TABLE VI
EXPERIMENTAL RESULTS

	SAT_4v		SAT		FAN(de)		FAN(long)		FAN(de)+SAT		FAN(long)+SAT	
Circuit	ab.	time	ab.	time	ab.	time	ab.	time	ab.	time	ab.	time
b14	0	1:00m	0	0:19m	107	0:11m	7	1:42m	0	0:12m	0	1:24m
b15	0	1:16m	0	0:24m	619	0:11m	318	26:25m	0	0:18m	0	23:02m
b17	0	4:36m	0	2:22m	1382	1:41m	622	56:54m	0	1:58m	0	50:55m
b18	0	27:33m	0	22:30m	740	19:16m	270	41:40m	0	20:34m	0	39:32m
b20	0	2:30m	0	0:56m	225	0:35m	42	7:46m	0	0:44m	0	6:46m
b21	0	2:41m	0	0:59m	198	0:39m	43	6:48m	0	0:43m	0	6:18m
b22	0	3:49m	0	1:35m	284	1:07m	52	9:34m	0	1:14m	0	9:58m
p44k	0	2:18h	0	26:01m	12	4:58m	0	4:59m	0	5:55m	0	8:03m
p49k	-	-	77	1:43h	3770	2:06h	162	2:38h	74	1:55h	2	2:49h
p80k	1	42:58m	0	9:43m	218	34:55m	21	39:13m	0	39:38m	0	57:20m
p88k	0	11:41m	0	9:33m	195	9:13m	38	12:40m	0	10:27m	0	18:56m
p99k	0	8:41m	0	6:50m	1398	6:02m	512	1:16h	0	7:25m	0	1:02h
p177k	941	10:28h	0	1:19h	270	16:06m	47	20:03m	0	19:03m	0	31:23m
p462k	129	3:31h	6	2:16h	1383	1:34h	423	2:07h	0	1:51h	0	3:07h
p565k	0	2:23h	0	2:23h	1391	2:21h	85	2:47h	0	2:47h	0	4:29h
p1330k	1	4:58h	1	5:05h	889	4:15h	144	4:28h	0	5:00h	0	7:30h

VI. CONCLUSION

Using structural information while transforming large industrial circuits into a CNF significantly reduces the size of the SAT instances for ATPG. As a consequence, the SAT solver needs less resources, which boosts the performance of the SAT-based ATPG approach. Furthermore, the integration of the SAT-based engine into the industrial ATPG framework of NXP Semiconductors improves the overall performance of the framework and leads to a fast and robust ATPG system.

REFERENCES

[1] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Trans. Comput.*, vol. C-30, no. 3, pp. 215–222, Mar. 1981.

[2] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. Comput.*, vol. C-32, no. 12, pp. 1137–1144, Dec. 1983.

[3] M. H. Schulz, E. Trischler, and T. M. Sarfert, "SOCRATES: A highly efficient automatic test pattern generation system," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 7, no. 1, pp. 126–137, Jan. 1988.

[4] I. Hamzaoglu and J. Patel, "New techniques for deterministic test pattern generation," *J. Electron. Test.—Theory and Applications*, vol. 15, no. 1/2, pp. 63–73, Aug.–Oct. 1999.

[5] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 11, no. 1, pp. 4–15, Jan. 1992.

[6] P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "Combinational test generation using satisfiability," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 9, pp. 1167–1176, Sep. 1996.

[7] J. Shi, G. Fey, R. Drechsler, A. Glowatz, F. Hapke, and J. Schlöffel, "PASSAT: Efficient SAT-based test pattern generation for industrial circuits," in *Proc. IEEE Annu. Symp. VLSI*, 2005, pp. 212–217.

[8] J. Shi, G. Fey, R. Drechsler, A. Glowatz, J. Schlöffel, and F. Hapke, "Experimental studies on SAT-based test pattern generation for industrial circuits," in *Proc. Int. Conf. ASIC*, 2005, pp. 967–970.

[9] J. Marques-Silva and K. Sakallah, "GRASP: A search algorithm for propositional satisfiability," *IEEE Trans. Comput.*, vol. 48, no. 5, pp. 506–521, May 1999.

[10] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in *Proc. Des. Autom. Conf.*, 2001, pp. 530–535.

[11] E. Goldberg and Y. Novikov, "BerkMin: A fast and robust SAT-solver," in *Proc.* Des. Autom. Test Eur., 2002, pp. 142–149.

[12] N. Eén and N. Sörensson, "An extensible SAT-solver," in *Proc. SAT*, 2003, vol. 2919, pp. 502–518.

[13] J. Marques-Silva and K. Sakallah, "Robust search algorithms for test pattern generation," Dept. Informatics, Tech. Univ. Lisbon, Lisbon, Portugal, Tech. Rep. RT/02/97, Jan. 1997.

[14] P. Tafertshofer, A. Ganz, and K. Antreich, "IGRAINE—An implication graph-based engine for fast implication, justification, and propagation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 8, pp. 907–927, Aug. 2000.

[15] E. Gizdarski and H. Fujiwara, "SPIRIT: A highly robust combinational test generation algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 12, pp. 1446–1458, Dec. 2002.

[16] J. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM J. Res. Develop.*, vol. 10, no. 4, pp. 278–291, Jul. 1966.

[17] R. T. Stanion, "Circuit synthesis verification method and apparatus," U.S. Patent 6 056 784, May 2, 2000.

[18] G. Fey, J. Shi, and R. Drechsler, "Efficiency of multi-valued encoding in SAT-based ATPG," in *Proc. Int. Symp. Multiple-Valued Logic*, 2006, pp. 25–30.

