Test Application Time and Volume Compression through Seed Overlapping

Wenjing Rao*
UC San Diego
CSE Department
wrao@cs.ucsd.edu

Ismet Bayraktaroglu
Sun Microsystem
Test Technology Group
ismet.bayraktaroglu@sun.com

Alex Orailoglu
UC San Diego
CSE Department
alex@cs.ucsd.edu

ABSTRACT

We propose in this paper an extension on the Scan Chain Concealment technique to further reduce test time and volume requirement. The proposed methodology stems from the architecture of the existing SCC scheme, while it attempts to overlap consecutive test vector seeds, thus providing increased flexibility in exploiting effectively the large volume of *don't-care* bits in test vectors. We also introduce modified ATPG algorithms upon the previous SCC scheme and explore various implementation strategies. Experimental data exhibit significant reductions on test time and volume over all current test compression techniques.

Categories and Subject Descriptors

B.7.3 [**Hardware**]: Integrated Circuits—*Reliability and Testing*

General Terms

Design, Reliability, Verification, Algorithm

Keywords

Test Compression, SOC Test, Deterministic Test, XOR Network, Scan Chain Concealment

1. INTRODUCTION

The rapid development of IC technologies has made it possible to put an increasing number of transistors on a single chip while at the same time delivering intense challenges to the test domain. Scan based design prevails as it enhances both controllability and observability with a limited number of input and output pins. However, in large circuits that necessitate long scan chains, scan-based design is confronted with the problem of highly increased testing time and memory requirement, which consequently leads to high test cost. To attack this problem, a variety of schemes have been proposed towards test time and volume reductions.

A straightforward yet effective approach is to cut down the number of test vectors. Through the process of test pattern generation, compaction is applied to compatible test cubes to decrease the size of the test vector set. The compaction of test vectors utilizes the fact that the majority of bits in most test cubes are unspecified, which increases the probability of mutual compatibility.

While compaction focuses on reducing the number of test vectors, compression schemes aim at reducing the number of bits needed to be stored and shifted per test vector, thus achieving a reduction in test volume and time. More generally, compression schemes try to apply a set of test vectors by inputting a reduced amount of data. Traditionally, data compression is accomplished by exploiting the redundant data or the unbalanced frequencies of various patterns. Statistical coding [1] and run-length coding [2] are well known software techniques that have been imported to the VLSI test domain for test volume compression purposes.

The fact that most bits in a test vector can be *don't-cares* turns out to be an important feature exploitable for compression as well. Hence, in the VLSI test domain, various schemes have been proposed to reduce test time and volume by utilizing the unspecified bits. The Scan Chain Concealment (SCC) compression scheme [3] uses a linear mapping network to drive a high number of internal scan chains with a short outer scan chain that is visible to the tester. The test vectors of the outer scan chain, denoted as *seed vectors*, are the compressed data that need to be stored and transferred from the tester. Each seed vector is decompressed by the linear mapping network into a much longer vector that feeds the internal scan chains. The feasibility of using a small amount of input data to generate the required test vectors stems from a well-constructed XOR linear mapping network and the large number of *don't-care* bits in the test vectors.

The ratio of input to output pins of the XOR decompression network determines the target compression ratio of the SCC design, and thus is fixed. The SCC scheme provides a threshold during the ATPG process, based on which a compromise is made between compaction and compression to attain a heuristic compression ratio. Since the ratio of *don't-care* bits in each test vector determines the maximum possible compression ratio, the flexibility in adjusting the compression ratio to the ratio of *don't-care* bits hereby promises further reductions in test time and volume. Introducing further overlapping between the seed vectors of the SCC design provides an additional dimension of compression ratio adjustment, one that is inherently determined by the number of *don't-care* bits in the test vectors to be applied.

In this paper we propose an approach to increase the compression rate for test time and test volume based on the SCC design. The approach enables the adjustment of the compression rate to fit the ratio of *don't-care* bits in test vectors. The experimental data shows evident improvements upon the previously suggested SCC technique [3], the hitherto leading test compression technique.

The paper is organized as follows: section 2 provides an outline

*The work of the first author is partially supported by a Cal-(IT)[2] Conexant Fellowship.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2003, June 2–6, 2003, Anaheim, California, USA.
Copyright 2003 ACM 1-58113-688-9/03/0006 ...$5.00.

