model interconnect with bends. Hence, the proposed cascading method is more appropriate.

It is important to note that the given formulas of the model parameters are fit to our specific test chip and process. Moreover, these formulas are only valid within a geometrical range, which is within the coverage of our test structures. For the corner segments, the valid range for the width is from 3 to 20 μm and the angles are 45° and 90°. However, as stated in Section II-A and [2], most of the on-wafer interconnects used in RFICs are within this range.

V. Conclusion

An equivalent circuit model is proposed for real case on-wafer CMOS RFIC interconnects. The equivalent circuit of the entire complex shaped interconnect is obtained by cascading basic sub-segment models according to the physical structure. The model parameters are extracted from the on-wafer S-parameter measurements and formulated into empirical expressions. The validity of the proposed model is proved by the measurements of the test structures within a tolerance of 9%. It is highlighted that with the geometrical information, i.e., the length, width and angel of the bend, most of the commonly used on-wafer RF CMOS interconnect can be characterized by using this proposed model. Moreover, given the compact nature of the model, it can be easily implemented into commercial EDA tools.

References

[1] X. Shi, J. Ma, B. H. Ong, K. S. Yeo, M. A. Do, and E. Li, "Equivalent circuit model of on-wafer interconnects for CMOS RFICs," in *Proc. IEEE RAWCON*, Atlanta, GA, Sep. 2004, pp. 95–98.

[2] X. Shi, J. Ma, K. S. Yeo, M. A. Do, and E. Li, "Equivalent circuit model of on-wafer CMOS interconnects for RFICs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 9, pp. 1060–1071, Sep. 2005.

[3] R. H. Havemann and J. A. Hutchby, "High-performance interconnects: An integration overview," *Proc. IEEE*, vol. 89, no. 5, pp. 586–601, May 2001.

[4] T. C. Edwards and M. B. Steer, *Foundations of Interconnect and Microstrip Design*, 3rd ed. New York: Wiley, 2000.

[5] R. J. Baker, H. W. Li, and D. E. Boyce, *CMOS Design, Layout and Simulation*. Piscataway, NJ: The Institute of Electrical and Electronics Engineers, Inc., 1997.

[6] X. Qi, "High frequency characterization and modeling of on-chip interconnects and RFIC wire bonds," Ph.D. dissertation, Dept. Elect. Eng., Stanford University, Stanford, CA, 2001.

[7] W. R. Eisenstadt and Y. Eo, "S-parameter-based IC interconnect transmission line characterization," *IEEE Trans. Compon., Hybrids, Manuf. Technol.*, vol. 15, no. 4, pp. 483–490, Aug. 1992.

Multilevel-Huffman Test-Data Compression for IP Cores With Multiple Scan Chains

Xrysovalantis Kavousianos, Emmanouil Kalligeros, and Dimitris Nikolos

***Abstract*—Various compression methods have been proposed for tackling the problem of increasing test-data volume of contemporary, core-based systems. Despite their effectiveness, most of the approaches that are based on classical codes (e.g., run-lengths, Huffman) cannot exploit the test-application-time advantage of multiple-scan-chain cores, since they are not able to perform parallel decompression of the encoded data. In this paper, we take advantage of the inherent parallelism of Huffman decoding and we present a generalized multilevel Huffman-based compression approach that is suitable for cores with multiple scan chains. The size of the encoded data blocks is independent of the slice size (i.e., the number of scan chains), and thus it can be adjusted so as to maximize the compression ratio. At the same time, the parallel data-block decoding ensures the exploitation of most of the scan chains' parallelism. The proposed decompression architecture can be easily modified to suit any Huffman-based compression scheme.**

***Index Terms*—Huffman encoding, test data compression.**

I. Introduction

In order to meet the tight time-to-market constraints, contemporary digital systems embed predesigned and preverified Intellectual Property (IP) cores. The structure of IP cores is often hidden from the system integrator and as a result, neither fault simulation, nor test pattern generation can be performed for them; only a precomputed test set is delivered along with such a core. Various methods have been proposed for reducing both the test-data volume and test-application time of unknown-structure IP cores [1]–[7], [9], [10], [12]–[18], [20]–[25]. Many of them encode directly the statically or dynamically compacted test sets using various (usually classical) compression codes like Golomb [2], [3], [21], alternating run-length [4], frequency-directed run-length (FDR) [5], [18], statistical codes [7], [10], [12], nine-coded-based [23], and combinations of codes [22]. Unfortunately, these code-based methods cannot exploit the existence of multiple scan chains in a core. This shortcoming stems from the fact that parallel decompression is not possible due to either the nature of the code or the way it is used in the specific method. For example, the decoding of the various run-length codes is performed in a strictly serial manner. On the other hand, the straightforward approach for parallelizing selective Huffman decoding (i.e., by encoding slice-sized data blocks and then by loading the scan chains in parallel with a whole slice after the decoding of every codeword) leads to both large decompressors and very low compression ratios. Since the vast majority of the cores have multiple scan chains, a serial-in, parallel-out register must be used for spreading the decoded data in them and thus, no test-time savings, despite the existence of multiple scan chains, are possible. The solution of using multiple decoders in parallel is too

Manuscript received May 6, 2007; revised August 8, 2007. This work was supported in part by the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II) and by the program PYTHAGORAS.

