Utilizing ATE Vector Repeat with Linear Decompressor for Test Vector Compression

Joon-Sung Yang, *Member, IEEE*, Jinkyu Lee, *Member, IEEE*, and Nur A. Touba, *Fellow, IEEE*

***Abstract*—Previous approaches for utilizing automatic test equipment (ATE) vector repeat are based on identifying runs of repeated scan data and directly generating that data using ATE vector repeat. Each run requires a separate vector repeat instruction, so the amount of compression is limited by the amount of ATE instruction memory available and the length of the runs (which typically will be much shorter than the length of a scan vector). In this paper, a new and more efficient approach is proposed for utilizing ATE vector repeat. The scan vector sequence is partitioned and decomposed into a common sequence which is the same for an entire cluster of test cubes and a unique sequence that is different for each test cube. The common sequence can be generated very efficiently using ATE vector repeat. Experimental results demonstrate that the proposed approach can achieve much greater compression while using many fewer vector repeat instructions compared with previous methods.**

***Index Terms*—Linear decompres, test vector compression, vector repeat.**

I. Introduction

TEST cost in the integrated circuit (IC) industry has increased dramatically to achieve high test quality as size and complexity continue to grow [1]. Large and complex ICs such as System-on-a-Chip (SoC) and three-dimensional ICs (3-D-ICs) have caused a drastic increase in test cost. One of the critical factors is large test data volume. To obtain high test quality, test data volume may exceed the memory capacity of the available automatic test equipment (ATE). Furthermore, the large amount of test data needs to be transferred from the ATE to a chip with limited tester bandwidth, which results in long test time. To overcome increased test memory requirements and tester data bandwidth requirements, test vector compression has become very important. Test vector compression provides a way of reducing both the tester memory requirement and the tester data bandwidth requirement. A number of test vector compression techniques have been proposed in the literature [30].

A special class of test vector compression schemes involves using a linear decompressor which uses only linear operations to decompress the test vectors. This includes techniques based on linear feedback shift register (LFSR) reseeding and combinational linear expansion circuits consisting of XOR gates. Linear compression schemes are very efficient in exploiting don't care values in test cubes to achieve large amounts of compression.

Linear decompressors expand seeds to deterministic test cubes [2]–[6], [7], [8], [10], [11]–[17], [28]. A seed is an initial state of a linear decompressor that is expanded by running the linear decompressor. Given a deterministic test cube, a corresponding seed can be computed by solving a set of linear equations (one equation for each specified bit) based on the feedback polynomial of linear decompressor. Since typically only 1–5% of the bits in a test vector are specified, most bits in a test cube do not need to be considered when a seed is computed because they are don't care bits. Therefore, the size of a seed is much smaller than the size of a test vector. Consequently, linear decompressors can significantly reduce test data storage and bandwidth.

The amount of compression achieved with linear compression schemes depends directly on the number of specified bits in test cubes. While linear decompressors are very efficient in exploiting don't cares in test set, they cannot exploit correlations in the test cubes. Hence, they cannot compress test data less than the total number of specified bits in the test data. Nonlinear decompressors on the other hand can exploit correlations in test cubes, but are not as efficient as linear decompressors in exploiting don't cares. Because test data is typically only 1–5% specified with the rest as don't cares, linear decompressors are generally more effective overall. This fact coupled with the simple and compact design of linear decompressors is the main reason why they are used in commercial tools.

One instruction that is commonly found in ATEs is vector repeat which allows the ATE to repeat a sequence n times. ATE stores a single test vector and an instruction to repeat this vector n times. A sequence of n identical test vectors is applied by vector repeat [22]. The approach taken in this paper is to utilize the vector repeat function on top of a linear decompressor to get the advantages of both the ATE vector repeat and the linear decompressor. The ATE vector repeat is used to exploit correlations in the specified bits to reduce the number of specified bits that the linear decompressor has to produce. Since the amount of compression achieved with a linear decompressor depends on the number of specified bits it needs to produce, this approach results in much greater compression than what the linear decompressor could achieve by itself (preliminary results were presented in [18]).

Manuscript received April 17, 2013; revised September 4, 2013 and January 7, 2014; accepted February 19, 2014. Date of current version July 15 2014. This paper was recommended by Associate Editor A. E. Gattiker.

J.-S. Yang is with Sungkyunkwan University, Suwon 440-746, Korea (e-mail: js.yang@skku.edu).

J. Lee is with Samsung Austin Research Center, Austin, TX 78730 USA (e-mail: jinkyu.l@samsung.com).

N. A. Touba is with the University of Texas at Austin, Austin, TX 78712 USA (e-mail: touba@ece.utexas.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2014.2314307

0278-0070 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Section II provides a review of the related work. Section III presents the proposed scan data decomposition method. Section IV describes the hardware architectures of the proposed scheme. Section V illustrates the clustering algorithm for test cubes, and the ATE architectures are shown in Section VI. Experimental results are shown in Section VII and conclusion is given in Section VIII.

II. Related Work

Previous research has proposed ways to use the ATE vector repeat mechanism to reduce vector memory requirements for scan testing. In this paper, a new and more efficient approach is proposed for utilizing ATE vector repeat.

A test cube is a deterministic test vector in which the inputs that are not assigned during automated test pattern generation (ATPG) are left as don't cares. Normally, random fill is performed where the don't cares are filled randomly with 1s and 0s to increase the chance of detecting additional faults. In [19], a methodology for utilizing ATE vector repeat was described. The idea is that instead of doing random fill of the test cubes, repeat fill is done where don't cares are filled by repeating the last specified bit within the same scan chain. This creates runs of repeated values in the scan chains. A scan slice is defined as the n-bits loaded in parallel from the tester into n scan chains each clock cycle. If multiple consecutive scan slices are identical, then ATE vector repeat can be used to generate them. Only one copy of the scan slice data needs to be stored in the ATE vector memory, and then a vector repeat instruction is stored in the ATE instruction memory. There is a limit to how much ATE vector repeat can be used based on the amount of instruction memory that is available. Consequently, ATE vector repeat is only used for the longest runs of repeated scan slices and not all runs.

In [20], ATE vector repeat is used for repeating full scan vectors during a transition test. For transition tests and other two-patterns tests, the same full scan vector may be repeated in the scan chain if it is used as the launch (V_2) vector for 1 two-pattern test and then as the initialization (V_1) vector for the next two-pattern test. In [21], transition test chains where only the very first and last vectors in a sequence are not repeated are formed to maximally utilize ATE vector repeat to reduce ATE storage.

In [22], ATE vector repeat per pin-group is investigated. In [19], a single ATE vector repeat instruction applies to all pins. However, some testers provide the ability to specify pin-groups for which to apply the ATE vector repeat instruction. It was shown in [22] that by using ATE vector repeat on smaller pin-groups, the length of the runs increases which allows greater compression.

In [23], ATE vector repeat is used in conjunction with a scan slice encoding scheme. Each scan slice is encoded into a sequence of smaller codewords, and an on-chip decoder is used to expand the codewords into the original scan slices. In this scheme, some codewords may be repeated, so ATE vector repeat is used for runs of repeated codewords.

In [28] and [29], compression of incompatible test pattern is proposed. It allows conflicts on certain positions among test cubes and generates one parent pattern. Control patterns and incremental patterns are also considered to recover content of the original test cube. Because the parent pattern includes conflicts in test cubes, control pattern, and incremental patterns help to recuperate the patterns like the original ones. Hence, the compression ratio varies with the number of allowed conflicts in test cubes and decompressor logic is fairly complicated to handle the parent pattern, incremental pattern, and control pattern.

III. Decomposing Scan Data

Previously proposed approaches for utilizing ATE vector repeat are all based on identifying runs of repeated scan data and directly generating that data using ATE vector repeat. The amount of compression is limited by the amount of ATE instruction memory available and the length of the runs. This paper proposes a new and more efficient way of utilizing ATE vector repeat. The idea is to exploit the fact that many test cubes have similar input assignments due to the fact that they are structurally related in the circuit. In the proposed scheme, the set of test cubes in a test set are partitioned into clusters that share many input assignments. The scan sequence is then decomposed into two components: the sequence of specified bits that is common across all the test cubes in a cluster, and the sequence of specified bits that is unique to each test cube.