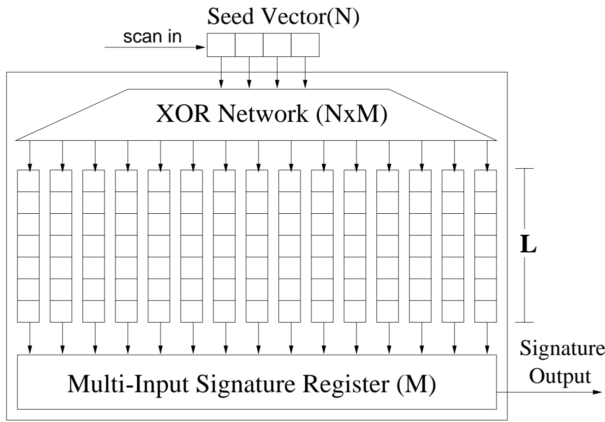

Figure 1: SCC architecture of hidden scan chains

of the previous work in the area of test time and volume reduction. In section 3, we briefly review the SCC compression scheme. The proposed approach is described in section 4 while section 5 provides the test vector generation algorithm for the proposed approach. We show some experimental data in section 6; section 7 summarizes the paper and draws a number of conclusions.

2. PREVIOUS WORK

A variety of approaches [4, 5, 6] focusing on improving test vector compaction during the test pattern generation process have been proposed. These approaches mainly aim at minimizing the test vector set by providing various heuristic ways of compacting test cubes, either statically or dynamically.

Statistical coding and run length coding such as Huffman code [1] and Golomb code [2] have been studied and applied to the area of scan chain based testing as well [1, 2]. These approaches take test vectors as data to be compressed and apply compression schemes that exploit the unbalanced frequency of different patterns. The large number of *don't-care* bits existing in test vectors are assigned to preferred values accordingly to achieve high compression ratios. Various implementations of hardware decoder architectures have been proposed as well [1, 2].

On the other hand, some compression schemes aim at exploiting the architecture of scan chain based designs [3, 7, 8, 9, 10]. A long scan chain is usually partitioned into multiple smaller internal scan chains that can be fed simultaneously. A variety of strategies are provided to supply test vectors to the multiple internal scan chains through the limited input pins. In [3] and [9], a linear decompression network composed of an XOR network [3] or a ring generator [9] is applied to provide test vectors to the internal scan chains. In the Parallel Serial Full Scan approach [7], the same test vector is shifted into all the internal scan chains in the stage called "parallel mode"; normal serial shifting is resumed in the late stage for full fault coverage. Through the Virtual Scan Chain approach [8], a set of LFSRs are assigned to each internal scan chain to supply test vectors. The seeds of the LFSRs are shifted into the chained LFSRs. In the compression approach of Test Data Mutation Code [10], every bit that needs to be flipped in a test vector is encoded. Test vectors are also reordered in a beneficial way to attain a high compression ratio. In these approaches with multiple internal scan chains, the output of the multiple inner scan chains is usually compressed by a MISR and the signature is shifted out through the main output pins.

Some approaches try to achieve compression by reusing the data within scan chains [11, 12]. By shifting in and out only a part of the scan chain, the number of bits needed to be shifted for each test cycle is reduced. In these approaches test vectors are either reordered statically by a heuristic algorithm to attain maximal overlapping [12] or generated dynamically to ensure certain overlapping with the previous response [11].

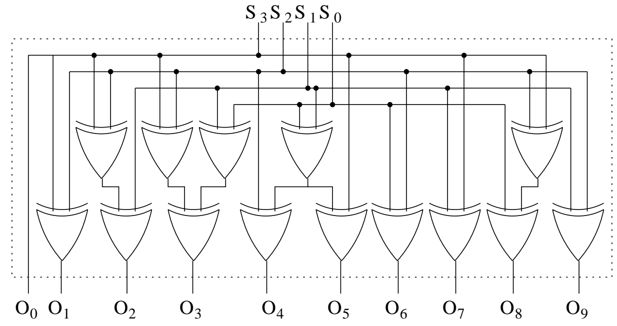

Figure 2: An example of an XOR network

3. PRELIMINARIES

The architecture of SCC design is described in this section as well as its ATPG algorithm and compression results. As is shown in Figure 1, the SCC scheme develops a decompression network which decodes an N bit seed vector and produces M outputs to feed M internal hidden scan chains on the circuit. The N-bit seed vector resides in an outer scan chain that is visible to the ATE. Each seed vector is shifted in by the ATE at the start of every test cycle. The XOR decompression network then decodes the N bits seed into an M bit output vector, denoted as an *output slice*, and shifts the slice into M internal hidden scan chains, one bit for each internal scan chain accordingly. Hence by the SCC scheme each seed vector is expanded into a slice of an internal test vector, i.e. one bit of every internal scan chain. If we suppose that the length of each internal hidden scan chain is L, then after L such test cycles, all the internal hidden scan chains will have been fed by L seeds with proper test vectors that are ready to be applied to the circuit. The test time and volume are thus reduced by the ratio of M to N.

