# Test Data Compression Using Dictionaries with Selective Entries and Fixed-Length Indices

LEI LI and KRISHNENDU CHAKRABARTY
Duke University
and
NUR A. TOUBA
University of Texas, Austin

We present a dictionary-based test data compression approach for reducing test data volume in SOCs. The proposed method is based on the use of a small number of ATE channels to deliver compressed test patterns from the tester to the chip and to drive a large number of internal scan chains in the circuit under test. Therefore, it is especially suitable for a reduced pin-count and low-cost DFT test environment, where a narrow interface between the tester and the SOC is desirable. The dictionary-based approach not only reduces test data volume but it also eliminates the need for additional synchronization and handshaking between the SOC and the ATE. The dictionary entries are determined during the compression procedure by solving a variant of the well-known clique partitioning problem from graph theory. Experimental results for the ISCAS-89 benchmarks and representative test data from IBM show that the proposed method outperforms a number of recently-proposed test data compression techniques. Compared to the previously proposed test data compression approach based on selective Huffman coding with variable-length indices, the proposed approach generally provides higher compression for the same amount of hardware overhead.

Categories and Subject Descriptors: B.7.3 [**Integrated Circuits**]: Reliability and Testing—*built-in tests*; *test generation*

General Terms: Algorithms, Design

Additional Key Words and Phrases: Embedded core testing, reduced pin-count testing, SoC testing, test data volume, test application time

This research was supported in part by the National Science Foundation under grants CCR-9875324 and CCR-0204077, and by a graduate fellowship from the Design Automation Conference.
A preliminary and abridged version of this article appeared in *Proceedings of the IEEE VLSI Test Symposium* (Napa Valley, Calif., Apr.). IEEE Computer Society Press, Los Alamitos, Calif., pp. 219–224.
Author's address: L. Li and K. Chakrabarty, Department of Electrical and Computer Engineering, 130 Hudson Hall, Box 90291, Duke University, Durham, NC 27708; email: {ll,krish}@ee.duke.edu; N. A. Touba, Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712-1084.

## 1. INTRODUCTION

Intellectual property (IP) cores are now being routinely used in large system-on-a-chip (SOC) designs. Higher circuit densities and a larger number of embedded cores lead to higher test data volume, which in turn leads to an increase in testing time. New techniques are therefore needed to reduce test data volume and testing time, as well as to overcome ATE memory and bandwidth limitation. In order to facilitate test reuse and the use of IP cores, these techniques should not require detailed structural models for additional fault simulation or test generation.

Built-in self-test (BIST) offers a promising alternative to ATE-based external testing. While BIST is now extensively used for memory testing, it is less common for logic testing. Problems with logic BIST include inadequate fault coverage due to random-resistant fault and bus contention during test application. While the fault coverage can be made quite high using methods such as reseeding [Hellebrand et al. 2000], bit-flipping [Wunderlich and Kiefer 1996] and bit-fixing [Touba and McCluskey 1996], these techniques require structural information for fault simulation and test generation.

Structural methods for reducing test data volume and testing time require design modifications. For example, the Illinois scan architecture (ILS) offers an alternative to conventional scan design [Hsu et al. 2001]. However, fault simulation and test generation are necessary in ILS as post-processing steps to get high fault coverage.

Test data compression is a nonintrusive method that can be used to compress the precomputed test set $T_D$ provided by the core vendor to a much smaller test set $T_E$, which is then stored in ATE memory. An on-chip decoder is used to generate $T_D$ from $T_E$ during pattern application. A number of techniques based on statistical coding [Iyengar et al. 1999; Jas et al. 1999], run-length coding [Jas and Touba 1998], Golomb coding [Chandra and Chakrabarty 2001b], FDR coding [Chandra and Chakrabarty 2001a], EFDR coding [El-Maleh and Al-Abaji 2002], and VIHC coding [Gonciari et al. 2002], have been proposed to reduce test data volume. Test data volume reduction techniques based on on-chip pattern decompression are also presented in Bayraktaroglu and Orailoglu [2001], El-Maleh et al. [2001], Reda and Orailoglu [2002], Schafer et al. [2002], and Volkerink et al. [2002]. Several dictionary-based compression methods have recently been presented to reduce SOC test data volume. In Jas et al. [1999], frequently occurring blocks are encoded into variable-length indices using Huffman coding. A dictionary with fixed-length indices is used to generate all the distinct output vectors in Reddy et al. [2002]. A test data compression technique based on LZ77 algorithm, which uses a dynamic dictionary, is proposed in Wolff and Papachristou [2002].

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 [Barnhart et al. 2001] and SmartBIST [Koenemann et al. 2001] tools from IBM and the TestKompress tool from Mentor Graphics [Rajski et al. 2002] reduce test data volume and testing time through the use of test data compression and on-chip decompression.

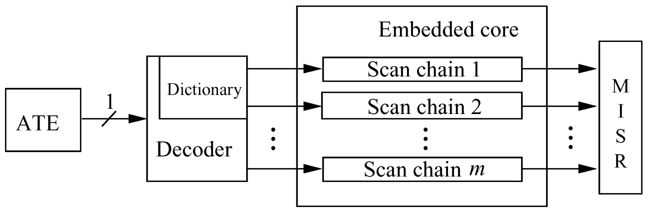


Fig. 1. Illustration of the proposed method for a single ATE channel.

In this article, we present a dictionary-based test data compression method for IP cores that provides significant compression for precomputed test sets. The dictionary uses fixed-length indices, and its entries are carefully selected such that the dictionary is efficiently utilized. The proposed method is based on the use of a small number of ATE channel to drive a large number of internal scan chains in the core under test; see Figure 1. (The test response can be compacted using a MISR or other techniques. We do not consider output compression in this article.) This technique does not require a gate-level circuit model for fault simulation or test generation; this is in contrast to BIST methods such as Touba and McCluskey [1996] and Wunderlich and Kiefer [1996] and commercial test data compression tools that interleave test cube compression with test generation [Koenemann et al. 2001; Rajski et al. 2002].

Unlike coding techniques such as Chandra and Chakrabarty [2001a, 2001b], this approach does not require multiple clock cycles to determine the decompressed test pattern after the last bit of the corresponding compressed data packet is transferred from the ATE to the chip. This dictionary-based approach therefore not only reduces testing time but it also eliminates the need for additional synchronization and handshaking between the SOC and the ATE. This approach is therefore targeted towards a reduced pin-count test and low-cost DFT tester [Vranken et al. 2001] environment, where a narrow interface between the tester and the SOC is desirable. We exploit the fact that a set of test cubes with close to 100% fault coverage for full scan circuits contains a large number of don't-cares [Rajski et al. 2002]. Test set relaxation techniques such as in El-Maleh and Al-Suwaiyan [2002] and Kajihara and Miyase [2001] can often be used to obtain an even larger number of don't-cares.

The rest of the article is organized as follows. In Section 2, we briefly review dictionary-based data compression. Section 3 shows how a dictionary can be used for test data compression. We describe how a variant of the clique partitioning problem from graph theory can be used for the compression procedure. In Section 4, we present upper and lower bounds on the amount of compression that can be obtained with a dictionary. These bounds are expressed in terms of the number of scan chains in the circuit under test, the length of the longest scan chains, the size of the dictionary, and the number of test patterns. Section 5 describes the decompression architecture. Experimental results and a comparison with related recent work are presented in Section 6. Finally, Section 7 concludes the article.

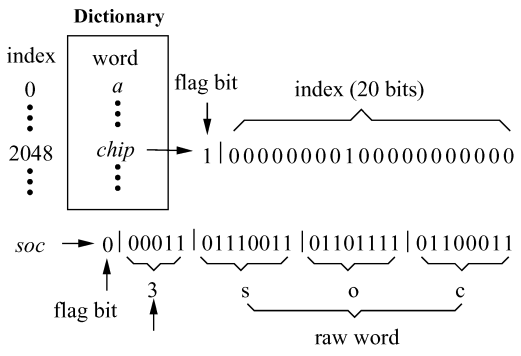


Fig. 2. An example of dictionary-based data compression.