An example is shown in Fig. 1 to illustrate how the scan data is decomposed. Assume that the eight test cubes shown in Fig. 1 are included in one cluster. Each bit position in a test cube cluster can be classified as either being a don't care if no test cube has a specified value in that bit position, having "common data" if all test cubes have compatible values in that bit position, or having "unique data" if two or more test cubes have conflicting specified values. In the example, the last bit position is a don't care. The 1st and 3rd bit positions have compatible value across all of the eight test cubes and thus are common data. The common data can be generated by the common sequence decompressor that operates based on ATE vector repeat since it is the same for each test cube. The 2nd, 4th, 5th, 6th, and 7th bit positions have conflicting values and thus are unique data. They must be generated by the unique sequence generator (USG). The common data for the test cube cluster is shown in Fig. 1 along with the unique data for each test cube.

Because the scan data is decomposed into common data and unique data, a control signal is required to indicate if a bit position should be filled from the common data or the unique data. This is illustrated in Fig. 2. The control signal is a don't care for any don't care bit position in a cluster (in the example in Fig. 1, only the last bit position is a don't care), and it has a specified value for all other bit positions. A key property is that the same control sequence can be used when decompressing all test cubes in a cluster and thus it is a "common control." This means that the control signal can be generated by the common sequence generator (CSG) using ATE vector repeat and thus the storage required for the common control is amortized across all the test cubes in the cluster. The common control sequence for the example test cube cluster is shown in Fig. 1.

Original								
Test cube 1	0	1	1	1	1	0	1	x
Test cube 2	0	0	1	1	1	0	1	x
Test cube 3	0	1	1	1	1	0	0	x
Test cube 4	0	1	1	0	0	1	1	x
Test cube 5	0	0	x	1	1	0	0	x
Test cube 6	0	1	1	0	1	1	1	x
Test cube 7	x	1	1	1	0	1	1	x
Test cube 8	x	1	1	1	1	1	1	x
Encoded								
Common control	1	0	1	0	0	0	0	x
Common data	0	x	1	x	x	x	x	x
Unique data 1	x	1	x	1	1	0	1	x
Unique data 2	x	0	x	1	1	0	1	x
Unique data 3	x	1	x	1	1	0	0	x
Unique data 4	x	1	x	0	0	1	1	x
Unique data 5	x	0	x	1	1	0	0	x
Unique data 6	x	1	x	0	1	1	1	x
Unique data 7	x	1	x	1	0	1	1	x
Unique data 8	x	1	x	1	1	1	1	x

Fig. 1. Example of proposed encoding scheme.

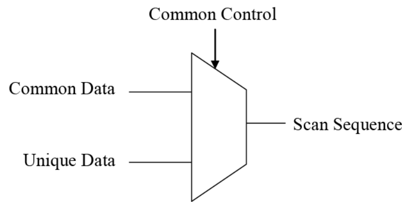

Fig. 2. Multiplexer implementation.

Typically a large number of specified bits can be included in the common data because many test cubes have similar input assignments due to the fact that they are structurally related in the circuit. The clustering procedure described in Section V selects the clusters to maximize the amount of compression for the cluster. During decompression, the CSG produces both the common data and common control. Only one copy of an input stream for the CSG needs to be stored in ATE vector memory since it can be applied using an ATE vector repeat instruction for all test cubes in the cluster. Therefore, a large reduction in storage requirements can be achieved.

In the example in Fig. 1, the number of specified bits in the original test cubes is 53, and the number of specified bits in the encoded test cubes is 49 (2 specified bits in the common data, 40 specified bits in the unique data, and 7 specified bits in the common control). The reduction in the example shown in Fig. 4 is small, but in real cases, the number of test cubes in a cluster is much greater than the number of test cubes in this example, so the number of specified bits generated from the common data is much higher, thereby making the reduction in the total number of specified bits larger.

IV. Decompression Hardware

Section III describes how the scan sequence is decomposed into common data and unique data. Two separate on-chip decompressors are then used to generate these two sequences

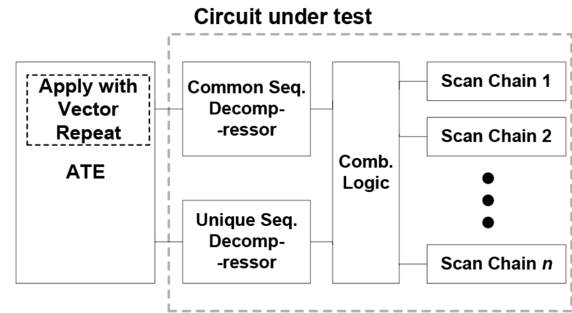

Fig. 3. Diagram of proposed decompression hardware.

as illustrated in Fig. 3. Since the common sequence is the same for all test cubes in a cluster, the input stream to its decompressor can be generated using ATE vector repeat. Only one copy of this input stream needs to be stored in the ATE vector memory, and only one ATE vector repeat instruction needs to be stored in the ATE instruction memory for decompressing the entire test cube cluster. For the unique sequence corresponding to each test cube, it is generated with its own decompressor. The unique sequence will only contain very few specified bits because most of the specified bits will be generated by the common sequence decompressor. If a linear decompressor based on dynamic LFSR reseeding such as those described in [6]–[9] is used, the amount of compression depends only on the number of specified bits in the sequence. Note that the design of these decompressors is independent of the test set, so they can be reused when testing multiple cores in a SoC design. The diagram of the decompression hardware is shown in Fig. 3.

In comparing the proposed approach for using ATE vector repeat with the previous approaches, it has several advantages. Previous approaches can only generate runs of repeated values which typically will be much shorter than the length of a scan vector, and each run requires a separate vector repeat instruction. The proposed approach generates the common data component for a cluster of test cubes, and only one vector repeat instruction is needed for each test cube cluster. Consequently, the proposed approach is much more efficient in utilizing the ATE vector repeat instructions. For the limited ATE instruction memory available, the proposed approach will be able to achieve greater compression. The cost of the proposed approach is the need for on-chip decompressors. However, the on-chip decompressors can efficiently exploit the large percentage of don't care bits in test cubes to achieve very high compression. Compared with a conventional linear decompression alone, the proposed use of ATE vector repeat provides a significant improvement in the amount of compression.

This paper proposes two decompression architectures to implement the proposed scheme depending on the ATE vector repeat functionality. Fig. 4 illustrates vector transfer through vector repeat from an ATE that supports repeat-per-pin-group to CUT. In this type of ATE, during execution of the vector repeat instruction, a group of tester pins can be loaded by the ATE vector repeat while the other pins are loaded in a conventional way. In some cases, a given ATE may support

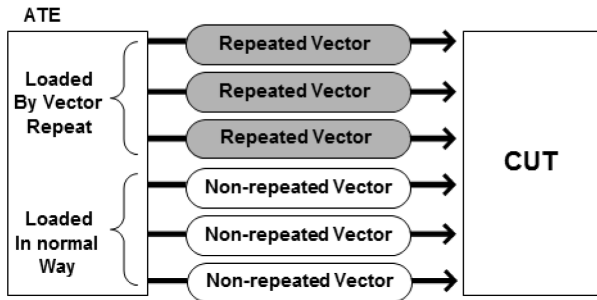

Fig. 4. ATE supporting repeat-per-pin-group.

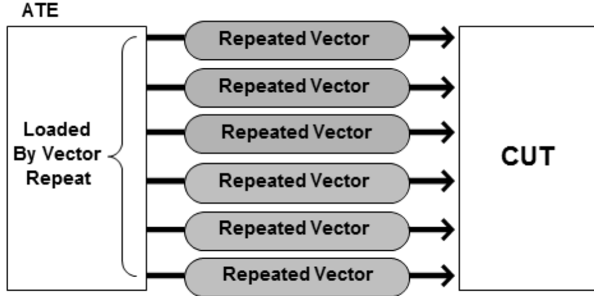

Fig. 5. ATE supporting only repeat-per-all-pins.

only repeat-per-all-pins, and does not support repeat-per-pin-group. In this case, all pins are always loaded through vector repeat during execution of the vector repeat instruction as illustrated in Fig. 5.