It has been previously shown that test cubes or test vectors with a large number of unspecified bits can be encoded effectively with an LFSR seed [13, 8]. The decompression network in SCC can be built based on an LFSR sequence and implemented in parallel using an XOR network. Thus, when large ratios of unspecified bits exist within test vectors, the decompression network can drive a high number of internal scan chains with very short seed vectors. An example of an XOR network is shown in Figure 2.

The seed for a certain output slice is calculated by solving a set of linear equations, the number of which is determined by the number of specified bits in the target output slice. The coefficients of the equations, on the other hand, are determined by the construction of the XOR network. When the ratio of specified bits in a test vector is low, an XOR network delivers appreciable compression; the solution of the associated set of linear equations identifies the suitable seed for the output slice.

The SCC test vector generation algorithm starts by generating a test cube for a fault in a fault list. Then it tries to compact the test cubes for other faults in the list as long as finding a sequence of seeds to acquire the test vector is feasible. Eventually, when no more test cubes can be compacted under the constraint of solving the seed equations, a test vector is generated with its sequence of seed vectors. All the *don't care* bits in the test vector are now filled pseudorandomly by its seeds and the decompression network;

hence the test vector is fully specified. Fault simulation is then utilized to drop additional faults that can be detected by the current generated test vector from the list. The above process iterates until all (or a certain percentage of) the faults have been caught.

With the SCC compression scheme, usually an over 80% reduction in test volume and test time can be achieved. Test application time and volume reductions ranging from 12.9% to 58.1% over current schemes are reported in [3].

4. OVERLAPPING OF SEEDS

The fixed compression ratio in the SCC design constrains its capability of exploiting *don't-care* bits, therefore the maximum compression ratio cannot be reached. In this section, we describe our approach of adding flexibility to the SCC compression scheme through the overlapping of seed vectors.

During the SCC test vector generation algorithm, the *don't-care* bits in test vectors are exploited through two stages: compaction and compression. Compaction leads to the reduction of *don't-care* bits in test vectors, and it is applied under the feasibility constraint of the subsequent compression step. Thus, compaction cannot be applied *ad infinitum* but is constrained to terminate at some point as a certain minimal ratio of *don't-care* bits needs to be observed so as to be exploited by the compression scheme.

The discussion in the last section shows that the number of specified bits in an M bit output slice determines the number of equations while the seed length N determines the number of variables. To make the set of equations solvable, the number of specified bits in each M bit slice should not exceed N. Hence, the compression stage always tries to exploit a fixed $M - N$ number of *don't-care* bits. In the SCC scheme, the compression stage is restrained from obtaining a higher ratio by the fixed exploitable bits. If the number of *don't-care* bits left over by compaction exceeds $M - N$, compression only exploits the fixed $M - N$ unspecified bits, resulting in an inability to attain the maximum possible compression rate. This situation usually occurs during the later part of the test vector generation process, where faults are not "clustered" together and test cubes cannot undergo a large amount of compaction, thus condemning test vectors to a large amount of unexploitable *don't-care* bits while each detecting only a small number of new faults.

We introduce the seed overlapping scheme, which aims at providing flexibility to fit the compression ratio to the number of *don't-care* bits left over by compaction. Unlike the inner scan chains or an ordinary scan chain, an outer scan chain contains always a seed vector to be decoded for feeding multiple inner scan chains. Consequently, no response is ever written back into the outer scan chain itself. This feature makes it feasible to apply overlapping between consequent seed vectors whenever possible to exploit further compression flexibly. With the overlapping among the seed vectors, the amount of data needed to be stored and shifted from the ATE is further reduced. The amount of overlapping that is achievable between two adjacent seed vectors is inherently determined by the number of *don't-care* bits for an output slice. We show next the feasibility of overlapping between seeds in terms of matrix operations with an example.

