Open access • Proceedings Article • DOI:10.1109/ISVLSI.2005.49

Low cost test vector compression/decompression scheme for circuits with a reconfigurable serial multiplier — Source link

A. Dutta, T. Rodrigues, Nur A. Touba

Published on: 11 May 2005 - IEEE Computer Society Annual Symposium on VLSI

Topics: Test compression, Test vector, Design for testing, Automatic test pattern generation and System on a chip

Low Cost Test Vector Compression/Decompression Scheme for Circuits with a Reconfigurable Serial Multiplier

Avijit Dutta, Terence Rodrigues, and Nur A. Touba

Computer Engineering Research Center
University of Texas, Austin, TX 78712-1084
{adutta,trodrigu,touba}@ece.utexas.edu

Abstract

Many chip designs contain one or more serial multipliers. A scheme is proposed to exploit this to compress the amount of data that needs to be stored on the tester and transferred to the CUT during manufacturing test. The test vectors are stored on the tester in a compressed format by expressing each test vector as a product of two numbers. While performing multiplication on these stored seeds in the Galois Field modulo 2, GF(2), the multiplier states (i.e. the partial products) are tapped to reproduce the test vectors and fill the scan chains. In contrast with other test vector decompression schemes that add significant test specific hardware to the chip, the proposed scheme reduces hardware overhead by making use of existing functional circuitry. Experimental results demonstrate that a high encoding efficiency can be achieved using the proposed scheme.

1. Introduction

System-on-a-chip (SOC) designs integrate a lot of functionality together on a single chip. This high level of integration has reduced manufacturing cost, but has greatly increased the cost of test. One of the increasingly difficult challenges in testing SOCs is handling the large amount of test data that must be transferred between the tester and the chip. The entire set of test vectors for all components of the SOC must be stored on the tester and transferred to the chip during testing.

A number of test data compression schemes have been developed using a variety of techniques to reduce the test storage on the tester and also to reduce the test data bandwidth between the tester and the chip. Code based compression techniques include run-length codes [1], Huffman codes [2], frequency directed codes [3], LZ77 [4], etc. Another very popular test vector compression technique is based on linear expansion where test data is decompressed on chip using only linear operations. This includes techniques based on linear feedback shift register (LFSR) reseeding [6], linear expansion networks consisting of XOR gates and seed overlapping [5]. These methods exploit the unspecified bits in the test cubes. Every decompressed bit is represented as a linear combination (modulo 2) of the stored compressed data. Every stored bit is denoted as a free variable. Free variables can be assigned any value (0, 1). The values are assigned such that when shifted into the scan chains through linear expansion networks, they can reproduce the desired test cubes. Each of the specified bits of the scan vectors can be represented by a linear equation in terms of the free variables depending on the decompressor circuit. Typically only 2-5% of the bits are specified and that provides a high degree of freedom while solving the linear equations. The choice of decompression network is vital as the encoding efficiency, which is defined as the ratio of specified bits to the number of bits stored on the tester (i.e. the compressed data) largely depends on it. The number of free variables used to solve a test cube can be fixed [8] or variable [9], [10]. Schemes with a fixed number of free variables require less control logic and control data. However the encoding efficiency is limited by the most specified test cube which is the hardest to solve. Schemes using a variable number of free variables require more complex control logic compared to those with fixed number of free variables.

One very economical option is to use the existing components in the circuit under test (CUT) to test other parts of the circuit. Since multiplication is very commonly used and is an expensive operation [11], rapid improvements of VLSI technologies have been used to increase the reliability and the speed of the multiplication units embedded in the circuit. This is particularly important for specialized chips supporting multiplication intensive operations, such as in digital signal processing and computer graphics. For the design of fixed-point digital signal processing systems, primarily two approaches are used: bit-serial and bit-parallel. Bit-parallel approaches are time efficient. On the other hand, bit serial approaches are highly area efficient. Since signal-processing integrated circuits typically require many applications over a short period, small area multiplier implementations are of particular interest. For serial multipliers, area in terms of adder requirement grows only linearly with the word size N. This compares favorably to the N^2 adder required for parallel multipliers. Serial multipliers allow a high degree of pipelining and hence very high throughput can be achieved. Another advantage is very small interconnect area is required for communication of numerical data with arbitrary precision. The serial multipliers can be easily reconfigured to

