Survey of Test Vector Compression Techniques

Nur A. Touba
University of Texas at Austin

Test data compression consists of test vector compression on the input side and response compaction on the output side. Test vector compression has been an active area of research, yielding a wide variety of techniques. This article summarizes and categorizes these techniques, explaining how they relate to one another. The goal is to provide a framework for understanding the theory and research in this area.

THE AMOUNT OF DATA required to test ICs is growing rapidly in each new generation of technology. Increasing integration density results in larger designs with more scan cells and more faults. Moreover, achieving high test quality in ever smaller geometries requires more test patterns targeting delay faults and other fault models beyond stuck-at faults. Conventional external testing involves storing all test vectors and test response on an external tester—that is, ATE. But these testers have limited speed, memory, and I/O channels. The test data bandwidth between the tester and the chip is relatively small; in fact, it is often the bottleneck determining how fast you can test the chip. Testing cannot proceed any faster than the amount of time required to transfer the test data:

Test time ≥ (*amount of test data on tester*) / (*number of tester channels* × *tester clock rate*)

Overcoming limited tester-chip bandwidth

Three general approaches help overcome this bottleneck: stand-alone BIST, hybrid BIST, and test data compression.

Stand-alone BIST

Traditional stand-alone BIST involves using on-chip hardware to perform all test pattern generation and output response analysis. Stand-alone BIST eliminates the need for tester storage. This is very useful for performing self-test in the field when there is no access to a tester. However, achieving high fault coverage with stand-alone BIST generally requires considerable overhead because of random-pattern-resistant (RPR) faults, which have low detection probabilities. Detecting such faults requires either test points or deterministic-pattern-embedding logic. Other issues with BIST include the need for a BIST-ready design, a way to handle false and multicycle paths, and the need to keep nondeterministic values from corrupting the final signature.

Hybrid BIST

If a particular chip design uses BIST only for manufacturing test, then hybrid BIST can be more cost-effective than stand-alone BIST. Hybrid BIST involves storing some data on the tester to help detect RPR faults. The simplest approach is to perform ATPG for RPR faults not detected by pseudorandom BIST to obtain a set of deterministic test patterns that "top up" the fault coverage to the desired level, and then store those patterns directly on the tester.

More efficient hybrid BIST schemes store the deterministic top-up patterns on the tester in a compressed form, then use the existing BIST hardware to decompress these patterns. Some schemes embed deterministic patterns by using compressed weight sets or by perturbing the pseudorandom sequence in some manner.

Test data compression

As Figure 1 illustrates, test data compression involves adding some additional on-chip hardware before and after the scan chains. This additional hardware decompresses the test stimulus coming from the tester; it also compacts the response after the scan chains and before it goes to the tester. This permits storing the test data in a compressed form on the tester. With test data compression, the tester still applies a precise deterministic (ATPG-generated) test set to the circuit under test (CUT).

0740-7475/06/$20.00 © 2006 IEEE Copublished by the IEEE CS and the IEEE CASS

This process differs from that of hybrid BIST, which applies a large number of patterns, including both pseudorandom and deterministic data. Although hybrid BIST can reduce the amount of test data on the tester more than test data compression can, hybrid BIST generally requires longer test application time because you must apply more patterns to the CUT than with test data compression (in essence, hybrid BIST trades off more test application time for less tester storage). The advantage of test data compression is that it generates the complete set of patterns applied to the CUT with ATPG, and this set of test patterns is optimizable with respect to the desired fault coverage. Test data compression is also easier to adopt in industry because it's compatible with the conventional design rules and test generation flows for scan testing.

Test data compression provides two benefits. First, it reduces the amount of data stored on the tester, which can extend the life of older testers that have limited memory. Second—and this is the more important benefit, which applies even for testers with plenty of memory—it can reduce the test time for a given test data bandwidth. Doing so typically involves having the decompressor expand the data from n tester channels to fill greater than n scan chains. Increasing the number of scan chains shortens each scan chain, in turn reducing the number of clock cycles needed to shift in each test vector.

Test data compression must compress the test vectors losslessly (that is, it must reproduce all the care bits after decompression) to preserve fault coverage. The output response, on the other hand, can use lossy compaction (which does not reproduce all data, losing information) with negligible impact on fault coverage. Ideally, the output response could be compacted using just a multiple-input signature register (MISR). However any unknown (nondeterministic) values in the output response would corrupt the final signature. Researchers have developed several schemes to address the problem of unknown values in the output response, including eliminating the source of the unknown values, selectively masking the unknown values in the output stream, or using an output compaction scheme that can tolerate the unknown values. Output compaction is an entire subject in itself, and I will not discuss it further in this article.

The subject here is test vector compression techniques. Test vectors are highly compressible because typically only 1% to 5% of their bits are specified (care) bits. The rest are don't-cares, which can take on any value with no impact on the fault coverage. A *test cube* is a deterministic test vector in which the bits that ATPG does not assign are left as don't-cares (that is, the ATPG does not randomly fill the don't-cares).

