Survey of VLSI Test Data Compression Methods

Usha Mehta

***Abstract*—It has been seen that the test data compression has been an emerging need of VLSI field and hence the hot topic of research for last decade. Still there is a great need and scope for further reduction in test data volume. This reduction may be lossy for output side test data but must be lossless for input side test data. This paper summarizes the different methods applied for lossless compression of the input side test data, starting with simple code based methods to combined/hybrid methods. The basic goal here is to prepare survey on current methodologies applied for test data compression and prepare a platform for further development in this avenue.**

***Index Terms*—VLSI, Testing, data compression**

I. Need of Test Data Compression

AS a result of the emergence of new fabrication technologies anddesign complexities,standard stuck-at scan tests are no longer sufficient. The number of tests and corresponding data volume increase with each new fabrication process technology.

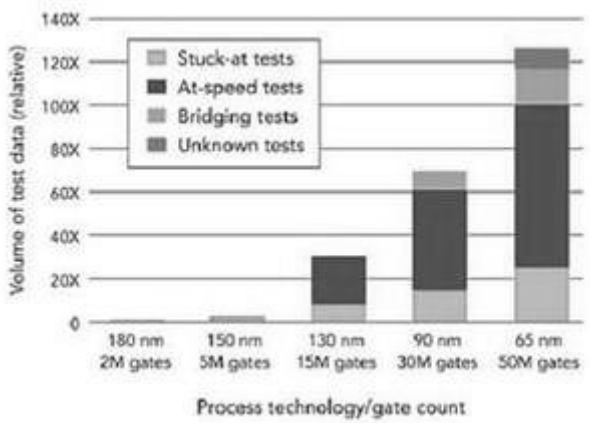

Fig. 1. Volume of Test Data

As fabrication technologies evolve, test application time and test data volume are drastically increasing just to maintain test quality requirements. New tests require: greater than 2X the test time to handle devices that double in gate count but maintain the same number of scan channels, 3X to 5X the number of patterns to support at-speed scan testing for the growing population of timing defects at 130-nm and smaller fabrication processes, and 5X the number of patterns to handle multiple-detect and new DFM-based fault models.

Thus, the starting point is 10X compression just to maintain tester throughput and 20X if new fault models are used, which becomes 40X if the next design doubles in size. If you consider reducing the block-level routing and top-level scan pins by 5X, that means you need 5X more compression on top of the existing compression. Supporting multisite testing or DFM-based fault models will triple the compression requirements at a minimum. A major benefit of compression is to reduce test pin count, which is a major cost benefit at manufacturing. As a result, some companies are already looking for compression well beyond 100X tester cycle reduction[4][5].

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 $\geq$*(amount of test data on tester) / (number of tester channels X tester clock rate)*[1]

The resurgence of interest in test data compression has also led to new commercial tools that can provide over 10X compression for large industrial designs. For example, the OPMISR and SmartBIST[3] tools from IBM and the TestKompress tool from Mentor Graphics[2] reduce test data volume and testing time through the use of test data compression and on-chip decompression.

II. Test Data Compression Techniques

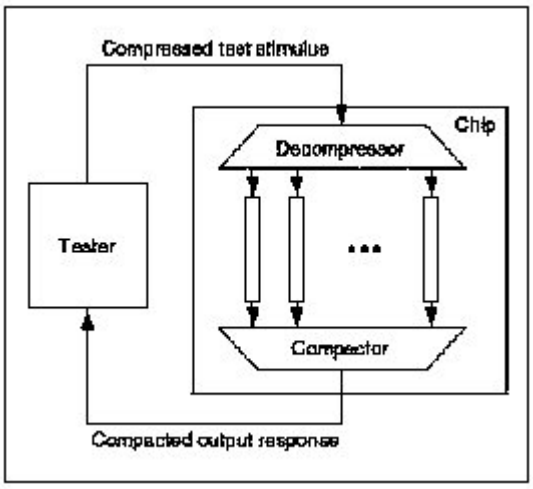

Fig. 2. Test data compression.

Usha Mehta is with the Department of Electronics & Communication Institute of Technology, Nirma University Ahmedabad-382481, e-mail:usha.mehta@nirmauni.ac.in

As Figure 2 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).

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 lossless (i.e. it must reproduce all the care bits after decompression) to preserve fault coverage. 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 (i.e. the ATPG does not randomly fill the don't-cares). 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.

Test vector compression schemes fall broadly into three categories[1]:

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

III. Code Based Test Data Compression Techniques

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

The quantity of test data rapidly increases, while, at the same time, the inner nodes of dense SoCs become less accessible from the external pins. The testing problem is further exacerbated by the use of *intellectual property* (IP) cores, since their structure is often hidden from the system integrator. In such cases, no modifications can be applied to the cores or their scan chains, whereas neither automatic test pattern generation nor fault simulation tools can be used. Only precomputed test sets are provided by the core vendors, which should be applied to the cores during testing. So in this case, any test data compression technique which is ATPG independent and fault simulation independent is most preferable. So code based test data compression technique satisfies both the requirements i.e. it applies directly to ready test patterns and doesn't require any ATPG. The same way it doesn't require any fault simulation also. The other advantages that can be achieved in some of the cases are difference patterns and reordering of test patterns.