Figure 3 illustrates a matrix representation of the XOR network example shown previously in Figure 2. If we denote the seed vector as S, the output slice vector as O and the XOR network matrix as X, then the relationship between an output slice and its seed can be expressed as:

$$[O] = [X][S]$$

Suppose the required output slice contains four specified bits, such as $O = (O_0O_1O_2O_3O_4O_5O_6O_7O_8O_9)$ =(x1xx0x1xx0).

$$\begin{aligned} O_0 &= S_3 \\ O_1 &= S_2 \oplus S_3 \\ O_2 &= S_1 \oplus S_2 \oplus S_3 \\ O_3 &= S_0 \oplus S_1 \oplus S_2 \oplus S_3 \\ O_4 &= S_0 \oplus S_1 \oplus S_2 \\ O_5 &= S_0 \oplus S_1 \oplus S_3 \\ O_6 &= S_0 \oplus S_2 \\ O_7 &= S_1 \oplus S_3 \\ O_8 &= S_0 \oplus S_2 \oplus S_3 \\ O_9 &= S_1 \oplus S_2 \end{aligned} \quad \Rightarrow \quad \begin{bmatrix} O_0 \\ O_1 \\ O_2 \\ O_3 \\ O_4 \\ O_5 \\ O_6 \\ O_7 \\ O_8 \\ O_9 \end{bmatrix} = \begin{bmatrix} 0\,0\,0\,1 \\ 0\,0\,1\,1 \\ 0\,1\,1\,1 \\ 1\,1\,1\,1 \\ 1\,1\,1\,0 \\ 1\,1\,0\,1 \\ 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 1\,0\,1\,1 \\ 0\,1\,1\,0 \end{bmatrix} \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix}$$

Figure 3: Matrix representation of an XOR network example

Thus, with only (O_1, O_4, O_6, O_9) specified to (1010), the seed can be determined by utilizing the four rows in matrix X corresponding to O_1, O_4, O_6 and O_9. The seed vector S is calculated by the reduced equation below corresponding to the rows for the specified bits (O_1, O_4, O_6, O_9):

$$S = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} 0\,0\,1\,1 \\ 1\,1\,1\,0 \\ 1\,0\,1\,0 \\ 0\,1\,1\,0 \end{bmatrix}^{-1} \begin{bmatrix} O_1 \\ O_4 \\ O_6 \\ O_9 \end{bmatrix} = \begin{bmatrix} 0\,1\,0\,1 \\ 0\,1\,1\,0 \\ 0\,1\,1\,1 \\ 1\,1\,1\,1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

The output slice is consequently fully specified by the pseudo-random fill of the XOR matrix as $[O] = [X][S] = (0100011110)$.

To show an example of overlapping, we now consider the next output slice, O', that follows O. Let O' be (xx0x1xxxxx) thus containing only two specified bits at O'_2 and O'_4. We can then form the next seed vector by only shifting in two bits upon the previous seed, where $S'_2 = S_0 = 0$ and $S'_3 = S_1 = 1$ derived from the shifting of S_0 and S_1 in the previous seed (0110). The new unspecified S'_0 and S'_1 can be calculated by solving the equation below.

$$\begin{bmatrix} O'_2 \\ O'_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0\,1\,1\,1 \\ 1\,1\,1\,0 \end{bmatrix} \begin{bmatrix} S'_0 \\ S'_1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0\,1 \\ 1\,1 \end{bmatrix} \begin{bmatrix} S'_0 \\ S'_1 \end{bmatrix} + \begin{bmatrix} 1\,1 \\ 1\,0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} S'_0 \\ S'_1 \end{bmatrix} = \begin{bmatrix} 0\,1 \\ 1\,1 \end{bmatrix}^{-1} \left(\begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 1\,1 \\ 1\,0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

By solving the equation, the output slice O' can be generated by shifting only half of the seed vector. The seed $S' = (0101)$ thus can specify the output slice O' as $O' = [X][S'] = (1100100011)$.

The example above shows the method applied so as to exploit the overlapping between seed vectors and also illustrates that the amount of overlapping between seeds matches the *don't-care* bit rate in the output slice. Instead of fully shifting in every seed vector, overlapping can be applied to every two adjacent seed vectors to reduce the required amount of shifting. In general, an XOR network can be designed in a way that introduces no linear dependency among a small number of rows in the matrix X, thus making it always feasible to generate a seed that overlaps with the previous one in the amount that matches the number of *don't-care* bits in the output slice. The flexibility of changing the compression ratio according to the variation in the *don't-care* bit ratio is therefore achieved through the proposed scheme.