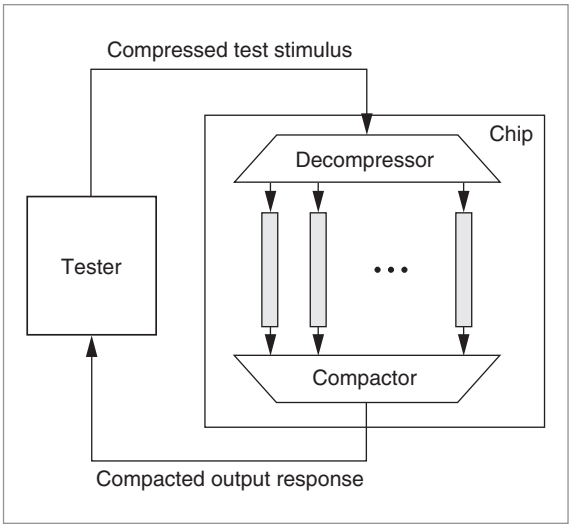

Figure 1. Test data compression.

In addition to containing a very high percentage of don't-cares, test cubes also tend to be highly correlated because faults are structurally related in the circuit.

Both of these factors are exploitable to achieve high amounts of compression. Recently, researchers have proposed a wide variety of techniques for test vector compression. Here, I summarize and categorize these techniques and explain how they relate to each other. The focus is on hardware-based test vector compression techniques for scan architectures; I do not discuss software-based, nonscan, and hybrid BIST techniques.

Test vector compression schemes fall broadly into three categories:

- *Code-based schemes* use data compression codes to encode test cubes.
- *Linear-decompression-based schemes* decompress the data using only linear operations (that is LFSRs and XOR networks).
- *Broadcast-scan-based schemes* rely on broadcasting the same values to multiple scan chains.

Code-based schemes

Code-based schemes use data compression codes to encode the test cubes. This involves partitioning the original data into symbols, and then replacing each symbol with a code word to form the compressed data. To perform decompression, a decoder simply converts

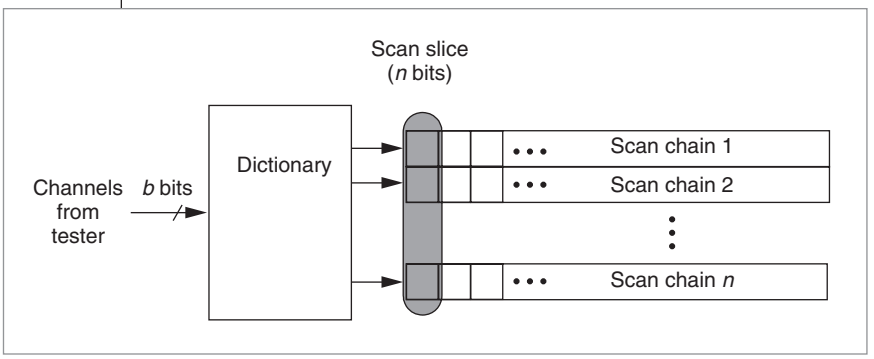

Figure 2. Test compression using a complete dictionary.

each code word in the compressed data back into the corresponding symbol.

Run-length-based codes

The first data compression codes that researchers investigated for compressing scan vectors encoded runs of repeated values. Jas and Touba proposed a scheme based on run-length codes that encoded runs of 0s using fixed-length code words.[1] To increase the prevalence of runs of 0s, this scheme uses a cyclical scan architecture to allow the application of difference vectors where the difference vector between test cubes t_1 and t_2 is equal to $t_1 \approx t_2$. Careful ordering of the test cubes maximizes the number of 0s in the difference vectors, thereby improving the effectiveness of run-length coding. Chandra and Chakrabarty proposed a technique based on Golomb codes that encode runs of 0s with variable-length code words.[2] The use of variable-length code words allows efficient encoding of longer runs, although it requires a synchronization mechanism between the tester and the chip. Further optimization is achievable by using frequency-directed run-length (FDR) codes[3] and variable-input Huffman codes (VIHC),[4] which customize the code based on the distribution of different run lengths in the data.

Dictionary codes

Another form of coding is *dictionary coding*, which partitions the original data into *n*-bit symbols and uses a dictionary to store each unique symbol. This technique compresses data by encoding each *n*-bits using a *b*-bit code word that corresponds to the symbol's index in the dictionary (*b* is less than *n* when all possible symbols do not occur in the data). Reddy et al. proposed a scan vector compression scheme, illustrated in Figure 2, that uses a complete dictionary.[5] It uses *n* scan chains, and stores each distinct *scan slice* (the *n*-bits loaded into the scan chains in each clock cycle) in the dictionary.

To minimize dictionary size, this technique modifies the test cubes to minimize the number of distinct scan slices. The size of each index equals $\lceil \log_2 n \rceil$, where *n* is the number of distinct scan slices.