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

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

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

Digital Object Identifier 10.1109/TVLSI.2008.2000448

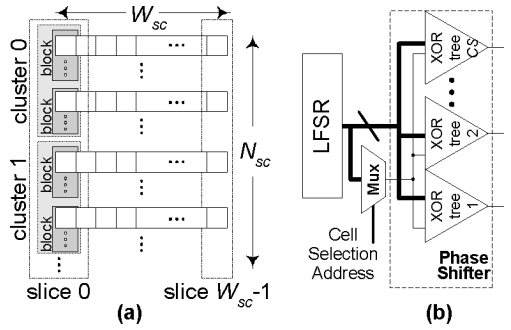

Fig. 1. (a) Scan chains, clusters, and blocks. (b) Pseudorandom generator.

expensive and thus inapplicable. Hence, although classical codes compare favorably, in terms of compression ratio, against many methods for multiple scan-chain cores, they are rarely used for such cores.

In this paper, we take advantage of the inherent parallelism of Huffman decoding (a whole block is decoded in a single clock cycle, after codeword identification), and we propose a Huffman-based compression scheme that exploits most of the parallelism offered by the multiple scan chains of a core. Specifically, we generalize the multilevel Huffman encoding of [12], and we present a novel and low overhead decompression architecture. Since the size of the encoded data blocks is independent of the slice-size (i.e., the number of scan chains), the former can be adjusted to maximize the compression ratios, while the parallel data-block decoding offers significant test-time improvements. Moreover, the decoder can be easily modified to support any Huffman-based compression scheme.

II. Compression Method

Consider a core with $N_{\rm sc}$ balanced scan chains of $W_{\rm sc}$ scan cells each [see Fig. 1(a)]. Each test cube (test vector with x values) is partitioned into $W_{\rm sc}$ consecutive slices of $N_{\rm sc}$ bits and every slice is partitioned into clusters of size *CS*. If $N_{\rm sc}$ is not divided exactly by *CS*, then the last cluster of all slices is shorter than the others. More formally, each test cube is partitioned into $W_{\rm sc} \cdot \lceil N_{\rm sc}/CS \rceil$ test clusters and the test set consists of totally $TC = W_{\rm sc} \cdot \lceil N_{\rm sc}/CS \rceil \cdot N_{\rm cubes}$ clusters ($N_{\rm cubes}$ is the number of test cubes).

The proposed encoding scheme is based on pseudorandom bit generation and multilevel Huffman coding. As pseudorandom generator we use a small internal-XOR linear feedback shift register (LFSR) and a phase shifter [see Fig. 1(b)], which can produce pseudorandom clusters of size *CS*. The phase shifter is initially designed as proposed in [19] and, then, for being able to choose among different sequences of pseudorandom clusters for the same time period, we add one extra input (i.e., one XOR gate) to each XOR tree [11]. A single MUX is used for selecting among various cells of the LFSR and its output is connected to the extra input of every XOR tree. For every different cell selected by the MUX, different linear functions are implemented by the phase shifter and a different sequence of pseudorandom clusters is generated. Hereafter, every time we refer to the pseudorandom cluster-sequence of (inverted) LFSR cell i, we will mean the cluster sequence generated when the (inverted) output of LFSR cell i is driven to the extra input of the XOR trees.

According to the proposed encoding scheme, the test clusters of each slice are encoded from cluster 0 to cluster $\lceil N_{\rm sc}/CS \rceil - 1$ [starting from slice $W_{\rm sc} - 1$ down to slice 0; see Fig. 1(a)]. Therefore, for every sequence of the test cubes, a sequence of *TC* test clusters is implied. Throughout the decoding process, the LFSR advances its state *TC* times and thus, the generator of Fig. 1(b) produces totally *TC* pseudorandom clusters (i.e., each pseudorandom cluster corresponds to one test cluster). If a test cluster is compatible with the corresponding pseudorandom cluster, then the former is produced by the pseudorandom generator. The volume of test clusters that are compatible with pseudorandom ones is maximized by considering the cluster sequences of more than one LFSR cells. Thus, if a test cluster cannot be matched by the corresponding pseudorandom cluster of a cell's cluster-sequence, it may be compatible with the respective cluster of another cell's sequence.