[19] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Univ. California, Berkeley, CA, Tech. Rep. No. UCB/ERL M92/41, 1992.

[20] D. Tille, G. Fey, and R. Drechsler, "Instance generation for SAT-based ATPG," in *Proc. IEEE Workshop Des. Diagnostics Electron. Circuits Syst.*, 2007, pp. 153–156.

Test Data Compression Based on Variable-to-Variable Huffman Encoding With Codeword Reusability

Xrysovalantis Kavousianos, Emmanouil Kalligeros, and Dimitris Nikolos

***Abstract*—A new statistical test data compression method that is suitable for IP cores of an unknown structure with multiple scan chains is proposed in this paper. Huffman, which is a well-known fixed-to-variable code, is used in this paper as a variable-to-variable code. The precomputed test set of a core is partitioned into variable-length blocks, which are, then, compressed by an efficient Huffman-based encoding procedure with a limited number of codewords. To increase the compression ratio, the same codeword can be reused for encoding compatible blocks of different sizes. Further compression improvements can be achieved by using two very simple test set transformations. A simple and low-overhead decompression architecture is also proposed.**

***Index Terms*—Embedded testing techniques, Huffman encoding, intellectual property (IP) cores, test data compression.**

Manuscript received October 12, 2007; revised January 22, 2008. This paper was recommended by Associate Editor K. Chakrabarty.

X. Kavousianos is with the Department of Computer Science, University of Ioannina, 45110 Ioannina, Greece (e-mail: kabousia@cs.uoi.gr).

E. Kalligeros is with the Department of Information and Communication Systems Engineering, University of the Aegean, 83200 Samos, Greece (e-mail: kalliger@aegean.gr).

D. Nikolos is with the Department of Computer Engineering and Informatics, University of Patras, 26500 Patras, Greece (e-mail: nikolosd@cti.gr).

Digital Object Identifier 10.1109/TCAD.2008.923100

I. Introduction

The high complexity of contemporary *systems-on-a-chip* (SoCs) makes their testing an increasingly challenging task. The quantity of test data rapidly increases, while, at the same time, the inner nodes of dense SoCs become less accessible from the external pins. The testing problem is further exacerbated by the use of *intellectual property* (IP) cores, since their structure is often hidden from the system integrator. In such cases, no modifications can be applied to the cores or their scan chains, whereas neither automatic test pattern generation nor fault simulation tools can be used. Only precomputed test sets are provided by the core vendors, which should be applied to the cores during testing.

Several methods have been proposed to minimize the test data volume of unknown-structure IP cores. The approaches in [3] and [23] embed the precomputed test vectors in long on-chip generated pseudorandom sequences, significantly reducing, this way, the test data volume. To minimize both test data volume and test application time, many methods directly encode the test sets by using various compression codes [4]–[7], [9], [12], [14]–[16], [26], [27], [29], [30]. Compression can be also performed on the difference vectors, but expensive cyclical shift registers should be incorporated in the system, or the scan chains of other cores must be reused [13]. The test application time can further be reduced by exploiting the multiple scan chains of the cores [1], [2], [8], [18]–[22], [24], [25], [28], [30]–[32]. There are also techniques that are based on dictionaries, whereas other techniques require the preexistence of various modules in the SoC (e.g., arithmetic modules and embedded processors). Due to the high hardware overhead of the former techniques and the embedded-module requirement of the latter techniques, we do not consider them further in this paper.

A statistical compression method that is based on the Huffman encoding of variable-length test set blocks is proposed in this paper. The encoding is performed in a selective manner, i.e., some blocks of the test set are left unencoded. Apart from the variable-to-variable nature of the proposed approach, the generated codewords are reusable in the sense that they can encode compatible blocks of different sizes. Two simple transformations are also presented to improve the statistical properties of the test set before compression. The proposed decompression architecture generates the decoded variable-length blocks in parallel, exploiting, this way, the test-application-time advantages that are offered by the existence of multiple scan chains in a core. Moreover, the decompressors are properly designed, so their hardware is kept low.

The remainder of this paper is organized as follows. Section II describes the proposed method, Section III presents the decompression architecture, and Section IV provides experimental results and comparisons. This paper is concluded in Section V.