## 2. DICTIONARY-BASED DATA COMPRESSION

Dictionary-based methods are quite common in the data compression domain [Salomon 2000]. While statistical methods use a statistical model of the data and encode the symbols using variable-size codewords according to their frequencies of occurrence, dictionary-based methods select strings of the symbols to establish a dictionary, and then encode them into equal-size tokens using the dictionary. The dictionary stores the strings, and it may be either static or dynamic (adaptive). The former is permanent, sometimes allowing for the addition of strings but no deletions, whereas the latter holds strings previously found in the input stream, allowing for additions and deletions of strings as new input is processed.

A simple example of a static dictionary is an English dictionary used to encode English text that consists of words. A word in the input text is encoded as an index to the dictionary if it appears in the dictionary. Otherwise it is encoded as the size of the word followed by the word itself. In order to distinguish between the index and the raw word, a flag bit needs to be added to each codeword. We present an example next to illustrate the encoding of a word. Suppose the dictionary contains $2^{20}$ words and thus needs a 20-bit index to specify an entry. A value of 0 for the flag bit indicates that this codeword is composed of the size of the word and the word itself following the flag bit. A value of 1 for the flag bit implies that the 20 bits of data following it is a dictionary index. Suppose a 5-bit field is used to specify the size of the word. As shown in Figure 2, the word *chip*, which is present in the dictionary with index 2048, is encoded as 1|00000000100000000000. The word *soc*, which is not in the dictionary, is encoded as 0|00011|01110011|01101111|01100011, where the 5-bit field 00011 indicates that three more bytes follow it.

The well-known compression algorithm LZ77 [Salomon 2000] is based on a dynamic dictionary. It uses part of the previously-seen input stream as the dictionary. Since our proposed method uses only a static dictionary, details of a dynamic dictionary are not discussed here.

## 3. DICTIONARY-BASED COMPRESSION OF TEST DATA

In this section, we describe the proposed dictionary-based test data compression method and illustrate it with an example. In the following description,

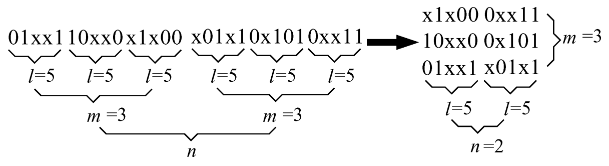


Fig. 3. An example of formatting the test data for multiple scan chains.

we assume that the precomputed SOC test data $T_D$ consists of $n$ test patterns $t_1, t_2, \ldots, t_n$. The scan elements of the core under test are divided into $m$ scan chains in as balanced a manner as possible. Each test vector can therefore be viewed as $m$ subvectors. If one or more subvectors are shorter than the others, don't-cares are padded to the end of these subvectors so that all the subvectors have the same length, which is denoted by $l$. The $m$-bit data at the same position of each subvector constitute an $m$-bit *word*. A total of $nl$ $m$-bit words thus are formed and encoded during the compression procedure. Figure 3 illustrates the formatting of the given test data for multiple scan chains. During test application, after a codeword is shifted into the decoder, an $m$-bit word $u_1, u_2, \ldots, u_m$ is immediately generated by the decoder and fed into the scan chains (one bit for each scan chain).

In the dictionary-based test data compression method, each codeword is composed of a prefix and a stem. The prefix is a 1-bit identifier that indicates whether the stem is a dictionary index or a word of uncompressed test data. If it equals 1, the stem is viewed as a dictionary index. On the other hand, if the prefix equals 0, the stem is an uncompressed word and it is $m$ bits long. The length of the dictionary index depends on the size of the dictionary. If $D$ is the set of the entries in the dictionary, the length of the index $l_{index} = \lceil \log_2 |D| \rceil$, where $|D|$ is the size of the dictionary. Since $l_{index}$ is much smaller than $m$, the compression efficiency is greater if more test data words can be obtained from the dictionary. However, the dictionary must be reasonably small to keep the hardware overhead low. Fortunately, since there are many don't-care bits in scan test data for typical circuits, we can appropriately map these don't-care bits to binary values and carefully select the entries for the dictionary, so that as many words as possible are mapped to the entries in the dictionary.

An important step in the compression procedure is that of selecting the entries in the dictionary. This problem can be easily mapped to a variant of the clique partitioning problem from graph theory [Cormen et al. 2001]. We next describe the clique partitioning problem and then show how the problem of determining dictionary entries can be mapped to this problem. We then present a heuristic algorithm for generating the dictionary entries. The proposed algorithm presents a graph-theoretic view of the procedure presented in Jas et al. [1999].

An undirected graph $G$ consists of a set of vertices $V$ and a set of edges $E$, where each edge connects an unordered pair of vertices. Given an undirected graph $G = (V, E)$, a *clique* of the graph is a subset $V' \subseteq V$ of vertices, each pair of which is connected by an edge in $E$ [Cormen et al. 2001]. Given a positive integer $K$, the clique partitioning problem refers to the partitioning of $V$ into

Table I. An Example of Test Data for Multiple Scan Chains

| Scan chain index | Word index | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | X | X | 0 | X | 0 | 1 |
| 2 | X | 0 | 1 | 1 | 0 | 0 | 1 | X | X | X | 1 | X | X | 0 | 1 | 0 |
| 3 | X | X | X | X | 0 | X | 0 | 1 | 0 | 0 | 1 | 1 | 0 | X | X | X |
| 4 | X | 0 | X | 0 | X | X | 0 | X | 0 | 0 | 0 | 0 | 0 | X | 0 | 1 |
| 5 | 0 | 0 | 0 | 0 | X | 0 | X | 0 | X | X | X | 0 | X | 1 | 0 | X |
| 6 | 0 | X | 1 | 0 | 1 | 0 | X | X | 1 | X | 0 | 0 | X | 0 | X | X |
| 7 | 1 | 0 | 1 | X | X | X | X | 1 | 1 | 0 | X | 1 | 0 | 0 | 1 | 0 |
| 8 | 1 | X | 0 | X | 0 | 1 | X | 1 | 0 | X | X | X | X | X | X | 1 |

$k$ cliques, where $k \leq K$. The clique partitioning problem is $\mathcal{NP}$-hard [Garey and Johnson 1979],[1] hence heuristic approaches must be used to solve it in reasonable time for large problem instances.

Recall that in dictionary-based data compression, we obtain $nl$ $m$-bit words after placing the test set in a multiple scan chain format. Two words $u_1u_2 \cdots u_m$ and $v_1v_2 \cdots v_m$ are defined to be *compatible* to each other if for any position $i$, $u_i$ and $v_i$ are either equal to each other or at least one of them is a don't-care bit. We construct an undirected graph $G$ to reflect the compatible relationships between the words as follows: First, a vertex is added to the graph for each word. Then, we examine each pair of words. If they are mutually compatible, an edge is added between the corresponding pair of vertices. A clique in $G$ refers to a group of test data words that can be mapped to the same dictionary entry. If the dictionary can have at most $|D|$ entries and the total number of words is $nl$, the goal of the compression procedure is to find the largest subset of $G$ that can be partitioned into $|D|$ cliques; the remaining vertices in $G$ denote test data words that are not compressed. This problem can easily be shown to be $\mathcal{NP}$-hard by contradiction. If the compression can be optimally solved in polynomial time then it provides a yes/no answer to the decision version of the clique partitioning problem in polynomial time. We therefore use the following heuristic procedure.

(1) Copy the graph $G$ to a temporary data structure $G'$.
(2) Find the vertex $v$ with the maximum degree in $G'$.
(3) Establish a subgraph that consists of all the vertices connected to $v$. Copy this subgraph to $G'$ and add $v$ to a set $C$. (The subgraph thus formed does not include the vertex $v$.)
(4) If $G'$ is not empty, go to Step (2). Otherwise, a clique $C$ has been formed consisting all the vertex found in Step (2).
(5) Remove the vertices in the clique $C$ from $G$ and copy $G - C$ to $G'$. Go to Step 2 and repeat until $|D|$ cliques are found.

The complexity of this procedure is $O(N^3)$, where $N = nl$ is the number of vertices in the graph. Table I shows an example of test data formatted for

[1]The decision version of the clique partitioning problem is $\mathcal{NP}$-complete.