The LFSR is loaded only once, initially, with a random seed and the pseudorandom cluster-sequences of all normal and inverted LFSR cells with length equal to *TC* are generated. Then the test cubes are ordered in $N_{\rm cubes}$ iterations, so as to form the sequence of test clusters (one test cube is selected in every iteration and its test clusters are appended at the end of the sequence). Specifically, the $W_{\rm sc} \cdot \lceil N_{\rm sc}/CS \rceil$ test clusters of every unordered test cube are compared at each iteration, against the corresponding $W_{\rm sc} \cdot \lceil N_{\rm sc}/CS \rceil$ pseudorandom clusters of every normal and inverted LFSR cell. Then, a cube weight is calculated as follows: for every test cluster of a cube that is compatible with the corresponding pseudorandom cluster of a (normal or inverted) cell's sequence, the test cluster's defined-bits volume is added to the cube weight. The cube with the maximum weight is selected.

Next, a predetermined number of LFSR cells are iteratively selected. At each iteration, a weight is calculated for every cell, which is equal to the sum of the defined-bit volumes of all test clusters that are compatible with the corresponding pseudorandom clusters of that cell. The cell with the maximum weight is selected and the test clusters which are compatible with the corresponding pseudorandom clusters of that cell are not further considered for the selection of the rest LFSR cells.

After the selection of the LFSR cells, each test cluster is either: (a) compatible with at least one of the corresponding pseudorandom clusters of the selected cells' sequences or (b) not compatible with any one of them. Clusters of category (a) are encoded using *Cell encoding*. Specifically for generating them, the selection address of the MUX of Fig. 1(b) is encoded, i.e., each Huffman codeword is used for enabling an LFSR cell to drive the XOR trees' extra input. Clusters of category (b) are labeled as failed and a single Huffman codeword is used for distinguishing them from those of category (a).

In many cases, consecutive test clusters (cluster groups) can be generated by using the same LFSR cell. We encode such groups with a Cell-encoding codeword that indicates the selected cell, followed by a *Length encoding* codeword indicating the number of consecutive clusters that will be generated by using that cell (group length). The group lengths are chosen from the following list of distinct lengths: $2^0, 2^1, \ldots, 2^{r-1}$, where $2^{r-1} < max\_length$ (*max_length* is the maximum number of consecutive test clusters that are compatible with the corresponding pseudorandom clusters of an LFSR cell). Every Cell-encoding codeword is followed by a Length-encoding one, if the encoded test cluster is not a failed cluster.

The association of the test clusters of category (a) with the selected LFSR cells is done in an iterative fashion: at each iteration, the pseudorandom-cluster sequences of all selected cells are compared against the test-cluster sequence. All successive pseudorandom clusters which are compatible with the corresponding test clusters are grouped and the largest group is selected. If its size is not equal to one of the distinct lengths in the length list, it is set equal to the greatest possible distinct length by removing some pseudorandom clusters from the end of the group. If there is more than one maximum-sized group generated by different cells, the one produced by the most frequently used cell is

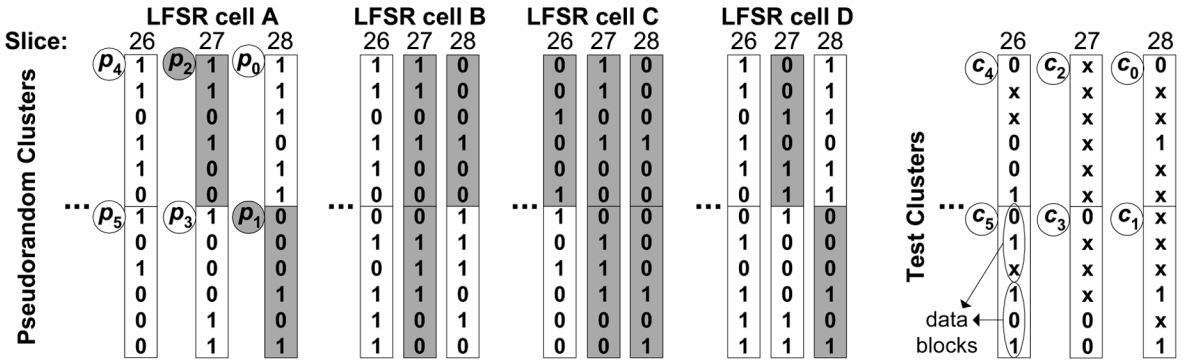

Fig. 2. Associating test clusters with LFSR cells (example).

selected, so as to skew the occurrence frequencies of the LFSR cells. The test clusters matched by the pseudorandom clusters of the selected group are not further considered in the remaining iterations.

For the test clusters of category (b) *Block encoding* is adopted. That is, all failed clusters are partitioned into blocks of size *BS*, and the blocks with the highest probabilities of occurrence are encoded using selective Huffman coding [10]. Some blocks remain unencoded (failed blocks) and are embedded directly in the compressed test set. Contrary to [10] where an extra bit is used in front of all codewords, in the proposed approach a single codeword is appended in front of only each failed block (as in the case of failed clusters).