II. Proposed Method

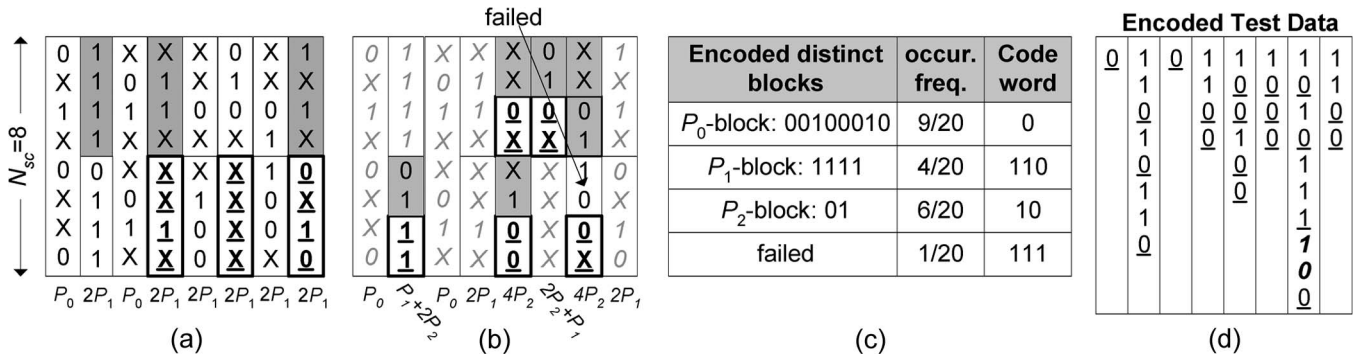
A. Encoding–Decoding Method

Let T be the test set of an IP core. T, which is of size $|T|$ (in bits), is partitioned into $|T|/l$ blocks of size l, hereafter called *test set parts* or, simply, *parts*. Each test set part consists of specified (0, 1) and unspecified bits (x) and is compatible with a number of fully specified blocks that are generated by substituting its x bits with all possible combinations of 0s and 1s. According to the selective Huffman coding [14], the m fully specified blocks that are compatible with most of the test set parts are Huffman encoded. We call these m fully specified blocks *distinct blocks*. Each test set part is either encoded by the codeword of a compatible distinct block or remains unencoded. If a test set part is compatible with more than one of the m encoded distinct blocks, the codeword of the most frequently occurring block is used for its encoding.

Assuming that m remains constant, the effectiveness of the selective Huffman coding is affected by block size l in two contradictory ways. As l increases, the test set is partitioned into fewer and larger parts, and, thus, the total number of codewords that are required for encoding the original test set decreases. As a result, better compression can be achieved. At the same time, however, the compression ratio is negatively affected by the fact that more test set parts remain unencoded (since, as block size l increases, fewer parts are compatible with the m encoded distinct blocks). Decreasing block sizes lead to exactly opposite behaviors. Consequently, to maximize the efficiency of the selective Huffman coding, the volume of the unencoded test set parts must be minimized, while, at the same time, the total number of codewords (and, hence, the total size of the encoded data) must be kept low. To achieve this goal, we can take advantage of the well-known characteristic that, in every test set, there are regions with many defined bits (i.e., densely specified) and regions with many x bits (i.e., sparsely specified). Densely specified regions are the main sources of unencoded data, and, therefore, their compression is favored by the usage of small distinct blocks. On the other hand, sparsely specified regions are more efficiently compressed using large distinct blocks, since, this way, many test set parts, despite their big size, are compatible with the encoded distinct blocks due to the great number of x bits that they contain. From the above analysis, we deduce that compression can be improved if the test sets are partitioned into variable-length parts, which means that variable-length distinct blocks should be encoded.

In the proposed approach, as a test set part, we consider a whole slice or a slice portion (the test bits that correspond to the ith scan cell of every scan chain constitute the ith slice of a test cube). Each Huffman codeword encodes a distinct block of a specific size. When a codeword is decoded, the corresponding distinct block is generated in parallel (after codeword identification), exploiting, this way, the parallelism that is offered by the multiple scan chains of a core. Note that, to select the m distinct blocks that will be encoded, the x values of the test set should first be replaced by constant binary values (i.e., 0s and 1s). To determine the proper x-bit assignment so that the occurrence frequencies of the encoded distinct blocks will be as skewed as possible, we use an extension of the second algorithm (Alg2) that was proposed in [14]. According to the original algorithm in [14], the two most frequently occurring test set parts that are compatible are merged, forming a more specified and frequently occurring part than its predecessors. The same procedure is iteratively repeated until no further merging is possible. If, after the parts' merging, there are any remaining x bits, they are filled with random values. This way, the various test set parts are gradually transformed into fully specified blocks, and the m most frequently occurring ones are selected for encoding (i.e., they are the encoded distinct blocks). The extensions on the original algorithm will shortly become clear.

Initially, the test set is partitioned into slices according to the scan-chain structure of the core (slices are also called P_0-parts). All P_0-parts are considered for the selection of the first distinct blocks that will be encoded (i.e., P_0-blocks of size equal to the number of scan chains N_{sc}). The first selected P_0-block is the block that is compatible with most of the P_0-parts, the second selected P_0-block is the block that is compatible with most of the P_0-parts that have not been encoded by the first P_0-block, etc., (the selected P_0-blocks are derived by merging the most frequently occurring P_0-parts, as explained above). When a number of P_0-blocks have been selected, each of the unencoded P_0-parts is partitioned into two portions of equal size, which are called P_1-parts. Again, a number of distinct P_1-blocks are selected, the unencoded P_1-parts are partitioned into P_2-parts, some P_2-blocks are selected, the unencoded P_2-parts are partitioned into P_3-parts, and so on. Finally, each of the $P_0, \ldots, P_{\max}$-blocks is compatible with some $P_0, \ldots, P_{\max}$-parts, respectively, where the max value is determined



Encoded distinct blocks	occur. freq.	Code word
P_0-block: 00100010	9/20	0
P_1-block: 1111	4/20	110
P_2-block: 01	6/20	10
failed	1/20	111

Fig. 1. Proposed encoding example.

by the designer ($P_{\max}$-parts are called primitive parts). The size of P_i-parts is either $\lceil N_{\rm sc}/2^i \rceil$ or $\lfloor N_{\rm sc}/2^i \rfloor$ bits (i.e., if P_{i-1}-parts cannot be partitioned into two equal P_i-parts, then half of the P_i-parts are 1 bit shorter than the rest).

