INTEGRATION, the VLSI journal ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Enhancement of test data compression with multistage encoding

S. Sivanantham [a], M. Padmavathy [b], Ganga Gopakumar [c], P.S. Mallick [a,*], J. Raja Paul Perinbam [d]

[a] School of Electrical Engineering, VIT University, Vellore 632014, Tamil Nadu, India
[b] School of Electronics Engineering, VIT University, Vellore 632014, Tamil Nadu, India
[c] Sankalp and KPIT Semiconductors Pvt. Ltd, Bangalore, Karnataka, India
[d] Department of ECE, KCG College of Technology, Chennai 600097, Tamil Nadu, India

ARTICLE INFO

Article history:
Received 9 July 2013
Received in revised form
6 December 2013
Accepted 6 December 2013

Keywords:
VLSI testing
Test data compression
Huffman coding
FDR coding
Design for testability
Low-power testing

ABSTRACT

In this paper, we present two multistage compression techniques to reduce the test data volume in scan test applications. We have proposed two encoding schemes namely alternating frequency-directed equal-run-length (AFDER) coding and run-length based Huffman coding (RLHC). These encoding schemes together with the nine-coded compression technique enhance the test data compression ratio. In the first stage, the pre-generated test cubes with unspecified bits are encoded using the nine-coded compression scheme. Later, the proposed encoding schemes exploit the properties of compressed data to enhance the test data compression. This multistage compression is effective especially when the percentage of do not cares in a test set is very high. We also present the simple decoder architecture to decode the original data. The experimental results obtained from ISCAS'89 benchmark circuits confirm the average compression ratio of 74.2% and 77.5% with the proposed 9C-AFDER and 9C-RLHC schemes respectively.

1. Introduction

Developments in process technology have led to the design of systems with millions of transistors on a single chip and it has resulted in an increase of test data required to test the circuits. Conventional external testing processes involve storing all test vectors and test responses on the automatic test equipment (ATE) memory. The test data volume for the scan-based test is normally very large due to its single pattern length generated using a combinational automatic test pattern generation (ATPG) tool. The test application time depends on the amount of test data stored on ATE, the time required to transfer the test data from ATE to the core and length of the scan chain. But these testers have limited memory, speed and I/O channels. The system-on-a-chip (SoC) revolution also gives challenges in the area of power dissipation, especially for design and test engineers. Generally, a circuit or system consumes more power in a test mode than in a normal mode. This high power consumption during testing affects the circuit reliability [1]. Test power depends on the number of scan elements present in the circuit. Efficient test data reduction techniques can reduce the testing time, test power and ATE memory requirements.

Linear compression schemes are very efficient at exploiting unspecified bits in the test cubes to achieve a large amount of compression. Several on-chip pattern decompression techniques were proposed to reduce test data volume such as linear feedback shift register (LFSR) reseeding [2–6], mutation encoding [7], scan-chain concealment [8], reconfigurable serial multiplier [9], packet-based compression [10] techniques are among these categories. A scan architecture called reconfigured scan forest was proposed to reduce test data volume and test application cost by [11]. Ward et al. [12] proposed a scheme which combines a linear decompressor with a non-linear decoder to provide very high levels of compression for test data. A technique for simultaneous reduction of both test data volume and test power named linear decompressor based test compression is presented in [13]. This scheme divides the test cubes into two blocks, namely test cube with low toggles and high toggles which feeds the scan-chain with a novel design-for-testability (DFT) architecture to reduce the scan-in transitions. Several other techniques such as embedded deterministic test (EDT) [14], smartBIST [15] and reconfigurable interconnection network (RIN) [16] were also proposed to reduce the test data volume. However, these methods are not suitable to test the embedded cores since structural information of the circuits is required for test generation and fault simulation. Test data compression based on time-multiplexing is presented in [17],

* Corresponding author.
E-mail addresses: ssivanantham@vit.ac.in (S. Sivanantham), psmallick@vit.ac.in (P.S. Mallick).

0167-9260/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.vlsi.2013.12.001

where the compressed seeds are passed to every embedded core by sharing the data channels. Solana [18] proposed a new scan architecture called virtual chain partition (VCP), which is useful for embedded cores to reduce the test application time, test data volume and test power. This scheme determines the maximum reduction in test cycles obtainable with the architecture and selects the most suitable configuration for each circuit. However, this scheme requires large area overhead.

The code-based schemes use data compression techniques such as statistical coding [19,20], Golomb coding [21] and run-length coding [22–30] to encode the test cubes. In these approaches, the original data are partitioned into symbols, and then each symbol is assigned to a codeword to form the encoded data. Each codeword is converted into the corresponding symbol with on-chip decompression hardware. These methods do not require the structural information about the circuit under tests (CUTs) and they are more suitable for intellectual property (IP) core based SoCs. Tehranipoor et al. [31] proposed to use only nine codewords to encode the test data. The frequency-directed run-length (FDR) code [30] is a variable-to-variable-length code which maps variable-length runs of 0 s to variable-length codewords. The FDR code is not suitable for the test sets which contains more number of 1 s. The compression methods such as alternating FDR (ALT-FDR) coding [23], extended FDR (EFDR) coding [24], alternating variable-length (AVR) coding [25], equal-run-length coding (ERLC) [26], shifted FDR (SFDR) coding [27] and modified FDR (MFDR) coding [28] consider both runs of 0 s and 1 s to form the codewords. Rosinger et al. [29] presented a minimum transition count (MTC) filling approach for simultaneous reduction of test data volume and power dissipation.

Statistical codes form the variable-length codewords for fixed-length of data blocks. Among the available statistical codes, Huffman code [32] provided a good compression efficiency because of its shortest average codeword length. The Huffman coding technique with fixed-length of blocks to reduce the test data volume is described in [19]. However, it requires complex decoder architecture to decode the large number of distinct blocks. Several Huffman based compression techniques such as variable-input Huffman coding [33], variable-to-variable Huffman coding [34], selective Huffman coding [35], optimal selective Huffman coding [36], complementary Huffman coding [37] and modified selective Huffman coding [38] are proposed to improve compression efficiency, area overhead and test application time. The technique exploits reordering of test patterns to minimize the shapes needed to encode the test data which is presented in [39]. Compression of the test data by geometric shapes is described in [39]. A technique based on merging consecutive compatible blocks of the test data is presented in [40]. Wolff et al. [41] proposed to use a popular software compression algorithm called LZ77 to reduce the test data volume. A method which iteratively encodes $2^{|n|}$ runs of compatible or inversely compatible patterns into a codeword is presented in [42].

Many code-based compression techniques have the objective of only reducing the test data volume without emphasis on test power reductions. For example, the compression techniques described in [24,31,33–36,43,44] focus mainly to reduce the test data volume. Several test independent compression techniques were used to reduce the test power and test data volume [24–26,45]. The zero-fill algorithm is used in [30,44] to maximize the 0-runs to reduce scan-in test power. The zero-fill algorithm fills the unspecified bits with 0 s. The X-bits are filled with 0 s or 1 s in order to improve skewing of the occurrence frequencies of the distinct blocks [35,43]. In ALT-FDR coding, all unspecified bits are filled to minimize the weighted transition metric (WTM) [23]. The ALT-FDR also reports a significant reduction in scan-out phase, but achieves less compression ratio since the unspecified bits are filled to reduce the test power.

Many authors gave their attention to multistage test data compression techniques to enhance the compression ratio. Mixing of run-length encoding and Huffman coding techniques to reduce the test data volume, test power and test application time is described in [45]. Lingappan et al. [46] presented multilevel compression techniques where the test data is compressed using the Huffman coding/embedded deterministic test technique and the compressed test set is further compressed using the LZ77 technique. The compression technique described in [43] uses multilevel Huffman coding to improve the compression ratio. Several hybrid test data compression/decompression schemes are presented in [47–49]. A technique to compress/decompress the test data using cyclical de-compressors and run-length coding is described in [47]. Most of the low-power compression techniques do not address the test power reduction in scan-out phase.

