International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 2 Issue 12, December - 2013

Overview of Hardware Based Test Compression

V. Jayapradha M.E., (Phd)
Asst Prof, SCSVMV University

Dr. S. Ravi
Dr.MGR University

R. Kamalakkannan M.E., (Phd)
Asst Prof, SCSVMV University

ABSTRACT

As devices growing in gate count, scan test data volume and application time also increasing. Many Test compression techniques have been developed to reduce test data volume and test application time. Test compression techniques are easy to adopt in industry because they are based on scan based method. Test compression is achieved by adding some additional on-chip hardware before the scan chains to decompress the test vectors coming from the tester and after the scan chins to compact the response going to the tester ,this paper gives the overview of various test compression schemes in VLSI testing and compares the various testing schemes

TEST COMPRESSION

Reducing the test vectors applied to the CUT is called test compression. ATE generates the suitable test vectors which is compressed by the test compressor while applying to the CUT is decompressed and the responses are compacted then which is sent to the ATE. 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. 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.

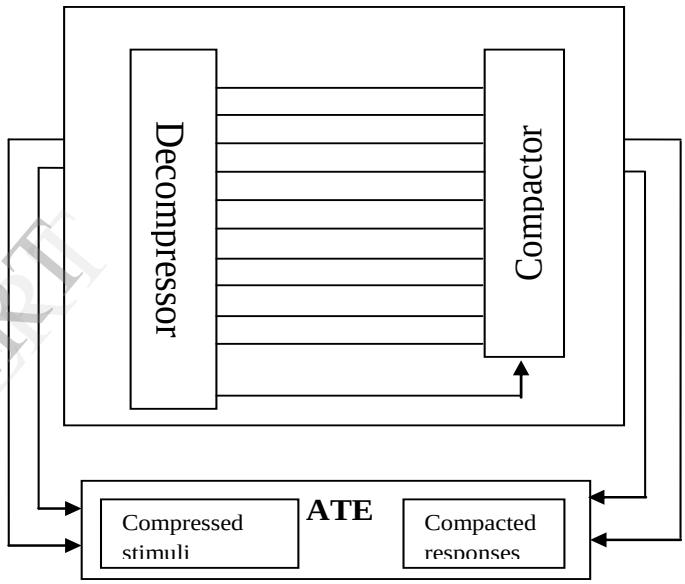

Fig 1 :Test compression block diagram

Various Test Compression Methods

Many researchers have been developed many algorithms for test data compression we can classify the methods in three main category.

(i) Code based scheme
(ii) Linear decompression based scheme
(iii)Broadcast scan based scheme

Code Based Scheme

It is the popular scheme which has been successfully adopted in digital communication. Well known Huffman coding, RLC etc., these methods coming under this category. 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 following coding techniques are coming under code based scheme.

(i) Dictionary code (fixed –fixed)
(ii) Huffman code (fixed –variable)
(iii)Run length code (variable –fixed)
(iv)Golomb code (variable –variable)

Dictionary Code

LZW uses fixed-length code words to represent variable-length strings of test bits that commonly occur together, the LZW encoder and decoder build up the same dictionary dynamically while receiving the data. LZW places longer and longer repeated entries into a dictionary, and then emits the *code* for an element, rather than the string itself, if the element has already been placed in the dictionary.

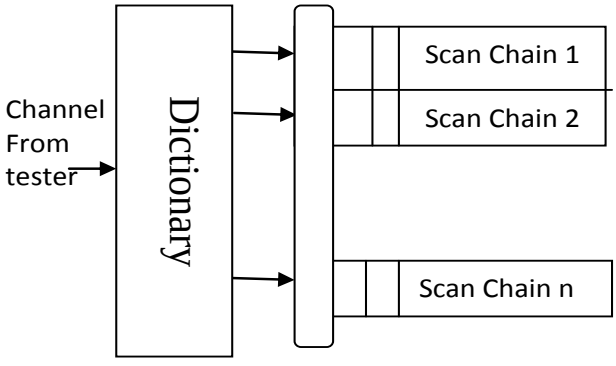

Fig2 : Dictionary Based Coding

Huffmann Coding

It is a lossless coding method which follows the principles of Variable length coding High frequency set of test bits are assigned for shorter codeword low frequency set of test bits Are assigned for longer codeword.

Table 1: Huffman coding

Symbol	Frequency	Pattern	Huffmann code
S_0	22	0010	10
S_1	13	0100	00
S_2	7	0110	110
S_3	5	0111	010
S_4	3	0000	0110
S_5	2	1000	0111
S_6	2	0101	11100
S_7	1	1011	111010
S_8	1	1100	111011
S_9	1	0001	111100
S_{10}	1	1101	111101
S_{11}	1	1111	111110
S_{12}	1	0011	111111

Run Length Coding

run : the repetition of a symbol.
run-length**:** number of repeated symbols
Instead of encoding the consecutive Symbols/test bits , it is more efficient to encode the run-length and the value that these consecutive symbols commonly share

Symbol	**Code**
000002222	0524

Golomb Code

Golomb coding is a lossless data compression method using a family of data compression codes invented by Solomon W. Golomb in the 1960s. Alphabets following a geometric distribution will have a Golomb code as an optimal prefix code, making Golomb coding highly suitable for situations in which the occurrence of small values in

the input stream is significantly more likely than large values.