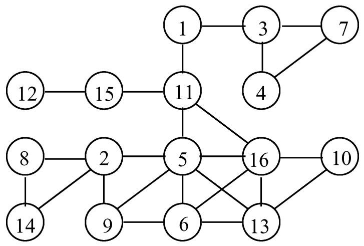


Fig. 4. The graph $G$ for the example of Table I.

multiple scan chains. The number of scan chains $m$ is 8 in this example. There are total of 16 words, each of which has 8 bits. Figure 4 shows the corresponding graph $G$ for the test data. Let us assume that a dictionary of size four is to be formed, that is, $|D| = 4$. Using the greedy algorithm described above, we obtain four cliques: {5, 6, 13, 16}, {2, 8, 14}, {3, 4, 7} and {1, 11}. (Here we use the word indices of Table I to represent the vertices.) After finding the cliques, we obtain the corresponding dictionary entry for each clique by merging the words in this clique. In this example, the four dictionary entries are {11100011, 01000110, 0000100X, 10X10001}. Three bits are then needed to encode the words in the cliques; an additional 1 bit is needed for the prefix, and 2 bits are required for the dictionary index. For words that are not in any clique, a total of 9 bits each must be transferred from the ATE. Since there are 12 words that can be generated from the dictionary, the size of the compressed data is $3 \times 12 + 9 \times 4 = 72$ bits, which corresponds to a compression of 43.75%. Moreover, the dictionary entries still contain some don't-care bits, which can reduce the hardware for the decoder, as explained next.

The clique partitioning procedure introduces a certain degree of randomness in the way the don't-care bits in $T_D$ are filled; the resulting "random fill" can be expected to increase the fortuitous detection of non-modeled faults. This is in contrast to coding methods such as Chandra and Chakrabarty [2001a, 2001b] in which the don't-cares are all mapped to 0s.

## 4. COMPRESSION BOUNDS

In this section, we derive simple lower and upper bounds on the amount of compression that can be obtained using the proposed dictionary. If the length of a test vector is $L$ bits, the length of each scan chain for a circuit with $m$ scan chains is $l = \lceil \frac{L}{m} \rceil$. Here we assume that the scan chains are balanced. Unbalanced scan chains can be balanced by adding dummy cells. As discussed in Section 2, for a test set $T_D$ with $n$ patterns, a total of $N = nl = n\lceil \frac{L}{m} \rceil$ $m$-bit words are generated and encoded during the compression procedure. If a data word matches an entry in the dictionary, i.e., it can be mapped to a dictionary entry by appropriately assigning 0 or 1 to its unspecified bits, a $1 + \lceil \log_2 |D| \rceil$-bit codeword is needed to encode it, where $|D|$ is the size of the dictionary; otherwise, $1 + m$ data bits are needed to encode this data word.

We denote the compressed test data set by $T_E$. A lower bound $B_L$ on the size of compressed test data $|T_E|$ is obtained by assuming that all the $N$ data words

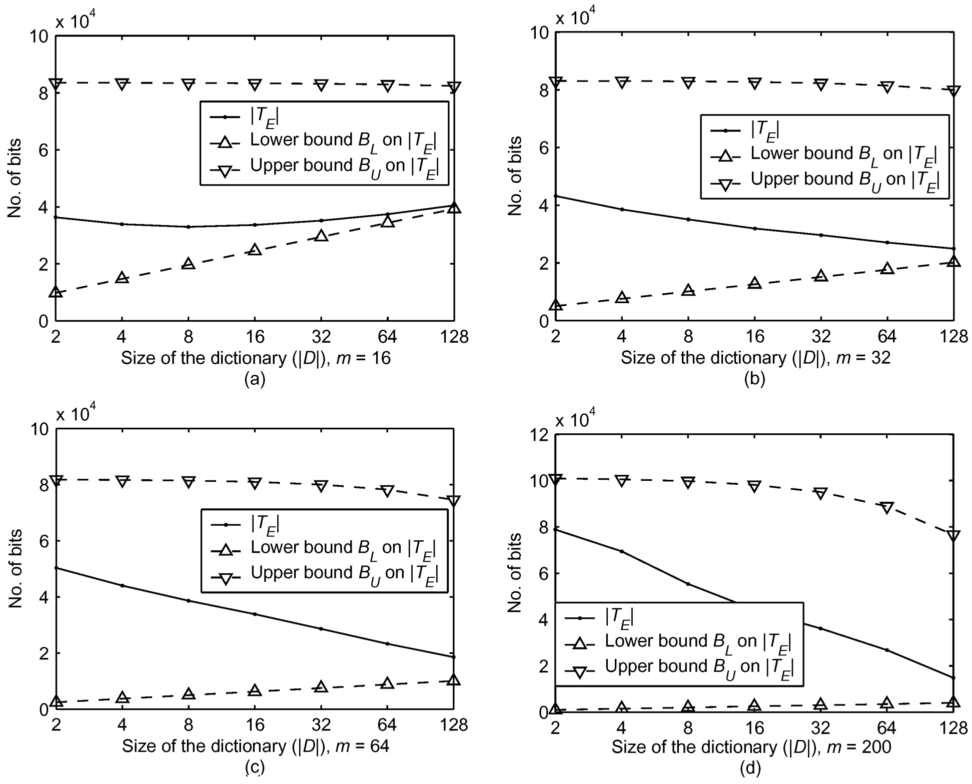


Fig. 5. Variation of the compression bounds and $|T_E|$ with dictionary size for s15850 with (a) $m = 16$ (b) $m = 32$ (c) $m = 64$ (d) $m = 200$.

are matched to dictionary entries. This implies that

$$\begin{aligned} B_L &= N(1 + \lceil \log_2 |D| \rceil) \\ &= n \left\lceil \frac{L}{m} \right\rceil (1 + \lceil \log_2 |D| \rceil). \end{aligned}$$

Similarly, an upper bound $B_U$ on $|T_E|$ is obtained by assuming that each dictionary entry matches only one data word, that is,

$$\begin{aligned} B_U &= (1 + \lceil \log_2 |D| \rceil)|D| + (N - |D|)(1 + m) \\ &= (1 + \lceil \log_2 |D| \rceil)|D| + \left( n \left\lceil \frac{L}{m} \right\rceil - |D| \right) (1 + m). \end{aligned}$$

In order to study these bounds further, we encode the test sets for two ISCAS-89 benchmark circuits, and compare the $T_E$ with these bounds. The test sets were obtained from the Mintest ATPG program [Hamzaoglu and Patel 1998]. Figure 5 shows the variations of these bounds and $|T_E|$ with dictionary size $|D|$ for various values of $m$ for the s15850 benchmark circuit. The number of entries in the dictionary $|D|$ is varied in powers of 2 from 16 to 128. We find that $|T_E|$ is very close to the lower bound $B_L$ when $m = 16$ and $|D| = 128$.

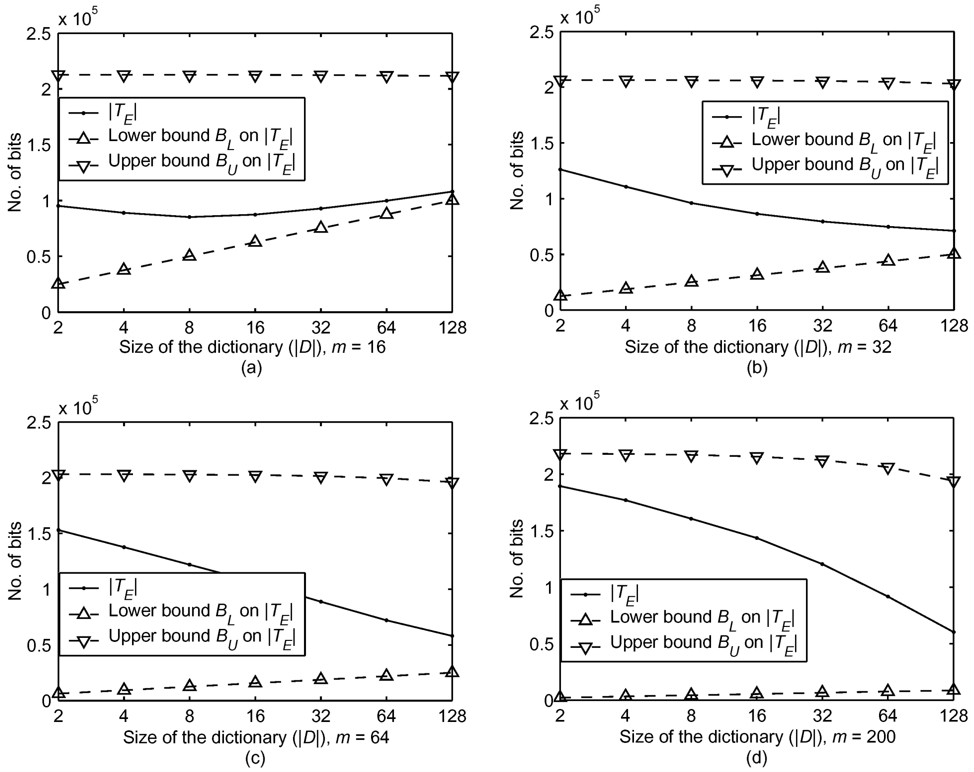


Fig. 6. Variation of the compression bounds and $|T_E|$ with dictionary size for s38584 with (a) $m = 16$ (b) $m = 32$ (c) $m = 64$ (d) $m = 200$.