To implement the seed overlapping scheme, a minor change needs to be made upon the existing SCC architecture. With the overlapping of the seed vectors, an extra information regarding the amount

of overlapping between every two seeds is required to be stored and transferred. Since the length of each seed vector is very short, the size of this information is negligible; thus no degradation in the effectiveness of the proposed scheme is incurred. From the architectural aspect, this information is passed to the circuit through either an extra pin or by making use of the signature output pin as a bi-directional pin as was originally proposed in [10].

5. TEST GENERATION ALGORITHM

In order to guarantee full coverage of faults, a test generation algorithm is needed to be incorporated within the proposed scheme. Since the proposed scheme is based on the SCC design, the main structure of the ATPG algorithm is similar to the SCC test pattern generation algorithm. However, to take into consideration the overlapping between seeds, the algorithm needs to be modified somewhat to generate proper test vectors.

It is worth noticing that, in the SCC ATPG algorithm, each test vector is made up of L output slices (assuming that the length of the internal hidden scan chains is L), which are obtained from the decompression of L consecutive seeds. The compaction stage performed in the SCC approach is based on a whole test vector, under the constraint that each output slice composing the current test vector should be generatable by a corresponding seed. The compression part, on the other hand, is always performed on a per-slice basis, which means every output slice is compressed into a seed by solving a set of equations.

With the overlapping approach, the compression part can now obtain a flexibility by compressing the output slice into a partially specified seed, which is the consequence of the partial overlap between the current and the previous seed. The overlap scheme can in an output slice exploit *don't-care* bits in the range $M - N$ to $M - 1$.

With this increased flexibility, we should consider now as to what extent compaction should be allowed to be performed. With the overlapping scheme, the constraint for compaction can be made stricter by terminating the compaction stage earlier, allowing compression to perform more exploitation and to encode a slice into a partially specified seed. We explore two strategies for the ATPG algorithm depending on how one chooses the constraint in the compaction stage.

One possible approach is to maintain the fixed constraint for compaction as was done in the SCC algorithm; thus no assumption for overlapping is presumed in the compaction stage. Compaction is always done on the basis of every test vector, i.e. L output slices; thus after the compaction every slice of a test vector should be guaranteed a seed for encoding. The compression part then tries to explore the overlapping capability between every two adjacent seeds; this is achieved by trying to perform as much overlap as possible while guaranteeing the equations to be solvable. This approach encourages compaction to be performed with the loosest constraint; thus we denote it as the *Increased Compaction Algorithm* and describe it further in subsection 5.1.

The other possible approach puts a stricter constraint on conpaction to ensure that a certain fixed amount of overlapping between adjacent seeds is always feasible. With this constraint, compaction is restrained while the compression part always attains a higher encoding ratio. We denote the second strategy as the *Increased Compression Algorithm* and provide more detail in subsection 5.2.

The computation overhead of both strategies is bounded by the complexity of the SCC ATPG algorithm, due to the short length of the seed vector.

5.1 Increased Compaction Algorithm

1. Select a fault, f, and generate a test cube C that detects it.
2. While there exists an undetected fault f' in the fault list
   (a) Generate a test cube C' for fault f'
   (b) If C' is compatible with C and the resultant test cube is compressible, then:
      i. Merge C' into C
      ii. Perform fault simulation and remove detected faults from the fault list
3. Perform maximal possible overlapping between adjacent seeds for the current test pattern C.
4. Perform fault simulation and drop detected faults from the fault list.
5. If undetected faults remain, go to step 1.

The *Increased Compaction Algorithm* is based on the strategy that tries to perform as much compaction as possible. The constraint for performing compaction is identical to that utilized in the SCC ATPG algorithm, where compression is applied assuming no overlap between seeds. Therefore, overlapping of seeds is not presumed during the compaction stage and is only subsequently attempted within every test vector during the compression stage.

In this algorithm, the iteration part of applying compaction first and compression afterwards is similar to the corresponding part in the SCC algorithm. The resultant test cube is decomposed into a sequence of output slices in step 3 and efforts are made to perform maximum overlap between adjacent seed vectors. The sequence of overlapped seeds is computed by solving a set of matrix equations. Through the expansion of the seeds performed by the decompression network, a sequence of output slices for every test cube is determined which thereby constitute a fully-specified test vector. Fault simulation is then applied to the generated test vector and fault dropping is performed to get rid of the detectable faults. This process is iterated until no detectable faults remain in the fault list, thus guaranteeing full fault coverage.

