reconfigurable linear decompressor and Section 4 shows how it can be used to improve the compression. Experimental results are presented in Section 5 and Section 6 has conclusions.

2. Reconfigurable Linear Decompressor

The idea of reconfiguring a linear decompressor is to modify the output space of the linear decompressor to ensure that a given test cube can be generated using the linear decompressor. The linear decompression network can be modified using the configuration bits. For each configuration, the output space of the linear decompressor will be different. Hence this increases the chances of a test cube being generated using the decompressor. The configuration bits can either be stored on the tester and transferred to the decompressor with the test cubes or stored on-chip using a ROM. The number of different configurations required depends on the test set. In the case when each test cube in a test set requires a different configuration, these need to be explicitly stored. In the case when all the test cubes can be generated using the same configuration, then the decompression network can be finalized for that configuration and no extra storage is required. This implies that the decompressor is simply *redesigned*.

There are several possible ways to modify a linear decompressor such that it can be reconfigured. A simple mod-

ification would be to add multiplexers to each output of the network with the select input coming from the configuration bits. The other inputs could be some other output of the original decompressor. For example, consider a combinational decompressor using XOR gates shown in Figure 2. Instead of directly sending the outputs to the scan chains, there is another stage with a multiplexer before each scan chain as illustrated in Figure 3. These multiplexers will select which output is connected to which scan chain based on the configuration bits.

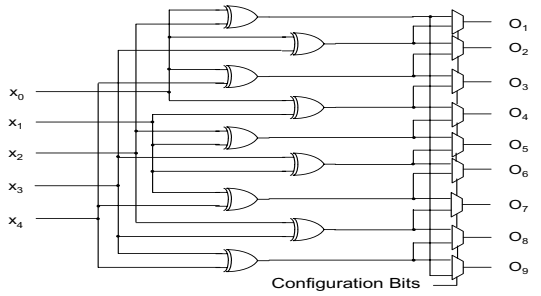

Figure 3. Making a Decompressor Reconfigurable

For a test cube to be solvable using the linear decompressor, a solution to the system of linear equations $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$ must exist where $\boldsymbol{x}$ is an assignment of values to the free-variables that are inputs to the decompressor when generating the test cube, and $\boldsymbol{b}$ is the value of each bit in the test cube. There is no need to solve the linear equations for the unspecified bits in the test cube, and hence only the linear equations (rows) corresponding to the specified bits in $\boldsymbol{b}$ need to be considered. Gaussian elimination [5] can be used to perform rows operations that transform a set of columns into an identity matrix. The elements that make the identity matrix are called the pivots.

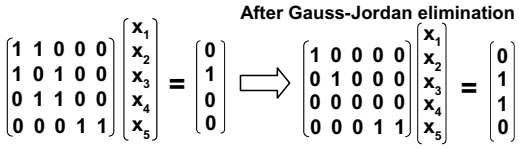

Figure 4. System of Equations for Test Cube t_1

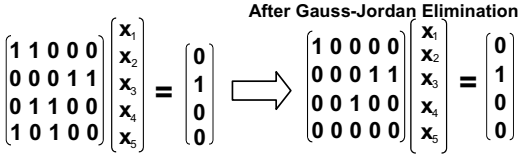

Figure 5. Reconfigured System of Equations for t_1

An example of a system of linear equations for a test cube t_1 and the corresponding system after Gaussian elimination is shown in Fig. 4. The rows after Gaussian elimination can be classified as either pivoted rows or linearly dependent rows. The pivoted rows have pivots while the linearly dependent rows are all 0. For the example in Fig. 4, the first two rows and the last row are pivoted while the third row is linearly dependent. If all rows are pivoted, then a solution to the system of linear equations exists, and hence the test cube can be decompressed using the linear decompressor. If some of the rows are linearly dependent, then a solution only exists if all of the corresponding values in $\boldsymbol{z}$ (the vector $\boldsymbol{b}$ after Gaussian elimination) are equal to 0 for the linearly dependent rows. If there is a linearly dependent row whose corresponding value in $\boldsymbol{z}$ is equal to 1, then no solution exists. In Fig. 4, the third row is linearly dependent but the corresponding value in $\boldsymbol{z}$ is 1, and thus there is no solution.