1.1. Main contribution

We present two multistage compression techniques called 9C-AFDER and 9C-RLHC to achieve high compression ratio, low scan-in and scan-out test power with small area overhead. To accomplish these objectives, we present two encoding methods, one is based on run-length coding and another is based on Huffman coding. Both the methods exploit the properties of encoded test obtained from 9C compression to enhance the compression ratio. First the test set with unspecified bits is compressed with the nine-coded compression technique and the resultant test set is further encoded with proposed alternating frequency-directed equal-run-length coding (AFDER) or run-length based Huffman coding (RLHC) schemes. The remaining unspecified bits in the 9C encoded test sets are filled with logic values to reduce test power. We also present the decompression architecture to decode the encoded test which will take small area overhead and overall test application time.

In Section 2, we describe the motivation behind this work and review the nine-coded compression technique. Then we describe our proposed encoding schemes with examples. Section 3 describes the decompression architectures, Section 4 presents the experimental results obtained for the proposed compression schemes, and Section 5 concludes.

2. Multistage encoding techniques

2.1. Motivation

The ATPG generated test cube to test the typical industrial circuits normally contain 95–98% of unspecified bits [15]. These bits can be freely filled with the logic values (0 or 1) in order to reduce the test data volume and/or the test power. However, filling of X-bits to reduce the test data volume may increase the test power or vice versa.

In the nine-coded compression technique [31], the unspecified bits in the test cubes were utilized in order to achieve better compression ratio. The compressed data still contain a large number of unspecified bits. These unspecified bits can be filled to reduce the test power without compromising the compression ratio. One interesting property observed with this method is that, it also provides better power reduction in test applications. To the best of our knowledge, this inherent property of test power reductions in the 9C technique was never reported in the literature so far. The encoded test set obtained from 9C coding consists of long runs of 0 s and 1 s, as well as a large number of repeated codewords can also be noted. For example, the circuit s13207 has 1423 runs of 0 s, 1947 runs of 1 s and 346 consecutive equal-runs. Hence, the encoded test volume can be further reduced by

Please cite this article as: S. Sivanantham, et al., Enhancement of test data compression with multistage encoding, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.12.001

exploiting the above said properties. These observations motivate us to use the nine-coded compression technique as the first stage in our multistage encoding techniques.

The second stage encoding can be decided based on the following criteria: (1) it should exploit the advantage of above said properties of encoded test set data obtained from the 9C technique and (2) it should use simple on-chip decompression hardware without increasing the test application time (TAT) or with small TAT overhead. Many existing methods are effective in reducing test data volume of precomputed test sets which could not exploit the properties of 9C encoded data. So, we have proposed two encoding techniques to fulfill the above said criteria. The first technique called alternating frequency-directed equal-run-length coding (AFDER) that exploits the long runs of 0 s and 1 s as well as consecutive alternating runs with equal-run-lengths. The second scheme called run-length based Huffman coding (RLHC) that exploits the higher probability in frequency of occurrence of distinct symbols in the compressed test set obtained from the nine-coded compression technique.

2.2. Overview of nine-coded compression technique

The nine-coded compression (9C) technique considers the input test data block size of fixed-length. Each input test vector is partitioned into groups of bits with particular size called as block, say K and the block size is user defined. Encoding is then performed on each of these groups with nine codewords in total. The block size is selected as even, so that each of these blocks can easily be divided into equal halves such that the halves may be either all 0 s, 1 s or a set of mismatched bits, that is, a mixed group of 0 s, 1 s and X-bits. Table 1 shows the code formation scheme in the 9C technique for block size $K=8$. Each half of the K-bits consists of either all 0 s or all 1 s for cases 1–4, but for cases 5–8, there is one half with either all 0 s or all 1 s and another half with all mismatched bits, indicated by *uuuu*. This means that the *uuuu* bits are combinations of 0 s and 1 s and may be the do not care bits. The case 9 has the entire K-bits to be mismatched. The third and fourth columns indicate the symbol and its description for the corresponding input blocks. The column five shows the codeword corresponding to each symbol or block. For the cases 5–9, when mismatched bits occur, they will also be passed along with the corresponding codeword. As shown in Table 1, the combinations are given in the input block and other similar blocks can be grouped accordingly. The column six shows some examples for each case. Since only nine codewords are used for encoding the input vectors, this compression technique is called the nine-coded compression technique.

The unspecified bits are considered during compression, where we can use the unfilled precomputed test sets directly as the input. It is noted that the encoded data still contain many unspecified bits. The unspecified bits appeared in the *uuuu* blocks are filled using the minimum-transition filling (MTM) technique to minimize the test power. Once the test data has been compressed using the 9C method, the fully specified bits in the compressed data have runs of 0 s, 1 s and equal-run-length of consecutive runs. The test data volume can be further reduced using the either AFDER or RLHC technique which exploits the properties of 9C encoded test set.

2.3. Alternating frequency-directed equal-run-length coding (AFDER)

The proposed coding strategy is based on the observation that a fully specified test set T_p obtained from the nine-coded technique is composed of alternating runs of zeros and runs of ones. The AFDER is variable-to-variable length code which is mapped as variable-length runs of 0 s or 1 s to variable-length codewords.

Table 2 shows the AFDER coding scheme. In most of the FDR coding schemes, the runs of 0 s were considered as strings of 0 s followed by a bit '1', while runs of 1 s as strings of 1 s followed by a bit '0'. In our scheme, we consider runs of 0 s as strings of 0 s and runs of 1 s as stings of 1 s only. For example, the pattern 111111 and 00000 is considered as runs of 1 s and 0 s with run-length of 6 and 5 respectively. So, the codeword for run-length of 0 is not included in our scheme. The major difference between other alternating FDR codes and proposed AFDER code is in assigning of a least-size codeword to the consecutive codes having equal-run-length. The codeword assigned to the repetitive code is smaller in length compared to the length of original codeword.

Table 2
Proposed AFDER coding scheme.

Group	Run	Prefix	Tail	Codeword	Code length
Repeated runs		–	–	01	2
A1	1	0	00	000	3
	2		01	001	
A2	3	10	00	1000	4
	4		01	1001	
	5		10	1010	
	6		11	1011	
A3	7	110	000	11 000	6
	8		001	110 001	
	9		010	110 010	
	10		011	110 011	
	11		100	110 100	
	12		101	110 101	
	13		110	110 110	
	14		111	110 111	
A4	15	1110	0000	11 100 000	8
	…		…	…	
	30		1111	11 101 111	
…	…	…	…	…	…

Table 1
9C code formation for block size $K=8$.

Case	Input block	Symbol (Si)	Description	Code word (C)	Examples
1	0000 0000	00	All 0′s	0	0000_0000, 0000_XXXX
2	1111 1111	11	All 1′s	10	1111_1111, 1111_XXXX
3	0000 1111	01	Left half 0′s, right half 1′s	11 000	0000_1111, X00X_11X1
4	1111 0000	10	Left half 1′s, right half 0′s	11 001	X11X_00X0, 111X_0000
5	1111 uuuu	1U	Left half 1′s, right half mismatched bits	11 010	1111_0010, 1111_X0X1
6	uuuu 1111	U1	Left half mismatched bits, right half 1′s	11 011	X0X1_X11X, 100X_X11X
7	0000 uuuu	0U	Left half 0′s, right half mismatched bits	11 100	00XX_0X01, 0000_X011
8	uuuu 0000	U0	Left half mismatched bits, right half 0′s	11 101	1XX0_0000,100X_X00X
9	uuuu uuuu	UU	All mismatched bits	1111	1011_0X01, X011_X011

Please cite this article as: S. Sivanantham, et al., Enhancement of test data compression with multistage encoding, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.12.001

In this scheme, we assign the codeword '01' for the repetitive runs. This method gives the best results in the cases where the frequencies of consecutive occurrence of similar runs are very high.