Note below that this is the Rice–Golomb encoding, where the remainder code uses simple truncated binary encoding, also named "Rice coding" (other varying-length binary encodings, like arithmetic or Huffman encodings, are possible for the remainder codes, if the statistic distribution of remainder codes is not flat, and notably when not all possible remainders after the division are used). In this algorithm, if the *M* parameter is a power of 2, it becomes equivalent to the simpler Rice encoding.[11]

1. Fix the parameter *M* to an integer value.
2. For *N*, the number to be encoded, find
    1. quotient = *q* = int[*N*/*M*]
    2. remainder = *r* = *N* modulo *M*
3. Generate Codeword
    1. The Code format : <Quotient Code><Remainder Code>, where
    2. Quotient Code (in unary coding)
        1. Write a *q*-length string of 1 bits
        2. Write a 0 bit
    3. Remainder Code (in truncated binary encoding)

1. If *M* is power of 2, code remainder as binary format. So $\log_2(M)$ bits are needed. (Rice code)
2. If *M* is not a power of 2, set $b = \lceil \log_2(M) \rceil$
    1. If $r < 2^b - M$ code *r* as plain binary using *b*-1 bits.
    2. If $r \geq 2^b - M$ code the number $r + 2^b - M$ in plain binary representation using *b* bits.

Linear Decompression Based Scheme

A second category of compression techniques is based on using a linear de-compressor. Any de-compressor that consists of only wires, XOR gates, and flip flops is a linear de-compressor 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 de-compressor 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 de-compressor 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 de-compressor is obtainable from symbolic simulation of the linear de-compressor; in this simulation a symbol represents each free variable from the tester.[7],[8] combinational.

Researchers described the use of a combinational linear de-compressor in which an XOR of some of the tester channels drives each scan chain.[9],[5] This approach uses simpler hardware and control logic than approaches based on sequential linear de-compressors. The drawback is that combinational linear de- compressors 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 de-compressor by

dynamically adjusting the number of scan chains loaded in each clock cycle.

Sequential linear de-compressors are based on linear finite-state machines such as LFSRs, cellular automata, or ring generators.[6] 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 de-compressors and helps avoid the problem of the worst-case, most highly specified scan slices limiting the overall compression.

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 de-compressor consists of only fan-out wires. Given a particular test cube, the probability of encoding it with a linear de-compressor 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 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.[10] 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 [2] (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[3] 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.[5]

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.

Table 2: Compression of various test compression methods

Category	Merits	De-merits
Code Based	Uses correlation in specified bits; usable on any set of test cubes	Not as efficient in handling don't cares; more complex control logic
Linear de compressors		
Combinational	Uses only combinational gates; very simple control logic	Each slice encoded independently;
Sequential	Low linear dependency; very high encoding flexibility	More complex de compressor;
Broadcast Scan		
Static reconfiguration	Simple de compressor;	High linear dependency; lower encoding flexibility
Dynamic reconfiguration	More flexible than static reconfiguration;	More control information than for static reconfiguration; less encoding flexibility than sequential linear de compressors

Conclusion

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. Various methods are compared in table 1. Merits and demerits of various techniques listed in the table.From the analysis it is found that dynamic reconfiguration can provide the better result in test compression and commercial tools also available for this.

References

. [1]Test Data Compression Using Efficient Bitmask and Dictionary Selection Methods Kanad Basu, *Student Member, IEEE,* and Prabhat Mishra, *Senior Member, IEEE,* IEEE Transactions On Very Large Scale Integration (Vlsi) Systems, Vol. 18, No. 9, September 2010

[2]. S. Samaranayake et al., "A Reconfigurable Shared Scan-In Architecture," Proc. 21th VLSI Test Symp. (VTS 03), IEEE CS Press, 2003, pp. 9-14.

[3] 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.

[4] N. Sitchinava et al., "Changing the Scan Enable During Shift," Proc. 22nd VLSI Test Symp. (VTS 04), IEEE CS Press, 2004, pp. 73-78.

[5]. 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.

[6]. 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.

[7]. B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs," Proc. European Test Conf. (ETC 91), VDE Verlag, 1991, pp. 237-242.

[8]. L.-T. Wang, C.-W. Wu, and X. Wen, VLSI Test Principles and Architectures: Design for Testability, Morgan Kaufmann, 2006.

[9]. 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.

[10]. 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

[11] http://en.wikipedia.org/wiki/Golomb_coding