For the example in Fig. 4, let the specified values in $\boldsymbol{b}$ correspond to scan cells c_1 through c_4. If the decompressor is reconfigured such that the equations corresponding to scan cell c_2 and scan cell c_4 were exchanged, the system of linear equations would become solvable. This is shown in Fig. 5. Here, the last row is now linearly dependent and the corresponding value in $\boldsymbol{z}$ is 0. This is an example of how reconfiguration can be used to make a test cube solvable.

3. Symbolic Gaussian Elimination

The previous section described how reconfiguration of a linear decompressor can increase the chances of a test cube being solvable. In this section we describe a systematic procedure to do reconfiguration using *Symbolic Gaussian Elimination*. The key idea is to form the matrix $\boldsymbol{A}$ in terms of the configuration bits (*i.e.* each entry in the matrix is a function of the configuration bits) and then find an assignment of the configuration bits that makes the system of linear equations $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$ solvable.

In traditional Gaussian elimination, elementary row operations are used to reduce the coefficient matrix (the matrix $\boldsymbol{A}$) to a set of pivoted and linearly dependent rows. This is done by going through each column of the matrix and choosing a non-zero element as the pivot. All other rows with a 1 in this column are *xor*-ed with this row as part of elementary row operation. The resultant matrix will either have pivoted rows or linearly dependent rows.

We extend this technique to co-efficient matrices that have *functions* as elements. The functions are Boolean functions on a given set of variables. This is illustrated in Fig. 6 where each element of the matrix $\boldsymbol{F}$ is a function of the variables $\{a,b,c\}$. In this matrix, each f_{ij} is a function such that

$$f_{ij} : \{0,1\}^3 \rightarrow \{0,1\}$$

In the trivial case when each function is a constant 1 or a constant 0, this matrix will degenerate to a boolean matrix with entries 0 or 1. The variables $\{a,b,c\}$ correspond to the configuration bits of the linear decompressor. Solving for a system of linear equations such as $\boldsymbol{F}\,\boldsymbol{x} = \boldsymbol{y}$, with $\boldsymbol{F}$ as the matrix shown in Fig. 6 and $\boldsymbol{y}$ is a vector implies that we are

looking for a solution for at least one combination of a, b & c, or in other words, for at least one configuration.

$$\begin{pmatrix} f_{11}(a,b,c) & f_{12}(a,b,c) & f_{13}(a,b,c) & \dots & f_{1m}(a,b,c) \\ f_{21}(a,b,c) & f_{22}(a,b,c) & f_{23}(a,b,c) & \dots & f_{2m}(a,b,c) \\ f_{31}(a,b,c) & f_{32}(a,b,c) & f_{33}(a,b,c) & \dots & f_{3m}(a,b,c) \\ \dots & \dots & \dots & \dots & \dots \\ f_{n1}(a,b,c) & f_{n2}(a,b,c) & f_{n3}(a,b,c) & \dots & f_{nm}(a,b,c) \end{pmatrix}$$

Figure 6. A Matrix with Functions as Elements

Given such a system of linear equations, the algorithm for symbolic Gaussian elimination is given below. First, each element in the $\boldsymbol{y}$ vector is converted to a function in terms of the variables in $\boldsymbol{F}$. If the element in $\boldsymbol{y}$ is 1 then the function is identically equal to 1 and if the element is 0, the function is identically equal to 0. Then, the algorithm proceeds column wise choosing pivots in each column. Suppose f_{11} is the first pivot. Row operations are performed next for each row and this pivot. In the row operations, every element in a row is XOR-ed with the result of the AND of its element in the pivot column and the corresponding element in the pivot row. This is illustrated in Fig. 7 that shows the matrix after row operations for pivot f_{11}. The idea is to take into account both cases when $f_{11} = 0$ and $f_{11} = 1$. This operation is repeated for every pivot.

$$\begin{pmatrix} f_{11} & f_{12} & \dots & f_{1m} \\ f_{21} \oplus f_{21} \bullet f_{11} & f_{22} \oplus f_{21} \bullet f_{12} & \dots & f_{2m} \oplus f_{21} \bullet f_{1m} \\ f_{31} \oplus f_{31} \bullet f_{11} & f_{32} \oplus f_{31} \bullet f_{12} & \dots & f_{3m} \oplus f_{31} \bullet f_{1m} \\ \dots & \dots & \dots & \dots \\ f_{n1} \oplus f_{n1} \bullet f_{11} & f_{n2} \oplus f_{n1} \bullet f_{12} & \dots & f_{nm} \oplus f_{n1} \bullet f_{1m} \end{pmatrix}$$