Example: Consider a core with $N_{\rm sc} = 12$ and $W_{\rm sc} = 29$. Four LFSR cells (A, B, C, and D) and four cluster-group lengths (1, 2, 4, and 8) are used for encoding the test set. Assume that $CS = 6$ and $BS = 3$. Fig. 2 presents the sequence of the first 6 pseudorandom clusters $p_0, \ldots, p_5$ of the selected cells (the cluster labels of cells B, C, and D have been omitted), as well as the sequence of the first 6 test clusters to be encoded ($c_0, \ldots, c_5$ of the first cube). Each cluster p_i in gray is compatible with the corresponding test cluster c_i. The largest group of successive pseudorandom clusters that are compatible with the corresponding test clusters is the group of p_0-p_4 of cell C. However, the greatest distinct length that does not exceed the length of the group is 4. Therefore, test clusters c_0-c_3 are encoded as a group of size 4 (and are generated by using cell C), whereas pseudorandom cluster p_4 is removed from the group and is used afterwards for encoding c_4. Failed test cluster c_5 is partitioned into two blocks. At the end of the encoding process, the four most frequently occurring blocks are encoded, whereas the rest are marked as failed. ■

For simplifying the decoder, the same codewords are used for all three encodings as proposed in [12] (the volumes of the selected cells, list lengths, and encoded data blocks are equal). Thus, a codeword, depending on the mode of the decoding process, corresponds to three different kinds of information (and hence it is decoded differently): to an LFSR cell (normal or inverted), to a cluster-group length or to a data block. For efficiently compressing the test data, the Huffman codewords are generated considering all three encodings. The proposed generalized approach for multiple scan chains combines the advantages of multilevel coding with the flexibility of choosing the best cluster and block sizes, independently of the slice size.

III. Decompression Architecture

The proposed decompression architecture is shown in Fig. 3. The Input Buffer receives the encoded data from the automatic test equipment (ATE) channels in parallel (*ATE_DATA*) with the frequency of the ATE clock (*ATE_CLK*), shifts them serially into the Huffman FSM

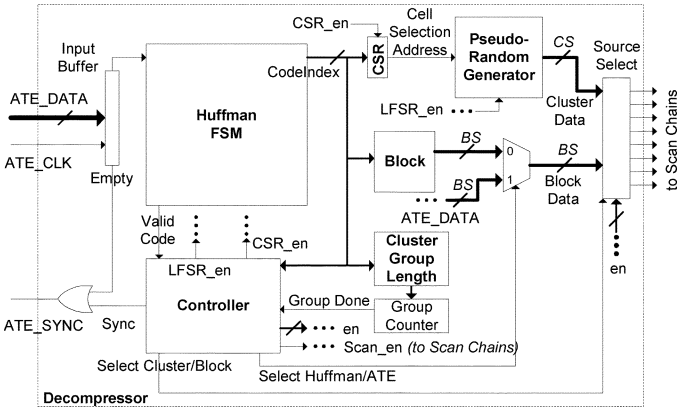

Fig. 3. Decompression architecture.

(finite-state machine) unit with the frequency of the system clock, and notifies the ATE to send the next test data (*Empty* $= 1$). When the Huffman FSM recognizes a codeword, it sets *Valid Code* $= 1$ and places on the bus *CodeIndex* a binary value. This value is used as the *Cell Selection Address* of the pseudorandom generator [see Fig. 1(b)] when the decoded cluster is not a failed one. Block and Cluster Group Length units are combinational blocks (or lookup tables) that generate respectively the encoded block or the group length corresponding to *CodeIndex*, when the received codeword is either a Length-encoding or a Block-encoding codeword. Since each block of a failed cluster is either Huffman encoded or not (i.e., embedded as is into the compressed data in the latter case), signal *Select Huffman/ATE* is used for distinguishing these two cases. The selected data are driven in parallel through a MUX to the Source Select unit. This unit receives pseudorandom clusters and blocks of failed clusters in parallel and, depending on the *Select Cluster/Block* signal, it gradually constructs the slice that will enter the scan chains. In other words, each slice is first formed in the Source Select unit in a number of clock cycles, and then is loaded in the scan chains (and hence, the scan chains are not fed with decoded data in every clock cycle).

The Controller, which is a state machine, waits until the first codeword has been received (*Valid Code* $= 1$), and if it indicates a non-failed cluster, it stores the cell address to the CSR register ($CSR\_en = 1$) and initiates the decoding of the next codeword (indicating the group length). When its decoding has been completed, the Controller loads Group Counter with the data returned by the Cluster Group Length unit and sets *Select Cluster/Block* $= 0$. The latter action enables pseudorandom data to enter the Source Select unit in parallel, for a number