The above-described procedure implies that, during the encoding of P_i-parts by P_i-blocks, the most sparsely specified parts are encoded, whereas the most densely specified ones are partitioned into P_{i+1}-parts (since densely specified parts have small occurrence frequencies and are not selected during the parts-merging procedure). As a result, the large blocks, which are derived by the merging of large test set parts, contain many x values. Instead of randomly replacing these values as in [14], we initially leave them unspecified, and we utilize them later to increase the selected blocks' encoding ability (and, hence, to further skew their occurrence frequencies). This is done by using the codewords of the already-selected blocks for encoding smaller test set parts (codeword reusability). More formally, a P_j-part is allowed to be encoded by the codeword of a larger P_i-block $(i < j)$, provided that the first ($\lceil N_{\rm sc}/2^j \rceil$ or $\lfloor N_{\rm sc}/2^j \rfloor$) bits of the P_i-block are compatible with the P_j-part (i.e., the smaller parts are always compared against the upper segments of the already-selected larger blocks). Thus, the upper segments of the selected P_i-blocks are initially left partially unspecified and will be defined later during the encoding of P_{i+1}, $P_{i+2}, \ldots, P_{\max}$-parts. It is obvious that, during the decoding process, only the necessary bits of the P_i-block will be generated, whereas the rest will be discarded.

Given that the codeword of a P_0, $P_1, \ldots, P_{\max -1}$-block can be used for encoding test set parts of various sizes, every such codeword must uniquely specify the actual encoded test set part during the decoding process. To keep the compressed data volume low, no information is stored about the size of the part encoded by a block's codeword. Instead, the encoding process is constrained according to the following condition.

Condition 1: "A P_i-block codeword can be used for encoding a smaller P_j-part (where $j > i$), if the P_j-part *is not the first* part of a larger test set part."

For example, the codeword of a P_0-block can be used for encoding a P_0-part (i.e., a whole slice; all of its bits will be generated during decoding). It can be also used for encoding **the second** P_1-part that has resulted from the partitioning of a P_0-part (the first half of the P_0-block will be decoded). Similarly, it can be used for the encoding of **the second** P_2-part of a P_1-part (the first quarter of the P_0-block will be decoded), and so on. Note, however, that a P_0-block cannot be utilized for the encoding of the first P_1-part of a P_0-part or of the first P_2-part of a P_1-part, even if its corresponding segments are compatible with those parts.

To improve compression, every test set part is encoded before it is partitioned into smaller parts, or, in other words, every selected distinct block is used for the encoding of as many large test set parts as possible. This is why P_0-parts are encoded first, followed by P_1-parts, P_2-parts, etc. Furthermore, we do not allow the encoding of a P_{i+1}-part, where $i \in [0, \max -1]$, by the codeword of a larger P_0, $P_1, \ldots, P_i$-block before the beginning of the encoding process of P_{i+1}-parts. For example, the encoding of a P_3-part by a P_0-block codeword before the selection of P_1 and P_2-blocks is not allowed. The reason is that this encoding may prevent a possible subsequent encoding of a (larger) P_1 or P_2-part that includes the aforementioned P_3-part.

The encoding process of P_i-parts stops, and that of P_{i+1}-parts begins when the total number of bits of the P_i-parts (TestBits$_i$) that are compatible with the next P_i-block that will be selected is fewer by a factor F than the bits of the P_{i+1}-parts (TestBits$_{i+1}$) that can be encoded by: 1) the codewords of the already-chosen P_0, $P_1, \ldots, P_i$-blocks and 2) the codeword of the first P_{i+1}-block that will be selected (i.e., when TestBits$_{i+1} \geq F \cdot$ TestBits$_i$). Factor F is used for achieving a balanced selection between large and small blocks [17]. The F value that maximizes the compression ratio for each circuit (which is a small positive integer greater than 1) can easily be tracked down, since the runtime of the encoding method is very short, and the set of possible F values is very small.

When all P_0, $P_1, \ldots, P_{\max}$-blocks have been selected, some of the $P_{\max}$-parts remain unencoded. Such parts are labeled as failed, and a separate Huffman codeword is assigned to all of them. In the compressed test set, these $P_{\max}$-parts are embedded unencoded, preceded by the aforementioned codeword. The Huffman tree is constructed when all P_i-blocks (where $0 \leq i \leq \max$) have been selected so that all occurrence frequencies are known. We illustrate the above-described process with an example.

Example 1: Consider a circuit with $N_{\rm sc} = 8$ scan chains and let $\max = 2$. The 64-bit test set, as shown in Fig. 1(a), is initially partitioned into P_0-parts, i.e., whole slices [ignore, for now, the partitioning in Fig. 1(a)]. Since all P_0-parts appear only once in the test set, the first two P_0-parts that are compatible (i.e., 0x1x0xx0 and x01xx01x, which are the first and third P_0-parts, respectively) are merged. The resulting part (i.e., 001x0010) cannot be merged with any of the rest P_0-parts, and, thus, it constitutes the first P_0-block. Assume now that factor F has been set to such a value, that the encoding of P_0-parts (i.e., the selection of P_0-blocks) has to stop, and the encoding of P_1-parts (i.e., the selection of P_1-blocks) has to begin. Thus, the unencoded P_0-parts are partitioned into P_1-parts [see Fig. 1(a)]. Before selecting the first P_1-block, all P_1-parts that are compatible with the first half of the selected P_0-block and, at the same time, satisfy Condition 1 (i.e., the second P_1-parts of all P_0-parts) are encoded using this block [these P_1-parts are boldfaced and underlined in Fig. 1(a)]. Note that, by using the first half of P_0-block 001x0010 for encoding P_1-part 0x10, the x bit of the P_0-block is set to 0. Then, the first P_1-block has to be selected. Since all the remaining P_1-parts appear only once, the first two P_1-parts that are compatible (i.e., 1111 and x11x) are merged. The