In this algorithm, the number of overlapping bits between two adjacent seeds can vary from 0 to $N-1$ in order to fit the *don't-care* bits ratio maximally; thus it is expected to obtain the most reduction in test time in terms of shifting cycles. Test volume, in terms of test data that need to be stored on the ATE, can be improved by this algorithm as well, although not as significantly as that of test time, since additional information is needed to be stored to indicate the amount of shifts needed for the seeds. Use of a limited number of discrete shifting steps enables minimization of the communication necessary for directing the amount of shift, albeit at the cost of a slight deterioration in application time improvement.

5.2 Increased Compression Algorithm

1. Select a fault, f, and generate a test cube C that detects it.
2. While there exists an undetected fault f' in the fault list
   (a) Generate a test cube C' for fault f'
   (b) If C' is compatible with C and the resultant test cube is compressible to a sequence of seeds overlapping by a prespecified amount, then:
      i. Merge C' into C
      ii. Perform fault simulation and remove detected faults from the fault list

	Increased Compression Algorithm					*Increased Compaction Algorithm*									
						none or half overlapping					any length of overlapping				
	Ptn	Vol compress		Time compress		Ptn	Vol compress		Time compress		Ptn	Vol compress		Time compress	
circuit	num	#bits	cmp	#bits	cmp	num	#bits	cmp	#bits	cmp	num	#bits	cmp	#bits	cmp
S13207	272	18828	88.9%	18012	89.4%	255	19137	88.7%	18372	89.2%	255	**17970**	**89.4%**	**14145**	**91.6%**
S15850	174	**15774**	**83.0%**	15252	83.5%	166	16002	82.7%	15504	83.2%	169	16454	82.2%	**13919**	**85.0%**
S35932	34	**5264**	**86.2%**	**4992**	**86.9%**	31	6692	82.4%	6444	83.0%	32	7312	80.8%	6032	84.2%
S38417	288	**60684**	**80.3%**	58380	81.0%	274	61052	80.2%	58860	80.9%	276	63833	79.2%	**52793**	**82.8%**
S38584	215	**31061**	**88.0%**	29556	88.5%	170	31814	87.6%	30624	88.1%	177	32839	87.2%	**26644**	**89.6%**
Ave	197	**26322**	**85.3%**	25238	85.9%	179	26939	84.3%	25961	84.9%	182	27682	83.8%	**22707**	**86.6%**

Table 1: Comparison among different strategies of test vector generation algorithms

3. Randomize the unspecified bits of the resultant test cube
4. Perform fault simulation and drop detected faults from the fault list.
5. If fault coverage threshold is not attained, go to step 1.
6. Perform the *Increased Compaction Algorithm* for all the undetected faults in the fault list.

This algorithm tries to impose a stricter constraint for compaction to guarantee improved compression. A fixed amount, typically half, of overlapping is set as a constraint for compaction up to a predefined percentage of fault coverage. However, it is occasionally hard to cover all the faults with the stricter constraint; thus the constraint is subsequently loosened to enable the full shifting of seed vectors and the previous strategy, *Increased Compaction Algorithm*, is applied to cover the remaining faults.

With the stricter constraint during the compaction stage, the number of test patterns, as well as test time, that are needed for this algorithm may increase slightly. Therefore, the threshold of the fault coverage percentage needs to be adjusted to attain an optimal solution. However, because of the fixed amount of overlapping, less information is required to indicate the overlapping amount; thus this algorithm is expected to obtain a higher reduction in test volume.

6. EXPERIMENTAL DATA

In our experimental framework, we utilize ATALANTA [14] as the test cube generation tool and HOPE [15] for fault simulation. Various seed overlapping compression algorithms are implemented in C programs and are based on a reimplementation of SCC compression scheme. We perform our scheme on the largest five ISCAS89 benchmark circuits with a randomly generated XOR network of size 24 × 200. Hereby, N equals to 24 and M equals to 200. The length of each internal scan chain, L, is 4 for the circuits S13207 and S15850, 9 for the circuits S35932 and S38417 and 8 for the circuit S38584.

In Table 1 we show a comparison among various test vector generation strategies. The *Increased Compression Algorithm* is shown on the left part and it is implemented by imposing half seed overlapping as the strict constraint for compaction. The *Increased Compaction Algorithm* is shown on the right part of Table 1. Within the *Increased Compaction Algorithm*, two kinds of implementations are explored. The one shown rightmost allows the compression stage to perform any length of overlapping from 0 to 23 between two adjacent seeds. Thus, this implementation exploits all the possibilities of compression ratios and is expected to obtain a maximum time reduction as discussed. The other one only allows a fixed amount of overlapping (half overlapping) between every adjacent seed when performed by the compression stage. Hence, either none or 12 bits of overlapping are applied, and the amount of data needed to indicate the length of shifting is decreased.