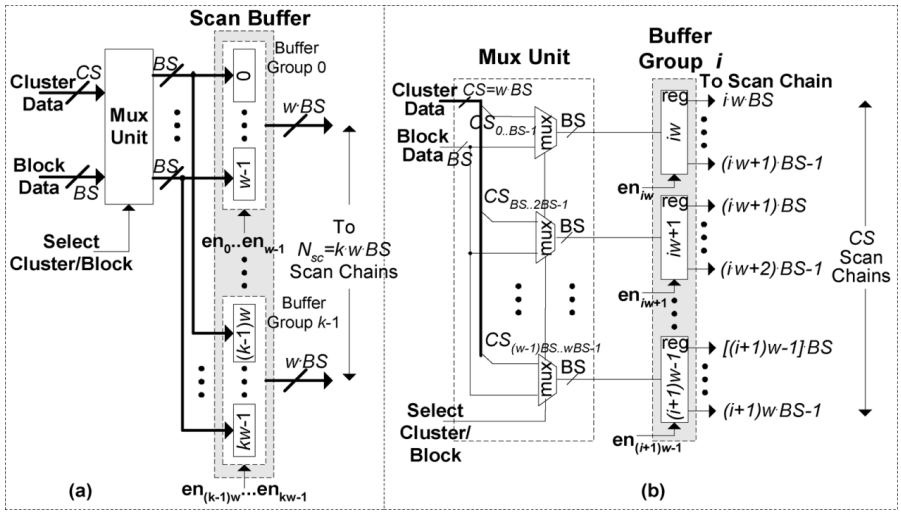

Fig. 4. Source Select unit.

of clock cycles equal to the length of the group (in every cycle, *CS* bits enter the Source Select unit). In each of these cycles, Group Counter decreases and the LFSR evolves ($LFSR\_en = 1$). The produced pseudorandom clusters are loaded in the Source Select unit until the end of the group (*Group Done* $= 1$). Every time a whole test slice is ready, it is loaded in the scan chains.

In case that the first codeword indicates a failed cluster, the subsequent codewords are decoded as data blocks. For every encoded block, the Controller sets *Select Huffman/ATE* $= 0$ and *Select Cluster/Block* $= 1$ for driving the output of the Block unit (which contains the decoded block) into the Source Select unit. For every failed block, the Controller sets *Select Huffman/ATE* $= 1$, *Select Cluster/Block* $= 1$, and $Sync = 1$ for signaling the ATE to send the failed block, which is driven in parallel to the Source Select unit (*ATE_CLK* is sampled, so as the Source Select unit to be enabled to receive the data at the right moment). When all blocks of the failed cluster have been generated, the LFSR is triggered once ($LFSR\_en = 1$) to pass from the state that produces the pseudorandom cluster which corresponds to the current failed cluster.

The Source Select unit [see Fig. 4(a)] receives cluster data from the pseudorandom generator (encoded clusters—*Cluster Data* bus), as well as block data either from the Block unit (Huffman encoded blocks—*Block Data* bus) or from the ATE (failed blocks). The received data are stored in the Scan Buffer which has size equal to that of a slice ($N_{\rm sc}$). The Scan Buffer consists of $\lceil N_{\rm sc}/BS \rceil$ registers with size equal to *BS*, grouped into $k = \lceil N_{\rm sc}/CS \rceil$ groups of $w = CS/BS$ registers (a group has the size of a cluster and includes w blocks). The Buffer Groups are loaded in a round robin fashion (Buffer Group i is loaded after Buffer Group $i-1$). When *Select Cluster/Block* $= 0$, the *Cluster Data* bus (of width *CS*) loads, through the MUX unit, all registers of a group simultaneously (in one clock cycle), whereas when *Select Cluster/Block* $= 1$, the *Block Data* bus (of width *BS*) is driven to every register (w clock cycles are needed for loading a whole group). This operation is handled by the Controller through the use of w enable signals $en_{iw}, \ldots, en_{(i+1)w-1}$, one for each register in the group [Buffer Group i is shown in Fig. 4(b)]. Totally, $k \cdot w$ enable signals are generated for the whole Scan Buffer. In order for a cluster to be loaded into Buffer Group i, all w enable signals of this group are activated. When a failed cluster is loaded into Buffer Group i, its registers are enabled one after the other, until all the blocks of the failed cluster are loaded into the corresponding registers. When Scan Buffer is full, the scan chains are loaded. Note that the Source Select unit allows the encoded-block sizes to be independent of the slice size. Thus, a similar unit can be utilized along with any Huffman-based compression method for multiple-scan-chain cores.

TABLE I
COMPRESSION RESULTS

Circuit	Min-test	N_{sc}=20		N_{sc}=40		N_{sc}=80		N_{sc}=100		Red. (%)
		16*c*	24*c*	16*c*	24*c*	16*c*	24*c*	16*c*	24*c*	
s5378	23754	9420	**9247**	9470	9261	9427	9338	9697	9521	61.1
s9234	39273	16056	15787	16201	**15722**	16330	15860	16358	15923	60.0
s13207	165200	20258	19400	18973	18543	18840	18381	18593	**18153**	89.0
s15850	76986	20143	19630	19754	19326	19763	**19313**	20162	19329	74.9
s38417	164736	63725	62227	60585	59078	60593	59026	60140	**58706**	64.4
s38584	199104	61891	59750	59699	**57801**	60776	58381	60584	58518	71.0

IV. EVALUATION AND COMPARISONS

We implemented the proposed method in the C programming language, and we performed experiments on a Pentium PC for the largest ISCAS'89 benchmarks circuits using the dynamically compacted Mintest test sets [8]. A primitive-polynomial LFSR of size 20 was used, while each XOR tree of the phase shifter comprised three gates. *BS* ranged from 5 to 10 (it was considered equal to the available ATE channels), whereas *CS* ranged from 20 to 50. The run time was a few seconds for each experiment.