The pattern is divided into runs of 0 s and 1 s for encoding the completely specified test. Runs of 0 s or 1 s are called as run-length and are mapped to the corresponding group. The size of the group is determined by the group number and is equal to 2^K, where K is the group number. That is, group A1 contains 2 members, group A2 contains 4 members, group A3 contains 8 members and so on. The codeword has a combination of two parts, a group prefix and a tail, both have the same bit size (except for group A1 where the prefix and tail sizes are different) and the bit size will be the same as group number. The use of prefix is to determine which group the run belongs and that of the tail is to determine the members in the group during decompression. For example, let the run-length r be the nth element of group K, the prefix will be the binary of last element in the $K-1$ group and the tail will be binary of $n-1$. In Table 2, the same codeword is assigned for identical runs of 1 s and 0 s. To decode such a sequence, it is only necessary to know whether the first run-length corresponds to a run of 0 s or 1 s.

Fig. 1 shows an example to encode the test set with AFDER coding. The 9C encoded data with 83-bits are now decomposed into runs of 0 s and runs of 1 s. As expected, the FDR code is not effective in reducing the data volume since it encodes only runs of 0 s. The ALT-FDR and MFDR methods encoded the data with the sizes of 72-bits and 70-bits respectively. Although, significant data reductions were achieved in both schemes, these reduction rates will be reduced further if the number of 0 runs increases. On the other hand, the size of the AFDER encoded data is only 65-bits since it exploits the properties of first stage encoding. The efficiency of AFDER can be increased if the 9C encoded test set contains a large number of repeated equal-run-length codes.

2.4. Run-length based Huffman coding (RLHC)

Huffman coding is a statistical data-coding method that reduces the average codeword length which represents the unique pattern of a set. It is important to note that the Huffman coding scheme is basically a fixed-to-variable scheme. The fixed-length input patterns restrict the exploitation of test set features for compression. This problem can be solved by the proposed coding scheme which allows an efficient exploitation of test set to achieve better compression. The efficiency of Huffman code mainly depends on the frequency of occurrence of all possible distinct symbols in the given encoded test set. The short codewords are assigned to most frequently occurred symbols and larger codewords are assigned to the less frequently occurred symbols. The average codeword length can be minimized in this way.

Let I be the test set of the IP core with fully specified bits and the test sets are partitioned into n distinct blocks each with a length of l. The frequencies (or probabilities) of occurrence of n distinct blocks $b_1, b_2, \ldots, b_n$ are represented as $p_1, p_2, \ldots, p_n$ respectively. The entropy of the test set $H(I)$ specifies the minimum average number of bits for each codeword and it can be defined as

$$H(I) = -\sum_{k=1}^{n} p_k(\log_2 p_k) \quad (1)$$

It is assumed that $c_1, c_2, \ldots, c_n$ are the codeword length of blocks $b_1, b_2, \ldots, b_n$ respectively. The average codeword length $C(I)$ is

$$C(I) = \sum_{k=1}^{n} p_k c_k \quad (2)$$

The Huffman code provides closely similar average codeword length of theoretical entropy bound described by using Eq. (1). If we skew the occurrence of n distinct blocks in the test sets as much as possible, the entropy value $H(I)$ can be further minimized. The higher probability of occurrence of distinct symbols in the compressed test set obtained from the nine-coded compression technique favors the targeted skewing, which can minimize the $H(I)$, and average codeword length.

The formation of Huffman tree and Huffman codes is as follows. Let m_h be the size of the group. The group size represents the maximum acceptable number of 0 s contained in a runs of 0 s of length smaller than or equal to m_h, which are referred to as patterns. These patterns are used as an input to the Huffman coding scheme where for each pattern, the number of occurrences is determined. For group size m_h, there can be maximum of m_h+1 symbols which is represented as $L_0, L_1, L_3, \ldots, L_{m_h}$, etc. For example, symbol and pattern formation with the group size $m_h = 4$ is shown in Table 3. The Huffman tree is built based on the patterns and the frequency of occurrences. To construct the Huffman tree, the patterns are arranged in the descending order of their occurrences. Then the sum of all the occurrences is calculated and then assigned to the root of the Huffman tree from which the branches are constructed. The symbols which are arranged in descending order, are directly assigned to the branches which reduce the length of the codeword.

The tree construction with fixed-to-variable Huffman and the proposed run-length based Huffman code (RLHC) codes are illustrated in Fig. 2 (a) and (b) respectively. The three test patterns with a total of 48-bits are partitioned into different symbols and the number of occurrences for each symbol is calculated. The Huffman tree is constructed and all the branches of the tree are marked with alternate 0 s and 1 s, as shown in Fig. 2. The codeword for each pattern or the symbol is computed by back tracing the path along the tree. The branches do not grow on both sides of the Huffman tree as described in the conventional Huffman algorithm. We are growing the branches only in the right-hand sides of tree which results in shorter codeword for most frequently occurred symbols and average codeword. This scheme is very effective when the number of symbols is limited. In our RLHC scheme, the maximum number of symbols is limited to m_h+1 for the group size of m_h. The number of symbols required to construct the Huffman tree is reduced to 5 as compared to 9 in the case of

Table 3
Representation of symbols and patterns for m_h=4.

Symbol	Pattern
L_0	1
L_1	01
L_2	001
L_3	0001
L_4	0000

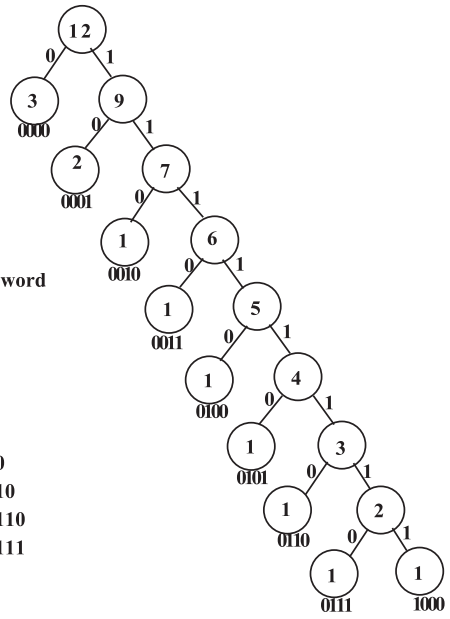
Test set : 1111100000111100 0000111111100000001111101110001111101111111111010111111100000000000**(83-bits)**
FDR : 00 00 00 00 00 1011 00 00 00 110001 00 00 00 00 00 00 110001 00 00 00 00 01 00 00 1001 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 01 01 00 00 00 00 00 00 110011 **(112-bits)**
ALT-FDR : 1011 1010 1001 110000 110000 110000 1010 00 1001 1000 1010 110100 00 01 00 110001 110011**(72-bits)**
MFDR : 1001 1000 0111 1010 1010 1010 1000 0100 0111 0110 1000 00110 0100 0101 0100 1011 00101**(70-bits)**
AFDER (ours): 1010 01 100 110000 10 10 1010 000 1000 10 1010 000 110011 000 01 01 **(65-bits)**

Fig. 1. Example of encoding procedure for AFDER code.

Please cite this article as: S. Sivanantham, et al., Enhancement of test data compression with multistage encoding, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.12.001

a

Test set	Block	Occ. Frequency	Codeword
	0000	3	0
	0001	2	10
	0010	1	110
1000 0010 0110 0001	0011	1	1110
0000 0100 0000 0001	0100	1	11110
0010 0101 0111 0000	0101	1	111110
	0110	1	1111110
	0111	1	11111110
	1000	1	11111111



b

Test set	Block	Occ. Frequency	Codeword
	0000	6/20	0
1000 0010 0110 0001	01	6/20	10
0000 0100 0000 0001	1	6/20	110
0010 0101 0111 0000	001	2/20	111
	0001	0	-

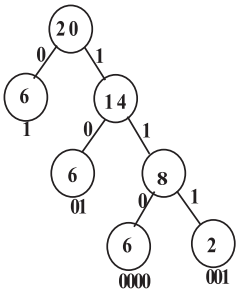


Fig. 2. Code formation and its tree construction. (a) Fixed-input Huffman code, (b) run-length based Huffman code.

fixed-input Huffman code. As a result, the average codeword length in full Huffman of $(3\times1)+(2\times2)+(1\times3)+(1\times4)+(1\times5)+(1\times6)+(1\times7)+(1\times8)+(1\times8)=48$ bits are reduced to $(4\times1)+(2\times2)+(1\times3)+(3\times3)=24$ bits in the RLHC method. The encoded data is 1111 0 10 110 10 1110 0 0 10 0 110.