Proceedings of the IEEE Computer Society Annual Symposium on VLSI
New Frontiers in VLSI Design
0-7695-2365-X/05 $20.00 © 2005 **IEEE**

function as a cyclic code generator, perform multiplication in GF(2), and function as an LFSR in addition to functioning as a pure binary multiplier. For circuits with a built-in serial multiplier unit, test data decompression can be performed using the multiplier itself without having to modify the scan chains. Our proposed scheme is based on a linear expansion technique and uses the intermediate states of a reconfigurable multiplier that is a functional component of the circuit itself to decompress the test data. Experimental results suggest that a high encoding efficiency is achievable using only the built-in serial multiplier. We also analyze how the encoding efficiency can be further improved by sharing operands across multiple test cubes.

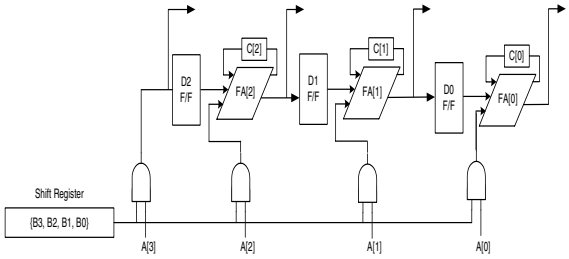

Figure 1. Architecture of a 4-bit wide reconfigurable multiplier

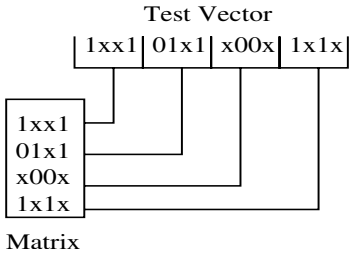

Figure 2. Test matrix

2. Decompression Using Multiplier

In this section, we describe the proposed technique to decompress deterministic test vectors on chip. Figure 1 shows the architecture of a 4 bit wide reconfigurable serial multiplier. When performing multiplication in GF(2), the carry register output is constantly set to 0 thereby transforming the full-adder into an XOR gate. The deterministic vectors are stored as compressed "seeds" and are decompressed by multiplying the seeds together in the multiplier which has been reconfigured to implement GF(2) multiplication. For an *(n×n)* test matrix, the objective is to find two *n*-bit numbers *A* and *B* which will be used as seeds, such that the states of the multiplier while performing product (*A* × *B*) in GF(2) using the shift-and-add algorithm generates the test matrix. The scheme is explained later with an example. The Gauss-Jordan elimination [7] method is used to solve the set of linear equations in GF(2). Consider the example in Fig. 2 where the test vector is converted into a (4x4) test matrix. Equation 1 shows the test matrix. The objective is to find two vectors *A* and *B* such that when multiplied in GF(2) using the built-in serial multiplier implementing the shift-and-add algorithm, the multiplier states can be tapped to produce the desired test matrix.

$$\begin{bmatrix} t_{14} & t_{13} & t_{12} & t_{11} \\ t_{24} & t_{23} & t_{22} & t_{21} \\ t_{34} & t_{33} & t_{32} & t_{31} \\ t_{44} & t_{43} & t_{42} & t_{41} \end{bmatrix} = \begin{bmatrix} 1 & x & x & 1 \\ 0 & 1 & x & 1 \\ x & 0 & 0 & x \\ 1 & x & 1 & x \end{bmatrix} \quad (1)$$

$$A = \begin{bmatrix} a_4 & a_3 & a_2 & a_1 \end{bmatrix} \quad (2)$$

$$B = \begin{bmatrix} b_4 & b_3 & b_2 & b_1 \end{bmatrix} \quad (3)$$

$$v_{ij} = a_i b_j \quad (4)$$

The matrix of partial products generated while computing $A \times B$ in GF(2) is:

$$\begin{bmatrix} & & & v_{14} & v_{13} & v_{12} & v_{11} \\ & & v_{24} & v_{23} & v_{22} & v_{21} & \\ & v_{34} & v_{33} & v_{32} & v_{31} & & \\ v_{44} & v_{43} & v_{42} & v_{41} & & & \end{bmatrix} \quad (5)$$