resulting part (i.e., 1111) can be also merged with P_1-part 1x1x. Thus, 1111 constitutes the first selected P_1-block, and the P_1-parts that are encoded using this block are highlighted in Fig. 1(a). Assume, again, that, after this step, inequality $\text{TestBits}_{i+1} \geq F \cdot \text{TestBits}_i$ is true, and, as a result, the encoding of P_1-parts stops, and the encoding of P_2-parts begins. After partitioning all unencoded P_1-parts into P_2-parts [see Fig. 1(b)], all P_2-parts that satisfy Condition 1 and are compatible with the first quarter of the selected P_0-block (i.e., 00) and the first half of the selected P_1-block (i.e., 11) are encoded using the corresponding blocks [they are boldfaced and underlined in Fig. 1(b), whereas every already encoded part is shown in light gray]. Then, P_2-part 01, which appears three times, is merged first with P_2-part xx that appears twice, and next with P_2-part x1. The resulting part (i.e., 01) is the first selected P_2-block, and the P_2-parts that are encoded by this block are highlighted in Fig. 1(b). One P_2-part (i.e., 10) is left unencoded, as shown in Fig. 1(b), and is labeled as failed. Note that, when the encoding process is complete, the test set has been partitioned into a total of 20 P_0, P_1, and P_2-parts. In Fig. 1(c), the selected distinct blocks, their occurrence frequencies, and the respective Huffman codewords that are generated by constructing the corresponding Huffman tree are reported. Finally, in Fig. 1(d), the encoded test set is shown, where the last bit of each codeword is underlined, and the unencoded test set part (i.e., 10) is boldfaced and italicized. ■

For decoding the test data, a counter s with a size that is equal to max bits is used. s counts from 0 to $2^{\max}-1$ and points to the next primitive part of a slice that has not yet been decoded (every slice consists of $2^{\max}$ primitive parts, and its decoding begins from the first, i.e., 0, primitive part and continues with primitive parts $1, 2, \ldots, 2^{\max}-1$). During codeword decompression, the largest possible test set part that can be decoded, independently of the received codeword, is first determined. This is a P_L-part if s is exactly divided by $2^{\max-L}$ but is not divided by $2^{\max-L+1}$, or, equivalently, if the volume of consecutive least significant bits of s that are 0 is equal to $\max-L$. In addition, the size of the distinct P_i-block that corresponds to the received codeword is determined. If it is equal or smaller than the size of the aforementioned P_L-part, then the actual test set part that will be decoded is identical to the whole P_i-block. Otherwise, the upper segment of the P_i-block whose size is equal to that of the P_L-part is decoded. When a whole slice has been generated, it is loaded into the scan chains.

B. Statistical Improvement of Test Data

In this section, we propose two simple and low-overhead test set transformations, which can optionally be applied before compressing the test data, to improve their statistical properties (no structural information of the core is required). This is achieved by increasing the difference between 0s and 1s in the test set. Specifically, all the bits of selected scan chains (transformation $\mathbf{T}_1$) and/or the values of selected scan cells (transformation $\mathbf{T}_2$) can be inverted. The transformed test set is, then, compressed, and, during decompression, the original test set is restored by removing, on the fly, the transformations. For example, suppose that, for a test set, the 0-bit volume is higher than the 1-bit volume. According to $\mathbf{T}_1$, the scan chains with more 1s than 0s, considering all test vectors, can be inverted to favor the 0s count. Similarly, $\mathbf{T}_2$ can be used for inverting a predefined number of scan cells with the highest difference of 0-bits from 1-bits for all test vectors.

III. Proposed Architecture

The proposed decompression architecture is presented in Fig. 2. The Input Buffer receives the encoded data in parallel from the automatic test equipment (ATE) with the ATE_CLK frequency and serially shifts them into the Huffman FSM (finite-state machine) with the

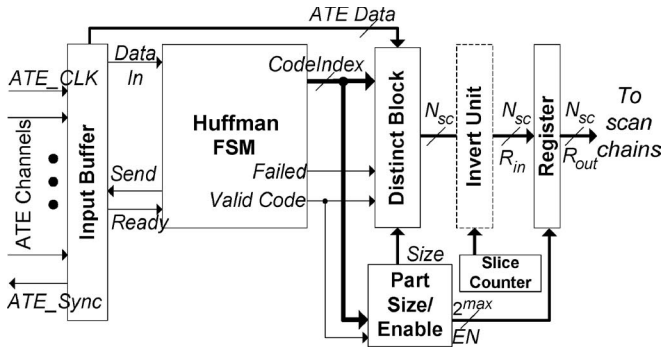


Fig. 2. Proposed decompression architecture.

system clock frequency. Upon the recognition of a codeword by the Huffman FSM, $Valid\,Code$ is set to 1, whereas the value of bus $CodeIndex$ indicates which codeword has been received. When the received codeword corresponds to an unencoded test set part, signal $Failed$ is also set to 1. The flow control between the buffer and the Huffman FSM is performed via signals $Ready$ and $Send$, whereas ATE synchronization can be achieved using a first-in first-out buffer between the decoder and the ATE [10].

The Distinct Block unit receives $CodeIndex$ and returns the proper portion of the distinct block that is encoded by the current codeword. Specifically, if the part that will be decoded has a size that is equal to that of 2^q primitive parts (this size is provided by bus $Size$ of the Part Size/Enable unit), then, at the outputs of the Distinct Block unit, $2^{\max-q}$ copies of the first $2^q \cdot |P_{\max}|$ bits of the encoded distinct block are generated ($|P_{\max}|$ is the primitive-parts size in bits). The enable signals that were activated by the Part Size/Enable unit ensure that only the proper bit positions of the register will be loaded with the generated data. Similarly, in case of a failed $P_{\max}$-part, its bits, which are directly received from the ATE (through the Input Buffer), are repeated $2^{\max}$ times at the outputs of the Distinct Block unit and are loaded in the proper bit positions of the register.