3. Decompression architecture

Fig. 3 shows the decompression architecture used to decompress the encoded data. It consists of two finite-state machine (FSM) blocks, one synchronization block, a counter, a multiplexer (MUX), and the control signals. The decoder operates on two clocks – the external clock *ATE_CLK* and the internal clock *SOC_CLK*. The FSM1 can be either AFDER-FSM or RLHC-FSM and the FSM2 represents the 9C-FSM.

The FSM1 receives the compressed data, *DATA_IN* from the ATE at *ATE_CLK* frequency. Once the FSM1 detects the codeword, decoding begins at the system clock frequency (*SOC_CLK*) and the *DEC_EN* is set to 1. When FSM1 decodes the data, it does not receive any data from the ATE. The *ACK_H* is set to 1, as soon as the FSM1 decoded the codeword and it is ready to receive the next codeword. The FSM2 receives the decoded data from FSM1 at the frequency of the system clock. Once FSM2 detects the codeword, it will decode the codeword also. For the codewords C_1, C_2, C_3 or C_4, the K output bits contain either 0 s or 1 s. For codewords C_5, C_6, C_7, C_8 or C_9, either $K/2$ or K bits in the output are expected to be received directly from *Data_in_u*. A 3–1 MUX is used to select

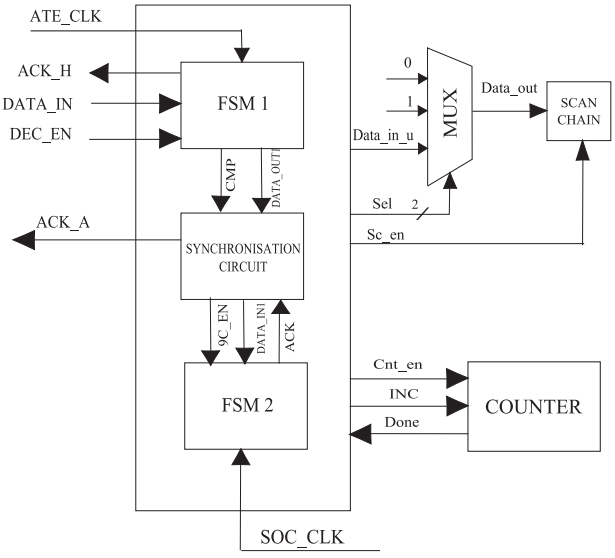


Fig. 3. Decompression architecture.

0,1 and *Data_in_u*. The two select lines, *Sel*1 and *Sel*0, come from the FSM to the MUX. The counter is used to control the transfer of $K/2$ bits from the output of MUX to the scan chain. The count begins when the FSM sends the *Cnt_en* signal and it gets incremented when it receives the *INC* signal. At the same time, it

Please cite this article as: S. Sivanantham, et al., Enhancement of test data compression with multistage encoding, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.12.001

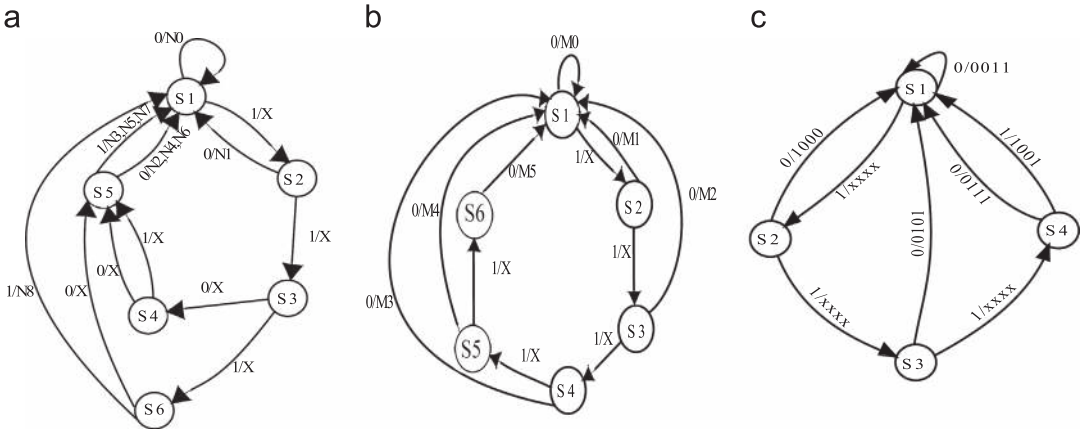


Fig. 4. Finite state machines for the decompression architecture. (a) 9C-FSM, (b) AFDER-FSM, (c) RLHC-FSM.

activates the *Sc_en* signal to enable the scan-chain. When the count reaches maximum, the *K*/2 bits are sent to the scan-chain through *data_out*. The counter sends the *Done* signal to the FSM so as to send the next value to *Sel* and *Cnt_en*. After the *Done* signal is sent for the second time by the counter, the FSM deactivates the *Sc_en*. A synchronization block is used to synchronize both the FSM′s.

Fig. 4(b) shows the state diagram for AFDER decoder FSM. It has six states from *S*1 to *S*6 and these states correspond to each group. The outputs are denoted as *M0–M5*. When the *DATA_IN* is given to FSM1 from ATE at the tester frequency, the *DEC_EN* signal is enabled when the decoding begins at the frequency of system clock. When the first codeword is decoded, the *CMP* signal goes high and the output *DATA_OUT1*, is given to the synchronization block. The *CMP* signal is the enable signal given to the counter in the synchronization circuit, to compare the data from the FSM1 and memory. It remains high until the entire input data are decoded. The synchronization block synchronizes the values and once 9*C_EN* goes high, the output *DATA_IN*1 from the block is taken as the input to the FSM2 to decode the 9C encoded data.

The state diagram used for the RLHC decoder with group size $m_h = 4$ is shown in Fig. 4(c). The number of states is equivalent to the total number of branches in the Huffman tree minus one. There are maximum of four states which represent the group size m_h. The FSM starts from state *S*1, and changes its state based on *DATA_in_ bit* from ATE. After detecting a codeword, decoding begins at the frequency of system clock and FSM back to its default state i.e. *S*1 state. For example, when the input data stream to be 01, the decoder changes its state from *S*1 to *S2* and again from *S2* to *S*1 and sets the decoder output to 1000 which indicate the decompressed output 0000. The length of the codeword is equal to the number of ATE clock cycles needed to detect a codeword. This FSM is activated as soon as the *DEC_EN* goes high and it receives input *DATA_IN* from the ATE. Once the decoding is done, the *CMP* signal goes high and the output *DATA_OUT*1 is given to the synchronization block.

Fig. 4(a) shows the state diagram for the 9C decoder FSM which is used as FSM2 and it consists of six states from *S*1 to *S*6. Since there are nine codewords used during compression, nine possible outputs also are there after decompression, which can be denoted by *N*0–*N*8. Each time a codeword is decoded by FSM2, the *Cnt_en* signal and the *scan_en* signal go high and the counter counts up to *K*/2 before sending a *Done* signal. When a *Done* signal received in the second time, the *Sc_en* signal goes low. For both the FSMs, each time the decoded output is obtained for one received codeword, the state controller starts again from starting state, *S*1. This happens when the *ACK_H* or *ACK_M* goes high for FSM1 and when the *ACK* signal goes high for FSM2.

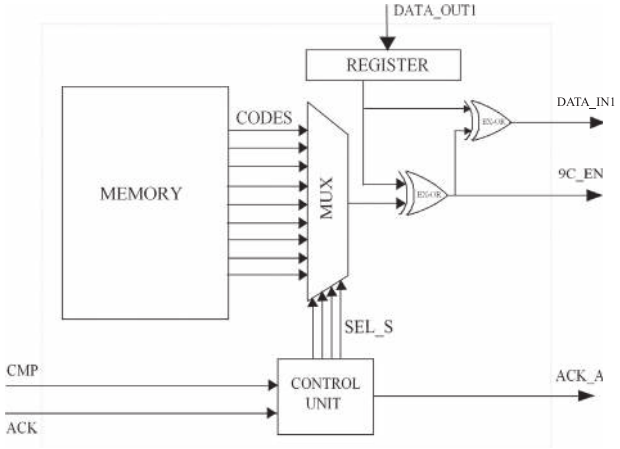


Fig. 5. FSM synchronization circuit.