The matrix in the right hand side of Eq. 6 shows the state of the multiplier after each clock cycle. After the first clock cycle the state of the multiplier while functioning in GF(2), is same as the first row of that matrix. On subsequent clock cycles, the state is determined by adding (modulo 2) the previous row right-shifted by 1 bit position, to the current row. In the following matrices, $\oplus$ implies (modulo 2) addition, which is equivalent to an xor operation. The objective is to choose *A* and *B* such that the matrix entries in Eq. 6 match the entries with a 0 or 1 specified in the matrix in Eq. 1. Equation 7 can be written in the form $Mx = y$.

$$\begin{bmatrix} t_{14} & t_{13} & t_{12} & t_{11} \\ t_{24} & t_{23} & t_{22} & t_{21} \\ t_{34} & t_{33} & t_{32} & t_{31} \\ t_{44} & t_{43} & t_{42} & t_{41} \end{bmatrix} = \begin{bmatrix} v_{14} & v_{13} & v_{12} & v_{11} \\ v_{24} & v_{14} \oplus v_{23} & v_{13} \oplus v_{22} & v_{12} \oplus v_{21} \\ v_{34} & v_{24} \oplus v_{33} & v_{14} \oplus v_{23} \oplus v_{32} & v_{13} \oplus v_{22} \oplus v_{31} \\ v_{44} & v_{34} \oplus v_{43} & v_{24} \oplus v_{33} \oplus v_{42} & v_{14} \oplus v_{23} \oplus v_{32} \oplus v_{41} \end{bmatrix}$$

(6)

$$\begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&1&0&0&1&0&0&1&0&0&1&0&0&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1
\end{bmatrix}\times\begin{bmatrix}v_{11}\\v_{12}\\v_{13}\\v_{14}\\v_{21}\\v_{22}\\v_{23}\\v_{24}\\v_{31}\\v_{32}\\v_{33}\\v_{34}\\v_{41}\\v_{42}\\v_{43}\\v_{44}\end{bmatrix}=\begin{bmatrix}t_{11}\\t_{12}\\t_{13}\\t_{14}\\t_{21}\\t_{22}\\t_{23}\\t_{24}\\t_{31}\\t_{32}\\t_{33}\\t_{34}\\t_{41}\\t_{42}\\t_{43}\\t_{44}\end{bmatrix} \quad (7)$$

Only the equations corresponding to the specified bits have to be solved. So finally we have,

$$\begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{bmatrix}\times\begin{bmatrix}v_{11}\\v_{12}\\v_{13}\\v_{14}\\v_{21}\\v_{22}\\v_{23}\\v_{24}\\v_{31}\\v_{32}\\v_{33}\\v_{34}\\v_{41}\\v_{42}\\v_{43}\\v_{44}\end{bmatrix}=\begin{bmatrix}t_{11}\\t_{14}\\t_{21}\\t_{23}\\t_{24}\\t_{32}\\t_{33}\\t_{42}\\t_{44}\end{bmatrix} \quad (8)$$

$$\begin{bmatrix}t_{11}\\t_{14}\\t_{21}\\t_{23}\\t_{24}\\t_{32}\\t_{33}\\t_{42}\\t_{44}\end{bmatrix}=\begin{bmatrix}1\\1\\1\\1\\0\\0\\0\\1\\1\end{bmatrix} \quad (9)$$

Next Gauss-Jordan Elimination method can be applied in GF(2) on Eq. 8 to obtain the vectors *A* and *B*. For this example, we finally get, *B = 1101* and *A = 1011* in binary, where *B* is applied serially with LSB applied first and the MSB applied last and *A* is applied in parallel. The states of the multiplier unit are shown in Fig. 4. 4 clock cycles are required to reproduce the test matrix. Here the storage requirement for the *4×4 (n=4)* test vector matrix is reduced from n^2 bits to $2n$ bits i.e. from 16 bits to 8 bits. For this particular example, the amount of compression *((16-8)/16)×100% = 50%* and encoding efficiency = *9/8 = 1.125*.

$$v_{44} = 1 \rightarrow a_4 = 1, b_4 = 1$$
$$v_{42} = 1 \rightarrow a_4 = 1, b_2 = 1$$
$$v_{32} = 1 \rightarrow a_3 = 1, b_2 = 1$$
$$v_{14} = 1 \rightarrow a_1 = 1, b_4 = 1$$
$$v_{11} = 1 \rightarrow a_1 = 1, b_1 = 1$$
$$v_{33} = 0 \rightarrow b_3 = 0$$
$$v_{24} = 0 \rightarrow a_2 = 0$$
$$v_{23} = 0$$
$$v_{12} \oplus v_{21} = 1 \rightarrow a_1 b_2 \oplus a_2 b_1 = 1$$