The proposed scheme requires relatively simple decompression hardware (two sequential linear decompressors, one-bit flip-flop, one NOR gate, and one MUX per scan chain).

A. Decompression Logic for ATE with Repeat-Per-Pin-Group

The decompression hardware is shown in Fig. 6. There are two types of memory in the ATE, instruction memory and vector memory. The instruction memory stores the ATE instructions including the vector repeat instructions, and the vector memory stores the data. The data is transferred to the decompressors based on ATE instructions. There are two sequential linear decompressors, *CSG* and *USG* in Fig. 6. *CSG* operates with ATE vector repeat and gets repeated vectors (*RV*) and the other decompressor operates without the ATE vector repeat and gets non-repeated vectors (*N-RV*). The decompressor that operates with vector repeat generates the common data and common control signals for each scan chain. Only one copy of the input stream for generating the common data and common control for all the test cubes in a cluster is stored in the ATE and applied repeatedly for each test cube in the cluster using an ATE vector repeat instruction. The other decompressor, *USG* generates the unique data and operates without vector repeat. This decompressor operates in the same way as the decompressor in conventional linear compression schemes. A 2-to-1 multiplexer is placed between the decompressors and each scan chain. Note that the size of the *USG* is much smaller than the size of *CSG* because the number of specified bits in the unique data of each test cube is much smaller than the number of specified bits in the common data and the common control.

There is a flip-flop that contains *repeat disable* bit. The *repeat disable* bit is used to load test cubes that have low correlation in the same way as the decompressor in conventional linear compression scheme. For the lowly-correlated test cubes, there is no benefit to use the proposed scheme. One specified bit per test cube is added to indicate if a test cube being loaded currently is a highly-correlated one or not. If it is a highly-correlated test cube, then the repeat disable bit is loaded with zero at the beginning of when the test cube is loaded and each MUX select bit is decided by the common control bit from *CSG*. If it is a lowly-correlated test cube, then the repeat disable is loaded with 1 and all the MUX select bits are set to 0, which makes the scan chains only loaded by *CSG* in a same way as a conventional linear decompressor regardless of the common control bit values.

One very nice property of sequential linear decompressors is that regardless of how many output signals they generate or how long of a sequence they generate, the number of input bits required for the decompressor depends only on the total number of specified bits that it needs to generate (the rest of the bits are essentially filled with random data). Thus, the architecture can be easily scaled to any number of scan chains limited only by the rate at which data from the ATE can be transferred to the sequential linear decompressors relative to the number of specified bits that the decompressor needs to generate. Note that the decompression hardware does not depend on the circuit or test set, which makes it possible to reuse it when testing multiple cores in a SoC design. The sequential linear decompressors that are used for this scheme could be any of the ones described in [6]–[9]. An example of the sequential linear decompressor is shown in Fig. 7.

B. Decompression Logic for ATE with Repeat-Per-All-Pins

The decompression logic for ATE with repeat-per-all-pins is almost the same as the decompression logic for ATE with repeat-per-pin-group and is shown in Fig. 8. In the ATE environment that supports repeat-per-pin-group, the tester pins that are transferred by vector repeat load the sequential linear decompressor (*CSG*) that generates common data and common control and the other test pins that are transferred conventionally load *USG* that generate unique data. However, in the ATE environment that does not support repeat-per-pin-group (support only repeat-per-all-pins), a sub-group of tester pins cannot be transferred via vector repeat. The vector repeat function in this type of ATE covers either all the tester pins or none of the pins.

In this ATE environment, the seed for *USG* for generating unique data is transferred in a conventional way first. The seed is to create the unique data not for one test cube, but for as many test cubes as the *USG* can. The number of test cubes whose unique data can be generated from a single *USG* seed depends on the size of the *USG* and the number of specified unique data bits in the test cubes. In Section V, it is described how to maximize the number of test cubes whose unique data bits are generated from a single *USG* seed.

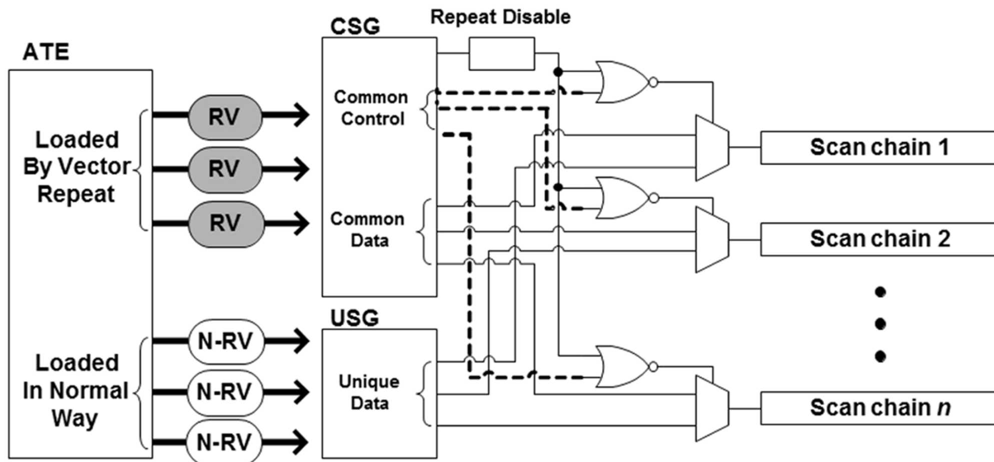

Fig. 6. Block diagram for decompression logic for ATE with repeat-per-pin-group.

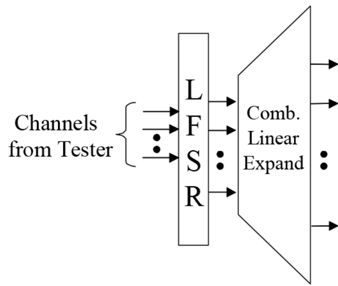

Fig. 7. Example of sequential linear decompressor.

Assume that the unique data bits for *m* test cubes can be generated from a given *USG* seed and *n* test cubes among *m* test cubes are in one cluster and (*m*-*n*) test cubes are in the other cluster. Then, after loading *USG* with a seed, *CSG* is loaded with one seed *n* times repeatedly and then with the other seed (*m*-*n*) times repeatedly via the vector repeat.

An example is shown in Fig. 9. There are seven test cubes ($t1 \sim t7$) in the test set and the test cubes are grouped into three clusters. *t*1 and *t*2 are in the first cluster, *t*3, *t*4, and *t*5 are in the second cluster, and *t*6 and *t*7 are in the third cluster. Assume that a given *USG* is able to generate at most six specified bits with single seed. With the *USG*, the unique data for the first cluster (*t*1, *t*2) and for the third cluster (*t*6, *t*7) can be created with one *USG* seed and the other seed creates the unique data for the second cluster (*t*3, *t*4, *t*5). The ATE instruction and data flow for this example is shown in Fig. 10. At first, the tester vector (*v*1, shaded in Fig. 10) to create the unique data for *t*1, *t*2, *t*6, and *t*7 is transferred to *USG*. When *v*1 is sent, the repeat disable signal is also being sent for CSG clock gating in order to prevent *v*1 being shifted to CSG. The next tester vector (*v*2) transferred to *CSG* is to generate common control and common data. Because *t*1 and *t*2 share the common control and common data, *v*2 is transferred to *CSG* one more time as shown in Fig. 10. When sending seeds to CSG, a repeat disable signal is transferred from ATE and this gates USG and holds USG data. The corresponding test cubes loaded into scan chains at a certain time are also shown in Fig. 10. Because *USG* loaded with *v*1 is able to generate unique data for *t*6, and *t*7, *v*3 generating common control and common data for *t*6 and *t*7 is followed by *v*2. Then the *USG* should be refreshed with new data (*v*4, shaded in Fig. 10) for creating unique data for *t*3, *t*4, and *t*5. After *USG* finishes getting *v*4, *v*5 that creates common control and common data for *t*3, *t*4, and *t*5 through *CSG* is transferred. Details about how to cluster test cubes and how to order the clusters are explained in Section V.