Figure 7. Matrix after Row Operations for Pivot f_{11}

$$\begin{pmatrix} f'_{11} & f'_{12} & f'_{13} & \dots & f'_{1m} \\ f'_{21} & f'_{22} & f'_{23} & \dots & f'_{2m} \\ f'_{31} & f'_{32} & f'_{33} & \dots & f'_{3m} \\ \dots & \dots & \dots & \dots & \dots \\ f'_{n1} & f'_{n2} & f'_{n3} & \dots & f'_{nm} \end{pmatrix}$$

Figure 8. Matrix after All Row Operations

The matrix after doing all the row operations for each pivot will look like Fig. 8 where f'_{ij} are again functions of $\{a,b,c\}$. For ease of explanation, assume that the pivot for each column i is given by element f'_{ii}. The next step would be to ensure that each pivot has atleast one minterm for which the function equates to one. If the pivot function does not have a single minterm that equates to one, then that row is equivalent to a linearly dependent row in the normal Gaussian eliminated matrix. Hence the corresponding element in the $\boldsymbol{y}$ matrix should be zero. This condition for a pivot f_{ii} can be written in mathematical form as

$$f'_{ii} + \bar{y_i} = 1$$

Since this condition must be valid for each pivot, the overall condition can be written in a product of sum form as

$$(f'_{11} + \bar{y_1})(f'_{22} + \bar{y_2}) \cdots (f'_{nn} + \bar{y_n}) = 1$$

If the above condition is satisfied, there exists a solution to the system of equations. The number of minterms of $\{a,b,c\}$ for which the above condition is satisfied will indicate the number of different configurations possible.

Note that there may be more than one possible pivot for each column and hence the pivot is selected using a heuristic. For the first column, the element that has the maximum number of minterms is chosen as the pivot since it has the best chances of having a solution. For the next columns, the element that has the most number of minterms in common with the current pivots is chosen. This ensures that the algorithms proceeds in such a way that maximum number of solutions (configurations) are possible.

The algorithm described above can be implemented with very little overhead with respect to the basic Gaussian elimination method. Each function is stored in terms of its minterms. For example, if three configuration bits are used, then there are eight possible minterms. Each element in the matrix consists of eight values, one corresponding to each minterm. The row operations are performed on the corresponding minterms. The only additional step in this procedure is evaluating the final product of sums condition. This can be achieved by simply performing bitwise operations on the entries so that the corresponding minterms are evaluated together. The number of entries in the product of sums condition depends on the number of pivots and hence on the size of the matrix. The complexity of the evaluation step increases linearly with the size, since the bitwise operations need to be performed for each additional pivot. The number of minterms for each function depend on the number of configuration bits. The number of configuration bits is a design parameter that can be decided depending on the compression required and the maximum allowed running time of the algorithm.

4. Increasing Compression by Reconfiguration

Given a linear decompressor, the systematic procedure given in the previous section can be used to reconfigure it and increase the chances of compressing any test set. This reconfiguration can be done in two ways. The first one is to search for a single configuration by which all the cubes in the test set can be compressed. In this case, the configuration can be hardwired into the decompressor *i.e.* the decompressor is *redesigned* and no explicit configuration bits are required. The other method would be to have one configuration for each cube which is loaded into the decompressor

every time a new test cube is loaded. The configuration bits for each test cube need to be stored explicitly.

Compression obtained using a linear decompression scheme can be improved using reconfiguration in several ways. One method would be to reduce the number of free-variables that the decompressor receives per test cube from the tester as much as possible while still keeping the test set compressible through reconfiguration. Any method can be used to design the initial decompressor. Then the number of free-variables that are input to the decompressor per test cube can be incrementally reduced and symbolic Gaussian elimination can be used to check whether it is possible to still solve for all the test cubes using reconfiguration. If so, then this process of incrementally reducing the number of free-variables and checking for a solution is repeated until a point is reached when no further reduction in the number of free-variables per test cube is possible while still being able to solve for all test cubes. The end result will be a linear decompressor that generates the exact same test set, but uses fewer tester channels thereby reducing tester storage and bandwidth requirements.