The two *n*-bit numbers *A* and *B* are called seeds and represent the compressed version of the test-vector matrix. In general, the reduction in storage is given by:

$$\left(\frac{n^2 - 2n}{n^2}\right) \times 100\% \quad (10)$$

In scan based designs, multiple clock cycles are required to generate each vector. Each test cube is partitioned into multiple scan chains as shown in Fig. 3. Each row is called a bit-slice and it corresponds to a row of the test matrix. The number of clock cycles required depends on the length of the largest scan chain. Figure 3 also illustrates how the multiplier can be integrated into the scan-based test environment. Once the test cube has been generated, a capture cycle applies the vector to the CUT. The response is captured into the scan cells. The response is serially sent to a multiple input signature register (MISR) for compaction. At the same time, the next vector is serially fed into the scan chains. Although the test matrices are not always solvable, the don't-care bits in the test vectors improve the chances of being able to solve them. The greater the number of don't-care bits, the fewer the number of equations, and hence the fewer the number of constraints on the v_{ij}'s. If the set of linear equations cannot be satisfied, then all n^2 bits have to be stored. In the tester the compressed data is stored first and next all the uncompressed data is stored. The reverse sequence is also possible. This eliminates the need to keep a marker bit with every test cube to indicate whether it is compressed or not. The number of compressed test cubes (COMPRESSED_COUNT) is also stored on the tester. As test cubes are applied to the CUT, the VECTOR _COUNT register in the tester is incremented. Every time the value of the VECTOR_COUNT register is compared with the value of COMPRESSED_COUNT. If they match then all the compressed test cubes have been applied and the tester switches mode to apply the remaining uncompressed test cubes. Encoding efficiency and compression can further be improved by sharing operands across multiple test cubes. However this requires indexing of the operands and for every compressed test cube two indices have to be stored to find the corresponding operands. If the number of distinct operands is *P* then $\log_2 P$ bits are required for indexing. If *K* is the number of compressible test cubes then for each of the

compressed test cubes two indices each consisting of $\log_2 P$ bits have to be stored. Let the total number of test cubes be *L*. The compression without operand sharing is given by:

$$\left(\frac{(L\times n^2)-(K\times 2n+(L-K)\times n^2)}{L\times n^2}\right)\times 100\% \quad (11)$$

If the sharing of operands is allowed then the compression is given by:

$$\left(\frac{(L\times n^2)-(P\times n+(L-K)\times n^2+2K\log_2 P)}{L\times n^2}\right)\times 100\% \quad (12)$$

Note that the *n*-bit shift register (*B*) is used to apply data serially to the multiplier and the *n*-bit register (*A*) is used to apply data in parallel to the multiplier. Both the registers are functional components of the CUT and are required. Hence no additional test specific hardware is required. However, a shadow register is required to maintain continuous flow of test data from the tester to the chip and to maximize the use of the tester bandwidth. To perform *n*×*n* multiplication *n* clock cycles are initially needed to serially fill in the *B* register and the shadow register using 2 tester channels. During the capture cycle the shadow register content is transferred in parallel to the *A* register. *n* clock cycles are required to decompress the particular test cube using the multiplier states. At the same time, next operands are loaded into the *B* register and shadow register serially. And the operation continues. Only 2 tester channels are sufficient and test data bandwidth is greatly reduced.