We list experimental data obtained from each strategy in Table 1. The column named "*Ptn num*" shows the number of test patterns required for each strategy. The remaining columns contain data for test volume and test time compression. The column indicated by "*#bits*" shows either the test volume in terms of the number of bits or test time in terms of the number of cycles. Test volume is usually higher due to the extra information needed for indicating the amount of overlapping. The column denoted "*cmp*" lists the reduced cycle or bit number of the proposed scheme as a percentage of the plain implementation which shifts all the test patterns sequentially with no compression scheme applied. The number of test patterns necessary for a plain implementation with no shifting is obtained from [3], and the test time and volume are calculated as the multiplication of the test pattern number and the scan chain length. The best result for each circuit is indicated in bold phase.

The experimental data is consistent with our analysis. In general, the *Increased Compression Algorithm* generates slightly more test patterns. On the other hand, as indicated by the number of bits in test volume, the *Increased Compression Algorithm* performs better than the *Increased Compaction Algorithm*. Also, the "none or half overlapping" strategy outperforms the "any length of overlapping" strategy in the *Increased Compaction Algorithm* in test volume as expected, due to the reduced information on the overlapping amount. However, with the increased freedom to perform any length of overlapping, the "any length of overlapping" strategy achieves the best test time compression rate in most cases. The circuit S35932 provides an exception due to the regularity characteristic of the circuit that leads to its faults being trivial to detect. Thus, the faults can be covered by a very small set of test vectors. In spite of the difference, all strategies show a high reduction of around 85% on both test time and volume compared to a plain scheme with no compression.

In Table 2 we show the comparison of the proposed scheme to previous results in test application time reduction. The results from the *Increased Compaction Algorithm* are used for the proposed scheme to aim at the best reduction for test application time. Other previous work included in the comparison are the *Parallel*

circuit	PSFS [7]	FDR [16]	SCC [3]	VSC [8]	Proposed	imp
S13207	82546	**20368**	25344	60894	14145	30.6%
S15850	76030	**21590**	22784	38010	13919	35.5%
S35932	9136	20946	**7218**	N/A	4492	38.2%
S38417	127932	**57066**	89856	154254	52793	7.5%
S38584	129580	70328	**38796**	101185	26644	31.3%
Ave	85045	38060	**36782**	88586	22705	28.6%
imp	73.3%	40.3%	38.3%	74.4%		

Table 2: Test time reduction comparison with previous work

circuit	SCC [3]	without over-lapping	overlapped scheme			
			test volume		test time	
			#bits	imp	#bits	imp
S13207	25344	24480	17970	26.6%	14145	42.2%
S15850	22784	15936	15774	1.0%	13919	12.7%
S35932	7128	6696	5264	21.4%	4492	32.9%
S38417	89856	58356	60684	-3.7%	52793	9.8%
S38584	38976	33024	31061	5.9%	26644	19.3%
Ave	36818	27734	26151	10.2%	22399	23.4%

Table 3: Reduction attributable solely to overlapping

Serial Full Scan (PSFS) [7], the *Frequency Directed Run-Length (FDR) coding* [16], the *Scan Chain Concealment (SCC)* [3] and the *Virtual Scan Chain (VSC)* [8]. As the *Embedded Deterministic Test* [9] utilizes industrial benchmarks not publically available, no comparison can be given here. The best previous result for each circuit is shown in bold font, and an improvement percentage is listed in the last column based on the comparison to the best previous result for each circuit. The last row shows the improvement on average made by the proposed scheme compared to each of the previous works. Significant improvements, ranging from 38.3% over *SCC* [3] to 74.4% over *VSC* [8], are observed.

To further investigate the reduction solely attributable to overlapping, we have performed a non-overlapping experiment with our implementation of the SCC scheme. Due to the difference[1] in implementation to ensure equitable comparison, the SCC approach we have re-implemented outperforms the original SCC on test time and volume in [3]. A significant improvement in test time due to the proposed seed overlapping can still be observed, nonetheless. Test volume, although slightly increased by the extra information needed to indicate the amount of overlapping, still shows strong improvements in general. The detailed results are shown in Table 3.