According to the proposed approach, even if a codeword corresponds to a P_i-block, it may be utilized for encoding $P_{i+1}, P_{i+2}, \ldots, P_{\max}$-parts. The proper decoded-part size is determined in the Part Size/Enable unit by a small combinational logic that examines the value of counter s (see Section II). If the volume of consecutive least significant bits of s that are 0 is equal to q, then the largest part that can be decoded is a $P_{\max-q}$-part. If the received codeword encodes a P_i-block and $\max-q \leq i$, then the whole P_i-block is decoded. Otherwise, the first bits of the P_i-block that form a $P_{\max-q}$-part are decoded. When s reaches $2^{\max}-1$, a whole slice has been loaded in the Register, and, then, it is transferred into the scan chains.

When transformations $\mathbf{T}_1$ and/or $\mathbf{T}_2$ are used, the Invert unit is placed between the Distinct Block unit and the Register. It consists of at most N_{sc} gates (i.e., one for each scan chain), which can be either inverters (for inverting all bits that enter a scan chain, i.e., $\mathbf{T}_1$) or XOR/XNOR gates (for selectively inverting specific bits that enter a scan chain, i.e., $\mathbf{T}_2$). If $\mathbf{T}_2$ is applied, the invert unit also incorporates a slice number decoding logic.

IV. Evaluation and Comparisons

The proposed compression method was implemented in the C programming language, and experiments were performed using the dynamically compacted Mintest test sets [11] for stuck-at faults. The runtime for each experiment is a few seconds.

In Table I, we present the test data compression results (in bits) of the proposed method for 16 and 128 scan chains, 24 selected distinct variable-length blocks, and primitive-parts size that is equal to 8 bits.

TABLE I
RESULTS OF THE PROPOSED ENCODING (IN BITS)

Circuit	Mintest (# bits)	16 Scan Chains			128 Scan Chains		
		No Transf.	T_1 + 50 T_2	T_1 + 100 T_2	No Transf.	T_1 + 50 T_2	T_1 + 100 T_2
s5378	23754	9776	9183	8955	9277	**7611**	7611
s9234	39273	15792	14514	14049	14493	**12765**	12765
s13207	165200	24039	23053	21623	16193	15092	**14578**
s15850	76986	22113	20685	20016	18611	17354	**16562**
s38417	164736	**58663**	61241	59748	59492	64963	63833
s38584	199104	60291	60101	59809	55612	55965	**55353**

TABLE II
COMPARISONS AGAINST METHODS FOR CORES WITH MULTIPLE SCAN CHAINS

Cir-cuit	Select. Huff. (# bits)	[18] (# bits)	[24] (# bits)	[28] (# bits)	Prop. (# bits)	Red. %* of prop. over:			
						Select. Huff.	[18]	[24]	[28]
s5378	11433	9247	14220	-	7611	33.4	17.7	46.5	-
s9234	19168	15722	30144	-	12765	33.4	18.8	57.7	-
s13207	28328	18153	20988	74423	14578	48.5	19.7	30.5	80.4
s15850	26873	19313	25140	26021	16562	38.4	14.2	34.1	36.4
s38417	70954	58706	85225	45003	58663	17.3	0.1	31.2	-30.4
s38584	71315	57801	57120	73464	55353	22.4	4.2	3.1	24.7

*Red. %=[1-(# bits of prop./ # bits of: Select. Huff., [18], [24], [28])] ·100

A few values of F were examined for each circuit (i.e., $1 \leq F \leq 15$), and the best result is shown in Table I (a thorough parameter analysis is provided in [17]). In almost all cases, compression improves as the number of scan chains increases. In addition, compared to the "No Transformations" case (i.e., the "No Transf." columns), almost always, we get better compression when transformations $\mathbf{T}_1$ and $\mathbf{T}_2$ for 50 cells are applied ("$\mathbf{T}_1 + 50\,\mathbf{T}_2$"); whereas, most of the times, a further increase in the number of cells that are inverted by $\mathbf{T}_2$ does not provide significant improvements.

We next compare the proposed approach against methods that target unknown-structure IP cores with multiple scan chains. Note that we do not compare against: 1) the approach in [8], since several conditions have to be satisfied by a core that is near the circuit under test so that the former can be used as a decompressor, and 2) methods that provide results for different test sets from those used in our experiments. In Table II, comparisons against the selective Huffman approach [14] [reimplemented here for multiple (i.e., 64) scan chains], [18], [24], and [28] are presented. For the selective Huffman approach, the number of selected fixed-length distinct blocks was set to 24, and three different block sizes that are equal to 8, 16, and 32 bits were examined. The best result for every circuit is reported in the second column in Table II. The third, fourth, and fifth columns present the best results of [18], [24], and [28], respectively. Note that, for the approach in [28], aside from the test data that are shown in column 4, an additional significant quantity of control data should be stored in the ATE. However, these data have not been reported by the authors in [28]. In the sixth column, we provide the best results of the proposed method. Finally, the seventh to tenth columns report the reduction percentages of the proposed method over the other methods. As we can see, in all cases, except for one (i.e., s38417 in [28], for which no control data have been reported), the proposed technique performs better than the other methods.

In Table III, we present the compressed-data reduction percentages of the proposed method against techniques that are applicable to cores with a single scan chain. The compression that was achieved by the proposed approach is higher than the compression of the rest of the methods, except for the s38584 case in [15] and the s38417 case in [16], which are marginally higher.