Fig. 5 is used to synchronize the operations between FSM1 and FSM2. It consists of memory, a register, a MUX, a control unit and XOR gates. The input data to the register is obtained from the FSM1. The control unit does the basic controlling of all the elements inside the unit. When the *CMP* signal goes high, the control unit sends select line value *SEL_S* to the MUX and the output of the MUX is then XORed with the output from the register. If the output of this XOR gate is 0, it means that a 9C codeword is available as the output from the synchronization block which is given as the input *DATA_IN*1 to the FSM2 to be decoded.

4. Experimental results

The experiments were conducted on six larger ISCAS′89 benchmark circuits. The proposed work was implemented in C language and compiled using gcc. We have used the dynamically compacted test sets generated by the mintest ATPG program [50] which are same as used in [21–26,31,33,35,36,38,43–45] for comparing our results with the published results.

4.1. Compression results with 9C-AFDER technique

In Table 4, we present the experimental results on compression obtained from the 9C-AFDER technique for ISCAS′89 benchmark circuits. In order to demonstrate the effectiveness of the proposed 9C-AFDER technique, we compare it with the number of code based compression techniques including Golomb [44], FDR [30],

Please cite this article as: S. Sivanantham, et al., Enhancement of test data compression with multistage encoding, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.12.001

Table 4
Comparison of compression obtained with 9C-AFDER with other run-length codes.

Circuit	SPL	Orig. vol.	Compressed test data volume							
			9C-AFDER (ours)	Golomb [44]	FDR [22]	EFDR [24]	ALT-FDR [23]	9C [31]	MFDR [28]	ERLC [26]
s5378	214	23 754	**6376**	14 085	12 346	11 419	11 694	11 713	11 528	12 389
s9234	247	39 273	**13 990**	22 250	22 152	21 250	21 612	19 279	16 597	22 210
s13207	700	165 200	**23 194**	41 658	30 880	29 992	32 648	33 353	27 390	32 044
s15850	611	76 986	**19 490**	40 717	26 000	24 643	26 306	25 882	25 459	25 844
s38417	1664	164 736	**50 423**	92 054	93 466	64 962	64 976	64 856	69 271	67 990
s38584	1464	199 104	**59 076**	104 111	77 812	73 853	77 372	68 631	80 996	76 473
Avg.	–	111 509	**28758**	52 479	43 776	37 687	39 101	37 288	38 540	39 492

Table 5
Compressed-data reduction percentage of 9C-AFDER technique over others.

Circuit	Mintest [50]	Golomb [44]	FDR [22]	EFDR [24]	ALT-FDR [23]	9C [31]	MFDR [28]	ERLC [26]
s5378	73.2	54.7	48.4	44.2	45.5	45.6	44.7	48.5
s9234	64.4	37.1	36.8	34.2	35.3	27.4	15.7	37
s13207	86	44.3	24.9	22.7	29	30.5	15.3	27.6
s15850	74.7	52.1	25	20.9	25.9	24.7	23.4	24.6
s38417	69.4	45.2	46.1	22.4	22.4	22.3	27.2	25.8
s38584	70.3	43.3	24.1	20	23.6	13.9	27.1	22.7
Avg.	74.2	45.2	34.3	23.7	26.5	22.9	25.4	27.2

EFDR [24], ALT-FDR [23], 9C [31], MFDR [28] and ERLC [26]. The columns 2 and 3 describe the single pattern length (SPL) in the test cube and uncompressed test data volume (Orig.vol.) of the CUTs respectively. The compressed test data volume obtained from the proposed 9C-AFDER technique and other compression techniques is presented from columns 4–11. From Table 4, it is evident that the proposed 9C-AFDER compression technique obtained better compression for all the circuits. Table 5 shows reduction percentage of compressed-data from the proposed 9C-AFDER method against various compression techniques available in the literature. The reduction percentage of the proposed technique over others is computed using

$$Reduction(\%) = \frac{Size(others) - Size(ours)}{Size(others)} \times 100 \quad (3)$$

where *Size*(*ours*) and *Size*(*others*) are the compressed data volume of the proposed method and the test data volume of other approaches respectively.

The 9C-AFDER compression scheme achieves a maximum of 86% compression ratio for s13207 circuit. On average, the 9C-AFDER technique achieves the compressed-data reduction percentage of 74.2%, 45.2%, 34.3%, 23.7%, 26.5%, 22.9%, 25.4% and 27.2% against [50,44,30,24,23,31,28,26] respectively. It is to be noted that, the comparisons are provided only for the compression techniques that do not require structural information of CUT.

4.2. Compression results with 9C-RLHC technique

Table 6 shows the compression results of 9C-RLHC method for different block sizes. The last column shows the best case compression ratio obtained for each circuit. The 9C-RLHC method achieves a maximum compression ratio of 85.3% for s13207 circuit. In order to show the effectiveness of the proposed 9C-RLHC compression technique on reduction of test data volume over other methods, we have compared our results with other Huffman based techniques. Table 7 shows the reduction percentage of test data volume for the proposed 9C-RLHC method against other

Table 6
Compression results for different block sizes in 9C-RLHC technique.

Circuit	Block size (m_h)						% CR
	4	5	6	7	8	9	
s5378	6836	**6664**	6736	6876	6753	6963	**71.9**
s9234	13 218	13 185	**13 176**	13 210	13 213	13 209	**66.5**
s13207	24 339	22 937	23 405	31 527	**22 553**	28 846	**86.3**
s15850	18 622	18 086	17 883	18 902	**17 629**	18 920	**77.1**
s38417	44 608	**43 097**	43 775	44 584	44 337	44 633	**73.8**
s38584	48 843	**47 399**	47 544	49 451	47 679	49 175	**76.2**

schemes like Huffman [32], selective Huffman [35], VIHC [33], opt. Huff [36], V2V Huffman [34] methods. On average, the proposed 9C-RLHC method achieves the compression ratio of 77.5%. Also, the 9C-RLHC provides the test data volume reduction of 58.4%, 35.1%, 36.2%, 29.6%, 21.1% against [32,35,33,36,34] methods respectively. We also demonstrate the effectiveness of the our multistage compression methods against other multistage/multilevel compression methods presented in the literature like RL-Huffman coding [45] and multilevel Huffman coding [43] methods. The 9C-RLHC method shows the reduction of 34.4% and 14.6% over [45,43] respectively. In the comparison, we have not included the multilevel compression technique presented in [46] since it uses different test sets. As shown, when both 9C-AFDER and 9C-RLHC methods yield better compression, the 9C-RLHC compression technique shows much higher reduction of test data volume.

4.3. Experimental results on power reductions

The average and peak-powers during scan-in and scan-out modes are computed based on the weighted transition metric (WTM) presented in [51]. Table 8 shows the scan-in average power of the proposed multistage compression technique with other published works. Our method provides significant power savings for all the circuits except s15850 circuit. However, the proposed

Please cite this article as: S. Sivanantham, et al., Enhancement of test data compression with multistage encoding, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.12.001

Table 7
Compressed-data reduction percentage of the proposed 9C-RLHC technique with others.

Circuit	Mintest [50]	Huff. [32]	Sel.Huff. [35]	VIHC [33]	Opt.Huff. [36]	V2V Huff. [34]	RL-Huff. [45]	Multi-Huff. [43]
s5378	71.9	54.6	37.5	41.8	37.8	31.8	39.3	28.8
s9234	66.5	42.9	26.7	36.4	24.9	16.6	36.0	15.1
s13207	86.4	60	40.6	17.2	17.8	6.2	21.9	−22.7
s15850	77.1	54.2	32.6	28.6	28.7	20.3	49.8	6.9
s38417	73.8	63.6	36.2	43.9	33.1	26.5	27.0	26.7
s38584	76.2	57.2	33.7	36.9	31.2	21.4	36.7	14.1
Avg.	77.5	58.4	35.1	36.2	29.6	21.1	34.4	14.6

Table 8
Comparison of scan-in average power with other schemes.