A drawback of using a complete dictionary is that the dictionary size can become very large, resulting in too much overhead for the decompressor. Li et al. proposed a partial-dictionary coding scheme that constrains the dictionary size based on the allocated area for the decompressor.[6] If the dictionary size is 2^b, then the 2^b scan slices that occur most frequently go into the dictionary. Scan slices that are not in the dictionary remain unencoded, and the decompressor bypasses the dictionary. An additional extra bit in each code word indicates whether or not to use the dictionary.

Würtenberger et al. proposed using a partial dictionary along with a "correction" network that flips bits to convert a dictionary entry into the desired scan slice.[7] Using the correction network reduces the dictionary size.

Statistical codes

Statistical coding partitions the original data into *n*-bit symbols and assigns variable-length code words based on each symbol's frequency of occurrence. It assigns shorter code words to symbols that occur more frequently, and longer code words to those that occur less frequently. This strategy minimizes the average length of a code word.

Jas et al. described a scan vector compression scheme based on selective Huffman coding.[8] A Huffman code is an optimal statistical code, but its decoder size grows exponentially with symbol size. Selective Huffman coding encodes only the most frequently occurring symbols, leaving the rest unencoded. This allows the use of larger symbol sizes, in turn allowing greater compression than what a similarly sized, full-Huffman decoder (with a smaller symbol size) could achieve.

Constructive codes

Constructive codes exploit the fact that each *n*-bit scan slice typically contains relatively few care bits. Each code word in a constructive code specifies a subset of the *n* bits in the scan slice. It is possible to construct the

scan slice by incrementally specifying all the care bits using a sufficient number of code words. Reda and Orailoglu proposed a scheme that constructs the current scan slice from the previous scan slice by flipping bits.[9] Each code word flips a single bit, so the number of code words used to construct each scan slice equals the number of care bits in the current scan slice that differ from those in the previous scan slice. This technique requires some control information to indicate when scan slice construction is complete and the slice is ready to be shifted into the scan chain. Wang and Chakrabarty proposed a constructive code method that first sets all bits in a scan slice to either 0 or 1 (whichever matches the largest number of care bits), and then incrementally loads the care bits with opposite value using either a single-bit or a group-copy mode.[10]

Linear-decompressor-based schemes

A second category of compression techniques is based on using a linear decompressor. Any decompressor that consists of only wires, XOR gates, and flip-flops is a linear decompressor and has the property that its output space (the space of all possible vectors that it can generate) is a linear subspace spanned by a Boolean matrix. A linear decompressor can generate test vector Y if and only if there exists a solution to the system of linear equations $AX = Y$, where A is the characteristic matrix for the linear decompressor and X is a set of free variables shifted in from the tester (you can think of every bit on the tester as a free variable assigned as either 0 or 1). The characteristic matrix for a linear decompressor is obtainable from symbolic simulation of the linear decompressor; in this simulation a symbol represents each free variable from the tester.[11,12] Encoding a test cube using a linear decompressor requires solving a system of linear equations consisting of one equation for each specified bit, to find the free-variable assignments needed to generate the test cube. If no solution exists, then the test cube is unencodable (that is, it does not exist in the output space of the linear decompressor). In this method, it is difficult to encode a test cube that has more specified bits than the number of free variables available to encode it. However, for linear decompressors that have diverse linear equations (such as an LFSR with a primitive polynomial), if the number of free-variables is sufficiently larger then the number of specified bits, the probability of not being able to encode the test cube becomes negligibly small. For an LFSR with a primitive polynomial, research showed that if the number of free variables is 20 more than the number of specified bits, then the probability of not finding a solution is less than 10^{-6}.[11]

Researchers have proposed several linear decompression schemes, which are either combinational or sequential.

Combinational linear decompressors

Researchers described the use of a combinational linear decompressor in which an XOR of some of the tester channels drives each scan chain.[13,14] This approach uses simpler hardware and control logic than approaches based on sequential linear decompressors. The drawback is that combinational linear decompressors must encode each scan slice using only the free variables shifted in from the tester in a single clock cycle, which is equal to the number of tester channels. The worst-case, most highly specified scan slices tend to limit the amount of achievable compression because the number of tester channels must be sufficiently large to encode the most highly specified scan slices.

Krishna and Touba proposed a method for improving the encoding efficiency of a combinational linear decompressor by dynamically adjusting the number of scan chains loaded in each clock cycle.[15]

Sequential linear decompressors

Sequential linear decompressors are based on linear finite-state machines such as LFSRs, cellular automata, or ring generators.[16] Their advantage lies in allowing the use of free variables from earlier clock cycles to encode a scan slice in the current clock cycle. This provides much greater flexibility than combinational decompressors and helps avoid the problem of the worst-case, most highly specified scan slices limiting the overall compression.

