Low Power Test-Compression for High Test-Quality and Low Test-Data Volume

Vasileios Tenentes, Xrysovalantis Kavousianos
Dept. of Computer Science, University of Ioannina, Greece, {tenentes,kabousia}@cs.uoi.gr

***Abstract*—Test data decompressors targeting low power scan testing introduce significant amount of correlation in the test data and thus they tend to adversely affect the coverage of unmodeled defects. In addition, low power decompression needs additional control data which increase the overall volume of test data to be encoded and inevitably increase the volume of compressed test data. In this paper we show that both these deficiencies can be efficiently tackled by a novel pseudorandom scheme and a novel encoding method. The proposed scheme can be combined with existing low power decompressors to increase unmodeled defect coverage and almost totally eliminate control data. Extensive experiments using ISCAS and IWLS benchmark circuits show the effectiveness of the proposed method when it is combined with state-of-the-art decompressors.**

***Index Terms*—Defect Coverage; Test Data Compression**

I. Introduction

Even though traditional test data compression techniques (like for example [2], [4], [12], [13], [14], [15], [19]) are very efficient in compressing test data, they elevate switching activity beyond acceptable levels. Increased power dissipation often causes good chips to fail testing, degrading thus production yield. At the same time, nanometer technologies introduce new types of defects which lead to rapid growth of test data volume, test time and inevitably power dissipation. It is therefore evident that modern chips require new test data compression techniques which offer low power dissipation, high compression and high defect coverage.

In the past there have been proposed symbol-based test data compression techniques which offer low shift power, like [6], [7], [8], [18]. However, these techniques suffer from low compression efficiency and additionally they are not suitable for modern cores consisting of large number of scan chains. Linear decompressors on the other hand achieve very high compression but they are not power-friendly as they fill the unspecified ('x') values in a random way [2], [4], [14], [15], [19]. Recently, linear decompressors emerged, that target low switching activity during scan testing [10], [11], [16], [17]. However, these techniques increase the volume of the encoded test data as they need additional data to control the low power operation of the linear decompressors. In addition they often adversely affect the unmodeled defect coverage of generated test vectors as they tend to fill the unspecified values of the encoded test cubes in a highly correlated manner for reducing switching activity. So far, improvement of unmodeled defect coverage during low power testing has only been targeted in the context of X-filling in [3], while [9] targets increased N-detection of the resulting test vectors. However, both techniques are inapplicable in a test-compression environment were Xs are necessary for encoding test cubes.

In this paper we propose a new low cost scheme which can be combined with classical decompresssors to improve the unmodeled defect coverage of the generated test vectors and at the same time to reduce shift power. The proposed method exploits inherent properties of test sets to generate multiple diverse power-efficient encodings of test cubes, and it selects those offering the highest unmodeled defect coverage using an evaluation metric based on output deviations [20]. Contrary to the state-of-the-art low power decompressors, the proposed scheme does not increase the volume of test data to be encoded and thus it achieves higher compression.

The proposed architecture is simple, test set independent and can be combined with linear and code-based decompressors. In particular, it can be combined with the state-of-the-art linear decompressors presented in [10], [11], [17] as well as with the symbol-based decompressors presented in [12], [13] to improve both their unmodeled defect coverage and their compression efficiency. Extensive experiments show the effectiveness of the proposed method in terms of shift power, test data volume (TDV), test application time (TAT) and unmodeled defect coverage measured as coverage of surrogate fault models. We note that, to the best of our knowledge, this is the first test data compression technique for low power testing which targets unmodeled defect coverage.

II. Background & Motivation

Excessive shift power during scan testing has been traditionally tackled by exploiting unspecified bits ('x') of test cubes (i.e. test vectors consisting of '0', '1' and 'x' logic values) in order to reduce the pairs of successive complementary test bits shifted into scan chains. For example, the Fill Adjacent technique [5] fills 'x' values in such a way as to load successive scan cells with the same logic value in order to minimize transitions during scan-in. Even though this (and other similar techniques) is very effective in reducing the shift power, the generated test vectors tend to suffer from low unmodeled defect coverage compared to the test vectors generated by randomly filling the 'x' values [3].

This research has been co-financed by the European Union (European Social Fund ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

1081-7735/11 $26.00 © 2011 IEEE
DOI 10.1109/ATS.2011.75

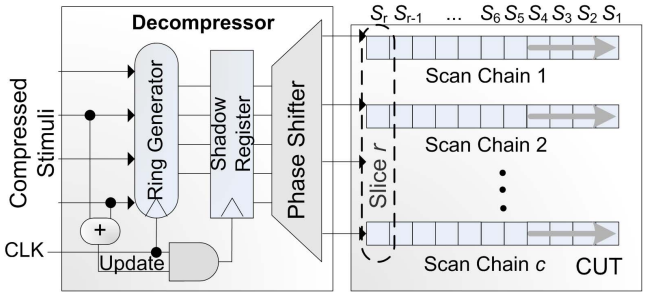

Fig. 1. Low power decompressors

Fig. 2. (a) presents a test cube for a circuit with 4 scan chains. (b) presents the test vector generated when this cube is encoded using a shadow register. The decompressor encodes slice 01x1 (this is the result of merging the compatible slices S_1,S_2,S_3), as well as the slice 1x10 (this is the result of merging slices S_4, S_5) and loads them into the shadow register using two Update operations at the 1st and 4th scan cycle highlighted in (b) (the 'x' values are randomly filled). The rest of the slices are generated using Hold operations. This encoding provides low switching activity but it is not unique. There are other groups of compatible successive slices that can be encoded, as shown in (c), (d) (the 2nd Update operation is applied sooner).