Compared to the decompression logic for ATE with repeat-per-pin-group, the decompression logic for ATE with repeat-per-all-pins is larger due to larger size of *USG*. The size of *CSG* is identical or slightly smaller. To reduce the hardware overhead of *USG*, the number of specified unique data bits needs to be reduced. A way to reduce the number of specified unique data bits with cost of increased number of clusters is introduced in Section V.

V. Forming Test Cube Cluster

The concept of clustering test cubes to exploit similar input assignments has been previously investigated in the context of built-in self-test (BIST). STAR-BIST [24] generates a parent pattern and then children patterns are generated by randomly flipping bits in the parent pattern. In [25], a folding counter is used to generate the children patterns. In [20], frequently occurring sequences shorter than a full pattern are stored on-chip and used to embed deterministic patterns in a semi-random sequence. The underlying concept of these approaches is similar to what is proposed here, but there are a number of significant differences. The proposed approach generates a specific precise deterministic test set whereas the previous methods embed a deterministic test set into a much larger set of test vectors. The proposed approach decomposes the vectors into a common sequence and unique sequence, and these two sequences are combined in a fundamentally different manner than what is done in [20], [24], and [25].

In the proposed scheme, test cubes are grouped into clusters. Each cluster requires the use of an ATE vector repeat instruction for generating the common sequence for the cluster. The clustering algorithm needs to be performed focusing two-fold.

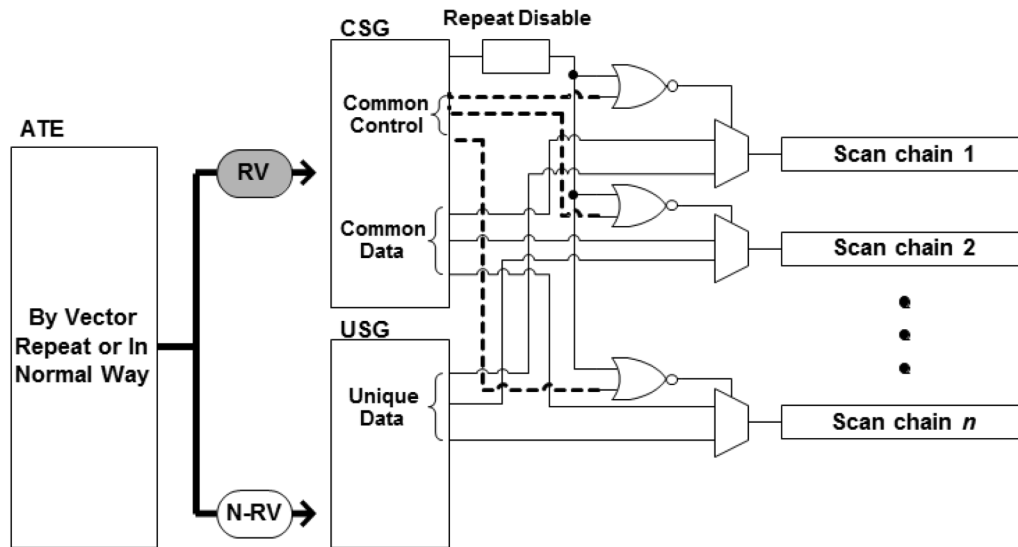

Fig. 8. Block diagram for decompression logic for ATE with repeat-per-all-pins.

Original Test Cubes						Encoded							
							C.C.	x	1	1	x	0	
							C.D.	x	1	1	x	x	
t1	x	1	1	x	0	t1	U.D.	x	x	x	x	0	Number of Specified unique data bits = 2
t2	x	1	1	x	1	t2	U.D.	x	x	x	x	1	
							C.C.	1	0	1	0	1	
							C.D.	0	x	1	x	1	
t3	x	0	1	1	1	t3	U.D.	x	0	x	1	x	
t4	0	0	1	0	1	t4	U.D.	x	0	x	0	x	Number of Specified unique data bits = 6
t5	0	1	1	1	1	t5	U.D.	x	1	x	1	x	
							C.C.	1	1	1	x	0	
							C.D.	1	1	1	x	x	
t6	0	0	0	x	0	t6	U.D.	x	x	x	x	0	Number of Specified unique data bits = 2
t7	0	0	0	x	1	t7	U.D.	x	x	x	x	1	

C.C. : Common Control, C.D. : Common Data, U.D. : Unique Data

Fig. 9. Block example of original/encoded test cubes.

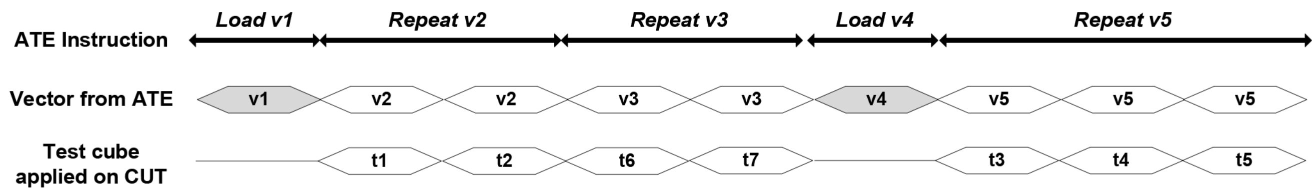

Fig. 10. Tester instruction & data flow.

1) Maximize the correlation in each cluster (to reduce the number of specified bits, thereby minimizing the tester storage).
2) Generate a small number of clusters (to minimize the number of ATE repeat instructions required).

Because the ATE instruction memory is limited, the number of clusters cannot exceed or equal to the amount of ATE instruction memory available.

Test cube clustering has been previously studied and some nice algorithms can be found in [26]. For the proposed scheme, a special benefit function is needed to account for both the control and data bits required to encode each cluster.

In order to maximize the compression achieved for each cluster, it is important that the test cubes in each cluster have many bit positions with compatible values. As more test cubes are added to a cluster, the number of clusters is reduced. This has the benefit of minimizing the number of the ATE instructions required for the vector repeat, but there is a tradeoff as more bit positions are likely to have conflicts thus reducing the effectiveness of each repeat instruction. A greedy clustering procedure taking this tradeoff into consideration is described here.

The flow diagram of the proposed clustering algorithm is shown in Fig. 11 and the benefit function that is used to assign

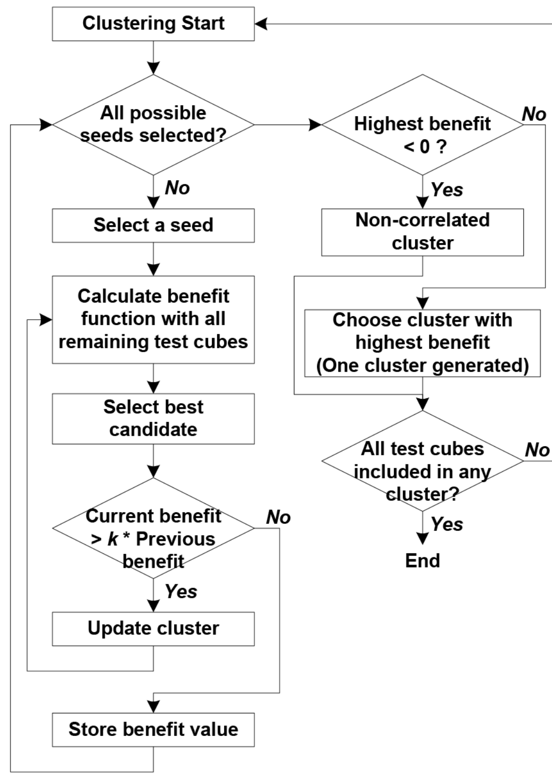

Fig. 11. Flow diagram of proposed clustering algorithm.