Another method is to keep the number of free-variables that the decompressor receives per test cube constant, but use reconfiguration to relax the constraints on ATPG (automatic test pattern generation) such that more specified bits per test cube can be generated. This will allow more static and dynamic compaction while still being able to solve for the test cubes. Some test compression methodologies (e.g., [1, 2], [17]) involve fixing the decompressor design and then constraining the ATPG so that the resulting test cubes will be in the output space of the decompressor. The constraints on the ATPG reduce the amount of static and dynamic compaction that are performed and therefore can result in more test cubes and hence more test time. Reconfiguration of the linear decompressor can be used to allow more specified bits per test cube while still being able to solve for the test cube. This can be used to relax the constraints on the ATPG and thereby allow more static and dynamic compaction which will in turn reduce the total number of test cubes and hence result in a reduction of both test time and tester storage requirements.

5. Experimental Results

The effectiveness of the proposed method was evaluated by performing two sets of experiments. The first set of experiments consisting those described in the previous section were performed on randomly generated test cubes for large industrial-size scan architectures. Circuits were assumed to have either 512 or 1024 scan chains and the number of channels available from tester to be 32. The length of the scan chains varied from 24 to 128. The initial decompressor is an XOR network with 32 inputs and 512 or 1024 outputs. For reconfiguration, a four input multiplexer was added to the output of the decompressor network with the select bits coming from the configuration bits. All the experiments were performed with a total of eight configuration bits. The results are presented in Table 1. The first two columns show the number of scan chains and length of each scan chain. The columns under "without reconfiguration" show the maximum compression that can be obtained using the given decompressor. For each randomly generated test cube, the number of specified bits was incrementally increased until it could no longer be solved using the given decompressor. The maximum percentage of specified bits per test cube that could be solved by the given decompressor is shown in column 4. The corresponding encoding efficiency and compression ratios are given in columns 5 and 6. Encoding efficiency is defined as the ratio of number of specified bits in the test set to the number of compressed bits that need to be stored on the tester. Compression ratio is the ratio of the original bits in the test set to the compressed bits.

The columns under "Reducing channels" show the results for reduction in the number of tester channels using reconfiguration. The reduced number of tester channels along with the corresponding compression ratio are shown under "Red. Chan." and "Compr. Ratio". The reduced channels include one channel from the tester for the configuration bits to be loaded before each test cube. The column "% Impr." shows the improvement in compression ratio due to the reconfiguration. The average percentage improvement for all the different scan sizes is around 39.4 % which is a significant improvement for very little extra hardware overhead. The columns under "Increasing Spec. Bits" show the results for increasing the percentage of specified bits that can be handled by a given decompressor using reconfiguration. The tester channels are kept constant at 32 and the number of specified bits are increased as much as possible until it is no longer possible to solve for all the test cubes. The new percentage of specified bits and encoding efficiency are shown as well as the percentage improvement in the encoding efficiency. The experiments assume a single configuration for each test cube and the configuration bits are taken into account while calculating the compression results. In this experiment, the average percentage improvement for all the different scan sizes is around 36.5 %.

The other set of experiments were performed on 100 % stuck-at fault coverage test sets for the largest ISCAS '89 [3] benchmark circuits. Table 2 compares the compression results obtained using the proposed scheme with some of the combinational decompressor techniques proposed earlier. The number of tester channels required and the amount of test data that need to be stored on the tester for the XOR network [1] and the adjustable width technique [13] are compared with those of the proposed scheme. Note that in [1] the decompressor design is integrated into the test

Table 1. Results for Randomly Generated Test Cubes

Num. Scan Chains	Scan Chain Length	Without Reconfiguration				Reducing Channels			Increasing Spec. Bits		
		Tester Chan.	Percent. Spec.	Encod. Effic.	Compr. Ratio	Red. Chan.	Compr. Ratio	% Impr.	Incr. % Spec.	Encod. Effic.	% Impr.
512	24	32	3.61 %	0.578	16.0	23	22.3	39.4 %	4.69 %	0.727	25.8%
	32	32	3.42 %	0.547	16.0	23	22.3	39.4 %	4.30 %	0.667	21.9%
	64	32	2.83 %	0.453	16.0	22	23.3	45.5 %	4.00 %	0.621	37.1%
	128	32	2.67 %	0.427	16.0	24	21.3	33.3 %	2.91 %	0.661	54.8 %
1024	24	32	1.66 %	0.531	32.0	24	42.7	33.3 %	2.34 %	0.727	36.9%
	32	32	1.66 %	0.531	32.0	24	42.7	33.3 %	2.25 %	0.697	31.3%
	64	32	1.49 %	0.476	32.0	22	46.5	45.5%	2.18 %	0.677	42.2%
	128	32	1.50 %	0.480	32.0	22	46.5	45.5%	2.20 %	0.680	41.7%