A few important factors to be considered with any compression technique are:

- The amount of compression possible,
- The area overhead because of decoding architecture. The on-chip decompression circuitry must be small so that it does not add significant area overhead. The properties of the code are chosen such that the decoder has a very small area and is guaranteed to be able to decode the test data as fast as the tester can transfer it.
- The reduction in test time. Transferring compressed test vectors takes less time than transferring the full vectors at a given bandwidth. However, in order to guarantee a reduction in the overall test time, the decompression process should not add much delay (which would subtract from the time saved in transferring the test data).
- The scalability of compression (does the compression technique work with various design sizes, with few or many scan channels, and with different types of designs?),
- Power dissipation is an important factor in today's chip design. Power dissipation in CMOS circuits is proportional to the switching activity in the circuit. During normal operation of a circuit, often a small number of flip flops change values in each clock cycle. However, during test operation, large numbers of flip flops switch, especially when test patterns are scanned into the scan chain. Compacting the test set often requires replacing (mapping) don't cares with specified bits "0" or "1". This process may increase switching activity of scan flip flops and eventually the scan-in power dissipation. There are usually plenty of don't cares in test patterns generated for scan. Test compression method should effectively use this opportunity for compression as well as power reduction.
- The robustness in the presence of X states (can the design

maintain compression while handling X states without losing coverage?),
- The ability to perform diagnostics of failures when applying compressed patterns.
- Type of Decoder: data-independent decoder or data dependant decoder. In the former category, the on-chip decoder or decompression program is universal, i.e., it is reusable for any test set. In contrast, the decoder of a data-dependent technique can only decompress a specific test vector. They have difficulties in terms of size and organization for improved compression and often require large on-chip memory[6] Hence, data-independency is a preferable property.

The data compression codes are generally classified into four categories based on symbol size and codeword size.

1) Fixed to Fixed Coding Schemes: Where symbol size as well as codeword size is fixed. Like Dictionary Code
2) Fixed to Variable Coding Schemes: Where symbol size is fixed but codeword size is variable. Like Huffman Code.
3) Variable to Fixed Coding Schemes: Where symbol size is variable but codeword size is fixed. Like Run Length Code
4) Variable to Variable Coding Schemes: Where symbol as well as codeword size is variable. Like Golomb Code.

During these years, the researchers have developed a large number of variants of above schemes. The following survey covers most of all such variants. Based on the basic schemes and evolved variants, these techniques are broadly divided into four different categories.

1) Run-Length Based Codes
2) Dictionary Based Codes
3) Statistical Codes
4) Constructive Codes

IV. Linear-decompression-based schemes decompress the data using only linear operations

A second category of compression techniques is based on using a linear decompressor. Any decompressor that consists of only wires, XOR gates, and flipflops 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.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 freevariable 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}.

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

V. 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. 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 scanslices) per seed.

VI. Conclusions

In this paper, the overview of wide variety of test data compression techniques proposed by researchers in current era is covered. This study analysis draws a conclusion that as the design complexity and hence test data volume continues to grow, the test data compression will be a major demand to reduce test time and test cost. The data independent compression methods, i.e. code based scheme can be most attractive sill in future avenues also. The hybrid methods combining code based scheme with other scheme like linear decopressor or broadcast based scheme can be further explored.

References

[1] Nur A. Tauba, ``Survey of Test Vector Compression Techniques'' IEEE transaction Design & Test of Computers , 2006.
[2] A. Chandra, K. Chakrabarty, ``Efficient test data compression and decompression for system-on-a-chip using internal scan chains and Golomb coding'', DATE '01: Proceedings of the conference on Design, automation and test in Europe, March 2001.
[3] Anshuman Chandra, Krishnendu Chakrabarty, ``Test Data Compression for System-on-a-Chip Using Golomb Codes'' VTS '00: Proceedings of the 18th IEEE VLSI Test Symposium, 2000.
[4] Anshuman Chandra, Krishnendu Chakrabarty, Frequency-Directed Run-Length (FDR) Codes with Application to System-on-a-Chip Test Data Compression, Proceedings of the 19th IEEE VLSI Test Symposium, March 2001.

[5] Anshuman Chandra, Krishnendu Chakrabarty, Test Data Compression and Test Resource Partitioning for System-On-Chip Using Frequency-Directed Run-Length (FDR) Codes IEEE Transactions on Computers , Volume 52 Issue 8, August 2003.

[6] Xrysovalantis Kavousianos, Emmanouil Kalligeros, Dimitris Nikolos, Multilevel-Huffman test-data compression for IP cores with multiple scan chains, IEEE Transactions on Very Large Scale Integration (VLSI) Systems , Volume 16 Issue 7 , July 2008.

Usha Mehta received B.E. degree in electronics engineering from Gujarat University and Master Degree in VLSI Design from Nirma University. Currently She is Associate Professor in Electronics and Communication Engineering Department, of Institute of Technology, Nirma University. Her research interests include ATPG, Test Data Compression and DFT. She is a member of the IEEE, IETE, VSI and ISTE.