Using a similar concept, the linear decompressors proposed in [16] decrease shift power by partitioning test data of each scan chain into blocks. Each unspecified value ('x') in blocks is filled with the last encountered specified value. When the block size is small the shift power is considerably reduced because many blocks are generated as repeated versions of the same specified bits. However, one additional control bit per block is needed which increases the test data.

A similar approach was adopted in [10], [11], [17]. Specifically, the decompressor generates the test data in slices i.e., groups of c bits concurrently loaded into the c scan chains. Whenever a group G_k of k ($k > 1$) successive test slices of a test cube are compatible (i.e., every slice in this group exhibits no bitwise incompatibilities with any other slice in this group) one test slice S_k which is compatible with all test slices of group G_k is encoded and it is loaded into the scan chains for k successive clock cycles. This is achieved by the use of a shadow register located between the ring generator and the phase shifter (see Fig. 1). When S_k has to be generated for the first time, the ring generator generates and transfers the test data corresponding to slice S_k to the shadow register by setting signal $Update$ to logic value '1' (Update operation). During the next k successive clock cycles, the shadow register holds its contents by setting the $Update$ signal to logic value '0' and thus it continues loading the scan chains with the same slice S_k (Hold operation). In this case, additional data are needed to control signal Update which are encoded within the compressed test data.

Even though these methods reduce the shift power they suffer from limited unmodeled defect coverage due to the correlation induced in the way the 'x' values of test cubes are filled during the decompression. We will show that the adverse effects of this correlation on the unmodeled defect coverage can be significantly reduced by following a different encoding method. An example is illustrated in Fig. 2.

Different power efficient encodings of test cubes generate different test vectors which detect different unmodeled defects. If the proper encoding for each test cube is selected then the unmodeled defect coverage of the generated vectors will improve. Higher volume of power efficient and diverse test vectors can be generated by partitioning scan chains into groups loaded by separate and independently controlled shadow registers. For example, suppose that scan chains SC_1, SC_2 in the example of Fig. 2 are loaded from shadow register A and scan chains SC_3, SC_4 are loaded from shadow register B. Then there are three possible power efficient encodings for test data of scan chains SC_1, SC_2 and another four encodings for scan chains SC_3, SC_4, providing thus 3×4 different encodings (we do not count the first Update operation as it is always applied before the loading of the first slice). One example is shown at Fig. 2e.

A similar (but for a different purpose) approach was proposed in [10]. Specifically, in [10] it was noted that multiple independently controlled shadow registers can be potentially used for further reducing the shift power during scan testing. However this approach causes test data volume expansion. Consider for example a core consisting of 100 scan chains and 100 scan slices (i.e., each scan chain consists of 100 scan cells) and let us assume a typical fill rate of 1% (i.e., in average each test cube consists of 100 specified and 9900 unspecified bits). Then the number of control bits per cube is equal to 100 (one control bit per slice) which have to be encoded in conjunction with the 100 specified bits of the test cube. This results to duplicating the test bits to be encoded and inevitably results to reduced overall compression. If we use 2, 3, 4, ... shadow registers in the same example the test data to be encoded increase by 3x, 4x, 5x, ... which renders this approach impractical.

In this paper we show that the large amount of unspecified bits in test cubes can be exploited to almost eliminate these control data. This enables the application of an advanced encoding method which offers a wide variety of unique power-efficient encodings. These encodings are screened by an output-deviation based metric which selects the encodings offering the highest unmodeled defect coverage. The proposed method is based on a pseudorandom scheme which controls the shadow register(s) independently of the decompressor at no additional overhead on control data even when a large number of shadow registers are used. When this pseudorandom scheme is combined with linear and symbol-based decompressors it achieves a significant reduction of the volume of compressed test data as it eliminates the need for controlling the shadow register. We note that the proposed method can be combined with techniques like scan chain disabling [10] to reduce capture and scan out power as well.

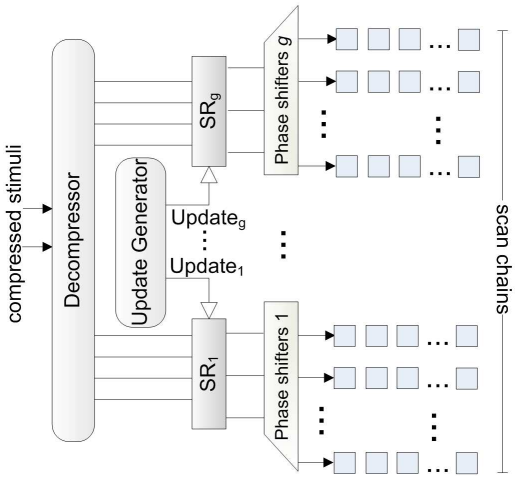

Fig. 3. Proposed Architecture

III. Proposed Method

A. Basic Concept

Consider a decompressor and a shadow register partitioned into g modules $SR_1, \ldots, SR_g$ as shown in Fig. 3. Each module SR_i drives a different group of scan chains and $Update_i$ is the Update signal driving module SR_i. Let TS be a set of test cubes generated using ATPG for a certain type of faults. The basic characteristic of the proposed method is that the Update operations of each module SR_i are determined prior to the encoding process by using a pseudorandom binary sequence[1] PS_i generated using a probability $P_{update-i}$. $P_{update-i}$ is the probability signal $Update_i$ to be set to logic value '1'. The encoding of test cubes of TS is adjusted to sequences PS_i. Specifically, if $PS_i(S_j) = 0$ (i.e. $Update_i = 0$, during generation of slice S_j) then SR_i holds its contents and the decompressor does not provide data to the SR_i (i.e. no test bits are encoded). If $PS_i(S_j) = 1$ ($Update_i = 1$, during generationg of slice S_j), then the contents of SR_i are updated with test data from the decompressor which are calculated in order to match all subsequent slices $S_{j+1}, S_{j+2}, \ldots$ which will be generated without updating the shadow register i.e. $PS_i(S_{j+1}) = PS_i(S_{j+2}) = \cdots = 0$.