Circuit	Scan-in average power							% of average power reductions over					
	mintest [50]	FDR [22]	EFDR [24]	ERLC [26]	RL-Huff. [45]	ALT-FDR [23]	ours	mintest [50]	FDR [22]	EFDR [24]	ERLC [26]	RL-Huff. [45]	ALT-FDR [23]
s9234	14 630	5692	3469	3500	3957	3466	2945	79.9	48.3	15.1	15.9	25.6	15.0
s13207	122 031	12416	7703	8115	7734	7703	7345	94	40.8	4.6	9.5	5.0	4.6
s15850	90 899	20 742	13 394	13 450	13 513	13 381	15 286	83.2	26.3	−14.1	−13.7	−13.1	−14.2
s38417	601 840	172 665	117 834	120 775	116 301	112 198	118 317	80.3	31.5	−0.4	2	−1.7	−5.5
s38584	535 875	136 634	89 138	89 356	85 655	88 298	87 614	83.7	35.9	1.7	1.9	−2.3	0.8
Avg.	–	–	–	–	–		–	84.2	36.6	1.38	3.1	2.7	0.14

Table 9
Comparison of scan-in peak-power with other schemes.

Circuit	Scan-in peak-power							% of peak-power reductions over					
	mintest [50]	FDR [22]	EFDR [24]	ERLC [26]	RL-Huff. [45]	ALT-FDR [23]	Ours	mintest [50]	FDR [22]	EFDR [24]	ERLC [26]	RL-Huff. [45]	ALT-FDR [23]
s9234	17 494	12 994	12 062	12 069	14 092	12 060	11 914	31.9	8.3	1.2	1.3	15.5	1.2
s13207	135 607	101 127	97 613	97 614	94 879	97 606	83 537	38.4	17.4	14.4	14.4	12	14.4
s15850	100 228	81 832	63 494	63 511	70 875	63 478	70 679	29.5	13.6	−11.3	−11.3	0.3	−11.3
s38417	683 765	505 295	404 617	404 693	411 718	404 617	380 136	44.4	24.8	6.1	6.1	7.7	6.1
s38584	572 618	531 321	479 573	479 573	481 158	479 530	471 580	17.6	11.7	1.7	1.7	2	1.7
Avg.	–	–	–	–	–		–	32.4	15.2	2.4	2.4	7.5	2.4

work saves 84.2%, 36.5%, 1.38%, 3.1%, 2.8% and 0.14% of average power against [50,22,24,26,45,23] respectively. Similarly, the peak-power of the ours and other schemes are given in Table 9. The proposed scheme achieves the percentage of reduction of 32.4%, 15.2, 2.4%, 2.4%, 7.5% and 2.4% of scan-in peak-power over [50,22,24,26,45,23] respectively. This desirable power reduction achieved in our work is mainly due to the inherent feature of 9C compression technique and minimum transition filling scheme adopted to selected cases as explained earlier in Section 2.

Next we evaluate the total power consumptions (i.e., sum of scan-in and scan-out power) in the proposed method. In [23], the unspecified bits are filled to achieve minimum WTM, both in scan-in and scan-out mode. However we are not comparing our work with [23] since it provides a very poor compression as compared to ours. Table 10 presents the comparison for total power consumption of our work with the zero-filled mintest test sets and found that the total average and peak-powers are reduced. It can be noted that, our scheme mainly focuses on the reduction of test power in scan-in phase. Power reduction in scan-out phase is an additional benefit obtained from our schemes. On an average, the total average and peak-power are 67% and 12.6% lesser than the mintest test sets respectively. This is reasonably acceptable reductions if we view each circuit as IP cores of SoC.

Table 10
Comparison of total power (scan-in and scan-out mode) reductions against [50].

Circuit	Total average power			Total peak-power		
	[50]	Ours	% red.	[50]	Ours	% red.
s9234	29 185	12 576	56.9	34 271	31 473	8.2
s13207	238 726	76 018	68.2	267 736	244 941	8.5
s15850	179 284	65 255	63.6	199 875	186 253	6.8
s38417	1 155 331	294 903	74.5	1 303 694	885 046	32.1
s38584	1 057 757	428 397	59.5	1 149 983	1 236 913	−7.6
Avg.	–	–	67.0	–	–	12.6

4.4. Analysis of test application time and area overhead

We now analyze the over all test application time (TAT) when a single scan chain is used by the decoding process. One of the main goals of any test data compression method is to reduce the overall

Please cite this article as: S. Sivanantham, et al., Enhancement of test data compression with multistage encoding, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.12.001

Table 11
Comparison of decoder area head and overall compression ratio.

Measuring components	FDR	EFDR	BM	Geometric	Ours	
	[22]	[24]	[40]	[39]	9C-AFDER	9C-RLHC
Decoder area (# of gates)	1380	1487	1931	3970	5521	1189
Average compression ratio (%)	57.22	63.3	64.47	66.6	74.2	77.5

test application time in addition to reduce the test data volume. The test application time depends on the time required to transfer the encoded test set from the tester to the chip and the time required to decode the encoded data to the scan chain. Let the ATE and the on-chip system are running at different frequencies. It is assumed that f_{ATE} and f_{SYS} are the operating frequencies of ATE and on-chip system respectively. Also, $f_{SYS} > f_{ATE}$, since slow speed testers are used to test the high speed systems. Where $f_{ATE} = f_{SYS}/\phi$, $\phi > 1$. The parameter ϕ is a power of two since it would be easier to synchronize the tester clock and the system clock.

In our scheme, the decoder consists of two-stages. The decoder receives the compressed data from the ATE at a frequency of f_{ATE}. The AFDER/RLHC codes as well as 9C codes are decoded at a frequency of f_{SYS}. The synchronizer is used to synchronize the AFDER/RLHC decoded output with 9C codewords. The proposed decoded scheme therefore decouples the internal scan chain from the ATE via the use of a decoder interface. This decoupling implies that the scan clock frequency is no longer constrained by the ATE clock frequency limitation. Thus, a low cost ATE running at a slower frequency f_{ATE} can be used to test a circuit with a higher scan test frequency f_{SYS}.

For the proposed scheme, let $TAT_{9C-AFDER}$ ($TAT_{9C-RLHC}$) be the test application time for the encoded test set. Let $T_{transfer}$ be the time required to transfer the encoded data from ATE to the chip and T_{decode} be the time required to decode the encoded test set. In the proposed decoder scheme, $T_{decoder}$ has two parts, T_{AFDER} (T_{RLHC}) is the time required to decode the AFDER (RLHC) encoded test set and T_{9C} be the test time required to decode the 9C encoded test set.

An upper bound on T_{ATE} can be obtained by making a pessimistic assumption that the decoding begins after the complete encoded test set is transferred from the ATE to the chip. This implies that

$$TAT = T_{transfer} + \{T_{AFDER} \text{ or } T_{RLHC}\} + T_{9C} \tag{4}$$

Let $|T_E|$ be the size of encoded test set. Since data is transferred from the ATE to the chip at the tester frequency, the time required to transfer the encoded test set is given by $T_{transfer} = |T_E|/f_{ATE}$. The time required to decode the encoded test set on-chip in the first stage is equal to T_{E1}/f_{SYS}, where T_{E1} is the size of 9C encoded test set. The test application time T_{9C} depends on the frequency of occurrence of each symbol (N_i). The 9C decoder codeword with size $|I_i|$ is entered into FSM at the frequency of f_{SYS} and K system clocks are needed for applying K bits into scan chain. The application time for the 9C decoder is given by

$$T_{9C} = \sum_{i=1}^{9} \left\{ \frac{K + |I_i|}{f_{SYS}} \right\} N_i \tag{5}$$

Let $N = \sum_{i=1}^{9} N_i$ and $|T_{E1}| = \sum_{i=1}^{9} |I_i| N_i$. The time required to decode the data obtained from AFDER/RLHC decoder is $T_{9C} = (K\ N + |T_{E1}|)/f_{SYS}$.