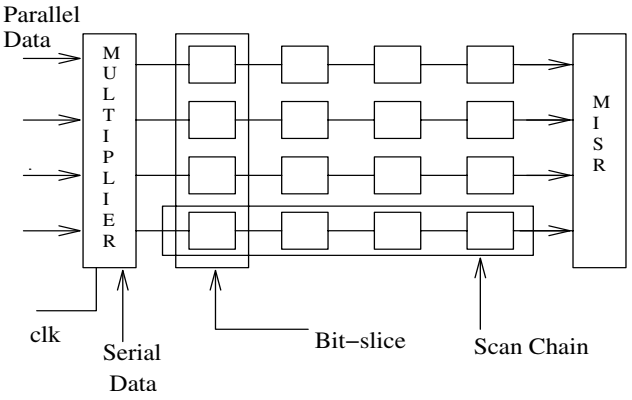

Figure 3. Integrating multiplier in scan test environment

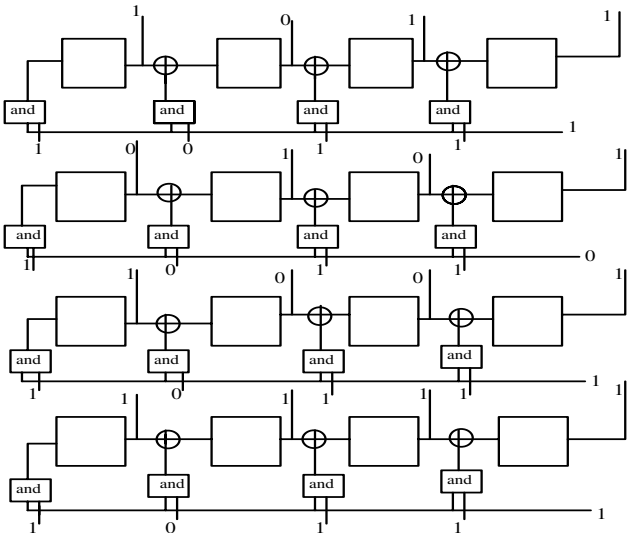

Figure 4. Multiplier states

If *K* test cubes are compressed using our proposed scheme and only *P* distinct operands are required and *(P < 2K+1),* then sharing will yield better encoding efficiency if and only if

$$(K)(2n)-(P)(n) > (2K)(\log_2 P) \quad (13)$$

Since typically the number of free variables is larger than the number of specified bits in the test cubes, there are a lot of degrees of freedom while solving the set of linear equations corresponding to the specified bits. The objective is to choose solutions which maximize the probability of sharing across different test cubes. However as operand size (*n*) increases it becomes increasingly difficult to explore exhaustively all possible solutions. Therefore we use two different strategies to increase operand sharing, one for smaller operand sizes (Strategy 1) and the other one for larger operand sizes (Strategy 2).

Strategy 1: For smaller operands *(n < 9)* it is possible to explore all the possible solutions for each of the test cubes. In this strategy, for a test cube, first the Gauss-Jordan elimination technique is applied to the set of linear equations and the pivoted matrix is obtained. Next from the pivoted matrix, mandatory assignments are found out. Mandatory assignments correspond to assignments of the from $v_{ij}=1$ or $v_{ij}=0$. These assignments could be directly derived from the singleton rows of the pivoted matrix or may result from some earlier mandatory assignments. A set of constraints are derived from these mandatory assignments e.g. $v_{ij}=1$ implies that both $a_i=1$ and $b_j=1$. Similarly $v_{ij}=0$ implies one or both of a_i *or* b_j is *0*. After fixing the bit values of the vectors *A* and *B* corresponding to the mandatory assignments, the rest of the bit positions are allowed all possible values. For each combination of *A* and *B*, the corresponding set of v_{ij}'s for all possible values of *i* and *j* is determined. Using the

0-7695-2365-X/05 $20.00 © 2005 **IEEE**

values of v_{ij}'s the set of values corresponding to t_{ij}'s is calculated. Due to large degrees of freedom, several choices of *A* and *B* vectors can match the current test cube at the specified bit positions and they all constitute the solution set for the current test cube. Out of all solutions, the one that matches with the largest number of distinct test cubes is chosen. The test cubes that are decompressible using the chosen *A* and *B* pair are removed from consideration and this process continues till the set of all test cubes becomes empty.

Strategy 2: However in larger dimensions i.e. *(n > 8),* strategy 1 becomes impractical as the number of all possible solutions becomes unacceptably large. In these cases, we followed a different strategy to maximize operand sharing across multiple test vectors. First the set of all test cubes is partitioned in smaller subgroups of size *(< 6)*. For each subgroup and for each test cube in the subgroup, the support set in terms of variables v_{ij} for each specified bit of all the test cubes in the subgroup are computed. The computed list gives the set of variables v_{ij} which impacts the specified bits and the corresponding a_i's and b_j's. The other bit positions are set to constant values and exhaustive search is initiated using only the variables in the support set. Next using a similar technique as in strategy 1, the minimal set of distinct operands is identified for each of the sub-groups. Finally all the sub-groups are merged and redundant operands are removed. Note that unlike strategy 1, this method is not optimal and different sharing can be achieved for different partitioning of the test set.