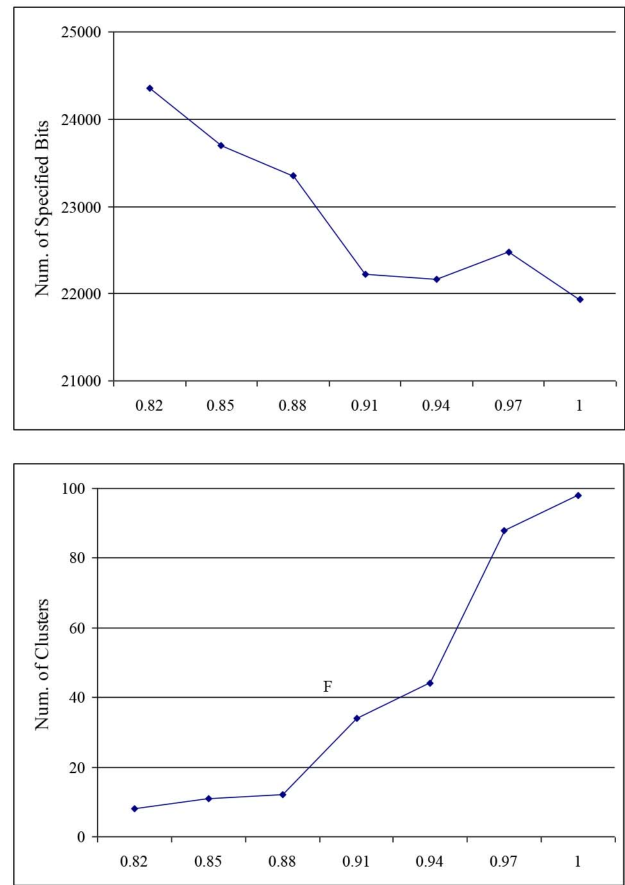

Fig. 12. Number of specified bits and clusters versus tuning variable.

a value to a cluster is shown below

$$Benefit = \frac{total_spec}{compatible_pos + spec_pos + total_unique_spec}$$

where *total_spec* is the total number of specified bits in the cluster, *compatible_pos* is the number of bit positions that are compatible in the cluster, *spec_pos* is the number of bit positions that have specified bits, and *total_unique_spec* is the total number of specified bits not in compatible bit positions. Essentially the numerator corresponds to the uncompressed storage requirements, and the denominator corresponds to the compressed storage requirements (the common data has a specified bit for each compatible bit position in the cluster, the common control has a specified bit for each bit position that has one or more specified bits in the cluster, and the unique data has one specified bit for each specified bit not in a compatible bit position in the cluster). The larger the benefit value is, the larger the amount of compression that is achieved when encoding the cluster. Note that while a greedy clustering procedure is described here, any clustering procedure in the literature can be used to maximize the benefit function defined here. The time complexity is $O(n^2)$ assuming n is the number of test cubes. The CPU time with 1.3 GHz machine for the overall encoding algorithm is about 20 minutes to 3 h for the four largest ISCAS-89 circuits.

One issue with the benefit function is that it may generate too many clusters in some cases. Note that one ATE repeat instruction is required in the proposed scheme for each cluster. Thus, there is a limit on how many clusters can be used based on the amount of ATE instruction memory that is available. To provide a mechanism for reducing the number of clusters generated by the clustering procedure, a tuning variable, k, is added to the algorithm as shown in Fig. 11. To increase the number of test cubes in each cluster, the condition for adding a test cube into a cluster can be loosened by making the value of k lower than 1. By lowering the value of k, the number of clusters reduces with a reasonable sacrifice in the number of specified bits. Note that if the value of k is reduced too much, at some point the number of specified bits in the encoded test cubes approaches the number of specified bits in original test cubes and hence no compression is achieved. The number of specified bits and the number of clusters generated versus value of k is shown in Fig. 12. Typically, as k approaches to 1.0, the number of specified bits decreases while the number of clusters increases. Based on the amount of the ATE instruction memory, a user can select the best value for k.

The tuning variable, k is also used to reduce the number of specified bits in the unique data. As explained in Section IV, the number of specified bits in unique data must be reduced so that a given *USG* in Fig. 8 (in an ATE environment supporting only repeat-per-all-pins, not repeat-per-pin-group) can generate unique data for as many test cubes as possible with a single seed. The smaller k value is, the smaller number of clusters and the larger number of test cubes are in each cluster. As the number of test cubes in a cluster goes up, more conflicts in bit positions happen. Hence, the number of specified bits in common control and common data decreases and the

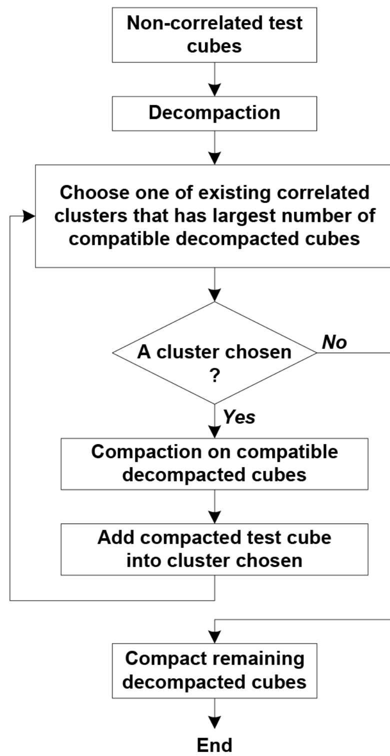

Fig. 13. Recompaction flow diagram.

number of specified bits in unique data increases. For an ATE supporting only repeat-per-all-pins, the tuning variable that is slightly higher than 1 is applied to the clustering algorithm, which results in a larger number of clusters and a smaller number of specified unique data bits.

Another potential issue is that the number of lowly-correlated test cubes (test cubes in the noncorrelated cluster in Fig. 11) may be large depending on a test compaction algorithm. An ATPG tool creates original test cubes and the test cubes are compacted if there is no conflict in any bit position, which makes the total number of test cubes smaller. Typically, a test compaction is performed in a way to minimize the total number of test cubes and that may cause test cubes conflicting in many bit positions with the other test cubes. This degrades efficiency of the proposed compression scheme. A recompaction scheme is proposed here to resolve the issue and shown in Fig. 13. A simple example is shown below to explain the recompaction scheme.

1) There are three clusters, two correlated clusters, and a noncorrelated cluster.
2) The noncorrelated cluster has two test cubes, $t1$ and $t2$.
3) The original test cubes before compaction for $t1$ and $t2$ (compacted test cubes: $t1$, $t2$) are $p1$, $p2$, $p3$, and $p4$ (de-compacted test cubes: $p1$, $p2$, $p3$, $p4$).

If a dynamic compaction algorithm is used, then it is harder to find de-compacted cubes for original test cubes. In order to get de-compacted cubes for the uncorrelated test cubes in the case where the dynamic compaction has been used, a fault grading is performed with correlated test cubes. For the faults not detected by the correlated test cubes, ATPG without dynamic compaction is performed. The newly generated test cubes are used as de-compacted test cubes in the recompaction scheme.

The de-compacted test cubes, $p1$, $p2$, $p3$, and $p4$ are checked if any of the test cubes can be included in any of two existing correlated clusters without modifying common control and common data. Assume that the maximum number of the de-compacted test cubes that can be included in one of the correlated clusters is 2 ($p1$, $p2$). $p1$ and $p2$ are compacted and the compacted test cube is added into the correlated cluster. The same process is performed on the remaining de-compacted test cubes ($p3$, $p4$). This process is repeatedly run until no correlated cluster is chosen to include any of the de-compacted test cubes. Assume that $p3$ is included in the other correlated cluster and $p4$ cannot be included in any of the two correlated clusters. Finally, there is just one test cube ($p4$) in the noncorrelated cluster instead of 2 ($t1$, $t2$) in the original noncorrelated cluster. Furthermore, $p4$ is the de-compacted test cube that typically has fewer specified bits than $t1$ and $t2$. The recompaction scheme reduces the number of specified bits in the noncorrelated cluster as well as the number of test cubes in the noncorrelated cluster. A drawback of the recompaction scheme is that it may increase the number of test cubes and test time. In this example, the two test cubes, $t1$ and $t2$ are converted to three test cubes: 1) one compacted test cube for $p1$; 2) $p2$, $p3$; and 3) $p4$. Therefore, the recompaction scheme is employed only when the number of test cubes in the noncorrelated cluster is relatively large.