This is because when the number of scan chains $m$, which also denotes the length of the data words, is small, the total number $2^m$ of possible data words is small. As a result, there is a strong likelihood that every data word matches a dictionary entry, a scenario that we used to derive the lower bound $B_L$. In Figure 5(a), it appears that 128 dictionary entries match most of the 16-bit data words obtained from the test set for s15850. A similar situation is observed in Figure 5(b) for $m = 32$. On the other hand, $|T_E|$ is very close to $B_U$ for $m = 200$ and $|D| = 2$ in Figure 5(d). When $m$ is large, the total number $2^m$ of possible data words is also large. Thus a small dictionary is likely to match only a few data words, which is analogous to the scenario used to derive the upper bound $B_U$ (only $|D|$ words are covered by the dictionary). Figure 6 shows similar results for the s38584 benchmark circuit.

In Table II, we list the *closeness* of $|T_E|$ to $B_L$, which is defined as $C = \frac{|T_E|-B_L}{B_U-B_L}$, for various values of the number of dictionary entries $|D|$ and the number of scan chains $m$. The closeness values listed in the table also show that $|T_E|$ is closer to $B_L$ for a small number of scan chains and a relatively large dictionary, and closer to $B_U$ for a large number of scan chains and a relatively small dictionary. As expected, the entries in the lower left corner of the table are close to zero.

Table II. The Closeness of $|T_E|$ to $B_L$ for (a) s15850 and (b) s38584

| | No. of scan chains | | | | | |
|---|---|---|---|---|---|---|
| $\|D\|$ | $m = 16$ | $m = 32$ | $m = 48$ | $m = 64$ | $m = 128$ | $m = 200$ |
| 2 | 0.36 | 0.49 | 0.56 | 0.60 | 0.71 | 0.78 |
| 4 | 0.28 | 0.41 | 0.50 | 0.52 | 0.63 | 0.69 |
| 8 | 0.21 | 0.34 | 0.40 | 0.44 | 0.52 | 0.55 |
| 16 | 0.15 | 0.28 | 0.34 | 0.37 | 0.42 | 0.44 |
| 32 | 0.11 | 0.22 | 0.27 | 0.29 | 0.33 | 0.36 |
| 64 | 0.06 | 0.15 | 0.20 | 0.21 | 0.24 | 0.27 |
| 128 | 0.03 | 0.08 | 0.11 | 0.13 | 0.15 | 0.15 |

(a)

| | No. of scan chains | | | | | |
|---|---|---|---|---|---|---|
| $\|D\|$ | $m = 16$ | $m = 32$ | $m = 48$ | $m = 64$ | $m = 128$ | $m = 200$ |
| 2 | 0.37 | 0.59 | 0.65 | 0.75 | 0.82 | 0.87 |
| 4 | 0.29 | 0.49 | 0.57 | 0.66 | 0.76 | 0.81 |
| 8 | 0.22 | 0.39 | 0.49 | 0.58 | 0.69 | 0.73 |
| 16 | 0.17 | 0.32 | 0.41 | 0.49 | 0.60 | 0.66 |
| 32 | 0.13 | 0.25 | 0.33 | 0.38 | 0.50 | 0.55 |
| 64 | 0.10 | 0.19 | 0.25 | 0.28 | 0.38 | 0.42 |
| 128 | 0.07 | 0.14 | 0.18 | 0.19 | 0.25 | 0.28 |

(b)

## 5. DECOMPRESSION ARCHITECTURE

In this section, we describe the decompression architecture for the dictionary-based test data compression method. Two architectures for the on-chip decoder are proposed; they are referred as Architecture I and Architecture II, respectively. Architecture I does not require any modification to the scan chains of the core under test. Architecture II is intended for cores with flexible scan chains since a slight modification to the scan chains is needed in this case. The latter requires less hardware than Architecture I.

Figure 7 illustrates the first decompression architecture. The decoder consists of a finite-state machine (FSM), an $m$-bit shifter, a $\log_2 m$-bit counter, a selector and the dictionary. The $m$-bit shifter takes data from *Data_in* when the signal *shift* equals 1. All the $m$ output bits pass to the selector while the dictionary only gets the higher order $l_{index}$ bits of the output as its index. Here, we use a combinational logic circuit to implement the dictionary. It outputs the $m$-bit word corresponding to the current value of the index.

The $\log_2 m$-bit counter is used to indicate whether the shift-in of a codeword has finished. It operates as follows.

—The signal *reset* resets the counter to 0.
—If $inc = 1$, the counter is incremented.
—When the value of the counter reaches $l_{index}$, the output *i_flag* equals 1.
—When the value of the counter reaches $m$, the output *m_flag* equals 1.

The FSM has only three states. It is enabled when the signal *Dec_en* is active. It starts by checking the first bit of the data shifted in from *Data_in*. If this bit

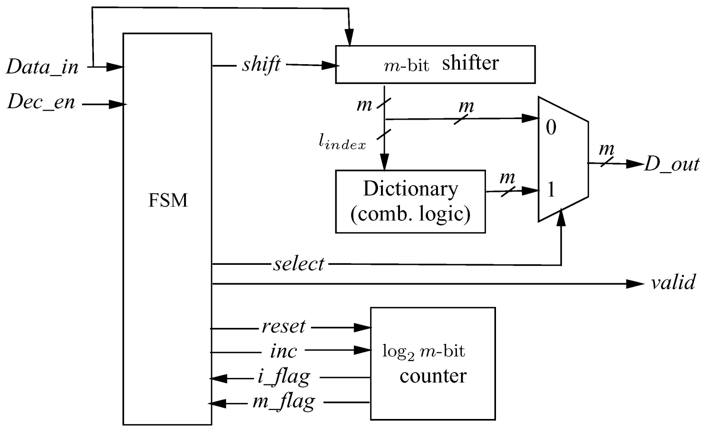


Fig. 7. Decompression Architecture I.

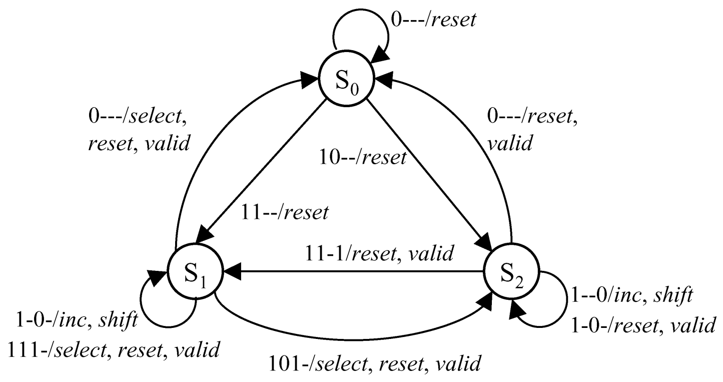


Fig. 8. State transition diagram for the FSM in Architecture I.

is 0, which implies that the next $m$ bits directly constitute a word, the FSM shifts this word into the $m$-bit shifter, which then feeds this word into the scan chains during the clock cycle in which the decoder checks the first bit of the next codeword. On the other hand, if the first bit is 1, the FSM shifts in the next $l_{index}$ bits and gets the decoded word from the dictionary. Only 19 gates and 2 flip-flops are required to implement the FSM using Synopsys Design Compiler.