In our method, both AFDER/RLHC and 9C decoders are simultaneously decoding the codewords. Therefore, the overall application time for 9C-AFDER/9C-RLHC can be given by

$$TAT = \frac{|T_E|}{f_{ATE}} + \text{Max}\left\{ \frac{|T_{E1}|}{f_{SYS}}, \frac{KN + |T_{E1}|}{f_{SYS}} \right\} \tag{6}$$

The time required to decode AFDER and RLHC codes is normally higher than the time required to decode the 9C codes since it output the fixed-length data. So the upper bound on TAT for 9C-AFDER and 9C-RLHC decoder can be

$$TAT = \frac{|T_E|}{f_{ATE}} + \frac{T_{E1}}{f_{SYS}} = \frac{\phi |T_E| + |T_{E1}|}{f_{SYS}} \tag{7}$$

Similarly, we can derive the lower bound on test application time. The lower bound ensures that the ATE never has to wait for the decoder to finish decoding the previous codeword. In other words, the ATE continuously supply the data to system without entering into its ideal state, to reduce the TAT. The lower bound on f_{ATE}/f_{SYS} for 9C-AFDER is limited by the largest decoded output of AFDER, L_{max} and block size used for the 9C method. Similarly the lower bound for 9C-RLHC is limited by the group size, m_h of RLHC scheme. The lower bound on f_{ATE}/f_{SYS} to obtain maximum TAT reduction for 9C-AFDER and 9C-RLHC methods is constrained by the inequalities $f_{ATE}/f_{SYS} \geq L_{max} \geq K$ and $f_{ATE}/f_{SYS} \geq m_h \geq K$ respectively.

The overall TAT for the single-stage 9C coding scheme [31] is given by

$$T_{9C} = \frac{KN + \phi |T_{E2}|}{f_{SYS}} \tag{8}$$

where T_{E2} is the size of encoded test set if single-stage compression is employed. It can be concluded from Eqs. (7) and (8) that the TAT of our multistage encoding is comparable with TAT of single-stage 9C coding, since $T_{E2} > T_{E2}$. This reduction of overall TAT is achieved at the expense of area overhead due to synchronization circuit in the decoder.

The decompression architecture presented in the last section was designed in Verilog HDL and synthesized using a commercial ASIC synthesis tool with 180 nm CMOS technology standard cells library. Table 11 presents the number of cells required to decompress the original data in both the compression schemes. The area overhead of other compression methods as well as their compression ratio is also included. It can be observed from Table 11 that, in-spite of the better compression ratio and significant power reductions, both the compression techniques demand only small area overhead. Since this decompression architecture is independent of the SoC, the area overhead with respect to total cells available in the SoC is acceptable.

5. Conclusions

Multistage encoding schemes to reduce the test data volume and test power are presented in this paper. The 9C-AFDER method exploits the runs of 0 s and 1 s in the first stage compression and the 9C-RLHC method exploits the frequency of occurrence of identical blocks. While both multistage techniques enhance the test data compressions in scan-based test applications, the 9C-RLHC

Please cite this article as: S. Sivanantham, et al., Enhancement of test data compression with multistage encoding, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.12.001

provides better compression ratio and lesser area overhead. The test application time is also reduced as single-stage compression scheme. Experimental results ensure that substantial reduction in test data volume, testing time and test power can be obtained. These techniques can be used to test SoC with IP cores since the compression and decompression are design independent. We can extend these schemes for multi-scan-based embedded core by modifying the decoder architecture to enhance the test application time.

Acknowledgment

The authors would like to thank Prof. Krishnendu Chakrabarty, Duke University, USA for providing Mintest test sets for ISCAS′89 benchmark circuits to carry this experimental work.