Our experimental results on ISCAS-89 circuits show that the number of uncorrelated test cubes is less than 20% of total number of test cubes in $s13207$, $s15850$, and $s38417$. The importance of the recompaction scheme is maximized in case of test set that has a large number of uncorrelated cubes. Depending on the test set and compaction scheme used, the number of uncorrelated cubes might be relatively large. In our test set for $s38584$, the number of uncorrelated test cubes is more than 50% of total number of test cubes depending on value on a tuning variable. However, even for the test cubes for $s38584$, the recompaction is not applied when the tuning variable, k is less than or equal to 1 because the number of uncorrelated test cubes is small enough to provide high compression ratio. Because the compaction algorithm is based on the pattern-wise correlation as well, the recompaction scheme is typically not needed. The time complexity is $O(n^2)$ assuming n is the number of decompacted test cubes in the noncorrelated cluster.

VI. ATE Architectures

This paper proposes vector repeat-based compression techniques for repeat per-pin-group and repeat per-all-pins, which is used depends on a particular ATE architecture's capability. For SoC testing, per-pin-group structure in ATEs is typically used to support multisite testing, AC delay testing, and so on [22], [31]. Modern ATEs generally support both per-pin-group and per-all-pin structures.

ATE memory can have two types of memory structures. One is to have a shared pool of memory. For low cost ATEs, a pooled memory is used to store patterns and they are distributed to each pin. The proposed *Decompression Logic for*

ATE with Repeat-Per-All-Pins method can be applied to ATEs with a pooled memory such as Advantest T6673 and T3347A. The other is to a have per-pin memory. This structure has the same amount of memory behind each channel. For high speed ATEs, a per-pin memory architecture is used to drive each channel. *Decompression Logic for ATE with Repeat-Per-Pin-Group* approach can be employed to ATEs with per-pin memory such as Teradyne UltraFlex and Advantest T2000. In this scenario, reducing the amount of memory on a subset of channels does not help to reduce test cost. The repeated data requires less storage since we only store one copy, whereas the unique data requires full storage. For example, in Fig. 4, there are six tester channels. First three channels are used for repeated data and the next three channels are assigned for unique data. If all channels have the same memory, then the channels storing unique data (bottom three channel memories in Fig. 4) are the limiting factor for fitting the test program in the memory. This might create uneven compression ratios on tester channels.

To balance the compression ratio across the pins, test vectors can be partitioned. For example, if we have two partitions, the first half of vectors getting the common data can be applied from one set of pins while the unique data are sent the other set of pins, then the other half of vectors getting the common data from a different subset of pins can be applied. This would spread the savings across the pins which would better balance the compression across all pins and thereby helping for a per-pin memory structure. The reconfiguration logic is implemented on the chip and this requires a MUX for each scan chain. The MUX allows an ATE channel get connected to one chain or to another chain through either CSG for common data or USG for unique data. If more balancing is desired, more than two partitions can be used at the cost of additional MUXes to implement the on-chip reconfiguration.

As an example of the capabilities of a few modern ATE architectures and the ATE development trends consider the following. Teradyne UltraFlex digital module has $1K$ word instruction memory and $512M$ word pattern memory. Advantest T2000 250M digital module provides 32 I/O channels and each channel supports fully featured per-pin measurement. There are $128M$ word (3 bits per word) main memory, $256M \sim 2G$ pattern memory (shared with a main memory), $4K$ subroutine memory, and $1K$ word instruction memory. Advantest T2000 500/800 Mb/s digital module provides $1K$ word instruction memory [31], [32], $4K \times 3$-*bits* subroutine memory per channel, $4K \times 512$ central capture memory, $8K$ fail capture memory per pin, $4K \times 3$ bits/pin pattern capture memory and, etc.

The number of ATE instructions required to implement the proposed method can be adjusted to fit into whatever instruction memory is available in the ATE being used. The number of clusters found by the clustering algorithm gives the number of repeat instructions. It can be adjusted by controlling the tuning variable (k).

VII. Experimental Results

Experiments were performed on the four largest ISCAS-89 benchmark circuits. The test cubes used in the experiments were generated in the following way. Atalanta (ATPG and fault simulator for stuck-at faults in combinational circuits) generated uncompacted test cubes and then bit-stripping was performed to maximize the number of don't cares. Finally, the test cubes were merged to minimize the number of specified bits. In Table I, the circuit name and the original number of specified bits in the deterministic test sets are shown in the first and second columns. The third column shows available ATE repeat vector function. r.p.g. means repeat-per-pin-group and r.p.a. means repeat-per-all-pins. The fourth column shows the number of test cubes. The number of test cubes in r.p.g. case in each circuit is always the same as the original number of test cubes. Except for *s*38584, the number of test cubes in r.p.a. case in each circuit is also the same as the original number of test cubes, which means that the test cube recompaction described in Section V has been performed only for *s*38584 r.p.a. case. As explained in Section V, the recompaction scheme is employed only in the case where the number of test cubes in a noncorrelated cluster is relatively large. In r.p.a. case (where the tuning variable is larger than 1) for *s*38584, the number of uncorrelated cubes is relatively large, which requires a recompaction scheme to enhance the performance of the proposed scheme. The number of test cubes has been increased from 296 to 315 when $k = 1.10$ and from 296 to 328 when $k = 1.15$. The value of the tuning variable is shown in the fifth column and the corresponding number of clusters generated by the clustering algorithm described in Section V is shown in the sixth column. The best results are typically shown when k is close 1. If an ATE instruction memory is very limited, such that the number of repeat instruction (which is the same as the number of clusters shown in the sixth column) is not allowable, the value of k can be adjusted to have less repeat instructions with a reasonable sacrifice of the number of specified bits. For each circuit, results are shown for two different numbers of clusters. The seventh and eighth columns show the number of specified bits in the unique sequence and the number of specified bits in common sequence. The ninth column shows the total number of specified bits in the encoded test cubes. The last column tells the reduction in the number of specified bits. The result is shown in the pooled memory ATE architecture. Note that the maximum number of clusters shown in Table I is 173, which means that the maximum number of ATE vector repeat instructions that have to be stored in the ATE instruction memory is only 173 or less for these circuits. In most circuits, the number of clusters is below 100. Of course, the number of clusters can also be reduced if necessary by lowering k. Since the number of specified bits with the proposed scheme is reduced, the number of free-variables that are needed to encode the data using linear decompressors will also reduce correspondingly.

As discussed in Section VI, in a per-pin memory architecture, the compression ratio is determined by the highest memory used on any channel. To balance the compression ratio across the channels, experiments were performed where the test vectors are partitioned two-ways and four-ways where the repeated data for each partition is generated from a different set of pins. The more we partition the test vectors, the more balanced compression ratio can be achieved by rotating the

TABLE I
RESULTS FOR COMPRESSION USING VECTOR REPEAT-PER-PIN-GROUP

Circuit	Original Specified Bits	ATE	Num. Test Cubes	Tuning var. (k)	Num. Clusters	Num. Specified In Unique Seq.	Num. Specified. In Common Seq.	Encoded Specified Bits	Reduction
s13207	9389	r.p.g.	266	0.90	18	5612	1881	7493	20.2%
				1.00	40	4551	3049	7600	19.1%
		r.p.a.	266	1.10	54	4007	3677	7684	18.2%
				1.20	72	3431	4605	8026	14.5%
s15850	10944	r.p.g.	269	0.90	15	6335	2103	8438	22.9%
				1.00	34	4650	3570	8229	24.8%
		r.p.a.	269	1.10	51	3821	4720	8541	22.0%
				1.20	80	3123	5634	8757	19.9%
s38417	30699	r.p.g.	376	0.90	19	12427	7207	19634	35.9%
				1.00	31	11044	8131	19175	37.5%
		r.p.a.	376	1.10	58	9830	12798	22628	26.3%
				1.20	107	7497	16962	24459	20.3%
s38584	26185	r.p.g.	296	0.89	18	17505	5849	23354	10.8%
				1.0	98	11293	10642	21935	16.2%
		r.p.a.	315	1.10	155	8422	14381	22803	12.9%
				1.15	173	7823	16058	23881	8.8%

TABLE II
REDUCTION COMPARISON BETWEEN POOLED MEMORY AND PER-PIN MEMORY

Circuit	Reduction in Pooled Memory ATE Architecture	Reduction in Per-Pin Memory ATE Architecture	
		2-Way Partition	4-Way Partition
s13207	18.5%	8.5%	18.2%
s15850	23.3%	21.6%	21.6%
s38417	27.5%	21.9%	26.2%
s38584	13.4%	6.9%	10.6%

channels used for repeated data. Table II shows a compression ratio in per-pin memory-based ATE architecture. First column shows benchmark circuits and the second column shows a compression ratio in the r.p.g ATE type with a pooled memory with different tuning variables. Last two columns give the reduction in the number of specified bits among channels, respectively, since it is a limiting factor in the compression.