The generation of the control sequences PS_i is very important for the effectiveness of the proposed method. Test cubes which exhibit bitwise incompatibilities in slices corresponding to successive Hold operations in sequence PS_i are not encode-able for PS_i (the potential of PS_i to encode the test cubes of a test set is hereafter referred to as the encode-ability of PS_i). A large number of Hold operations (i.e., a small value of $P_{update-i}$) degrades the encode-ability of PS_i while a large value of $P_{update-i}$ improves it (note that a value $P_{update-i} = 1$ can encode every test cube). However, a large value of $P_{update-i}$ increases the number of Update operations and thus the number of complementary bits shifted into the scan chains. So, when $P_{update-i}$ increases, shift power increases too. On the other hand, a small value of $P_{update-i}$ introduces high correlation in test data and the unmodeled defect coverage tends to drop (many adjacent scan cells are assigned the same logic value). Note that PS_i, determines only the groups of compatible slices for any encoded test cube (their specified bits are not affected). The generation of each PS_i sequence is part of the encoding method described next.

[1]The term *pseudorandom sequence* is used in a different meaning than in the rest literature. We use this term to refer to the way the shadow registers are controlled. The encoding of test cubes **still remains deterministic**

B. Encoding Method

At first we introduce a metric which is representative of the incompatibilities of test cubes and shows the likelihood a test cube to be encode-able using a pseudorandom sequence PS_i. Let t be a test cube. The volume of incompatibilities, $I(t, m)$, of scan chain $m \in [1, SC]$ for t, is defined as the number of successive complementary bits of t corresponding to scan chain m. Note that test cubes consist also of 'x' logic values which affect measure $I(t, m)$ based on the way they are filled. Since this is not known before the encoding, we adopt the following approximation: every 'x' logic value shifted into the scan chain is considered to be equal to the last specified logic value '0' or '1' which was encountered during the loading of this scan chain for t. This is a reasonable approximation as the proposed encoding tends to fill test cubes in a similar manner. Note that the 'x' values of test cubes are not actually filled which remain unaffected by this process. For example, for the test cube t of Fig. 2a we have $I(t, 1) = I(t, 4) = 1$, $I(t, 2) = I(t, 3) = 0$. The volume of incompatibilities $I(t)$ for test cube t is defined as the maximum value $I(t, m)$ for any of its scan chains $m \in [1, SC]$.

A test cube with a high (low) value of $I(t)$ is considered as a hard-to-encode (easy-to-encode) test cube due to its high (low) volume of incompatibilities between successive scan cells[2]. The same classification is done among scan chains. Specifically, for every scan chain m, $IS(m) = \sum_{t \in TS} I(t, m)$ is the measure of its incompatibilities among all test cubes of a test set TS. A scan chain m with high (low) value of $IS(m)$ is considered a hard-to-encode (easy-to-encode) scan chain. Both these measures can be used to improve the encoding process. The $IS(m)$ values are used to partition the scan chains into groups where each group is driven by its own shadow register. The $I(t)$ values are used to bias the encoding process towards the early encoding of the most hard-to-encode test cubes which can decrease the overall volume of test data.

At first the scan chains are partitioned into a pre-determined (selected by the designer) number of groups, g, according to their IS values (scan chains with similar IS values are grouped together). Since every group is independently controlled by a separate shadow register module, groups consisting of scan chains with small IS values are assigned a low initial value $P_{update-i}$, as the encode-ability of the corresponding pseudorandom sequences is not affected and the gains in switching activity reduction are high. For groups

[2]Among two test cubes t_1, t_2 with $I(t_1) = I(t_2)$ the most hard to encode is the cube with the highest value among $\sum_m I(t_1, m)$, $\sum_m I(t_2, m)$.

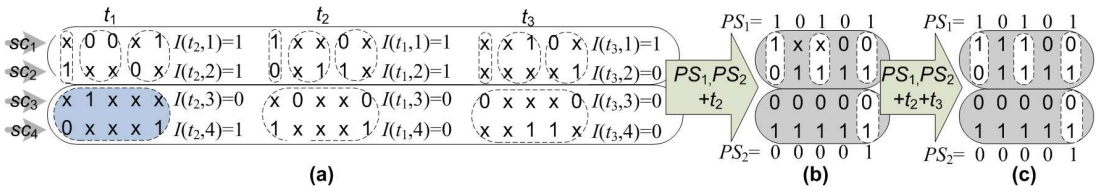

Fig. 4. Encoding example

consisting of scan chains with large IS values large initial $P_{update-i}$ values are assigned to enhance the encode-ability of the respective sequences. Let G_i be the number of incompatibilities of group i defined as the sum of the IS values of the scan chains comprising group i. Let SR_w ($w \in [1, g]$) be the module driving the group of scan chains with the highest volume (G_{worst}) of incompatibilities. Then the $P_{update-w}$ value for SR_w is set equal to a parameter P_{init} selected by the designer among a number of discrete values $P_1, P_2, \ldots, P_k$. The value of P_{init} is set according to the design objectives for shift power - test sequence length. A low value of P_{init} provides low shift power but increases the test sequence length. A high value of P_{init} increases the shift power but offers shorter test sequences. The initial probabilities $P_{update-i}$ of the rest modules are set to lower values which are calulated proportionally to P_{init}. Specifically $P_{update-i}$ is set equal to the rounding of the value $P_{init} \times G_i/G_{worst}$ to a discrete value in the set $P_1, P_2, \ldots, P_k$ (note that $G_i/G_{worst} < 1$).