Figure 8 shows the state transition diagram for the FSM in Architecture I. There are four inputs to the FSM in the order *Dec_en*, *Data_in*, *i_flag* and *m_flag*. The outputs of the FSM are set to 1 only when they appear in the output list in the state transition diagram.

In Architecture I, an $m$-bit shifter is needed for the on-chip decoder. If the number of scan chains is large, then the hardware overhead resulting from this shifter is also large. If a small modification to the scan chain is permitted, e.g., in the case of flexible scan chains, this $m$-bit shifter can be replaced by a small $l_{index}$-bit shifter. Figure 9 illustrates this decompression architecture, referred to as Architecture II. Here, the scan chains are slightly modified such

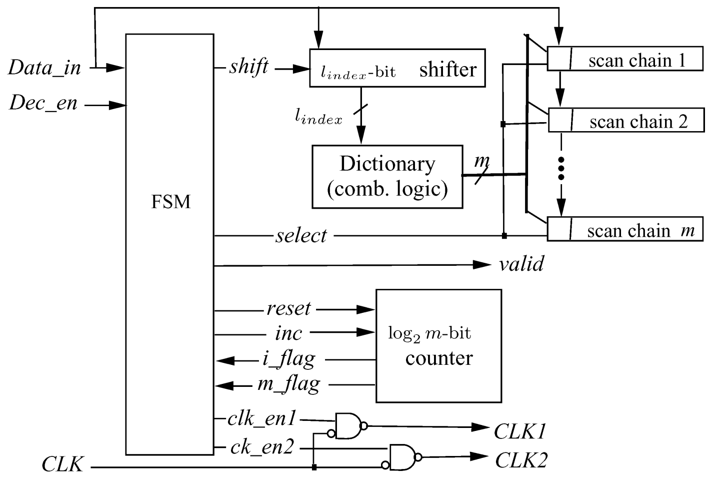


Fig. 9. Decompression Architecture II.

that the first cell in each scan chain can receive the input test data either from the dictionary or from the first bit of the previous scan chain. The signal *select* is used to select the source of the input test data for the first cell of the scan chains. The modification to a scan chain is therefore limited to its first scan cell. This can be implemented in a non-intrusive fashion by adding a multiplexer at the scan input pin. During decompression, the controlled clock signal *CLK*1 is used to drive the first cells of the scan chains, and *CLK*2 drives the remaining flip-flops in the scan chains and the MISR. The operation of the FSM is now slightly different from that of Architecture I for the codewords that consist of a 0 followed by a word. If the first bit of the codeword is 0, *CLK*1 is enabled for the next $m$ cycles and *CLK*2 is enabled only for the next cycle. Thus in the next cycle, the response in the scan chains are shifted one bit towards the MISR and one bit data is shifted into the first cell of the first scan chain. During the subsequent $m - 1$ cycles, the remaining $m - 1$ bits data are shifted into the first cells of the scan chains and other cells in the scan chains are frozen by disabling the clock signal *CLK*2. The FSM for Architecture II has only 4 states (as shown in Figure 10), and 21 gates and 2 flip-flops are required to implement it using Synopsys Design Compiler.

We can further reduce hardware overhead by using a single dictionary for several cores. We concatenate the test sets for the cores and view the concatenated test data as a single entity during compression. The cores that use the same dictionary are tested serially during the test session.

In the above decompression architectures, it is assumed that only one ATE channel is used to transfer the compressed data. The extension to multiple channels is straightforward. For example, if the length of the dictionary index is 7, then a codeword consists of 8 bits (1-bit prefix and 7-bit index). For such a configuration, we can use either 2, 4 or 8 ATE channels and an on-chip register of appropriate size without wasting ATE channel bandwidth.

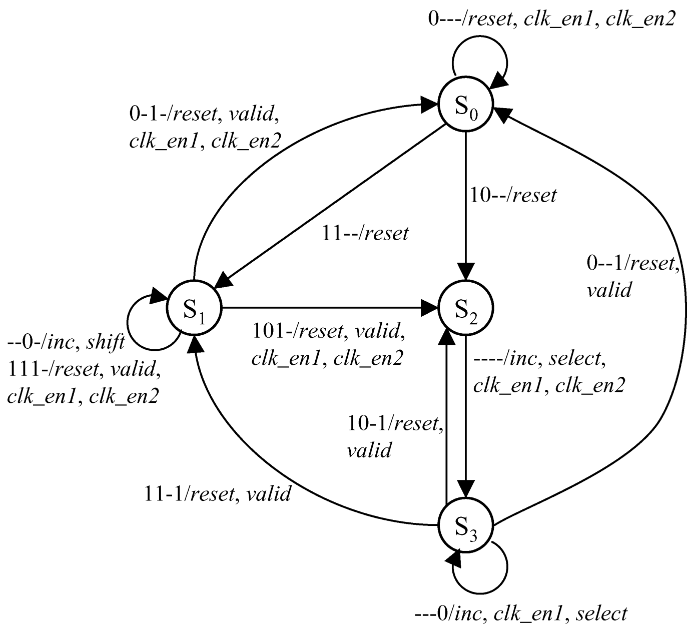

0---/reset, clk_en1, clk_en2
S0
0-1-/reset, valid, clk_en1, clk_en2
10--/reset
11--/reset
0--1/reset, valid
S1
S2
--0-/inc, shift
111-/reset, valid, clk_en1, clk_en2
101-/reset, valid, clk_en1, clk_en2
----/inc, select, clk_en1, clk_en2
10-1/reset, valid
11-1/reset, valid
S3
---0/inc, clk_en1, select

Fig. 10. State transition diagram for the FSM in Architecture II.

Table III. Compression Results with Varying Number of Scan Chains

| Circuit | No. of test vectors | No. of scan cells | Size of $T_D$ (bits) | Size of $T_E$ for varying no. of scan chains (bits) | | | | | | No. of gates for the dictionary |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $m=16$ | $m=32$ | $m=48$ | $m=64$ | $m=128$ | $m=200$ | |
| s5378 | 111 | 214 | 23754 | 12999 | 7791 | 6572 | **6345** | 8794 | 12970 | 656 |
| s9234 | 159 | 247 | 39273 | 21189 | 13551 | 13659 | 12783 | **11498** | 16826 | 951 |
| s13207 | 236 | 700 | 165200 | 83882 | 43936 | 31231 | 24074 | 13990 | **8517** | 1118 |
| s15850 | 126 | 611 | 76986 | 40491 | 24960 | 19705 | 18573 | **13873** | 14840 | 1093 |
| s35932 | 16 | 1763 | 28208 | 17214 | 8818 | 11583 | 7403 | 3123 | **1400** | 640 |
| s38417 | 99 | 1664 | 164736 | 92304 | 84009 | **62939** | 94350 | 104192 | 114243 | 581 |
| s38584 | 136 | 1464 | 199104 | 107962 | 71148 | 63740 | 58027 | 58189 | **53287** | 1469 |

## 6. EXPERIMENTAL RESULTS

In this section, we first apply the dictionary-based test data compression method to the test cubes for the seven largest ISCAS-89 circuits. These test cubes were obtained from the Mintest ATPG program [Hamzaoglu and Patel 1998]. We refer to the uncompressed test set as $T_D$ and the compressed test set as $T_E$. For each test set, we determine a dictionary that contains 128 entries, which provides good tradeoff between compression and hardware overhead. Thus, the length of the dictionary index is 7 bits.