In Table I the compression results of the proposed method for $N_{\rm sc} = 20, 40, 80,$ and 100, and for 16 and 24 LFSR cells (labeled 16*c* and 24*c*) are shown (various *CS* and *BS* values were examined and the best results are reported). In column 2, the sizes of the original Mintest test sets are presented. It is obvious that compression improves as the number of cells increases. The last column presents the reductions over Mintest, considering the best results of the proposed method (boldfaced).

We will next present comparisons against techniques suitable for unknown structure IP cores, which have reported results for the Mintest test sets. Methods [1], [14], [15], [17], [24], and [25] provide results for different test sets, and thus no direct comparison can be made with them. This is also the case for [13]. However, since [13] is a method that utilizes LFSRs and Huffman encoding like the proposed one, we implemented it and we performed experiments for the Mintest test sets, using LFSR sizes between 40 and 80 and scan-chain volumes in the

TABLE II
COMPRESSED DATA REDUCTION PERCENTAGES OVER OTHER TECHNIQUES

Circ.	[2]	[3]	[4]	[5]	[7]	[10]	[18]	[21]	[22]	[23]	[13]	[16]	[20]
s5378	-	-	20.9	25.1	19.3	13.3	19.0	36.7	15.8	12.0	24.0	35.0	-
s9234	29.3	30.1	27.3	29.0	24.1	12.6	26.0	34.3	23.6	11.5	16.0	47.8	-
s13207	56.4	48.3	44.4	41.2	33.4	52.2	39.5	52.2	37.2	25.8	61.7	13.5	75.6
s15850	52.6	36.8	26.6	25.7	21.8	26.2	21.6	38.3	23.2	12.7	35.6	23.2	25.8
s38417	36.2	35.6	9.6	37.2	23.6	13.1	9.6	20.1	0.5	4.0	37.1	31.1	-30.4
s38584	44.5	35.7	25.3	25.7	23.0	19.1	21.7	33.1	22.8	8.1	29.0	-1.2	21.3

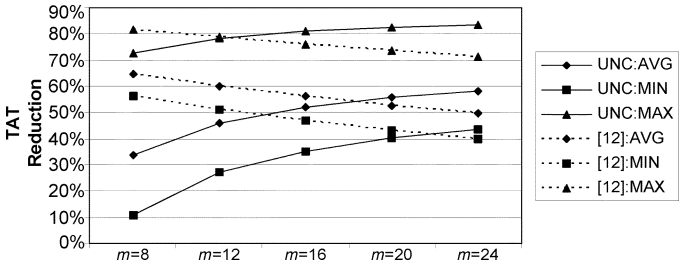

Fig. 5. TAT reduction.

range 5–40. No comparisons are provided against: 1) [6] since several conditions have to be satisfied by a core nearby the CUT so as to be used as decompressor and 2) [9] since its performance depends on the scan-chain structure of the cores. Note that we do not compare against approaches which employ fault simulation, and/or specially constrained ATPG processes. Also, we do not consider methods that use dictionaries, since they suffer from high hardware overhead.

In Table II, we present the improvements of the proposed approach over methods suitable for cores: 1) with a single scan chain (Columns 2–11) and 2) with multiple scan chains (Columns 12–14). The proposed approach performs better than all the other methods except for the case of s38584 of [16], which provides a marginally better result, and that of s38417 of [20]. However, the results reported in [20] do not include control information of significant volume should be also stored in the ATE for every core. Moreover, note that the reduced performance of [13] is due to the application of LFSR reseeding to the highly compacted Mintest test sets.

For assessing the test application time (TAT) improvements of our method, we performed two sets of experiments, for the best cases of Table I. In the first one, we study the TAT reduction achieved over the case in which the test set is downloaded uncompressed (UNC) to the core, using the same number of ATE channels. Fig. 5 presents the average (UNC:AVG), min (UNC:MIN), and max (UNC:MAX) improvements for various values of $m = f_{\rm SYS}/f_{\rm ATE}$ for all benchmarks. It is obvious that as m increases, the test-time gain is greater. In the second set of experiments, we compare the test application time of the proposed method against the single-scan-chain Multilevel Huffman approach of [12] for the ATE-channel volumes used in this paper. The best results of the proposed method and [12] have been compared and the average ([12]:AVG), min ([12]:MIN) and max ([12]:MAX) improvements are shown in Fig. 5. The TAT gain is very high in all cases (40%-81.6%). However, although the gain attributed to the parallel loading of multiple scan chains is constant, the serialization of the decoded data in [12] is carried out faster as m increases and thus the test-time reduction drops. We also compared the proposed method against methods [13], [16], and [20], which are suitable for cores with multiple scan chains. The average TAT reduction achieved by the proposed method over [13], [16], and [20] ranged, respectively, from 39.8% to 44.2%, from 82.7% to 89.1%, and from 15.2% to 46.3%, as m varied from 8 to 24.