After the initial value of every $P_{update-i}$ is determined, the sequence PS_i of each group corresponding to the first generated vector (i.e., for the first r cycles, where r is the length of the longest scan chain) is generated. This is achieved by the means of a trivial LFSR-based pseudorandom unit which will be presented in section III-D. The test cubes of TS are then examined for encode-ability for the given sequences. The encode-able cubes are those cubes which consist of slices without bitwise incompatibilities when they are successively loaded using Hold operations. These cubes are encoded as follows: every test slice S_j corresponding to an update operation ($PS_i(S_j) = 1$) and its following slices $S_{j+1}, S_{j+2}, \ldots$ corresponding to hold operations ($PS_i(S_{j+1}) = PS_i(S_{j+2}) = \cdots = 0$) are merged into one test slice which is encoded by the decompressor and it is loaded into SR_i when the update operation is applied. The encoding begins from the most hard-to-encode test cubes in order to minimize both the test data volume and the test sequence length. Additional test cubes can be encoded by the same sequence PS_i provided that a) they are encode-able for the sequence PS_i at hand, b) they are bitwise compatible with the previously encoded (by the same sequence PS_i) cubes and c) the decompressor has variables left to encode them. When no more test cubes can be encoded this process continues to the next vector (i.e., the sequence PS_i for each SR_i is generated for the next r cycles and the encoding continues with the remaining cubes). The following example illustrates the encoding process.

Example 1. Fig. 4 shows three test cubes t_1, t_2, t_3 and their I values. Based on I values the IS values are: $IS(sc_1) = 1+1+1 = 3$, $IS(sc_2) = 2$, $IS(sc_3) = 0$, $IS(sc_4) = 1$. Scan chains sc_1, sc_2 form the first group with $G_1 = 3 + 2 = 5$ and scan chains sc_3, sc_4 form the second group with $G_2 = 0 + 1 = 1$. Let $P_{init} = 1/2$, $k = 8$ and $[P_1, P_2, \ldots, P_k]$= $[1/16, 1/8, 1/4, 1/2, 3/4, 7/8, 15/16, 1]$. Since $G_1 > G_2$ we have $P_{update-1} = P_{init} = 1/2$ and $P_{update-2} = P_{init} \times G_2/G_1 = P_{init} \times 1/5 = 1/10$ which is rounded to the closest discrete P_i value, that is $P_2 = 1/8$. Let us assume that based on these probabilities the sequences $PS_1 = 10101$, $PS_2 = 00001$ are generated i.e. for the shadow register driving the first group three Update operations occur at the 1st, 3rd and 5th scan slice; for the shadow register driving the second group one Update operation occurs at the 1st slice. The test slices that must be compatible in order the test cubes to be encoded using PS_1, PS_2 are shown for t_1, t_2, t_3 inside dotted lines. Test cube t_1 is not encodeable for PS_2 as the test slices of the second group are not compatible (they are shown highlighted in Fig. 4). On the contrary, t_2, t_3 are both encode-able for PS_1, PS_2. Test cube t_2 is more hard-to-encode than t_3 because $I(t_2) = I(t_3)$ but $\sum_m I(t_2, m) > \sum_m I(t_3, m)$ and thus it is encoded first. Only the contents of the shadow registers at the Update operations (shown inside dotted lines in Fig. 4b) are encoded by the decompressor. Tthe rest of the slices are generated using Hold operations. After encoding t_2, few unspecified bits still exist which offer the potential for encoding also cube t_3. The final test vector is shown in Fig. 4c. ■

Certain incompatibilities in scan chains prohibit the encoding of some test cubes. When no test cubes can be further encoded for a number of successive test vectors, we increase $P_{update-i}$ of every group to the next higher discrete value and we initiate a new pseudorandom session. Each pseudorandom session is retained for as long as test cubes are encoded. In every successive session a different sequence is used for every signal $Update_i$ with increased rate of Update operations and thus more test cubes become encode-able.

As the values of $P_{update-i}$ increase in successive pseudorandom sessions, the switching activity increases (see section III-A) and its peak value may reach a predetermined limit. This happens because the remaining test cubes have many incompatibilities and thus they need a large number of Update operations which cannot be easily matched by pseudorandom sequences unless probabilities $P_{update-i}$ increase a lot. This means that the pseudorandom mode fails to further adhere with the power specifications of the circuit and it terminates.

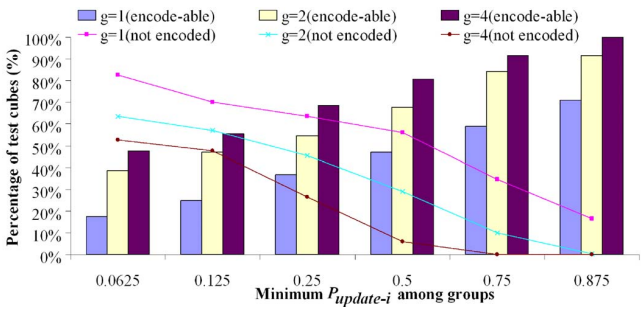

Fig. 5. Percentage of encode-able test cubes for the *Ethernet* benchmark

Then the deterministic mode is initiated with a global signal controlling all shadow registers like being one (the control data are encoded in this case as proposed in [10]).

The above encoding process owns its efficiency to the low fill rates of test sets. Specifically, as it is common in test sets, the vast majority of test cubes are sparsely specified while only a very small fraction of them are densely specified. The proposed method efficiently encodes the first ones during the pseudorandom sessions and the second ones during the deterministic session. In order to show the effectiveness of pseudorandomly generated sequences PS_i to encode large test sets we performed an experiment using the *Ethernet* circuit of IWLS suite [1]. This circuit consists of more than 10,000 scan cells and a dynamically compacted test set for complete coverage of stuck-at faults for this circuits is almost 12 Mbits in size; therefore it is more representative of realistic industrial designs than the rest of the benchmark circuits. Fig. 5 presents the percentage of test cubes which are encode-able for various sequences PS_i generated pseudorandomly. We run three different experiments by using $g = 1$, 2 and 4 shadow register modules. The x-axis presents the minimum value $P_{update-i}$ used among the groups (this value is increased from left to right of the x-axis as successive pseudorandom sessions are applied). At each pseudorandom session 100 different pseudorandom sequences PS_i were generated and the percentage of test cubes which are encode-able for at least one of them is reported by the means of bars.