Static reseeding. The earliest work in this area was based on static LFSR reseeding, a technique that computes a seed (an initial state) for each test cube. This seed, when loaded into an LFSR and run in autonomous mode, will produce the test cube in the scan chains.[11] This technique achieves compression by storing only the seeds instead of the full test cubes.

One drawback of using static reseeding for compressing test vectors on a tester is that the tester is idle while the LFSR is running in autonomous mode. One way around this is to use a shadow register for the LFSR to hold the data coming from the tester while the LFSR is running in autonomous mode.[17,18]

Another drawback of static reseeding is that the

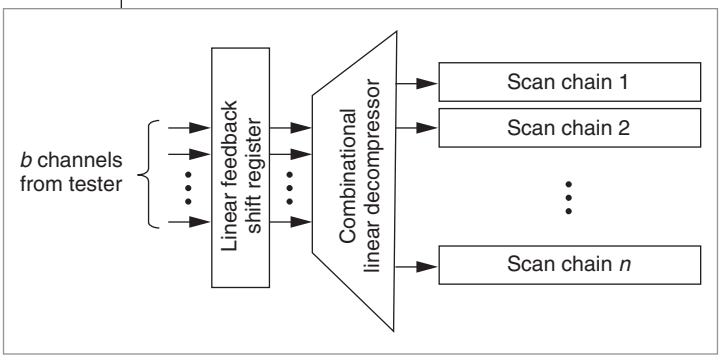

Figure 3. Example sequential linear decompressor.

LFSR must be at least as large as the number of specified bits in the test cube. One way around this is to only decompress a *scan window* (a limited number of scan-slices) per seed.[18-20]

Dynamic reseeding. Könemann,[21] Krishna et al.,[22] and Rajski et al.[23] proposed dynamic reseeding approaches. Dynamic reseeding calls for the injection of free variables from the tester into the LFSR as it loads the scan chains. Figure 3 shows a generic example of a sequential linear decompressor that uses *b* channels from the tester to continuously inject free variables into the LFSR as it loads the scan chains through a combinational XOR network. This network expands the LFSR outputs to fill *n* scan chains. The advantages of dynamic reseeding compared with static reseeding are that it allows continuous flow operation in which the tester is always shifting in data as fast as it can and is never idle, and it allows the use of a small LFSR.

Rajski et al. described a methodology for scan vector compression based on a sequential linear decompressor.[23] Instead of using an LFSR, this work uses a ring generator,[16] which improves encoding flexibility and provides performance advantages. A fixed number of free variables are shifted in when decompressing each test cube. In this case, the control logic is simple because this methodology decompresses every test cube in exactly the same way. Constraining the ATPG generates test cubes that are encodable using the fixed number of free variables.

Könemann described a methodology for scan vector compression in which the number of free variables used to encode each test cube varies.[21] This method requires having an extra channel from the tester to gate the scan clock. For a heavily specified scan slice, this extra gating channel stops the scan shifting for one or more cycles, allowing the LFSR to accumulate a sufficient number of free variables from the tester to solve for the current scan slice before proceeding to the next one. This approach makes it easy to control the number of free variables that the decompressor uses to decompress each test cube. However, the additional gating channel uses some test data bandwidth.

Combined linear and nonlinear decompressors

The amount of compression achievable with linear decompression is limited by the number of specified bits in the test cubes. Although linear decompressors are very efficient at exploiting don't-cares in the test set, they cannot exploit correlations in the specified bits. Hence, they cannot compress the test cubes to less than the total number of specified bits in the test cubes. The specified bits tend to be highly correlated, so one strategy to take advantage of this fact is to combine linear and nonlinear decompression to achieve greater compression than either one can alone.

One method encodes the inputs to a linear decompressor using a statistical code and selects the solution to the system of linear equations for each test cube in such a way that they can be effectively compressed by the statistical code.[19] The statistical code reduces the number of bits that the tester must store for the linear decompressor.

In another method, Sun, Kinney, and Vinnakota combine dictionary coding with a linear decompressor.[24] This methods either uses the dictionary to generate each scan slice or, if it is not present in the dictionary, uses the linear decompressor.

Another method places a nonlinear decompressor between the linear decompressor and the scan chains to compress the number of specified bits that the linear decompress must produce.[25] Doing so allows greater compression because the linear decompressor requires fewer free variables from the tester to solve the linear equations.

Broadcast-scan-based schemes

A third category of techniques is based on the idea of broadcasting the same value to multiple scan chains (a single tester channel drives multiple scan chains). This is actually a special degenerate case of linear decompression in which the decompressor consists of only fan-out wires. Given a particular test cube, the probability of encoding it with a linear decompressor that uses XORs is higher because it has a more diverse output space with fewer linear dependencies than a fan-out network. However, the fact that faults can be detected by many different test cubes provides an additional degree of freedom.