3. Experimental Results

To verify how the encoding efficiency varies for the proposed scheme for different distributions of specified bits, we first performed experiments with randomly generated test cubes. Only 2 tester channels were assumed in all the experiments and they were used to transfer the operands to the shadow register and the shift register from the tester. Test cubes were randomly generated with different distributions of specified bits. The generated test cubes were encoded using the proposed scheme. In all the experiments, 100 randomly generated vectors were used. In Table 1, scan architectures with 8, 16, 32 and 64 scan chains were used. The number of scan cells was assumed to be n^2 where *n* is the number of scan chains. For each of the scan architectures the existence of a serial multiplier unit (MU) with word-width the same as that of the number of scan chains was assumed. The randomly generated test cubes were encoded and the encoding efficiency was calculated. Table 1 shows the results with and without operand sharing. As is expected, if the number of specified bits per test cube is small then better encoding efficiency can be achieved using operand sharing as this makes use of the degrees of freedom while solving the linear equations. In some cases (denoted by *), the encoding efficiency with operand sharing is worse than without sharing. In these cases the condition of Eq. 13 is violated. For all the reported results in Table 1, all the test cubes were solvable because the number of free variables was significantly more than the number of specified bits per test cube. The results presented in Table 1 also provide an insight into choosing a particular multiplier word-size for any test set with a particular distribution of specified bits if the CUT contains multiple multiplier units. If the CUT contains only one multiplier, then its word-size is used. Higher encoding efficiency can be achieved using larger multiplier unit when the number of specified bits per test cube is small. On the other hand, if the test cubes are densely specified then smaller multiplier units provide better encoding efficiency. The empty cells in the Table 1 refer to the cases when one or more of the test cubes were not solvable. We performed a second set of experiments to compare our proposed scheme with other schemes. Note that the proposed compression/decompression scheme is targeted for circuits with a built-in serial multiplier component present, since this allows good encoding efficiency with significantly reduced test hardware. For the sake of comparison, we encoded the test cubes corresponding to four of the largest ISCAS 89 benchmark circuits. The test cubes for non-redundant faults were generated using ATALANTA**.** The multiplier size was chosen so as to maximize encoding efficiency based on the distribution of the specified bits in the test cubes. Table 2, shows the results for the proposed scheme corresponding to the ISCAS circuits. The number of scan chains is equal to the word-size of the multiplier unit being chosen. Note that using a smaller number of scan chains increases test time. Consider the circuit *s15850*. Here the number of scan cells is 611 and the existence of a *8×8* multiplier was assumed. Since the specified bit distribution is 12%, this provides the best encoding efficiency. Figure 5 shows the overall organization with 8 scan chains and 2 tester channels. The length of each test cube was made a multiple of 64 by adding extra padding bits. With padding, every test cube becomes 640-bit long. Next each test cube is partitioned into 10 64-bit blocks and each block is decompressed using the multiplier. The number of clock cycles (including capture cycle) required to decompress a particular test cube is *((10×8) + 1)).* The total number of clock cycles required to decompress the entire test set (including the setup time to fill the shift-register and shadow-register for the first time is *((142×81) + 8)* i.e. 11510. Table 3 shows a comparison of the results for the proposed scheme with a variety of other compression schemes. The best results in terms of test storage requirement are highlighted. Note that the technique proposed in [10] provides a higher encoding efficiency than all the results which are compared here. But for that scheme scan chain modification, dedicated test hardware, and a larger number of tester channels are

required. As can be seen from Table 3, in 3 out of 4 cases, the proposed scheme required least test data storage.