It is obvious that at each successive session more test cubes become encode-able for the generated sequences. In addition, as the number of shadow register modules increases more test cubes become encode-able as the pseudorandom sequences match in a better way the specific characteristics of each group of scan chains. The curves show the test cubes which remain not encoded at the end of each pseudorandom session (test cubes which are encode-able for any generated sequence are immediately dropped in this case). It is obvious that the vast majority of test cubes are easily encoded at the first sessions which offer very low switching activity. Especially in the case of $g = 4$ shadow register modules all test cubes are encoded very fast and the deterministic mode can be eliminated (no test cubes remain unencoded after the 5th session). Thus it is evident that the effectiveness of the proposed pseudorandom encoding depends on the specified bits density of test cubes, which is fairly low in large circuits, and not on the size or amount of test cubes. Therefore, we conclude that the proposed method is scalable to very large test sets.

C. Unmodeled Defect Coverage Improvement

The encoding of test cubes is done in two steps: a) at first n different encodings are generated which all offer the same high compression and low shift power and b) the n test vectors corresponding to the n encodings are screened for detecting unmodeled defects and the most promising one is selected. Specifically, the n most hard-to-encode test cubes $t_1, t_2, \ldots, t_n$ which are encode-able by the current pseudorandom sequence PS are selected and n candidate encodings $e_1, e_2, \ldots, e_n$ are generated. Each candidate encodes one of the n selected test cubes and as many additional test cubes as possible. All n-candidates offer high compression as they encode hard-to-encode cubes and the same low switching activity as they use the same sequence of Update/Hold operations. However, they generate different test vectors which detect different defects. The best candidate is selected using a metric based on output deviations. Output deviations [20] are based on a probabilistic fault model, in which a probability map (referred to as the confidence-level vector) is assigned to every gate in the circuit. Signal probabilities p_i^0, p_i^1 are associated with each line i for every input pattern, where p_i^0, p_i^1 are the probabilities for line i to be at logic '0', '1' respectively. The confidence level of a gate G with m inputs is a vector with 2^m components $R = (r_{0\ldots00}, r_{0\ldots01}, \ldots, r_{1\ldots11})$, where each component of R denotes the probability that the gate output is correct for the corresponding input combination. Let y be the output of a 2-input NAND gate and a, b be its inputs. Then:
$p_y^0 = p_a^1 \cdot p_b^1 r_{11} + p_a^0 \cdot p_b^0 (1 - r_{00}) + p_a^0 \cdot p_b^1 (1 - r_{01}) + p_a^1 \cdot p_b^0 (1 - r_{10})$
$p_y^1 = p_a^0 \cdot p_b^0 r_{00} + p_a^0 \cdot p_b^1 r_{01} + p_a^1 \cdot p_b^0 r_{10} + p_a^1 \cdot p_b^1 (1 - r_{11})$.

For any gate G, let its fault-free output value for an input pattern t_j be d, $d \in 0, 1$. The output deviation G_j of G for t_j is defined as $P_G^{\bar{d}}$ where $\bar{d}$ is the complement of d. Intuitively, the deviation for an input pattern is a measure of the likelihood that the gate output is incorrect for that input pattern. The deviation values at the circuit outputs or pseudo-outputs are indicative of the probability arbitrary defects to be detected at these outputs. Output deviations are determined without explicit fault grading; hence the computation (linear in the number of gates) is feasible for large circuits and large test sets (further details can be found in [20]).

Before we present the details of the output-deviation based metric we note that as it is common in industry, each test vector is applied using the Launch-On-Capture scheme and thus both its first and second response r_1, r_2 are used for detecting defects. Initially, every observable output f is assigned four weights $w(f, r_1, 0)$, $w(f, r_1, 1)$, $w(f, r_2, 0)$, $w(f, r_2, 1)$, corresponding to error-free logic values '0', '1' at output f at responses r_1 and r_2. All these weights are initially set equal to the number of lines in the fan-in logic cone of output f as this is indicative of the volume of defects that can be potentially detected at this output. Then the largest deviation values expected for the test cubes of TS at each output at both

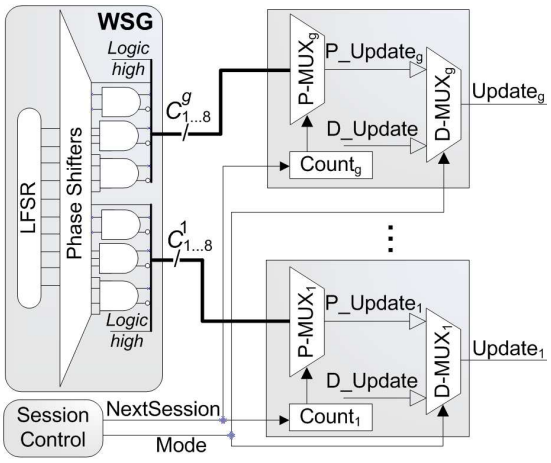

Fig. 6. Update Generation Module

responses r_1, r_2 and for both error-free logic values '0', '1' are estimated as follows: we temporarily fill the 'x' values of test cubes randomly (note that output deviations are calculated only for fully specified vectors) and for every test vector we calculate the deviation values at circuit outputs at both responses r_1, r_2 according to [20]. Among these values we identify the largest one for each output, response (r_1, r_2) and error-free logic value, which is denoted as $MaxDev(f, r_j, v)$ where $j = 1, 2$ and $v = 0, 1$ (test vectors are then discarded).