Table 2. Results for ISCAS'89 Benchmarks

Circuit	[1]		[13]		Prop. Scheme	
	Tester Chan.	Test Data	Tester Chan.	Test Data	Tester Chan.	Test Data
s13207	24	25344	19	14307	15	14098
s15850	32	22784	19	15067	18	18080
s38417	32	89856	19	49001	19	54020
s38584	24	38976	19	28994	19	31436

pattern generation, while in [13] compression is performed on already generated test patterns. The proposed scheme uses the same test patterns as [13]. As expected, the proposed scheme performs better than [1] for all circuits, both in terms of the number of tester channels required and the amount of compressed data. In comparison with [13], the number of channels required are smaller or the same but the amount of compressed data is slightly higher. Note that the proposed scheme doesn't need to generate separate scan clocks for each group of scan chains as required in [13].

6. Conclusions

A symbolic Gaussian elimination method to solve a constrained boolean matrix equation is presented. An application of the above method to reconfigure any linear decompressor is proposed. The reconfiguration can be implemented with very little hardware overhead. Experimental results show that compression obtained using a linear decompressor can be significantly improved using reconfiguration.

References

[1] I. Bayraktaroglu and A. Orailoglu. Test volume and application time reduction through scan chain concealment. In *Proc. of Design Automation Conference*, pages 151–155, 2001.

[2] I. Bayraktaroglu and A. Orailoglu. Decompression hardware determination for test volume and time reduction through unified test pattern compaction and compression. In *Proc. of VLSI Test Symposium*, pages 113–118, 2003.

[3] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *Proc. of International Symposium on Circuits and Systems*, page 19291934, 1989.

[4] M. Chandramouli. How to implement deterministic logic built-in self-test (BIST). *Complier: A Monthly Magazine for Technologists Worldwide*, Jan. 2003.

[5] C. Cullen. *Linear Algebra with Applications*. Addison-Wesley, 1997.

[6] I. Hamzaoglu and J. Patel. Reducing test application time for full scan embedded cores. In *Proc. of Int. Symposium on Fault Tolerant Computing*, pages 260–267, 1999.

[7] F. Hsu, K. M. Butler, and J. H. Patel. A case study on the implementation of the illinois scan architecture. In *Proc. of International Test Conference*, pages 538–547, 2001.

[8] A. Jas, B. Pouya, and N. Touba. Virtual scan chains: A means for reducing scan length in cores. In *Proc. of VLSI Test Symposium*, pages 73–78, 2000.

[9] A. Khoche and J. Rivoir. I/O bandwidth bottleneck for test: Is it real? In *Proc. of International Workshop on Test Resource Partitioning*, 2000.

[10] B. Konemann. LFSR-coded test patterns for scan designs. In *Proc. of European Test Conference*, pages 237–242, 1991.

[11] B. Konemann. A smartBIST variant with guaranteed encoding. In *Proc. of Asian Test Symposium*, pages 325–330, 2001.

[12] C. Krishna and N. Touba. Reducing test data volume using LFSR reseeding with seed compression. In *Proc. of International Test Conference*, pages 321–330, 2001.

[13] C. Krishna and N. Touba. Adjustable width linear combinational scan vector decompression. In *Proc. of International Conference on Computer-Aided Design (ICCAD)*, pages 863–866, 2003.

[14] C. Krishna and N. Touba. 3-stage variable length continuous-flow scan vector decompression scheme. In *Proc. of VLSI Test Symposium*, pages 79–86, 2004.

[15] L. Li and K. Chakrabarty. Deterministic BIST based on a reconfigurable interconnection network. In *Proc. of International Test Conference*, pages 460–469, 2003.

[16] S. Mitra and K. Kim. Xmax: X-tolerant architectures for maximal test compression. In *Proc. of International Conference on Computer Design*, pages 326–330, 2003.

[17] J. Rajski and et al. Embedded deterministic test for low cost manufacturing test. In *Proc. of Int. Test Conference*, pages 301–310, 2002.

[18] W. Rao, I. Bayraktaroglu, and A. Orailoglu. Test application time and volume compression through seed overlapping. In *Proc. of Design Automation Conference*, pages 732–737, 2003.

[19] H. Tang, S. Reddy, and I. Pomeranz. On reducing test data volume and test application time for multiple scan chain designs. In *Proc. of International Test Conference*, pages 1079–1088, 2003.