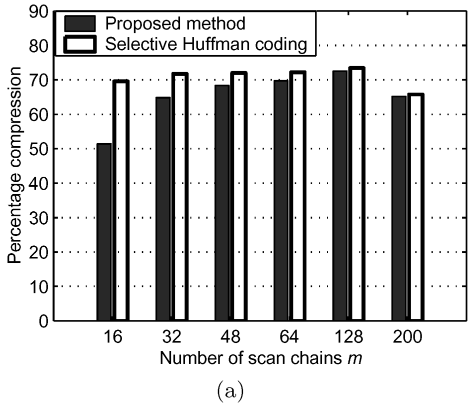
Table III shows the compression results for a varying number of scan chains. The second and third columns list the number of test vectors in each test set and the number of scan cells in each circuit, respectively. The minimum size of $T_E$ for each circuit is shown in boldface. Clearly, the amount of compression

Table IV. Comparison of Compression Results

| Circuit | Size of $T_D$ (bits) | Size of ATPG-compacted test set (bits) | Size of $T_E$ (bits): Proposed method | EFDR | VIHC | XOR network | Mutation encoding | Mutation encoding + XOR network |
|---|---|---|---|---|---|---|---|---|
| s5378 | 23754 | 20758 | 6345 | 11419 | 11516 | N/A | N/A | N/A |
| s9234 | 39273 | 25935 | 11498 | 21250 | 17736 | N/A | N/A | N/A |
| s13207 | 165200 | 163100 | 8517 | 29992 | 27737 | 25344 | 74423 | 15783 |
| s15850 | 76986 | 57434 | 13873 | 24643 | 30271 | 22784 | 26021 | 10798 |
| s35932 | 28208 | 19393 | 1400 | 5554 | 9458 | 7128 | 7222 | 3972 |
| s38417 | 164736 | 113152 | 62939 | 64962 | 74938 | 89856 | 45003 | 42264 |
| s38584 | 199104 | 161040 | 53287 | 73853 | 85674 | 38976 | 73464 | 22636 |

depends on the number of scan chains for any given circuit. The last column of the table shows the hardware overhead for the dictionary corresponding to the value of $m$ for which the minimum size of $T_E$ is obtained. For instance, the choice of $m = 64$ minimizes the size of $T_E$ for circuit s5378, and 656 gates are required to implement the corresponding dictionary, which has 7 inputs and 64 outputs. Since no additional handshaking is required between the ATE and the SOC, the testing time (in ATE clock cycles) is simply equal to the size of $T_E$. This is usually higher than the $nl$ ATE clock cycles required if no encoding is done. However, the proposed approach uses only a single ATE data channel instead of $m$ channels needed to drive $m$ scan chains in parallel. If additional data channels are used, the testing time can be reduced significantly for the proposed method.

Table IV presents a comparison of the compression results with EFDR coding [El-Maleh and Al-Abaji 2002], variable-length input Huffman coding (VIHC) [Gonciari et al. 2002], XOR network encoding [Bayraktaroglu and Orailoglu 2001], and test data mutation encoding [Reda and Orailoglu 2002]. Of these methods, only El-Maleh and Al-Abaji [2002] is based on a circuit-independent and test-independent decoder. The best reported results for the competing methods are listed in Table IV. The sizes of the ATPG-compacted test sets are also listed for the sake of comparison. The size of $T_E$ listed for the dictionary-based method is the minimum size shown in boldface in Table III. We find that the dictionary-based method outperforms EFDR coding and VIHC coding for all circuits. Compared to XOR network encoding, the compression is higher for four out of five circuits. There are two sets of compression results for test data mutation encoding in Reda and Orailoglu [2002]. They are obtained by encoding Mintest test sets and Atalanta-generated test sets, respectively. We used the first set of results for comparison since it is based on the same test sets that are used in our experiments. The last column lists the compression results obtained by combining test data mutation encoding and XOR network encoding, as well as additional structural information for test generation and compaction. Compared to these three sets of results, the dictionary-based method achieves higher compression for at least three out of five circuits for each case. Note that for mutation encoding, an additional test enable signal is necessary. This adds to the overhead for the scheme—additional storage (not included in Table III) and an additional tester channel are needed.



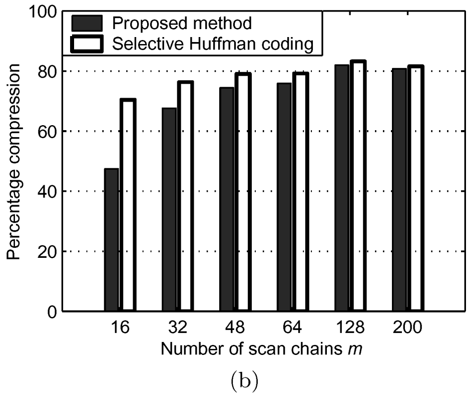
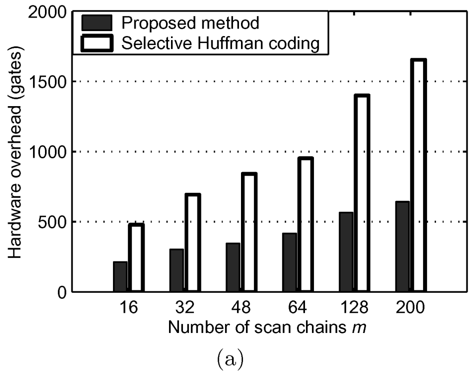


Fig. 11. Compression versus the number of scan chains for circuit s15850 with (a) $|D| = 64$ (b) $|D| = 128$.



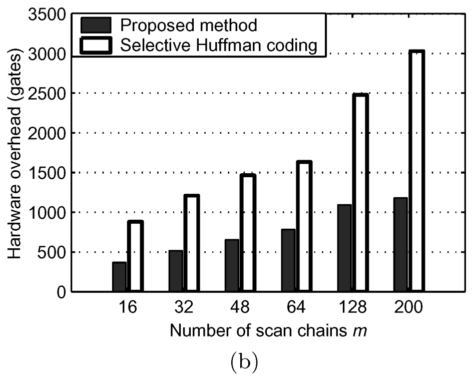


Fig. 12. Hardware overhead versus the number of scan chains for circuit s15850 with (a) $|D| = 64$ (b) $|D| = 128$.

In Reddy et al. [2002], all distinct output $m$-bit words for $m$ scan chains are included in the dictionary. This leads to a prohibitively large dictionary, especially for large values of $m$. The statistical coding method of Jas et al. [1999] is based on dictionaries with variable-length indices. For comparison, we implemented the selective Huffman coding algorithm in Jas et al. [1999] and used it to encode the dictionary entries that are obtained by the heuristic procedure described in Section 2. Figure 11 shows the percentage compression versus the number of scan chains for s15850 for two different dictionary sizes $|D| = 64$ and $|D| = 128$. Figure 12 shows the hardware overhead required to implement the corresponding dictionaries. The method based on Huffman coding provides higher compression than the proposed method. However, the difference in the compression for the two method is insignificant for large values of $m$, where the highest compression is achieved for both methods. Furthermore, the finite-state machine decoder for selective Huffman coding requires 2–3 times more chip area to implement than the proposed fixed-length-index dictionary. In order to compare the two methods accurately, we set a priori limit on the hardware

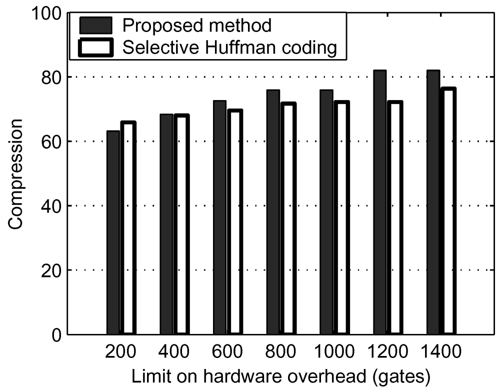


Fig. 13. Compression versus the limit on the hardware overhead for circuit s15850.