Let t be the test vector generated by a candidate-encoding. A weight $WT(t)$ is calculated for t as the sum of the weights $w(f, r_j, v)$ of outputs which have deviation value close to the $MaxDev(f, r_j, v)$ where v is the error free logic value at output f and response r_j when t is applied. Among the evaluated test vectors, the one with the highest weight is selected since it is the most promising for defects detection. Then the encoding process continues with the remaining test cubes. The weights of outputs that gave near-maximum deviation values at either response r_1, r_2 are divided by a constant factor F after each selection. This is motivated by the fact that many defects will probably be detected at these outputs by the selected candidate, and thus these outputs are expected to offer less defect coverage during the application of the next vectors. By reducing their weights the selection process is biased towards other outputs with high weights which are more promising for detecting undetected defects.

D. Proposed Architecture

The proposed architecture consists of the decompressor and the shadow register modules SR_i shown in Fig. 3 as well as of the Update Generation unit shown in Fig. 6. This module consists of the weighted signals generation unit (WSG), multiplexers (P-MUX$_i$) selecting between pseudorandom signals with different probabilities, a control unit which triggers the initiation of each next session and multiplexers (D-MUX$_i$) used for switching from pseudorandom to deterministic mode. At each clock cycle any number of 0 up to g modules may concurrently update their contents (note that as it is shown in Fig. 3 different outputs of the decompressor feed each shadow register module, thus the decompressor can even load all modules at the same cycle if necessary).

The weighted signals generator (WSG) consists of a small LFSR which is initially loaded with a known random seed and a very small combinational logic which generates pseudorandom signals with various probabilities in the range [0, 1]. This is achieved by feeding the outputs of different LFSR cells to combinational gates. For example, the output of a two-input AND gate driven by two LFSR cells has probability $P_{out} = (1/2)^2 = 25\%$. We verified that k =8 signals $C_1, C_2, \ldots, C_8$, with probabilities $0 < P_1 < P_2 < \ldots < P_8 \leq 1$ respectively, are sufficient to implement our scheme with negligible cost. Note that a phase shifter is also included in the WSG unit in order to provide multiple groups of linearly independent pseudorandom signals $C^1_{1...8}, \ldots, C^g_{1...8}$. Each of these groups of signals is used to drive a different shadow register module.

One among the signals $C^i_1, \ldots, C^i_8$ is selected by each P-MUX$_i$ (which is an $8 \rightarrow 1$ multiplexer in our case) for generating the PS_i sequence to drive signal $Update_i$. Signals $C^i_1, \ldots, C^i_8$ are connected to the inputs of P-MUX$_i$ in ascending order of signal probability, i.e, C^i_1 is connected to the first multiplexer input, C^i_2 is connected to the second input etc. Thus, in order to increase the probability that controls each group, a higher order input of P-MUX$_i$ is selected using a counter Count$_i$ which is very small (equal to 3 bits each for the case at hand). Count$_i$ stores the selection address of P-MUX$_i$ (let say value $1 \leq sel \leq 8$) for the entire session (it selects C_{sel}, thus $P_{update-i} = P_{sel}$). The value of Count$_i$ remains unchanged throughout every session and increases by one every time a new session is initiated.

In order to simplify the decompression process, at every successive session all counters simultaneously increase by one and thus every value $P_{update-i}$ increases to the next higher probability of WSG. This is triggered by internal registers of Session Control unit which are loaded from the ATE before the decompression process begins with the number of test vectors applied at each session. Since at most 8 pseudorandom sessions are applied ($k = 8$ probabilities are used) the area required for these session-registers is negligible. After the last pseudorandom session, Session Control switches the D-MUX$_i$ to the input D_Update which is a global signal common for all groups and the deterministic mode begins. The control data for signal D_Update are encoded by the decompressor.

The proposed scheme shown in Fig. 6 operates as a "low power converter" of the test cubes. It converts the test data from the decompressor into low power vectors compatible with the test cubes of TS. It is independent of the decompressor used to encode the converted data and thus it can be combined with linear as well as symbol-based decompressors.

IV. Experimental Results

We implemented the standard dynamic reseeding (SDR), the state-of-the-art low power dynamic reseeding proposed in [17] (LPDR) and the proposed method using the C++ programming language. For all the methods we used the same ring generators as decompressors and their size was selected by the s_{max} +

TABLE I
COMPARISONS TSL, TDV, TDF & BF (%)

Circuit	TDV (in Kbits)			TSL (# of vectors applied)			ASA			TDF Cov. (%)			BF Cov. (%)		
	SDR	LPDR	Prop.	SDR	LPDR	Prop.	SDR	LPDR	Prop.	SDR	LPDR	Prop.	SDR	LPDR	Prop.
s5378	5.6	10	6.9	273	305	331	50.1	5.8	12.4	63.54	62	66.72	94.9	94.39	95.11
s9234	11.3	20.9	13.9	477	504	597	49.6	11.6	19.3	47.41	49.81	52.59	88.47	88.17	88.81
s13207	10.9	20.8	13.4	342	419	415	50.1	5.4	13.3	62.27	61.25	69.43	92.55	92.09	93.05
s15850	14.7	26.8	18.4	498	552	611	50.1	7	11.7	55.4	54.68	58.39	95.96	94.42	94.67
s38417	64	97.5	69.3	1685	1548	1875	50	6.2	17.9	87.38	87.81	88.32	98.1	98.21	98.19
s38584	51.1	89.7	59.4	1115	1179	1281	50	7	13	67.68	67	68.52	91.65	91.57	91.74
ac97_ctrl	41.2	67.2	44.4	1547	1543	1665	50	3.8	3.9	57.74	57.44	66.88	99.54	99.49	99.53
pci_bridge	148.7	233	154.9	3614	3435	3731	53.3	2.6	5.7	83.83	81.88	84.6	98.6	98.56	98.6
tv80	40.3	72.5	47.6	2257	2330	2684	49.9	10.8	12.8	61.06	59.88	64.27	91.24	91.06	91.37
usb_funct	73.9	123.7	84.9	1709	1748	1895	50	5.2	11.7	74.67	74.32	77.02	97.35	97.32	97.44
ethernet	299.3	494.9	322	2385	2501	2574	50	3.3	12	53.19	53.21	57.01	93.47	93.35	93.61

20 rule, where s_{max} is the number of specified bits of the most specified test cube. We run experiments on a 4-CPU Linux workstation. The CPU time of the proposed method was almost 2.5 times the CPU time of the LPDR method.