References

[1] P. Girard, Survey of low-power testing of VLSI circuits, IEEE Design Test Comput. 19 (3) (2002) 82–92.
[2] C. Krishna, N. Touba, Reducing test data volume using LFSR reseeding with seed compression, in: Proceedings of the International Test Conference, 2002, pp. 321–330.
[3] W.-C. Lien, K.-J. Lee, T.-Y. Hsieh, A test-per-clock LFSR reseeding algorithm for concurrent reduction on test sequence length and test data volume, in: Proceedings of the Asian Test Symposium, 2012, pp. 278–283.
[4] H.-S. Kim, S. Kang, M.S. Hsiao, A new scan architecture for both low power testing and test volume compression under SOC test environment, J. Electron. Test.: Theory Appl. (JETTA) 24 (4) (2008) 365–378.
[5] W. Wang, J. Kuang, Z. You, P. Liu, Reducing test-data volume and test-power simultaneously in LFSR reseeding-based compression environment, J. Semicond. 32 (7) (2011).
[6] Z. Wang, H. Fang, K. Chakrabarty, M. Bienek, Deviation-based LFSR reseeding for test-data compression, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 28 (2) (2009) 259–271.
[7] S. Reda, A. Orailoglu, Reducing test application time through test data mutation encoding, in: Proceedings on Design, Automation and Test in Europe Conference and Exhibition, 2002, 2002, pp. 387–393.
[8] I. Bayraktaroglu, A. Orailoglu, Test volume and application time reduction through scan chain concealment, in: Proceedings on Design Automation Conference, 2001, pp. 151–155.
[9] S. Sivanantham, M. Padmavathy, S. Divyanga, P.V. Anitha Lincy, System-on-a-chip test data compression and decompression with reconfigurable serial multiplier, Int. J. Eng. Technol. 5 (2) (2013) 973–978.
[10] E.H. Volkerink, A. Khoche, S. Mitra, Packet-based input test data compression techniques, in: IEEE International Test Conference (TC), 2002, pp. 154–163.
[11] D. Xiang, Y. Zhao, K. Chakrabarty, H. Fujiwara, A reconfigurable scan architecture with weighted scan-enable signals for deterministic BIST, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 27 (6) (2008) 999–1012.
[12] S. Ward, C. Schattauer, N. Touba, Using statistical transformations to improve compression for linear decompressors, in: 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2005, 2005, pp. 42–50.
[13] W. Wang, J. Kuang, Z. You, Achieving low capture and shift power in linear decompressor-based test compression environment, Microelectron. J. 43 (2) (2012) 134–140.
[14] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, Embedded deterministic test, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 23 (5) (2004) 776–792.
[15] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, D. Wheater, A. SmartBIST variant with guaranteed encoding, in: Proceedings of the Asian Test Symposium, 2001, pp. 325–330.
[16] L. Li, K. Chakrabarty, Test set embedding for deterministic BIST using a reconfigurable interconnection network, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 23 (9) (2004) 1289–1305.
[17] A.B. Kinsman, N. Nicolici, Time-multiplexed compressed test of SOC designs, IEEE Trans. Very Large Scale Integr (VLSI) Syst. 18 (8) (2010) 1159–1172.
[18] J.M. Solana, Reducing test application time, Integr. VLSI J. 42 (3) (2009) 385–399.
[19] A. Jas, G.-D. Jayabrata, N.A. Touba, Scan vector compression/decompression using statistical coding, in: Proceedings of IEEE VLSI Test Symposium, IEEE, Los Alamitos, CA, United States, 1999, pp. 114–120.
[20] H. Ichihara, Y. Iwamoto, Y. Yoshikawa, T. Inoue, Test compression based on lossy image encoding, in: Proceedings of the Asian Test Symposium, 2011, pp. 273–278.
[21] A. Chandra, K. Chakrabarty, Test data compression and decompression based on internal scan chains and Golomb coding, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 21 (6) (2002) 715–722.
[22] A. Chandra, K. Chakrabarty, Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes, IEEE Trans. Comput. 52 (8) (2003) 1076–1088.
[23] A. Chandra, K. Chakrabarty, A unified approach to reduce SOC test data volume, scan power and testing time, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 22 (3) (2003) 352–362.
[24] A.H. El-Maleh, Test data compression for system-on-a-chip using extended frequency-directed run-length code, IET Comput. Digital Tech. 2 (3) (2008) 155–163.
[25] B. Ye, Q. Zhao, D. Zhou, X. Wang, M. Luo, Test data compression using alternating variable run-length code, Integr. VLSI J. 44 (2) (2011) 103–110.
[26] W. Zhan, A. El-Maleh, A new scheme of test data compression based on equal-run-length coding (ERLC), Integr. VLSI J. 45 (1) (2012) 91–98.
[27] S. Sivanantham, J. Manuel, K. Sarathkumar, P.S. Mallick, J.R.P. Perinbam, Reduction of test power and test data volume by power aware compression scheme, in: International Conference on Advances in Computing and Communications (ICACC), August, 2011, pp. 158–161.
[28] J. Feng, G. Li, A test data compression method for system-on-a-chip, in: Proceedings of 4th IEEE International Symposium on Electronic Design, Test and Applications, DELTA 2008, 2008, pp. 270–273.
[29] P. Rosinger, P.T. Gonciari, B.M. Al-Hashimi, N. Nicolici, Simultaneous reduction in volume of test data and power dissipation for systems-on-a-chip, Electron. Lett. 37 (24) (2001) 1434–1436.
[30] A. Chandra, K. Chakrabarty, Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression, in: Proceedings of the IEEE VLSI Test Symposium, 2001, pp. 42–47.
[31] M. Tehranipoor, M. Nourani, K. Chakrabarty, Nine-coded compression technique for testing embedded cores in SoCs, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 13 (6) (2005) 719–730.
[32] D. Huffman, A method for the construction of minimum-redundancy codes, Proc. IRE 40 (9) (1952) 1098–1101.
[33] P.T. Gonciari, B.M. Al-Hashimi, N. Nicolici, Variable-length input Huffman coding for system-on-a-chip test, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 22 (6) (2003) 783–796.
[34] X. Kavousianos, E. Kalligeros, D. Nikolos, Test data compression based on variable-to-variable Huffman encoding with codeword reusability, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 27 (7) (2008) 1333–1338.
[35] A. Jas, J. Ghosh-Dastidar, M.-. Ng, N. Touba, An efficient test vector compression scheme using selective Huffman coding, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 22 (6) (2003) 797–806.
[36] X. Kavousianos, E. Kalligeros, D. Nikolos, Optimal selective Huffman coding for test-data compression, IEEE Trans. Comput. 56 (8) (2007) 1146–1152.
[37] S.-K. Lu, H.-M. Chuang, G.-Y. Lai, B.-T. Lai, Y.-C. Huang, Efficient test pattern compression techniques based on complementary Huffman coding, in: 2009 IEEE Circuits and Systems International Conference on Testing and Diagnosis, ICTD′09, 2009.
[38] U.S. Mehta, K.S. Dasgupta, N.M. Devashrayee, Modified selective Huffman coding for optimization of test data compression, test application time and area overhead, J. Electron. Test.: Theory Appl. (JETTA) 26 (6) (2010) 679–688.
[39] A. El-Maleh, S. Al Zahir, E. Khan, Test data compression based on geometric shapes, Comput. Electr. Eng. 37 (3) (2011) 376–391.
[40] A.H. El-Maleh, Efficient test compression technique based on block merging, IET Comput. Digital Tech. 2 (5) (2008) 327–335.
[41] F.G. Wolff, C. Papachristou, Multiscan-based test compression and hardware decompression using LZ77, in: IEEE International Test Conference (TC), 2002, pp. 331–339.
[42] L.-J. Lee, W.-D. Tseng, R.-B. Lin, C.-H. Chang, 2n pattern run-length for test data compression, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 31 (4) (2012) 644–648.
[43] X. Kavousianos, E. Kalligeros, D. Nikolos, Multilevel Huffman coding: an efficient test-data compression method for IP cores, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 26 (6) (2007) 1070–1083.
[44] A. Chandra, K. Chakrabarty, System-on-a-chip test-data compression and decompression architectures based on Golomb codes, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 20 (3) (2001) 355–368.
[45] M. Nourani, M.H. Tehranipour, RL-Huffman encoding for test compression and power reduction in scan applications, ACM Trans. Design Autom. Electron. Syst. 10 (1) (2005) 91–115.
[46] L. Lingappan, S. Ravi, A. Raghunathan, N. Jha, S. Chakradhar, Test-volume reduction in systems-on-a-chip using heterogeneous and multilevel compression techniques, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 25 (10) (2006) 2193–2205.
[47] A. Jas, N.A. Touba, Test vector decompression via cyclical scan chains and its application to testing core-based designs, in: IEEE International Test Conference (TC), 1998, pp. 458–464.
[48] S. Chun, Y.J. Kim, J. Im, S. Kang, MICRO: a new hybrid test data compression/decompression scheme, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 14 (6) (2006) 649–654.
[49] A. El-Maleh, A hybrid test compression technique for efficient testing of systems-on-a-chip, in: Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems, vol. 2, 2003, pp. 599–602.
[50] S.I. Hamzaoglu, J.H. Patel S, Test set compaction algorithms for combinational circuits, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 19 (8) (2000) 957–963.
[51] R. Sankaralingam, R.R. Oruganti, N.A. Touba, Static compaction techniques to control scan vector power dissipation, in: Proceedings of the IEEE VLSI Test Symposium, 2000, pp. 35–40.

Please cite this article as: S. Sivanantham, et al., Enhancement of test data compression with multistage encoding, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.12.001

Sivanantham S. received B.E. degree in Electronics and Communication Engineering from the University of Madras, India and M.Tech. degree in VLSI Design from SASTRA University, India in 1997 and 2002 respectively. He is currently working as an Assistant Professor (Selection Grade) in the School of Electronics Engineering, VIT University, Vellore, India and pursuing Ph.D. degree in the School of Electrical Engineering from the same University. He served as a Leader for VLSI and Embedded System Division at VIT University during 2007–2009. Previously he was associated with J.J. College of Engineering and Technology, Trichirappalli, India and Bannari Amman Institute of Technology, Satyamangalam, India as a faculty in the Department of Electronics and Communication Engineering. His area of research interest includes the design for testability, reconfigurable architectures and low power VLSI design. He is the member of IEEE, IEICE, VLSI Society of India (VSI) and Indian Society for Technical Education (ISTE).

M. Padmavathy received B.E. degree in Electronics and Communication Engineering from the University of Madras, India and M.Tech. degree in VLSI Design from VIT University, Vellore, India in 2003 and 2013 respectively. She has worked in ZTE Telecom as a Residential Engineer in 2007. Her area of interest includes design for testability and digital integrated circuits design.

Ganga Gopakumar received B.E. degree in Electronics and Communication Engineering from Amrita Vishwa Vidyapeetham, Kerala, India and M.Tech. degree in VLSI Design from VIT University, Vellore, India in 2011 and 2013 respectively. Currently she is working as a Design Engineer at Sankalp and KPIT Semiconductors Pvt. Ltd, Bangalore, India. Her area of interest includes design for testability and low-power IC design.

P.S. Mallick received M.S. degree from the University of Chittagong, Bangladesh and Ph.D. from Jadavpur University, India. He worked 4.5 years in a Sweden based electronics industry named IAAB Electronics as a Technical Head. He has published about 50 research papers in different Journals and Conferences of International repute and authored a book on Matlab and Simulink. Currently he is working for School of Electrical Engineering, VIT University, Tamil Nadu, India, as a Professor and Dean. He has received the prestigious Jawaharlal Nehru Scholarship in 1998 for his doctoral research work. He is one of the enlisted technical innovators of India in 2007. His current area of research interest includes Optoelectronic materials and Devices, Nanoscale CMOS and VLSI Engineering.

J. Raja Paul Perinbam received B.E. degree in Electrical Engineering and M.Sc. (Engg.) in Applied Electronics both from the University of Madras, India and Ph.D. from Indian Institute of Technology Madras, India in the years 1970, 1973 and 1984 respectively. He worked as a professor in the College of Engineering, Anna University, Chennai, India from 1975 to 2008. He was also with Government College of Technology, Tamil Nadu, India and Indian Institute of Science, Bangalore, India. Currently he is working as a Professor in KCG College of Technology, Chennai, India. He is a renowned electronic circuit designer and is associated with many industries. His research areas includes VLSI Design and Embedded systems.

Please cite this article as: S. Sivanantham, et al., Enhancement of test data compression with multistage encoding, INTEGRATION, the VLSI journal (2013), http://dx.doi.org/10.1016/j.vlsi.2013.12.001