The area overhead for decompression logic is shown in Table III. For overhead calculation, the standard library for TSMC 0.18 μm process [27] is used. The second column shows the size of original linear decompressor for each circuit. The size of the original linear decompressor has been calculated with the test set used in the experiment for [9]. The third column shows the number of scan chains. The fourth column actually shows the sum of widths of all standard cells in each ISCAS-89 benchmark circuits with a original linear decompressor. The height for all cells is 5 μm. The number of 2-to-1 multiplexers and OR gates added that is the same as the number of scan chains is shown in the sixth column. The sizes of the CSG and the USG are shown in the seventh and eighth columns, respectively, which are the dominant factor in a decompression area overhead. Note that there is a big difference between sizes of the USG in r.p.g. and the USG in r.p.a. As explained in Section IV, the USG for r.p.a. case is created so that it can generate unique sequences for as many clusters as possible with one seed while the USG for r.p.g. case is created in a typical way (based on the maximum number of specified bits in a single unique sequence). The overall area for decompression logic in the proposed approach is shown in the ninth column and the overhead ratio is shown in the last column. For r.p.g. cases, the area overhead is less than 5% in all the circuits. For r.p.a. cases, the area overhead is bigger due to the large size of the USG than the area overhead in r.p.g. cases and is $6.1 \sim 9.4\%$.

To get results for the actual tester storage requirements using the proposed approach, we did experiments using the linear decompressor described in [9] (other linear decompressors could also be used). The results are shown in Table IV. The same test sets are compressed using the linear decompressor in [9] by itself, and using it in conjunction with the proposed scheme to utilize ATE vector repeat. The results show that the amount of data that needs to be stored in the ATE vector memory is significantly reduced with the proposed scheme. The second to last column shows the vector memory reduction compared with [9].

In Table V, the proposed scheme is compared with the scheme described in [22] for utilizing ATE vector repeat. [22] is based on repeat-per-pin-group function, so in this comparison, we compare only results on r.p.g. cases. In [22], the best results were obtained when a sequencer controls two pins, so we also assumed that a sequencer controls two pins. And as suggested in [22], only the vectors that can be repeated at least 16 times are encoded by the ATE vector repeat to reduce the number of the repeat instructions. Using this criteria, we generated experimental results on our test sets in the manner described in [22] (note that results published in [22] were for test sets that are not publicly available). The number of ATE repeat instructions is shown in the second and the fourth columns, and the amount of data stored in the vector memory is shown in the third and fifth columns. As can be seen, much larger reductions in the vector memory can be obtained with the proposed approach using an order of magnitude fewer ATE repeat instructions compared with [22]. Of course, it should

TABLE III
AREA OVERHEAD FOR DECOMPRESSION LOGIC

Circuit	Original Linear Decomp. (Bit)	Num. Scan Chains	Original Area	ATE	Num. 2-to-1 MUX + OR gate	Common Sequence Generator (Bit)	Unique Sequence Generator (Bit)	Decomp. Logic Area	Area Overhead
s13207	64	20	21467.9	r.p.g.	20	86	27	746.8	3.4%
				r.p.a.	20	78	150	2034.8	9.4%
s15850	66	20	23595.2	r.p.g.	20	115	28	1060.4	4.4%
				r.p.a.	20	93	142	2090.8	8.8%
s38417	120	40	58045.7	r.p.g.	40	275	41	2591.2	4.4%
				r.p.a.	40	231	261	4562.4	7.8%
s38584	120	40	56671.4	r.p.g.	40	145	48	1213.6	2.1%
				r.p.a.	40	112	284	3487.2	6.1%

TABLE IV
RESULTS FOR USING LINEAR DECOMPRESSOR IN [9] ALONE VERSUS USING IT WITH PROPOSED SCHEME

Circuit	[9]		Proposed Scheme			
	Num. Spec.	Vector Mem.	ATE	Num. Spec.	Vector Mem.	Reduc. (%)
s13207	9389	9872	r.p.g.	7493	7750	21.5
			r.p.a.	7684	8132	17.6
s15850	10944	11322	r.p.g.	8229	8694	23.2
			r.p.a.	8541	8933	21.1
s38417	30669	31245	r.p.g.	19175	20769	33.5
			r.p.a.	22628	24256	22.4
s38584	26185	28312	r.p.g.	21935	24268	14.3
			r.p.a.	22803	25830	8.8

TABLE V
RESULTS COMPARISON WITH [22]

Circuit	[22]		Proposed			
	Num. Repeat Inst.	Vector Mem.	Num. Repeat Inst.	Vector Mem.	Reduction	
					Inst. Mem.	Vector Mem.
s13207	976	8464	18	7750	98.1%	8.4%
s15850	894	15974	34	8694	96.2%	45.6%
s38417	2518	41506	31	20769	98.8%	50.0%
s38584	3264	53952	98	24268	96.9%	55.1%

be pointed out that the method in [22] does not require any on-chip hardware, whereas the proposed method requires two on-chip linear decompressors. However, note that the linear decompressors used in the proposed scheme will require a very small amount of area with current chip densities, and they can be reused when testing multiple cores.

In Table VI, a comparison is made with the scheme described in [23] that also utilizes ATE vector repeat. Here, results are shown for the exact same test cube files for two of the industrial circuits that were used in [23] (the others were not publicly available). Circuit information is shown in the first, second, and third columns. The fourth column shows the best results in terms of vector memory requirements reported in [23]. For the proposed method, results are shown for two different numbers of ATE vector repeat instructions for r.p.g. and one number of ATE vector repeat instructions for r.p.a. Note that the number of repeat instructions used in [23] is not reported in the paper and thus is not shown in Table VI. Vector memory required is shown in the seventh column. Last column shows the reduction in the vector memory required. There is a substantial reduction. A significant reduction comes from the fact that the proposed scheme is based on linear decompression. A major advantage of the proposed scheme is that it is compatible with linear decompression which is known to be highly efficient.

TABLE VI
RESULTS COMPARISON WITH [23]

Circuit			[23]	Proposed			
Name	Test Cubes	Scan Cells	Vector Memory	ATE	Repeat Inst.	Vector Memory	Reduction
ckt-4	1529	43414	3264850	r.p.g.	121	1518409	53.5%
					404	1341943	58.9%
				r.p.a.	677	1860371	43.0%
ckt-5	4900	26970	6079410	r.p.g.	283	2579402	57.6%
					841	2268261	62.7%
				r.p.a.	1476	3146437	48.2%

VIII. CONCLUSION

The proposed scheme harnesses the power of linear and nonlinear decompression together using a simple and compact decoder whose design is independent of the test set. Note that the compression could be significantly improved if scan chain reordering was employed along with the proposed scheme to increase bit-wise correlation. The design of the decompressor for the proposed scheme is independent of the test set or CUT and thus can be reused when testing multiple cores.