TABLE III
HARDWARE OVERHEAD (IN GATE EQUIVALENTS)

[2]	[4]	[7]	[12]	Proposed
125-307	320	136-296	203-432	**8*c*:** 377, **16*c*:** 473, **24*c*:** 582

For assessing the area overhead of the proposed method, we synthesized three decompressors for 8, 16, and 24 LFSR cells, assuming 10 ATE channels, $N_{\rm sc} = 40$, $CS = 20$ bits, and $BS = 10$ bits. The Block and Cluster Group Length units were implemented as combinational circuits. The overhead of the proposed approach, as well as that of [2], [4], [7], and [12] are shown in Table III in gate equivalents (a gate equivalent corresponds to a two-input NAND gate). The hardware overhead of [10] is greater than that of [7]. Note that in our results we have not considered the overhead of the Scan Buffer, since such a structure is always required for reforming the decoded data (LFSR-based techniques use a phase shifter). Furthermore, it can be avoided if the core is equipped with a separate scan enable for each scan chain (the enable signals shown in Fig. 4 can be used for driving the scan enables). It is obvious that the area occupied by the proposed decompressor is comparable to that of the above techniques, even though the latter perform the simpler and less hardware intensive serial decoding. The methods of [16] and [20] have low hardware overhead but they do not offer as high compression ratios as the proposed one.

One common decompressor can be used for testing, one after the other, several cores of a chip. Units Huffman FSM, CSR, the LFSR and the phase shifter can be implemented only once. If all cores have the same number of scan chains, the Controller and the Source Select unit can be also shared among them. Otherwise, the Source Select unit is implemented by taking into account the core with the greatest number of scan chains and the Controller is slightly modified so as to be able to load only the proper part of the Scan Buffer for the rest cores (through the activation of the proper subset of the enable signals). The rest decoder units (Block, Cluster Group Length, and the MUX of the pseudorandom generator) must be implemented for every core. The area of the latter units is equal to 7.7%, 14%, and 19.7% of the total area of the decompressor for 8, 16, and 24 cells, respectively. Therefore, only a small amount of hardware should be added for each additional core, which can be almost eliminated if the Block and Cluster Group Length units are implemented as lookup tables, and loaded with the specific data of every core at the beginning of each test session. As shown in [12], by sharing the same Huffman FSM unit, the compression ratio suffers only a marginal decrease.

V. CONCLUSION

In this paper, we generalized the multilevel Huffman test-data compression approach for IP cores with multiple scan chains, and we presented a low-overhead decompression architecture capable of generating whole clusters of test bits in parallel. The proposed method offers reduced test-application times and high compression ratios. Also, by making the sizes of the encoded data blocks independent of the slice size, we demonstrated how Huffman-based test-data compression can be applied to multiple scan-chain cores.

REFERENCES

[1] K. Balakrishnan, S. Wang, and S. Chakradhar, "PIDISC: Pattern independent design independent seed compression technique," in *Proc. Int. Conf. VLSI Des.*, 2006, pp. 811–817.