We conducted experiments on the largest ISCAS'89 and a subset of the IWLS'05 [1] benchmark circuits. We examined various scan chain configurations and we selected the one that yielded the best result for the baseline SDR method and then all other methods used that scan chain configuration. For each circuit a test set TS was generated using a commercial ATPG engine targeting complete coverage of stuck-at faults.

For LPDR a single shadow register was used to keep its TDV low. We implemented the shadow register control using both techniques proposed in [10], [17] (internal XOR tap or one additional ATE channel) and the best result is reported. For the proposed method we used four shadow register modules, n = 30 candidate encodings and the threshold on peak switching activity was set close to that of LPDR. Various initial values of $P_{update-i}$ were used and the best results are reported. The WSG unit implements k=8 probabilities: 1/16, 1/8, 1/4, 1/2, 3/4, 7/8, 15/16, 1. The ATE-repeat command was utilized to reduce the TDV for all methods. We further improve the TDV of both LPDR and SDR methods by filling free variables in a repeat-friendly way similar to [19]. In the proposed method all free variables are filled in a non-repeat-friendly way to improve output-deviations. For all the results we present the test data volume (TDV), the test sequence length (TSL) and the average scan-in switching activity (ASA) measured using the metric of [10], [17].

For evaluating the unmodeled defect coverage we used two surrogate fault models, namely the transition delay (TDF) and the bridging fault model (BF). **None of these models were targeted by the stuck-at test sets encoded**. For detecting transition faults each stuck-at test vector generated by the decompressors is applied on the circuit using two capture cycles according to Launch-On-Capture (LOC) technique. For the bridging fault model 100K pairs of lines were selected randomly for each circuit. For each pair, four bridging faults were simulated by considering both lines as aggressors and victims, and both logic values '0' and '1' at the aggressors. Fault simulations were carried out using a commercial tool. Note that similar approaches were adopted in many techniques (e.g. [3], [20]) for evaluating the unmodeled defect coverage.

Table I presents the TDV in Kbits, the TSL in number of vectors applied and the ASA values for each method (note that the same number of clock cycles is needed in all cases to generate, load and apply each test vector). Columns 2-10 present the TDV, TSL and ASA values of SDR, LPDR and the proposed method. For the proposed method various initial values of P_{update_i} were used (the best results are reported). The SDR approach offers the best compression but its ASA is unacceptable. LPDR offers very low ASA, but increases the TDV compared to SDR considerably due to the additional data required for controlling the shadow register. The proposed method offers short TSL and small TDV, which approach the respective values of SDR method, and very low ASA which approaches that of LPDR. The superiority of the proposed method compared to LPDR in respect to both TDV and TSL stems from the fact that almost no control data are required by the proposed method (the proposed method requires control data only during the deterministic mode which constitutes a very small portion of the test mode). The ASA of the proposed method is a little higher than that of LPDR, but it is still very low and it most probably complies with the power specifications of the circuit which is the most critical goal for scan testing.

The last 6 columns of Table I present defect coverage comparisons. As it was expected, in the majority of the cases the LPDR method offers reduced defect coverage compared to the SDR approach. In almost all cases the proposed method achieves much higher TDF and higher BF coverage than both LPDR and SDR methods. We also note that the improvement of the proposed method against the other methods in terms of BF coverage is less than the improvement in terms of TDF coverage. However, this is due to the fact that the bridging fault coverage is very high in all cases and thus there is no much potential for further improvement. In particular, the average (over all circuits) number of bridging faults that remain undetected after the application of the proposed method is less than 2.6% of the total number of faults simulated. This clearly show that the proposed technique has already achieved very high bridging fault coverage.

Fig. 7 presents the TDF coverage ramp-up achieved for the representative ac97_ctrl benchmark circuit. It is obvious that

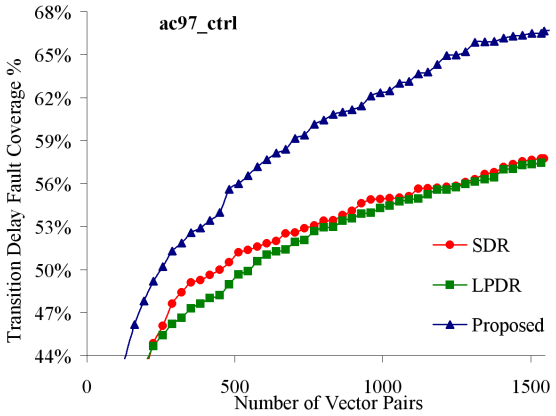

Fig. 7. Transition delay fault coverage ramp-up

the use of the proposed encoding combined with the output deviation-based metric offers higher coverage and coverage ramp-up than the rest methods reducing thus the TAT in an abort-at-first-fail environment.

For evaluating the hardware overhead we synthesized the proposed scheme for a) one and b) four shadow register modules. The proposed decompressors including all units (i.e., ring generator, shadow register, phase shifter, WSG, P-MUX, etc) is 15% larger in case (a) and 55% larger in case (b) than the decompressors of LPDR which are admittedly very small. Thus, the cost of the proposed scheme is also very small.