The advantage of broadcast scan is that it is easy to incorporate the decompressor-imposed constraints during ATPG to exploit this degree of freedom: Simply tie dependent inputs together in the circuit description given to the ATPG so that the ATPG algorithm will produce only encodable test cubes. It is not easy to do this for linear decompressors that use XORs, because incorporating XOR gates can significantly degrade ATPG performance. For linear decompressors that use XOR gates, the ATPG first produces a test cube and then must solve the linear equations to check whether the test cube is encodable. So each approach has its advantages. Linear decompressors that use XORs can encode a wider range of test cubes than broadcast scan, but broadcast scan can harness the ATPG to search for encodable test cubes more efficiently. Commercial tools based on both approaches are available.

Broadcast scan (for independent scan chains)

Lee et al. originally proposed broadcast scan in the context of testing independent circuits.[26] The idea is to use a single tester channel to load multiple scan chains that each drive independent circuits. When ATPG targets a fault in one circuit, it generally leaves many inputs unassigned. It can then assign the unassigned inputs for one circuit to target faults in the other circuits. Because the circuits are independent, the fault coverage of this broadcast structure will be the same as the original fault coverage with independent scan chains. If T_i is the number of scan vectors required by the ith independent scan chain, then the number of scan vectors with broadcast scan would be in the range from $Max(T_i)$ to $Sum(T_i)$.

Illinois scan (for dependent scan chains)

Using broadcast scan for multiple scan chains that drive the same circuit might result in reduced fault coverage because some scan cells always hold identical values. To address this shortcoming, Hamzaoglu and Patel proposed a parallel/serial scan architecture that has come to be known as *Illinois scan*.[27] It has two modes of operation:

- *broadcast*, which broadcasts one tester channel to multiple scan chains, and
- *serial,* which loads the scan chains serially.

For faults undetectable in broadcast mode, testing can use the serial mode. For multiple tester channels, you implement Illinois scan by broadcasting each tester channel to a subset of the scan chains. One methodology selects the set of scan chains that each tester channel broadcasts to based on compatibility analysis.[28]

Reconfigurable broadcast scan

For multiple tester channels, one way to increase the number of faults detected by broadcast mode is to reconfigure the set of scan chains that each tester channel broadcasts to. This changes the ATPG constraints, possibly allowing the detection of additional faults without having to resort to serial mode. This reconfiguration can be either static or dynamic.

Static reconfiguration. In static reconfiguration, the configuration changes between test cubes (reconfiguration occurs on a per scan, rather than per shift, basis). Pandey and Patel proposed a static reconfiguration technique that places multiplexers within the scan chains to reconfigure their composition and length.[29] These changes, in turn, alter the constraints imposed by the broadcast structure.

Samaranayake et al. described a static reconfiguration technique that places multiplexers only at the scan chain inputs to reconfigure the set of scan chains that each tester channel broadcasts to.[30] (This technique does not reconfigure the scan chains themselves.) This technique uses compatibility analysis to select the set of configurations. Tang, Reddy, and Pomeranz described using Omega networks to allow CUT-independent design of the reconfiguration network.[31]

Mitra and Kim described a static reconfiguration methodology that forms the configurations by setting different subsets of inputs to 0 in an XOR network instead of using a multiplexer network.[14]

Dynamic reconfiguration. In dynamic reconfiguration, the configuration can change for each scan slice (that is, the reconfiguration occurs on a per-shift basis). This provides much greater flexibility to detect more faults, but requires more control information to indicate when to perform the reconfiguration. With static reconfiguration, the reconfiguration only occurs a few times (only after the tester applies all the test cubes for a particular configuration), whereas dynamic configuration uses multiple reconfigurations for each test cube, thereby requiring more control information. Sitchinava et al. proposed a dynamic reconfiguration scheme in which some tester channels drive multiplexer control signals, which in turn establishes the configuration to use in each clock cycle.[32]

Wang et al. proposed a VirtualScan scheme that generalizes the dynamically reconfigurable broadcast network

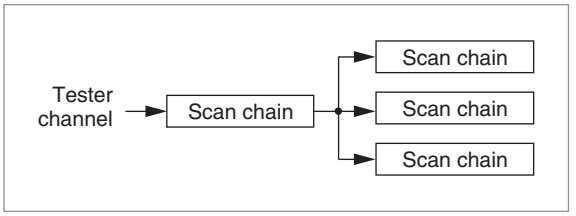

Figure 4. Example scan forest.

and permits the use of both multiplexers and XOR gates.[33] VirtualScan incorporates the network-imposed constraints directly into the ATPG backtrace. Another paper describes a time-division demultiplexing and multiplexing technique for use with VirtualScan to further reduce test application time and test pin-point count in situations where the tester I/O is faster than the scan shift frequency.[34]

Scan forest

Conventional broadcast scan broadcasts values only at the scan chain inputs. Miyase et al. proposed the idea of using a *scan forest*—that is, a scan architecture that broadcasts the outputs of scan cells, thereby creating a tree structure.[35] For example, as Figure 4 shows, this technique would replace a single scan chain of 100 scan cells with a scan tree in which the first 10 scan cells form the root, and the 10th scan cell broadcasts (fans out) to three branch scan chains, each having 30 scan cells. The height of this example scan tree would be 40 (10 for the root plus 30 for the branches), and dependency constraints would exist for only the three branch scan chains and not the stem. The added flexibility of using internal broadcasting in the scan chains can be used to design a scan forest that achieves high fault coverage without reconfiguration. The drawback is that the scan stitching might not be optimal for layout.