REFERENCES

[1] A. Khoche and J. Rivoir, "I/O bandwidth bottleneck for test: Is it real?" in *Proc. Int. Workshop Test Resource Partitioning*, 2000, pp. 2.3-1–2.3-6.

[2] B. Könemann, "LFSR-coded test patterns for scan designs," in *Proc. Eur. Test Conf.*, 1991, pp. 237–242.

[3] S. Hellebrand, S. Tarnuck, J. Rajski, and B. Courtois, "Generation of vector patterns through reseeding of multiple-polynomial liear feedback shift registers," in *Proc. Int. Test Conf.*, 1992, pp. 120–129.

[4] 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 Trans. Comput.*, vol. 44, no. 2, pp. 223–233, Feb. 1995.

[5] N. Zacharia, J. Rajski, and J. Tyszer, "Deompression of test data using variable-length seed LFSRs," in *Proc. VLSI Test Symp.*, Princeton, NJ, USA, 1995, pp. 426–433.

[6] J. Rajski *et al.*, "Embedded deterministic test for low cost manufacturing test," in *Proc. Int. Test Conf.*, 2002, pp. 301–310.

[7] B. Könemann, "A SmartBIST variant with guaranteed encoding," in *Proc. Asian Test Symp.*, Kyoto, Japan, 2001, pp. 325–330.

[8] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Trans. Computer Aided Des. Integr. Circuit Syst.*, vol. 23, no. 5, pp. 776–792, May 2004.

[9] C. V. Krishna, A. Jas, and N. A. Touba, "Test vector encoding using partial LFSR reseeding," in *Proc. IEEE Int. Test Conf.*, Baltimore, MD, USA, 2001, pp. 885–893.

[10] C. V. Krishna and N. A. Touba, "3-stage variable length continuous-flow scan vector decompression scheme," in *Proc. IEEE VLSI Test Symp.*, 2004, pp. 79–86.

[11] I. Hamzaoglu and J. H. Patel, "Reducing test application time for full scan embedded cores," in *Proc. Int. Symp. Fault Tolerant Comput.*, Madison, WI, USA, 1999, pp. 260–267.

[12] I. Bayraktaroglu and A. Orailoglu, "Test volume and application time reduction through scan chain concealment," in *Proc. Des. Autom. Conf.*, 2001, pp. 151–155.

[13] S. Mitra and K. Kim, "XMAX: X-tolerant architectures for maximal test compression," in *Proc. Int. Conf. Comput. Des.*, 2003, pp. 326–330.

[14] C. Krishna and N. Touba, "Adjustable width linear combinational scan vector decompression," in *Proc. ICCAD*, 2003, pp. 863–866.

[15] A. Jas, B. Pouya, and N. Touba, "Virtual scan chains: A means for reducing scan length in cores," in *Proc. VLSI Test Symp.*, Montreal, QC, Canada, 2000, pp. 73–78.

[16] E. Volkerink and S. Mitra, "Efficient seed utilization for reseeding based compression," in *Proc. VLSI Test Symp.*, 2003, pp. 232–237.

[17] W. Rao, I. Bayraktaroglu, and A. Orailoglu, "Test application time and volume compression through seed overlapping," in *Proc. Des. Autom. Conf.*, 2003, pp. 732–737.

[18] J. Lee and N. A. Touba, "Efficiently utilizing ATE vector repeat for compression by scan vector decomposition," in *Proc. Asian Test Symp.*, Fukuoka, Japan, 2006, pp. 237–244.

[19] C. Barnhart *et al.*, "OPMISR: The foundation for compressed ATPG vectors," in *Proc. Int. Test Conf.*, Baltimore, MD, USA, 2001, pp. 748–757.

[20] L. Li and K. Chakrabarty, "Hybrid BIST based on repeating sequences and cluster analysis," in *Proc. DATE*, 2005, pp. 1142–1147.

[21] X. Liu, M. Hsiao, S. Chakravarti, and P. J. Thadikaran, "Techniques to reduce data volume and application time for transition test," in *Proc. Int. Test Conf.*, 2002, pp. 983–992.

[22] H. Vranken, F. Hapke, S. Rogge, D. Chindamo, and E. Volkerink, "ATPG padding and ATE vector repeat per port for reducing test data volume," in *Proc. Int. Test Conf.*, 2003, pp. 1069–1078.

[23] Z. Wang, and K. Chakrabarty, "Test data compression for IP embedded cores using selective encoding of scan slices," in *Proc. Int. Test Conf.*, 2005, pp. 581–590.

[24] K.-H. Tsai, J. Rajski, and M. Marek-Sadowska, "Star test: The theory and its applications," *IEEE Trans. Computer-Aided Design*, vol. 19, no. 9, pp. 1052–1064, Sep. 2000.

[25] H.-G. Liang, S. Hellebrand, and H.-J. Wunderlich, "Two-dimensional test data compression for scan-based deterministic BIST," in *Proc. Int. Test Conf.*, Baltimore, MD, USA, 2001, pp. 894–902.

[26] R. Alleyne, "Clustering of test cubes: A procedure for the efficient encoding of complete test sets based on the intelligent reseeding of LFSRs," Master thesis, McGill University, Montreal, QC, Canada, 1994.

[27] Artisan Components, "TSMC 0.18μm Process 1.8-volt SAGE-X standard cell library databook," 2001, DB-SX-TSM003-3.1/18.

[28] G. Mrugalski, N. Mukherjee, J. Rajski, D. Czysz, and J. Tyszer, "Compression based on deterministic vector clustering of incompatible test cubes," in *Proc. Int. Test Conf.*, 2009, pp. 1–10.

[29] D. Czysz *et al.*, "Deterministic clustering of incompatible test cubes for higher power-aware EDT compression," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 30, no. 8, pp. 1225–1238, Aug. 2011.

[30] N. A. Touba, "Survey of test vector compression techniques," *IEEE Design Test Mag.*, vol. 23, no. 4, pp. 294–303, Jul. 2006.

[31] Advantest TT92 Memory Test System, Manual Number 8350379-04 [Online]. Available: http://www.advantest.com/US/Products

[32] Advantest T2000 500/800 Mbps Digital Module Data Sheet [Online]. Available: http://www.advantest.com/US/Products

Joon-Sung Yang (S'05–M'09) received the B.S. degree from Yonsei University, Seoul, Korea, in 2003, and the M.S. and Ph.D. degrees from the University of Texas at Austin, Austin, TX, USA, in 2007 and 2009, respectively, all in electrical and computer engineering.

After graduation, he was with Intel Corporation, Santa Clara, CA, USA, for four years. He is currently an Assistant Professor with SungKyunKwan University, Suwon, Korea. His current research interests include VLSI testing, silicon debug and nanometer scale test, and design methodologies.

Dr. Yang was a recipient of the Korea Science and Engineering Foundation Scholarship in 2005. He received the Best Paper Award at the 2008 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems and was nominated for the Best Paper Award at 2013 IEEE VLSI Test Symposium.

Jinkyu Lee (S'03–M'06) received the B.S. degree from Yonsei University, Seoul, Korea, in 2001 and the M.S. and Ph.D. degrees from the University of Texas at Austin, Austin, TX, USA, in 2004 and 2006, respectively, all in electrical and computer engineering.

Currently, he is a Staff CPU Design Engineer at Samsung, Austin, TX, USA.

Nur A. Touba (SM'05–F'09) received the B.S. degree from the University of Minnesota, Minneapolis, MN, USA, in 1990, and the M.S. and Ph.D. degrees from Stanford University, Stanford, CA, USA, in 1991 and 1996, respectively, all in electrical engineering.

He is currently a Professor with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA.

Dr. Touba was a recipient of the National Science Foundation Early Faulty CAREER Award in 1997, the Best Paper Award at the 2001 VLSI Test Symposium, and the 2008 Defect and Fault Tolerance Symposium. He served as a Program Chair for the 2008 International Test Conference and as a General Chair for the 2007 Defect and Fault Tolerance Symposium. Currently, he serves on the Program Committee for the Design Automation and Test in Europe Conference, International On-Line Test Symposium, European Test Symposium, Asian Test Symposium, and Defect and Fault Tolerance Symposium.