To assess the hardware overhead of the proposed method, we synthesized three different decompressors for the test set of s9234 by

TABLE III
PROPOSED METHOD VERSUS METHODS FOR SINGLE-SCAN-CHAIN CORES (REDUCTION %)

Circuit	[4]	[5]	[6]	[7]	[9]	[14]	[15]	[16]	[26]	[27]	[29]	[30]
s5378	-	-	34.9	38.4	33.5	28.6	18.7	20.7	33.3	30.7	47.9	27.6
s9234	42.6	43.3	40.9	42.4	38.4	29.0	17.7	18.6	39.9	38.0	46.7	28.1
s13207	65.0	58.5	55.3	52.8	46.5	61.6	20.7	39.9	51.4	49.5	61.6	40.4
s15850	59.3	45.8	37.0	36.3	32.9	36.7	12.5	22.7	32.8	34.1	47.1	25.1
s38417	36.3	35.6	9.7	37.2	23.6	13.1	0.2	-0.2	9.7	0.6	20.1	4.0
s38584	46.8	38.4	28.5	28.9	26.3	22.6	-0.3	8.9	25.0	26.1	35.9	12.0

TABLE IV
HARDWARE OVERHEAD OF MULTIPLE-SCAN-CHAIN METHODS (NUMBER OF GATE EQUIVALENTS)

Proposed			Selective Huffman			[18]
No Transf.	T_1 + 50T_2	T_1 + 100T_2	*BS*=8	*BS*=16	*BS*=32	
606	646	670	445	514	635	582

applying: 1) no transformations; 2) $\mathbf{T}_1$ and $\mathbf{T}_2$ for 50 selected cells; and 3) $\mathbf{T}_1$ and $\mathbf{T}_2$ for 100 selected cells. We also synthesized the decompressor of the implemented parallel selective Huffman approach with (fixed) block size (BS) that is equal to 8, 16, and 32 bits. In all experiments, the number of scan chains was set to 64, and the number of selected distinct blocks to 24. The results are shown in the first six columns in Table IV in gate equivalents (where a gate equivalent corresponds to a two-input NAND gate). Compared to the well-known selective Huffman approach, the proposed method imposes slightly higher hardware overhead. The hardware overhead in [18] for 24 selected cells is 582 gate equivalents (see the seventh column), and, thus, it is very close to the hardware overhead of the proposed method. As far as the approaches in [24] and [28] are concerned, their hardware overhead is low, but, as shown earlier, their compression ratios are much smaller than those of the proposed method. Finally, the hardware overhead of the single-scan-chain methods in [4], [6], [9], [14], [15], and [30] is between 125 and 769 gate equivalents. However, these techniques require much longer test application times and much greater test data storage.

V. CONCLUSION

In this paper, we have proposed an efficient compression method that is suitable for multiple-scan-chain IP cores of an unknown structure. Huffman was used as a variable-to-variable code for compressing variable-length blocks. To increase the compression ratio, codeword reusability and two transformations that improve the statistical properties of the original test set were introduced. Finally, a simple and low-overhead architecture was proposed to perform the decompression.

REFERENCES

[1] N. Badereddine *et al.*, "Power-aware test data compression for embedded IP cores," in *Proc. ATS*, 2006, pp. 5–10.
[2] K. Balakrishnan, S. Wang, and S. Chakradhar, "PIDISC: Pattern independent design independent seed compression technique," in *Proc. 19th VLSID*, 2006, pp. 811–817.
[3] K. Chakrabarty, B. Murray, and V. Iyengar, "Deterministic built-in test pattern generation for high-performance circuits using twisted-ring counters," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 5, pp. 633–636, Oct. 2000.
[4] A. Chandra and K. Chakrabarty, "System-on-a-chip test-data compression and decompression architectures based on Golomb codes," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 3, pp. 355–368, Mar. 2001.
[5] A. Chandra and K. Chakrabarty, "Test data compression and decompression based on internal scan chains and Golomb coding," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 6, pp. 715–722, Jun. 2002.