Comparison of schemes

Table 1 summarizes some general qualitative comparisons between the different categories of schemes. The advantage of the nonlinear code-based schemes is that they can efficiently exploit correlations in the specified bits and are usable on any set of test cubes (they do not require ATPG constraints). Hence, they are effective for IP cores for which no structural information is available. The drawback is that these schemes are generally not as efficient in exploiting don't-cares as linear techniques. Because industrial test cubes typically have more than 95% don't-care bits, linear techniques generally provide greater compression.

Combinational linear decompressors are simple to implement, requiring very little control logic. The main drawback is that the decompressor must produce each scan slice using only the free variables that come from the tester in a single clock cycle. Consequently, the most heavily specified scan slices tend to limit the amount of compression. Sequential linear decompressors avoid this obstacle by combining free variables across multiple clock cycles such that the linear equations for the scan cells have very low linear dependency. This provides high encoding flexibility, increasing the proba-

Table 1. General characteristics of different categories of schemes.

Category	Pros	Cons	Commercial tools available?
Code based			
	Exploits correlation in specified bits; usable on any set of test cubes	Not as efficient in exploiting don't-cares; more complex control logic	No
Linear decompressors			
Combinational	Uses only combinational gates; very simple control logic	Each slice encoded independently; two-step ATPG	Yes
Sequential	Low linear dependency; very high encoding flexibility	More complex decompressor; two-step ATPG	Yes
Broadcast scan			
Static reconfiguration	Simple decompressor; efficient one-step ATPG	High linear dependency; lower encoding flexibility	Yes
Dynamic reconfiguration	More encoding flexibility than static reconfiguration; one-step ATPG	More control information than for static reconfiguration; less encoding flexibility than sequential linear decompressors	Yes

bility of being able to encode a test cube.

Broadcast scan has less encoding flexibility than linear decompressors that use XOR gates because it has much greater linear dependency (some scan cells receive identical values). Thus, the space of encodable test cubes is smaller. However, the advantage of broadcast scan is that the ATPG backtrace can incorporate constraints for the decompressor; this way, the ATPG will produce only encodable test cubes. With sequential linear decompressors, the constraints are too complex to incorporate into the ATPG backtrace, requiring a two-step process: generate the test cube, and then solve the linear equations to see whether or not the test cube is encodable. Both schemes have advantages, and commercial tools based on both approaches are available.

Broadcast scan can also employ either static or dynamic reconfiguration. The static approach requires less control information, but has less encoding flexibility. The dynamic approach requires control information to select the configuration for each clock cycle, but provides greater encoding flexibility.

RESEARCHERS HAVE PROPOSED a wide variety of techniques for test vector compression, and vendors have developed and successfully deployed commercial tools based on these techniques. As the ratio of test data volume to I/O pins continues to grow rapidly, industry will require further advances in test compression technology to keep pace.

Future avenues for increasing compression might include developing new ATPG procedures better optimized for the compression scheme, using adaptive techniques that adjust for different portions of the circuit or test set, and moving toward hybrid BIST schemes. ■

References

1. A. Jas and N.A. Touba, "Test Vector Compression via Cyclical Scan Chains

Glossary

Broadcast scan. Using a fan-out network to broadcast the same value to multiple scan chains.

Constructive code. Fixed-to-variable code in which each code word specifies a subset of the bits in a symbol. The code constructs a symbol by using a sufficient number of code words to specify all of the symbol's care bits.

Dictionary code. Fixed-to-fixed code that stores only unique symbols in a dictionary; each code word is an index into the dictionary.

Dynamic reseeding. Incrementally changing the state of an LFSR each clock cycle as it is running.

Free variable. Each bit on the tester that the ATPG can assign as either a 0 or 1 to encode a test cube.

Hybrid BIST. Combining deterministic data stored on an external tester with pseudorandom BIST to achieve high fault coverage (can be as simple as using top-up vectors or a more elaborate scheme for embedding deterministic patterns in a pseudorandom sequence).

Illinois scan. Broadcast scan architecture that has both a broadcast and serial mode.

Linear decompressor. A decompressor consisting of only wires, XOR gates, and flip-flops; it has the property that a Boolean matrix spans its output space. So, solving a system of linear equations will check for whether these decompressors can encode a particular test cube or not.

Output response compaction. Using lossy compression techniques to compact the output response.