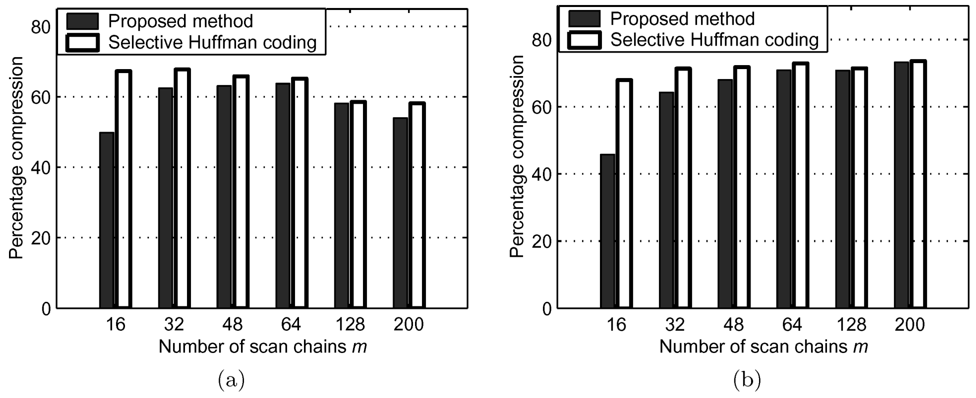
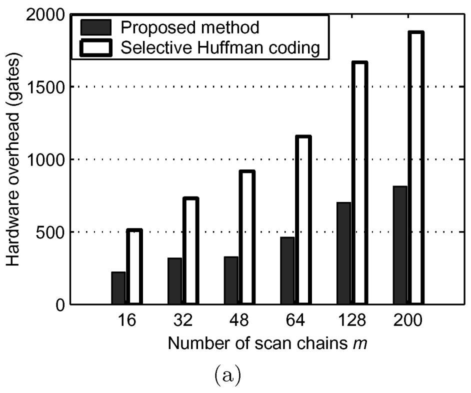


Fig. 14. Compression versus the number of scan chains for circuit s38584 with (a) $|D| = 64$ (b) $|D| = 128$.

overhead and determined the compression for both methods. The best compression results are shown in Figure 13 over all possible values of $m$ and $|D|$ that meet the constraints on hardware overhead. As shown in Figure 13, the proposed method provides higher compression for all the but one cases. It is only when the limit on hardware overhead is set to 200 gates that the method based on Huffman coding provides higher compression. Similar results are observed for s38584 (shown in Figures 14–16) and the other benchmark circuits.

Note that even smaller test data volume has been reported recently for these circuits using a combination of packetization and data compression codes [Volkerink et al. 2002]. These numbers were however obtained after a preprocessing step involving pseudorandom patterns, therefore a direct comparison can be misleading. Test data compression of up to 100X has also been reported recently for industrial designs using commercial tools [Rajski et al. 2002]. This approach however relies on the use of test generation together with test data compression—less compression might be expected for precomputed tests. A direct comparison with Rajski et al. [2002] is also difficult due to the proprietary nature of the underlying compression method and the industrial designs. In



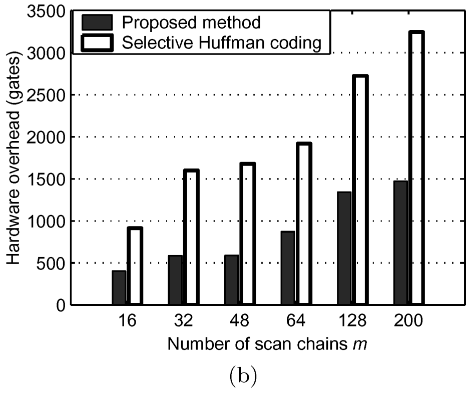


Fig. 15. Hardware overhead versus the number of scan chains for circuit s38584 with (a) $|D| = 64$ (b) $|D| = 128$.

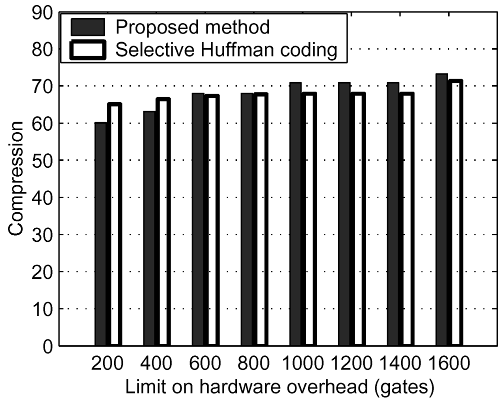


Fig. 16. Compression versus the limit on the hardware overhead for circuit s38584.

our work, we neither use a preprocessing step involving pseudorandom patterns nor do we interleave test generation with test data compression. These additional steps can also be used to enhance the effectiveness of dictionary-based compression.

Table V shows compression results when a single dictionary is used for pairs of ISCAS-89 circuits. We consider pairs of circuits that have nearly the same number of scan cells. The third column in the table lists the number of scan chains with which the dictionary-based method is used. The test data volume for the competing methods is calculated by summing the test data volumes for the individual circuits. We see that even though a single dictionary is used for the two circuits, the compression is greater than that for VIHC coding in all cases, also better than that for EFDR coding in two out of three cases, and it is higher than XOR-network encoding and test data mutation encoding in one out of two cases.

In order to evaluate the compression efficiency of the dictionary-based method for large test sets, we applied the method to two real test sets from

Table V. Compression Results on Using a Single Dictionary for Two Circuits

| Circuit pair | Size of $T_D$ (bits) | No. of scan chains ($m$) | No. of gates for the dictionary | Size of $T_E$ (bits) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Proposed method | | EFDR | VIHC | XOR network | Mutation encoding | Mutation encoding + XOR network |
| | | | | Joint | Separate | | | | | |
| {s5378, s9234} | 63027 | 32 | 525 | 24642 | 21342 | 32669 | 29252 | N/A | N/A | N/A |
| {s13207, s15850} | 242186 | 128 | 1175 | 40810 | 27863 | 54635 | 58008 | 48128 | 100444 | 26581 |
| {s38417, s38584} | 363840 | 48 | 689 | 150090 | 126679 | 138815 | 160612 | 128832 | 118467 | 64900 |

Table VI. Compression Results for Test Data from IBM

| Circuit | Size of $T_D$ (bits) | No. of scan chains ($m$) | Size of $T_E$ (bits) | Percentage compression | No. of gates for the dictionary | Size of $T_E$ (bits) for [Jas et al. 1999], and compression | No. of gates for the FSM in [Jas et al. 1999] |
|---|---|---|---|---|---|---|---|
| CKT1 | 11613472 | 400 | 232824 | 98.00 | 1541 | 122407 (98.95%) | 6256 |
| | | 300 | 309848 | 97.33 | 1721 | 205057 (98.23%) | 5039 |
| | | 200 | 464880 | 96.00 | 1198 | 273983 (97.64%) | 3255 |
| CKT2 | 4124288 | 256 | 164752 | 96.01 | 2455 | 150793 (96.34%) | 4963 |
| | | 200 | 172519 | 95.82 | 1917 | 136118 (96.70%) | 4283 |
| | | 128 | 276668 | 93.29 | 1516 | 177744 (95.69%) | 2866 |

industry. The test set for the first circuit (CKT1) from IBM consists of 32 statically-compacted scan vectors (a total of 362921 bits of test data per vector). This microprocessor design consists of 3.6 million gates and 726000 latches. The test set for a second microprocessor circuit (CKT2) from IBM consists of a set of 4 scan vectors (a total of 1031072 bits of test data per vector); this design contains 1.2 million gates and 32200 latches. Since we do not have access to the gate-level models for these circuits, we are unable to report fault coverage values for these test sets. Table VI lists the information about the test sets and the compression results for these two industrial circuits. In the absence of information about the number of scan chains for CKT1 and CKT2, two cases are assumed for each circuit: a total of 400 and 200 scan chains for CKT1, and 200 and 128 scan chains for CKT2. The sixth column of Table VI lists the number of gates required to implement the corresponding dictionaries with Synopsys Design Compiler. For CKT1 and CKT2, we obtain 50X and 25X compression, respectively, and the hardware overhead due to the dictionaries is negligible. We applied the technique of Jas et al. [1999] to these test sets for block sizes of 400 and 200, respectively. As shown in Table VI, the compression is greater, but the hardware overhead is substantially higher for the coding method of Jas et al. [1999].

Note that, as in Gonciari et al. [2002], Jas et al. [1999], and Reddy et al. [2002], the dictionaries presented are specific to the circuit under test. It is often desirable however to use a circuit-independent dictionary, which as in Chandra and Chakrabarty [2001a, 2001b], makes the decompression logic independent of the precomputed test set. This is especially useful in cases of circuit redesign or test set modifications. In the proposed approach, the decompression logic is test-independent if the dictionary is transferred from the ATE to an embedded

RAM at the start of a test session. If this is not the case, and the dictionary is implemented as a custom combinational logic, we can still use the same dictionary for several circuits, albeit with a potential decrease in the amount of compression. The negative impact on compression can however be minimized if a dictionary for a small circuit (CKT A) is used for a large circuit (CKT B). The relatively smaller number of test patterns for CKT A leaves many don't-cares in the dictionary, which in turn can be used to efficiently match the test patterns for CKT B. For example, we compressed $T_D$ for s13207 using the dictionary for s5378 and found that $T_E$ in this case contains 23571 bits, which is still less than the test data volume for several competing methods. In another experiment aimed at evaluating the effect of test set modifications, we first generated a dictionary for s38584 with $m = 32$ and then randomly altered 10% of the test vectors in $T_D$. The resulting compression obtained with the original dictionary is only 6% less than before.