Table 1. Encoding efficiency for different specified bit distributions

Operand Sharing	Specified Bit Distribution	n=8 8×8	n=16 16×16	n=32 32×32	n=64 64×64
YES	1-3	.14	.30	.48	**.70**
	3-5	.23	.40	**.70**	
	5-10	.40	.58*	-	-
	10-20	.48*	-	-	-
NO	1-3	.08	.16	.32	.64
	3-5	.16	.32	.64	
	5-10	.30	**.60**	-	-
	10-20	**.60**	-	-	-

Table 2. Results for ISCAS 89 circuits

Circuit Name	Mult Used	Num. Test Cubes	Total Num Bits	Num Spec Bits	Test Storage (Bits)	Enc Eff.
s13207	32x32	255	178500	9335	16320	.58
s15850	8x8	142	86762	10452	20720	.51
s38417	8x8	105	174720	29847	48680	.61
s38584	16x16	192	281088	25636	29872	.85

Table 3. Comparing test data storage for different encoding schemes

Circuit Name	FDR Codes [3]		Seed Overlapping [5]		Proposed	
	Num. Vect.	Total Bits	Num. Vect.	Total Bits	Num. Vect.	Total Bits
s13207	236	30880	272	17970	255	**16320**
s15850	126	26000	174	**15774**	142	20720
s38417	99	93466	288	60684	105	**48680**
s38584	136	77812	215	31061	192	**29872**

4. Conclusions

In this paper, a novel low cost compression/decompression scheme has been proposed using a reconfigurable serial multiplier which is a functional component of the circuit under test. The encoding efficiency can further be improved by minimizing number of 1s in the ATPG generated vectors.

Acknowledgement

This material is based on work supported in part by Intel and in part by the National Science Foundation under Grant No. CCR-0306238.

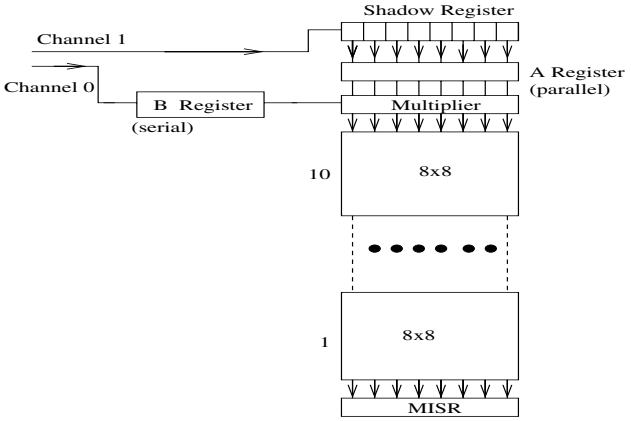

Figure 5. Scan architecture for *s15850* using 8×8 multiplier unit as decompressor

References

[1] Jas, A., and N.A. Touba, "Test Vector Decompression via Cyclical Scan Chains and Its Application to Testing Core-based Designs", *Proc. of Int. Test Conference*, pp. 458-464, 1998.

[2] Jas, A., J. Ghosh-Dastidar, and N. Touba, "Scan Vector Compression/Decompression using Statistical Coding", *Proc. of IEEE VLSI Test Symposium*, pp. 114-120, 1999.

[3] Chandra, A., and K. Chakrabarty, "Frequency-Directed Run Length (fdr) Codes with Application to System-on-a-Chip Test Data Compression", *Proc. of VLSI Test Symposium*, pp. 42-47, 2001.

[4] Wolff, F.G., and C. Papachristou, "Multiscan-based Test Compression and Hardware Decompression using lz77", *Proc. of International Test Conference*, pp. 331-339, 2002.

[5] Rao, W., I. Bayraktaroglu, and A. Orailoglu, "Test Application Time and Volume Compression through Seed Overlapping", *Proc. of Design Automation and Test in Europe*, pp. 732-737, 2003.

[6] Konemann, B., "LFSR-Coded Test Patterns for Scan Designs", *Proc. of European Test Conf.*, pp. 237-242, 1991.

[7] Cullen, C. G., *Linear Algebra with Applications*, Addison-Wesley, ISBN 0-673-99386-8, 1997.

[8] Rajski, J., et al., "Embedded Deterministic Test for Low Cost Manufacturing Test," *Proc. of Int. Test Conf.*, pp. 301-310, 2002.

[9] Konemann, B., "A SmartBIST Variant with Guaranteed Encoding", *Proc. of Asian Test Symposium.*, pp. 325-330, 2001.

[10] Krishna, C.V., and N. Touba, "3-stage Variable Length Continuous-flow Scan Vector Decompression Scheme", *Proc. of VLSI Test Symposium*, 2004.

[11] Stelling, P., C. Martel., V. Oklobdzija., and R. Ravi, "Optimal Circuits for Parallel Multipliers", *IEEE Trans. On Computer*, Vol. 47 No. 3, pp. 273-285, Mar. 1998.