Output space of decompressor. Set of all test cubes that the decompressor can encode.

Run-length code. Variable-to-fixed code that encodes runs of repeated values with code words.

Scan forest. Generalized broadcast scan technique that broadcasts the output of internal scan cells to drive multiple scan chains.

Scan slice. Set of scan cells that load from the tester in the same clock cycle. For n scan chains of length m, there are m scan slices, each consisting of n scan cells.

Stand-alone BIST. Traditional notion of BIST, in which all test pattern generation and output response analysis occurs on-chip, without the need for an external tester.

Static reseeding. Loading a specific initial state into an LFSR, and running it in autonomous mode to generate a particular test vector.

Statistical code. Fixed-to-variable code that encodes symbols with variable-length code words. It uses shorter code words for more frequently occurring symbols.

Test data compression. Compressing the data stored on the tester using both test vector compression and output response compaction.

Test vector compression. Using lossless compression techniques to reduce the amount of data that the tester must store to reproduce a deterministic test set.

Test cube. Test vector in which inputs that the ATPG does not assign are left as don't-cares.

and Its Application to Testing Core-Based Designs," *Proc. Int'l Test Conf.* (ITC 98), IEEE CS Press, 1998, pp. 458-464.

2. A. Chandra and K. Chakrabarty, "System-on-a-Chip Test-Data Compression and Decompression Architectures Based on Golomb Codes," *IEEE Trans. Computer-Aided Design*, vol. 20, no. 3, Mar. 2001, pp. 355-368.
3. A. Chandra and K. Chakrabarty, "Test Data Compression and Test Resource Partitioning for System-on-a-Chip Using Frequency-Directed Run-Length (FDR) Codes," *IEEE Trans. Computers*, vol. 52, no. 8, Aug. 2003, pp. 1076-1088.
4. P.T. Gonciari, B.M. Al-Hashimi, and N. Nicolici, "Variable-Length Input Huffman Coding for System-on-a-Chip Test," *IEEE Trans. Computer-Aided Design*, vol. 22, no. 6, June 2003, pp. 783-796.
5. S.M. Reddy et al., "On Test Data Volume Reduction for Multiple Scan Chain Designs," *Proc. 20th VLSI Test Symp.* (VTS 02), IEEE CS Press, 2002, pp. 103-108.
6. L. Li, K. Chakrabarty, and N.A. Touba, "Test Data Compression Using Dictionaries with Selective Entries and Fixed-Length Indices," *ACM Trans. Design Automation Electrical Systems*, vol. 8, no. 4, Apr. 2003, pp. 470-490.
7. A. Würtenberger, C.S. Tautermann, and S. Hellebrand, "Data Compression for Multiple Scan Chains Using Dictionaries with Corrections," *Proc. Int'l Test Conf.* (ITC 04), IEEE CS Press, 2004, pp. 926-935.
8. A. Jas et al., "An Efficient Test Vector Compression Scheme Using Selective Huffman Coding," *IEEE Trans. Computer-Aided Design,* vol. 22, no. 6, June 2003, pp. 797-806.
9. S. Reda and A. Orailoglu, "Reducing Test Application Time Through Test Data Mutation Encoding," *Proc. Design, Automation, and Test in Europe, Conf. and Exhibition* (DATE 02), IEEE CS Press, 2002, pp. 387-393.
10. Z. Wang and K. Chakrabarty, "Test Data Compression for IP Embedded Cores Using Selective Encoding of Scan Slices," *Proc. Int'l Test Conf.* (ITC 05), IEEE Press, 2005, pp. 581-590.
11. B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs," *Proc. European Test Conf.* (ETC 91), VDE Verlag, 1991, pp. 237-242.
12. L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*, Morgan Kaufmann, 2006.
13. I. Bayraktaroglu and A. Orailoglu, "Concurrent Application of Compaction and Compression for Test Time and Data Volume Reduction in Scan Designs," *IEEE Trans. Computers,* vol. 52, no. 11, Nov. 2003, pp. 1480-1489.
14. S. Mitra and K.S. Kim, "XPAND: An Efficient Test Stimulus Compression Technique," *IEEE Trans. Computers*, vol. 55, no. 2, Feb. 2006, pp. 163-173.
15. C.V. Krishna and N.A. Touba, "Adjustable Width Linear Combinational Scan Vector Decompression," *Proc. Int'l Conf. Computer-Aided Design* (ICCAD 03), IEEE CS Press, pp. 863-866.
16. G. Mrugalski, J. Rajski, and J. Tyszer, "Ring Generators—New Devices for Embedded Test Applications," *IEEE Trans. Computer-Aided Design*, vol. 23, no. 9, Sept. 2004, pp. 1306-1320.
17. P. Wohl et al., "Efficient Compression and Application of Deterministic Patterns in a Logic BIST Architecture," *Proc. 41st Design Automation Conf.* (DAC 03), ACM Press, 2003, pp. 566-569.
18. E.H. Volkerink and S. Mitra, "Efficient Seed Utilization for Reseeding Based Compression," *Proc. 21st VLSI Test Symp.* (VTS 03), IEEE CS Press, 2003, pp. 232-237.
19. C.V. Krishna, A. Jas, and N.A. Touba, "Reducing Test Data Volume Using LFSR Reseeding with Seed Compression," *Proc. Int'l Test Conf.* (ITC 02), IEEE CS Press, 2002, pp 321-330.
20. P. Wohl et al., "Efficient Compression of Deterministic Patterns into Multiple PRPG Seeds," *Proc. Int'l Test Conf.* (ITC 05), IEEE Press, 2005, pp. 916-925.
21. B. Koenemann et al., "A SmartBIST Variant with Guaranteed Encoding," *Proc. 10th Asian Test Symp.* (ATS 01), IEEE CS Press, 2001, pp. 325-330.
22. C.V. Krishna, A. Jas, and N.A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding," *Proc. Int'l Test Conf.* (ITC 01), IEEE CS Press, 2001, pp. 885-893.
23. J. Rajski et al., "Embedded Deterministic Test," *IEEE Trans. on Computer-Aided Design*, vol. 23, no. 5, May 2004, pp. 776-792.
24. X. Sun, L. Kinney, and B. Vinnakota, "Combining Dictionary Coding and LFSR Reseeding for Test Data Compression," *Proc. 42nd Design Automation Conf.* (DAC 04), ACM Press, 2004, pp. 944-947.
25. J. Lee, and N.A. Touba, "Combining Linear and Non-Linear Test Vector Compression using Correlation-Based Rectangular Coding," *Proc. 24th VLSI Test Symp.* (VTS 06), IEEE CS Press, 2006, pp. 252-257.
26. K.-J. Lee, J.J. Chen, and C.H. Huang, "Using a Single Input to Support Multiple Scan Chains," *Proc. Int'l Conf. Computer-Aided Design* (ICCAD 98), IEEE CS Press, 1998, pp. 74-78.
27. I. Hamzaoglu and J.H. Patel, "Reducing Test Application Time for Full Scan Embedded Cores," *Digest of Papers 29th Ann. Int'l Symp. Fault-Tolerant Computing* (FTCS 99), IEEE Press, 1999, pp. 260-267.
28. M.A. Shah and J.H. Patel, "Enhancement of the Illinois