[6] A. Chandra and K. Chakrabarty, "A unified approach to reduce SOC test data volume, scan power and testing time," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 3, pp. 352–363, Mar. 2003.
[7] A. Chandra and K. Chakrabarty, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes," *IEEE Trans. Comput.*, vol. 52, no. 8, pp. 1076–1088, Aug. 2003.
[8] R. Dorsch and H.-J. Wunderlich, "Tailoring ATPG for embedded testing," in *Proc. ITC*, 2001, pp. 530–537.
[9] P. Gonciari, B. Al-Hashimi, and N. Nicolici, "Variable-length input Huffman coding for system-on-a-chip test," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 6, pp. 783–796, Jun. 2003.
[10] P. Gonciari, B. Al-Hashimi, and N. Nicolici, "Synchronization overhead in SOC compressed test," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 1, pp. 140–152, Jan. 2005.
[11] I. Hamzaoglu and J. Patel, "Test set compaction algorithms for combinational circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 8, pp. 957–963, Aug. 2000.
[12] V. Iyengar, K. Chakrabarty, and B. Murray, "Deterministic built-in pattern generation for sequential circuits," *J. Electron. Test.—Theory and Applications*, vol. 15, no. 1/2, pp. 97–114, Aug./Oct. 1999.
[13] A. Jas and N. A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based designs," in *Proc. ITC*, 1998, pp. 458–464.
[14] A. Jas, J. Ghosh-Dastidar, M.-E. Ng, and N. Touba, "An efficient test vector compression scheme using selective Huffman coding," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 6, pp. 797–806, Jun. 2003.
[15] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Multilevel Huffman coding: An efficient test-data compression method for IP cores," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 6, pp. 1070–1083, Jun. 2007.
[16] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Optimal selective Huffman coding for test-data compression," *IEEE Trans. Comput.*, vol. 56, no. 8, pp. 1146–1152, Aug. 2007.
[17] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Test-data compression based on variable-to-variable Huffman encoding with codeword reusability," Dept. Comp. Sci., Univ. Ioannina, Greece, Tech. Rep. [Online]. Available: http://charon.cs.uoi.gr/~tech_report/hci/publications/tech_rep__01.pdf
[18] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Multilevel-Huffman test-data compression for IP cores with multiple scan chains,"*IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, to be published.
[19] C. Krishna and N. Touba, "Reducing test data volume using LFSR reseeding with seed compression," in *Proc. ITC*, 2001, pp. 321–330.
[20] C. Krishna and N. Touba, "Adjustable width linear combinational scan vector decompression," in *Proc. ICCAD*, 2003, pp. 863–866.
[21] J. Lee and N. Touba, "Low power test data compression based on LFSR reseeding," in *Proc. ICCD*, 2004, pp. 180–185.
[22] J. Lee and N. Touba, "Combining linear and nonlinear test vector compression using correlation-based rectangular encoding," in *Proc. 24th VTS*, 2006, pp. 252–257.
[23] L. Li and K. Chakrabarty, "Test set embedding for deterministic BIST using a reconfigurable interconnection network," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 23, no. 9, pp. 1289–1305, Sep. 2004.
[24] L. Li, K. Chakrabarty, S. Kajihara, and S. Swaminathan, "Efficient space/time compression to reduce test data volume and testing time for IP cores," in *Proc. 18th Int. Conf. VLSI Des.*, 2005, pp. 53–58.
[25] S. Lin, C. Lee, and J. Chen, "Adaptive encoding scheme for test volume/time reduction in SOC scan testing," in *Proc. ATS*, 2005, pp. 324–329.
[26] A. El-Maleh and R. Al-Abaji, "Extended frequency-directed run-length code with improved application to system-on-a-chip test data compression," in *Proc. ICECS*, 2002, vol. 2, pp. 449–452.
[27] M. Nourani and M. H. Tehranipour, "RL-Huffman encoding for test compression and power reduction in scan applications," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, no. 1, pp. 91–115, Jan. 2005.
[28] S. Reda and A. Orailoglu, "Reducing test application time through test data mutation encoding," in *Proc. DATE*, 2002, pp. 387–393.
[29] P. Rosinger, P. Gonciari, B. Al-Hashimi, and N. Nicolici, "Simultaneous reduction in volume of test data and power dissipation for systems-on-a-chip," *Electron. Lett.*, vol. 37, no. 24, pp. 1434–1436, Nov. 2001.
[30] M. Tehranipour, M. Nourani, and K. Chakrabarty, "Nine-coded compression technique for testing embedded cores in SoCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 6, pp. 719–731, Jun. 2005.
[31] Z. Wang and K. Chakrabarty, "Test data compression for IP embedded cores using selective encoding of scan slices," in *Proc. ITC*, 2005, pp. 1–10.
[32] S. Ward, C. Schattauer, and N. Touba, "Using statistical transformations to improve compression for linear decompressors," in *Proc. IEEE Int. Symp. DFT*, 2005, pp. 42–50.

State-Sensitive X-Filling Scheme for Scan Capture Power Reduction

Jing-Ling Yang and Qiang Xu

***Abstract*—Based on the operation of a state machine, this paper elucidates a comprehensive frame for probability-based primary-input-dominated X-filling methods to minimize the total weighted switching activity (WSA) during the scan capture operation. Experimental results demonstrate that the proposed approach significantly reduces both average and peak WSAs.**

***Index Terms*—Scan test, sequential circuits, switching activity (SA), test generation.**

I. Introduction

Full scan is the most utilized test strategy in the semiconductor industry. Applying a scan test, however, results in the switching activity (SA) of a circuit under test (CUT) during test mode that is far beyond that during normal operational mode [1], [2]. Various techniques such as scan chain reordering, scan chain segmentation, clock gating, and low-power automatic test pattern generation (ATPG) have been developed to reduce scan shift power dissipation (e.g., [3]–[7]). Some techniques, including circuit modification [8], ATPG algorithm [9], and X-filling techniques [10]–[13], focused on scan capture power reduction. Among these scan capture power reduction methods, X-filling techniques do not require a modification in the CUT and do not need to rerun the time-consuming ATPG process and, hence, are widely accepted.

As well as having no effect on CUT and ATPG, X-filling techniques are compatible with those shift power reduction techniques that use or do not use X-bits. Procedures for generating X-bits for all the steps of the scan test (which are, namely, scan in, scan capture, and scan out) can be found in [10]. Examples of X-filling capture power reduction techniques that are compatible with non X-filling shift power reduction techniques can be found in [13].

Sankaralingam and Touba [10] introduced unspecified values (X-bits) in the scan vector and reassigned them to reduce scan peak power, which may be caused by scan-in, scan capture, and/or scan-out problems. To decrease scan peak power, first, X-bits are introduced in the scan vector and then reassigned to minimize the number of state changes in the scan flip-flops (SFFs) between two consecutive operation steps. For scan capture peak power reduction, incremental fault-free simulations are used in the procedure.

Manuscript received February 27, 2007; revised August 23, 2007, November 11, 2007, and March 1, 2008. This paper was recommended by Associate Editor S. Vrudhula.

The authors are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: jlyang@cse.cuhk.edu.hk; qxu@cse.cuhk.edu.hk).

Digital Object Identifier 10.1109/TCAD.2008.923418