[2] A. Chandra and K. Chakrabarty, "System-on-a-chip test-data compression and decompression architectures based on Golomb codes," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 3, pp. 355–368, Mar. 2001.
[3] A. Chandra and K. Chakrabarty, "Test data compression and decompression based on internal scan chains and Golomb coding," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 21, no. 6, pp. 715–72, Jun. 2002.
[4] A. Chandra and K. Chakrabarty, "A unified approach to reduce SOC test data volume, scan power and testing time," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 3, pp. 352–363, Mar. 2003.
[5] 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.
[6] R. Dorsch and H. -J. Wunderlich, "Tailoring ATPG for embedded testing," in *Proc. Int. Test Conf.*, 2001, pp. 530–537.
[7] P. T. Gonciari, B. M. Al-Hashimi, and N. Nicolici, "Variable-length input Huffman coding for system-on-a-chip test," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 6, pp. 783–796, Jun. 2003.
[8] I. Hamzaoglu and J. Patel, "Test set compaction algorithms for combinational circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 19, no. 8, pp. 957–963, Aug. 2000.
[9] Y. Han, Y. Hu, X. Li, H. Li, and A. Chandra, "Embedded test decompressor to reduce the required channels and vector memory of tester for complex processor circuit," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 5, pp. 531–540, May 2005.
[10] 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 Des. Integr. Circuits Syst.*, vol. 22, no. 6, pp. 797–806, Jun. 2003.
[11] E. Kalligeros, X. Kavousianos, and D. Nikolos, "Multiphase BIST: A new reseeding technique for high test-data compression," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 10, pp. 1429–1446, Oct. 2004.
[12] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Multilevel Huffman coding: An efficient test-data compression method for IP cores," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 6, pp. 1070–1083, Jun. 2007.
[13] C. Krishna and N. Touba, "Reducing test data volume using LFSR reseeding with seed compression," in *Proc. Int. Test Conf.*, 2002, pp. 321–300.
[14] C. Krishna and N. Touba, "Adjustable width linear combinational scan vector decompression," in *Proc. Int. Conf. CAD*, 2003, pp. 863–866.
[15] J. Lee and N. Touba, "Combining linear and non-linear test vector compression using correlation-based rectangular encoding," in *Proc. IEEE VLSI Test Symp.*, 2006, pp. 252–257.
[16] 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. Int. Conf. VLSI Des.*, 2005, pp. 53–58.
[17] S. Lin, C. Lee, and J. Chen, "Adaptive encoding scheme for test volume/time reduction in SoC scan testing," in *Proc. IEEE Asian Test Symp.*, 2005, pp. 324–329.
[18] A. Maleh and R. Abaji, "Extended frequency-directed run-length code with improved application to system-on-a-chip test data compression," in *Proc. Int. Conf. Electr. Circuits Syst.*, 2002, vol. 2, pp. 449–452.
[19] J. Rajski, N. Tamarapalli, and J. Tyszer, "Automated synthesis of phase shifters for built-in self-test applications," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 19, no. 10, pp. 1175–1188, Oct. 2000.
[20] S. Reda and A. Orailoglu, "Reducing test application time through test data mutation encoding," *Proc. Des. Autom. Test Eur.*, pp. 387–393, 2002.
[21] 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.
[22] M. Tehranipour, M. Nourani, K. Arabi, and A. Afzali-Kusha, "Mixed RL-Huffman encoding for power reduction and data compression in scan test," in *Proc. Int. Symp. Circuits Syst.*, 2004, vol. 2, pp. 681–684.
[23] 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.
[24] E. Volkerin and S. Mitra, "Efficient seed utilization for reseeding based compression," in *Proc. IEEE VLSI Test Symp.*, 2003, pp. 232–237.
[25] S. Ward, C. Schattauer, and N. Touba, "Using statistical transformations to improve compression for linear decompressors," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, 2005, pp. 42–50.

Improving the Transition Fault Coverage of Functional Broadside Tests by Observation Point Insertion

Irith Pomeranz and Sudhakar M. Reddy

***Abstract*—Functional broadside tests were defined to address overtesting that may occur due to high peak current demands when tests for delay faults take the circuit through states that it cannot visit during functional operation (unreachable states). The fault coverage achievable by functional broadside tests is typically lower than the fault coverage achievable by (unrestricted) broadside tests. A solution to this loss in fault coverage in the form of observation point insertion is described. Observation points do not affect the state of the circuit. Thus, functional broadside tests retain their property of testing the circuit using only reachable states to avoid overtesting due to high peak current demands. However, the extra observability allows additional faults to be detected. A procedure for observation point insertion to improve the coverage of transition faults is described. Experimental results are presented to demonstrate that significant improvements in transition fault coverage by functional broadside tests is obtained.**

***Index Terms*—Broadside tests, functional broadside tests, observation points, overtesting, transition faults.**

I. Introduction

Detection of delay faults requires the application of two-pattern tests. In scan-based circuits, a broadside test [1] specifies a scan-in state denoted by s, and two primary input vectors denoted by a_1 and a_2. We denote a broadside test by $\langle s, a_1, a_2 \rangle$. Application of the test starts by scanning in s. The primary input vectors a_1 and a_2 are then applied in functional mode. The scan-in state s together with a_1 define the first pattern of the test. The next state reached under s and a_1, together with a_2, define the second pattern of the test. The final state reached after the application of a_2 is scanned out.

Functional scan-based tests were defined as scan-based tests that detect target faults using only states that the circuit can visit during functional operation [2]. Based on this definition, functional broadside tests were defined as broadside tests that detect faults using only states that the circuit can visit during functional operation [3]. States that the circuit can visit during functional operation are also called reachable states. In general, a reachable state is a state that can be reached from all the states of the circuit. An important property of a reachable state

Manuscript received March 11, 2007; revised May 24, 2007. The work of I. Pomeranz was supported in part by Semiconductor Research Corporation under Grant 2004-TJ-1244. The work of S. M. Reddy was supported in part by Semiconductor Research Corporation under Grant 2004-TJ-1243.

I. Pomeranz is with the School of Electrical and Computer Engineering, Purdue University, W. Lafayette, IN 47907 USA (e-mail: pomeranz@ecn.purdue.edu).

S. M. Reddy is with the Electrical and Computer Engineering Department, University of Iowa, Iowa City, IA 52242 USA.

Digital Object Identifier 10.1109/TVLSI.2008.2000453