Scan Architecture for Use with Multiple Scan Inputs," *IEEE Computer Soc. Ann. Symp. VLSI* (ISVLSI 04), IEEE CS Press, 2004, pp. 167-172.
29. A.R. Pandey and J.H. Patel, "Reconfiguration Technique for Reducing Test Time and Test Volume in Illinois Scan Architecture Based Designs," *Proc. 20th VLSI Test Symp.* (VTS 02), IEEE CS Press, 2002, pp. 9-15.
30. S. Samaranayake et al., "A Reconfigurable Shared Scan-In Architecture," *Proc. 21th VLSI Test Symp.* (VTS 03), IEEE CS Press, 2003, pp. 9-14.
31. H. Tang, S.M. Reddy, and I. Pomeranz, "On Reducing Test Data Volume and Test Application Time for Multiple Scan Designs," *Proc. Int'l Test Conf.* (ITC 03), IEEE CS Press, 2003, pp. 1079-1088.
32. N. Sitchinava et al., "Changing the Scan Enable During Shift," *Proc. 22nd VLSI Test Symp.* (VTS 04), IEEE CS Press, 2004, pp. 73-78.
33. L.-T. Wang et al., "VirtualScan: A New Compressed Scan Technology for Test Cost Reduction," *Proc. Int'l Test Conf.* (ITC 04), IEEE CS Press, 2004, pp. 916-925.
34. L.-T. Wang et al., "UltraScan: Using Time-Division Demultiplexing/Multiplexing (TDDM/TDM) with VirtualScan for Test Cost Reduction," *Proc. Int'l Test Conf.* (ITC 05), IEEE Press, 2005, pp. 946-953.
35. K. Miyase, S. Kajihara, and S.M. Reddy, "Multiple Scan Tree Design with Test Vector Modification," *Proc. 13th Asian Test Symp.* (ATS 04), IEEE CS Press, 2004, pp. 76-81.

Nur A. Touba is an associate professor in the Department of Electrical and Computer Engineering at the University of Texas at Austin. His research interests include DFT and dependable computing. Touba has a BS from the University of Minnesota, and an MS and a PhD from Stanford University, all in electrical engineering. He is a senior member of the IEEE.

■ Direct questions and comments about this article to Nur A. Touba, Computer Engineering Research Center, Dept. of Electrical and Computer Eng., Engineering Science Bldg., University of Texas at Austin, Austin, TX 78712-1084; touba@ece.utexas.edu.

For further information on this or any other computing topic, visit our Digital Library at http://www.computer.org/publications/dlib.