In this table, the column "*SCC [3]*" shows the results from the original SCC scheme as outlined in [3] while the column "*without overlapping*" indicates the enhanced implementation of SCC yet without performing any overlapping. The rest of the columns show the results of the proposed approach from the perspectives of both test volume and test time. Test volume and time in terms of bit number and shifting cycles are shown in column "*#bits*". The column "*imp*" shows improvement ratios upon the approach without overlapping (i.e. column 3).

7. CONCLUSION

A compression scheme based on the SCC design for test time and volume reduction is proposed. Through the overlapping of seeds, the compression ratio is capable to be adjusted according to the *don't-care* ratio of test vectors. The flexibility thereby achieved makes it possible to maximize the compression ratios of test time and test volume. The methodology of performing seed overlapping is provided in this paper and several strategies for test vector generation algorithms are discussed accordingly.

We provide three sets of experimental results. The set of results for comparison among various test pattern generation algorithm strategies exhibits consistency with our analysis, thus providing valuable information for a practical approach. Another set of data provides comparisons between the proposed scheme and previous work, proving attainment of remarkably elevated compression ratios with the proposed scheme as well as significant improvement compared to previous related work. The third set of results clearly confirms the promising improvement achieved through the proposed overlapping scheme.

The suggested approach can be incorporated within the SCC scheme as well as other decompression network based compression techniques, while from the ATE point of view the test procedure remains unaltered. Consequently, the significant improvement in test time and volume reduction made by the proposed scheme as well as its compatibility to other techniques make it promising in the test compression domain.

[1] The variance originates from a difference in the outer scan chain length and code parameter definitions.

8. REFERENCES

[1] A. Jas, J. Ghosh-Dastidar and N. A. Touba, "Scan Vector Compression/Decompression Using Statistical Coding", in *VTS*, pp. 25–29, 1999.

[2] A. Chandra and K. Chakrabarty, "System-on-a-Chip Test-Data Compression and Decompression Architectures Based on Golomb Codes", *TCAD*, vol. 20, n. 3, pp. 355–368, March 2001.

[3] I. Bayraktaroglu and A. Orailoglu, "Test Volume and Application Time Reduction Through Scan Chain Concealment", in *DAC*, pp. 151–155, 2001.

[4] I. Hamzaoglu and J. Patel, "Test Set Compaction Algorithms for Combinational Circuits", in *ITC*, pp. 283–289, 1998.

[5] I. Pomeranz, L. Reddy and S. Reddy, "COMPACTTEST: a Method to Compact Test Sets for Combinational Circuits", *TCAD*, vol. 12, n. 7, pp. 1040–1049, July 1993.

[6] S. Bhatia and P. Varma, "Test Compaction in a Parallel Access Scan Environment", in *ATS*, pp. 300–305, 1997.

[7] I. Hamzaoglu and J. Patel, "Reducing Test Application Time for Full Scan Embedded Cores", in *FTCS*, pp. 260–267, 1999.

[8] A. Jas, B. Pouya and N. A. Touba, "Virtual Scan Chains: A Means for Reducing Scan Length in Cores", in *ITC*, pp. 73–78, 2000.

[9] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide and J. Qian, "Embedded Deterministic Test for Low Cost Manufacturing Test", in *ITC*, pp. 301–310, 2002.

[10] S. Reda and A. Orailoglu, "Reducing Test Application Time Through Test Data Mutation Encoding", in *DATE*, pp. 387–393, 2002.

[11] W. Rao and A. Orailoglu, "Virtual Compression through Test Vector Stitching for Scan Based Designs", in *DATE*, 2003.

[12] C. Su and K. Hwang, "A Serial Scan Test Vector Compression Methodology", in *ITC*, pp. 981–988, 1993.

[13] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman and B. Courtois, "Built-in Test for Circuits with Scan Based on Reseeding of Multiple-polynomial Linear Feedback Shift Registers", *IEEE Transactions on Computers*, vol. 44, n. 2, pp. 223–233, February 1995.

[14] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits", Technical report 12-93, Dep't of Electrical Eng., Virginia Polytechnic Institute and State University, 1993.

[15] H. K. Lee and D. S. Ha, "HOPE: An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits", in *DAC*, pp. 336–340, 1992.

[16] A. Chandra and K. Chakrabarty, "Test Resource Partitioning for SOCs", *IEEE Design & Test of Computers*, vol. 18, n. 5, pp. 80–91, September/October 2001.

[17] I. Bayraktaroglu and A. Orailoglu, "Decompression Hardware Determination for Test Volume and Time Reduction through Unified Test Pattern Compaction and Compression", in *VTS*, 2003.