We also investigated the impact of incremental changes to the circuit netlist and the test set on the effectiveness of dictionary-based compression. We considered the ISCAS-89 benchmark circuit s5378, which consists of 1775 inverters and 1004 other logic gates (239 OR gates and 765 NOR gates). We used Atalanta to generate a test set $T_D$ for s5378, containing 1125 test cubes. This test set was then compressed using a dictionary of 128 entries to obtain $T_E$. For a scan configuration of 128 scan chains, the percentage compression was found to be 92.36%. We next modified circuit s5378 to s5378* by changing 5% of the 1004 logic gates to another type (12 OR gates to AND gates, and 38 NOR gates to NAND gates), and used Atalanta to generate a test set $T_D^*$ for s5378* containing 943 test cubes. The dictionary used to compress the test set $T_D$ was then used to compress $T_D^*$. The percentage compression in this case was 91.20%, which is only marginally less than the compression percentage for the original circuit.

Finally, we generated a single dictionary for four ISCAS-89 benchmark circuits, namely s5378, s9234, s13207, and s15850, and compared the combined test data volume for compression for these circuits to that obtained using separate dictionaries and using the EFDR and VHIC methods. The total test data volume based on separate dictionaries is 48155 bits, a significant reduction from the original test data volume of 305213 bits. With a single dictionary for all four circuits, the test data volume is 85665 bits, which still amounts to a significant amount of compression. More importantly, this test data volume is less than the 87304 bits and 87260 bits, obtained with the EFDR and VHIC methods, respectively.

## 7. CONCLUSION

We have shown how dictionary-based test data compression can be used to reduce test data volume for SOCs. The proposed method delivers compressed patterns from the tester to the chip and drives a large number of internal scan chains using only a single ATE channel. Hence, the dictionary-based compression technique is especially suitable for reduced-pin count testing, multi-site testing, as well as a low-cost DFT test environment. In contrast to techniques based on data compression codes, this approach does not rely on additional synchronization between the SOC and the ATE. Experimental results for the ISCAS-89 and representative test data from IBM show that the dictionary can be implemented with a small amount of hardware and the test data volume for the proposed method is less than that for a number of recently proposed test data compression techniques. Compared to the previously proposed test data compression based on selective Huffman coding, the proposed method achieves higher compression for the same amount of hardware overhead, except for the special case when the constraint on the hardware overhead is very small.

## REFERENCES

Barnhart, C., Brunkhorst, V., Distler, F., Farnsworth, O., Keller, B., and Koenemann, B. 2001. OPMISR: The foundation for compressed ATPG vectors. In *Proceedings of the International Test Conference*. 748–757.

Bayraktaroglu, I. and Orailoglu, A. 2001. Test volume and application time reduction through scan chain concealment. In *Proceedings of the ACM/IEEE Design Automation Conference*. ACM, New York, 151–155.

Chandra, A. and Chakrabarty, K. 2001a. Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression. In *Proceedings of the VLSI Test Symposium*. 42–47.

Chandra, A. and Chakrabarty, K. 2001b. System-on-a-chip test data compression and decompression architectures based on Golomb codes. *IEEE Trans. Computer-Aided Design 20*, 355–368.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. 2001. *Introduction to Algorithms*. MIT press, Cambridge, London, England.

El-Maleh, A. and Al-Abaji, R. 2002. Extended frequency-directed run-length codes with improved application to system-on-a-chip test data compression. In *Proceedings of the International Conference on Electronics, Circuits and Systems*. 449–452.

El-Maleh, A. and Al-Suwaiyan, A. 2002. An efficient test relaxation technique for combinational and full-scan sequential circuits. In *Proceedings of the VLSI Test Symposium*. 53–59.

El-Maleh, A., al Zahir, S., and Khan, E. 2001. A geometric-primitives-based compression scheme for testing systems-on-chip. In *Proceedings of the VLSI Test Symposium*. 54–59.

Garey, M. R. and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York.

Gonciari, P. T., Al-Hashimi, B., and Nicolici, N. 2002. Improving compression ratio, area overhead, and test application time for system-on-a-chip test data compression/decompression. In *Proceedings of the Design, Automation and Test in Europe Conference*. 604–611.

Hamzaoglu, I. and Patel, J. H. 1998. Test set compaction algorithms for combinational circuits. In *Proceedings of the International Conference on CAD*. 283–289.

Hellebrand, S., Liang, H.-G., and Wunderlich, H.-J. 2000. A mixed-mode BIST scheme based on reseeding of folding counters. In *Proceedings of the International Test Conference*. 778–784.

Hsu, F. F., Butler, K. M., and Patel, J. H. 2001. A case study on the implementation of illinois scan architecture. In *Proceedings of the International Test Conference*. 538–547.

Iyengar, V., Chakrabarty, K., and Murray, B. T. 1999. Deterministic built-in pattern generation for sequential circuits. *J. Elect. Test: Theory Appl. 15*, 97–115.

Jas, A., Ghosh-Dastidar, J., and Touba, N. A. 1999. Scan vector compression/decompression using statistical coding. In *Proceedings of the VLSI Test Symposium*. 114–120.

Jas, A. and Touba, N. A. 1998. Test vector decompression via cyclical scan chains and its application to testing core-based design. In *Proceedings of the International Test Conference*. 458–464.

Kajihara, S. and Miyase, K. 2001. On identifying don't care inputs of test patterns for combinational circuits. In *Proceedings of the International Conference on CAD*. 364–369.

Koenemann, B., Barnhart, C., B.Keller, Snethen, T., Farnsworth, O., and Wheater, D. 2001. A SmartBIST variant with guaranteed encoding. In *Proceedings of the Asian Test Symposium*. 325–330.

RAJSKI, J., TYSZER, J., KASSAB, M., MUKHERJEE, N., THOMPSON, R., TSAI, H., HERTWIG, A., TAMARAPALLI, N., MRUGALSKI, G., EIDE, G., AND QIAN, J. 2002. Embedded deterministic test for low-cost manufacturing test. In *Proceedings of the International Test Conference*. 301–310.

REDA, S. AND ORAILOGLU, A. 2002. Reducing test application time through test data mutation encoding. In *Proceedings of the Design, Automation and Test in Europe Conference*. 387–393.

REDDY, S. M., MIYASE, K., KAJIHARA, S., AND POMERANZ, I. 2002. On test data volume reduction for multiple scan chain design. In *Proceedings of the VLSI Test Symposium*. 103–108.

SALOMON, D. 2000. *Data Compression: The Complete Reference*. Springer-Verlag New York, Inc., New York, NY.

SCHAFER, L., DORSCH, R., AND WUNDERLICH, H.-J. 2002. Respin++—Deterministic embedded test. In *Proceedings of the European Test Workshop*. 37–44.

TOUBA, N. A. AND MCCLUSKEY, E. J. 1996. Altering a pseudo-random bit sequence for scan based BIST. In *Proceedings of the International Test Conference*. 167–175.

VOLKERINK, E. H., KHOCHE, A., AND MITRA, S. 2002. Packet-based input test data compression techniques. In *Proceedings of the International Test Conference*. 154–163.

VRANKEN, H., WAAYERS, T., FLEURY, H., AND LELOUVIER, D. 2001. Enhanced reduced pin-count test for full-scan designs. In *Proceedings of the International Test Conference*. 738–747.

WOLFF, F. G. AND PAPACHRISTOU, C. 2002. Multiscan-based test compression and hardware decompression using LZ77. In *Proceedings of the International Test Conference*. 331–339.

WUNDERLICH, H.-J. AND KIEFER, G. 1996. Bit-flipping BIST. In *Proceedings of the International Conference on Computer-Aided Design*. 337–343.

Received March 2003; revised July 2003; accepted August 2003