Finally, in order to show the advantages of the proposed scheme when combined with other decompressors, we implemented the statistical encoding proposed in [13] as follows: the test data corresponding to each scan slice are partitioned into multiple constant-length blocks and each block is encoded using the selective Huffman code. Data sent by the ATE are decoded using a Huffman decompressor, and the decoded blocks fill a register with length equal to the number of scan chains. When all blocks of a test slice are loaded into the register the slice is loaded into the scan chains. The proposed scheme is applied to this decompressor as follows: the aforementioned register plays the role of the shadow register which is partitioned into modules. Each codeword is used to encode the test data required to load a shadow register module whenever it is updated according to the pseudorandom sequences. The parts of the register corresponding to shadow register modules which are not updated at a scan cycle require no codewords (no encoding is done for these modules similar to the encoding method described in section 3.1). Thus, the proposed method in this case reduces both TDV and TAT (less codewords and clock cycles are required for loading each test slice into the scan chains). Due to space limitations we provide results only for the ac97_ctrl benchmark circuit for 32 scan chains, block size equal to 8 bits and number of encoded blocks equal to 16. When the proposed scheme is applied to this decompression architecture, the TDV drops from 55.3Kbits to 24.7 Kbits, the ASA drops from 36.4% to 4.6% and the test application time is reduced by 25.6%. At the same time TDF coverage increases from 41.8% to 50.24% and BF coverage increases from 95.8% to 98.6%. Thus the proposed method offers considerable gains in this case too.

V. Conclusions

We presented a new decompression scheme and a novel encoding method which can be combined with various decompressors to offer low shift power, high unmodeled defect coverage and high compression. Extensive experiments showed that when the proposed method is combined with state-of-the-art linear and statistical code based decompressors, both compression and unmodeled defect coverage improve while shift power is retained at very low levels.

References

[1] IWLS'05 circts., online: http://www.iwls.org/iwls2005/benchmarks.html.
[2] K. J. Balakrishnan and N. A. Touba, "Improving linear test data compression," *IEEE Trans. on CAD*, pp. 1227–1237, 2006.
[3] S. Balatsouka, V. Tenentes, X. Kavousianos, and K. Chakrabarty, "Defect aware x-filling for low-power scan testing," in *Proc. IEEE/ACM DATE*, march 2010, pp. 873–878.
[4] I. Bayraktaroglu and A. Orailoglu, "Test volume and application time reduction through scan chain concealment," in *Proc. ACM/IEEE DAC*, 2001, pp. 151–155.
[5] K. Butler, J. Saxena, A. Jain, T. Fryars, J. Lewis, and G. Hetherington, "Minimizing power consumption in scan testing: pattern generation and dft techniques," in *Proc. ITC*, 2004, pp. 355–364.
[6] A. Chandra and K. Chakrabarty, "Low-power scan testing and test data compression for system-on-a-chip," *IEEE Trans. on CAD*, vol. 21, no. 5, pp. 597–604, may 2002.
[7] ——, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (fdr) codes," *IEEE Trans. Comput.*, vol. 52, pp. 1076–1088, August 2003.
[8] ——, "A unified approach to reduce soc test data volume, scan power and testing time," *IEEE Trans. on CAD*, vol. 22, no. 3, pp. 352–363, Mar. 2003.
[9] Z. Chen, D. Xiang, and B. Yin, "Segment based x-filling for low power and high defect coverage," in *Proc. VLSI-DAT*, april 2009, pp. 319–322.
[10] D. Czysz, M. Kassab, X. Lin, G. Mrugalski, J. Rajski, and J. Tyszer, "Low-power scan operation in test compression environment," *IEEE Trans. on CAD*, vol. 28, no. 11, pp. 1742–1755, 2009.
[11] D. Czysz, G. Mrugalski, N. Mukherjee, and J. R. J. Tyszer, "Low-power compression of incompatible test cubes," in *ITC*, 2010, pp. 1–10.
[12] A. Jas, J. Ghosh-Dastidar, M.-E. Ng, and N. A. Touba, "An efficient test vector compression scheme using selective huffman coding," *IEEE Trans. on CAD*, vol. 22, no. 6, pp. 797–806, 2003.
[13] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Optimal selective huffman coding for test-data compression," *IEEE Trans. on Comput.*, vol. 56, no. 8, pp. 1146–1152, 2007.
[14] B. Könemann, "Lfsr-coded test patterns for scan designs," in *Proc. ETC*, 1991, pp. 237–242.
[15] C. V. Krishna, A. Jas, and N. A. Touba, "Achieving high encoding efficiency with partial dynamic lfsr reseeding," *ACM TODAES*, vol. 9, no. 4, pp. 500–516, 2004.
[16] J. Lee and N. A. Touba, "Lfsr-reseeding scheme achieving low-power dissipation during test," *IEEE Trans. on CAD*, vol. 26, no. 2, pp. 396–401, 2007.
[17] G. Mrugalski, J. Rajski, D. Czysz, and J. Tyszer, "New test data decompressor for low power applications," in *Proc. ACM/IEEE DAC*, 2007, pp. 539–544.
[18] M. Nourani and M. H. Tehranipour, "Rl-huffman encoding for test compression and power reduction in scan applications," *ACM TODAES*, vol. 10, pp. 91–115, January 2005.
[19] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Trans. on CAD*, vol. 23, no. 5, pp. 776–792, 2004.
[20] Z. Wang and K. Chakrabarty, "Test-quality/cost optimization using output-deviation-based reordering of test patterns," *IEEE Trans. on CAD*, vol. 27, no. 2, pp. 352–365, feb. 2008.