Test Data Compression Using Selective Encoding of Scan Slices

Zhanglei Wang, *Member, IEEE*, and Krishnendu Chakrabarty, *Fellow, IEEE*

***Abstract*—We present a selective encoding method that reduces test data volume and test application time for scan testing of Intellectual Property (IP) cores. This method encodes the slices of test data that are fed to the scan chains in every clock cycle. To drive N scan chains, we use only c tester channels, where $c = \lceil \log_2(N+1) \rceil + 2$. In the best case, we can achieve compression by a factor of N/c using only one tester clock cycle per slice. We derive a sufficient condition on the distribution of care bits that allows us to achieve the best-case compression. We also derive a probabilistic lower bound on the compression for a given care-bit density. Unlike popular compression methods such as Embedded Deterministic Test (EDT), the proposed approach is suitable for IP cores because it does not require structural information for fault simulation, dynamic compaction, or interleaved test generation. The on-chip decoder is small, independent of the circuit under test and the test set, and it can be shared between different circuits. We present compression results for a number of industrial circuits and compare our results to other recent compression methods targeted at IP cores.**

***Index Terms*—ATE pattern repeat, IP cores, scan slice, test data compression.**

I. Introduction

TEST DATA volume is now recognized as a major contributor to the cost of manufacturing testing of integrated circuits (ICs) [1]–[4]. Recent growth in design complexity and the integration of embedded cores in system-on-chip (SoC) ICs has led to a tremendous growth in test data volume; industry experts predict that this trend will continue over the next few years [5]. For example, the 2005 ITRS document predicted that the test data volume for integrated circuits will be as much as 30 times larger in 2010 than in 2005 [6].

High test data volume leads to an increase in testing time. In addition, high test data volume may also exceed the limited memory depth of automatic test equipment (ATE). Multiple ATE reloads are time consuming because data transfers from a workstation to the ATE hard disk, or from the ATE hard disk to ATE channels are relatively slow; the upload time ranges from tens of minutes to hours [7]. Test application time for scan testing can be reduced by using a large number of internal scan chains. However, the number of ATE channels that can directly drive scan chains is limited due to pin count constraints.

Logic built-in self-test (LBIST) [8] has been proposed as a solution for alleviating these problems. LBIST reduces dependencies on expensive ATEs and allows precomputed test sets to be embedded in test sequences generated by BIST hardware to target random pattern resistant faults. However, the memory required to store the top-up patterns for LBIST can exceed 30% of the memory used in a conventional automatic test pattern generation (ATPG) approach [8]. With increasing circuit complexity, the storage of an extensive set of ATPG patterns on-chip becomes prohibitive [1]. Moreover, BIST can be applied directly to SoC designs only if the embedded cores are BIST-ready; considerable redesign may be necessary for incorporating BIST in cores that are not BIST-ready.

Test data compression offers a promising solution to the problem of increasing test data volume. A test set T_D for the circuit under test (CUT) is compressed to a much smaller data set T_E, which is stored in ATE memory. An on-chip decoder is used to generate T_D from T_E during test application. A popular class of compression schemes relies on the use of a linear decompressor. These techniques are based on LFSR reseeding [9]–[12] and combinational linear expansion networks consisting of XOR gates [13]–[15], and they have been implemented in commercial tools such as TestKompress from Mentor Graphics [1], SmartBIST from IBM/Cadence [3], and DBIST from Synopsys [16]. These compression schemes exploit the fact that scan test vectors typically contain a large fraction of unspecified bits even after compaction. However, the on-chip decoders for these techniques are specific to the test set, which necessitates decompressor redesign if the test set changes during design iterations. Finally, in order to achieve the best compression, these methods resort to fault simulation and test generation. As a result, they are less suitable for test reuse in SoC designs based on Intellectual Property (IP) cores.

Another category of compression methods uses statistical coding, variants of run-length coding, dictionary-based coding, and hybrid techniques [17]–[22]. These methods exploit the regularity inherent in test data to achieve high compression. However, most of these schemes target single scan chains and they require synchronization between the ATE and CUT.

We present a selective encoding method that reduces test data volume and test application time for the scan testing of IP cores. This method encodes the slices of test data that are fed to the scan chains in every clock cycle. Unlike many prior methods, the proposed method does not encode all the specified (0 s and 1 s) and unspecified (don't care) bits in a slice. For example, if a slice contains more 1's than 0's, only the 0's are encoded and all don't cares are mapped to 1. We use only c tester channels,

Manuscript received January 13, 2007; revised September 3, 2007. First published August 12, 2008; current version published October 22, 2008. This work was supported in part by the National Science Foundation under Grant CCR-0204077. A preliminary version of this paper was published in the *Proceeding of the IEEE International Test Conference*, pp. 581590, 2005. This research was carried out when Zhanglei Wang was a Ph.D. student at Duke University.

Z. Wang was with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA. He is now with Cisco Systems Inc., San Jose, CA 95134 USA (e-mail: zhawang@cisco.com).

K. Chakrabarty is with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: krish@duke.edu).

Digital Object Identifier 10.1109/TVLSI.2008.2000674

where $c = \lceil \log_2(N+1) \rceil + 2$, to drive N scan chains. The logarithmic reduction in the number of tester channels allows us to use a large number of internal scan chains, thereby reducing test application time significantly. In the best case, we can achieve compression by a factor of N/c using only one tester clock cycle per slice. We derive a sufficient condition on the distribution of care bits that allows us to achieve the best-case compression. We also present a probabilistic lower bound on the compression for a given care bit density.

The proposed technique does not require dedicated test pins for each core in an SoC. If cores are tested sequentially, only one common test interface is needed. If some cores are tested in parallel, then they can together be viewed as a larger core with more scan chains. For example, if five cores are tested in parallel, and each core has 255 scan chains, it is equivalent to one core with 255 × 5 scan chains. The proposed technique will therefore only require 13 pins. Meanwhile, the on-chip decoder must be modified to handle the scan-in and capture for the five different cores.

The pattern decompression is of the continuous-flow type because no complex handshakes are required between the tester and the chip, and there is no need to introduce tester stall cycles. Unlike popular compression methods, such as Embedded Deterministic Test (EDT) [1], the proposed approach is suitable for IP cores because it does not require structural information for fault simulation, dynamic compaction, or interleaved test generation. The on-chip decoder is small, independent of the circuit under test and the test set. We present compression results for a number of industrial circuits, and compare our results to other recent compression methods targeted at IP cores.

The steady increase in clock frequencies over recent years has led to designs with a small number of gates between latches, or between latches and input/output (I/O) pins [23]. As a result, logic circuits today have very short combinational logic depth, and many logic cones with very little overlap. This is in contrast to older circuits such as the ISCAS'85 benchmarks that tend to have a smaller number of overlapping logic cones. A consequence of the shallow logic depth is that test patterns in present day circuits contain many don't care bits; e.g., it has been reported recently that test sets for industrial circuits contain only 1%–5% care bits [24]. After a desired stuck-at coverage is obtained (using methods such as dynamic compaction and fault grading to reduce pattern count), a commercial test pattern generator typically uses random fill to increase the likelihood of surreptitious detection of unmodeled faults. However, if the test sets for the cores are delivered with the don't care bits to the system integrator, an appropriate compression method can be used at the system level to reduce test data volume and testing time. This imposes no additional burden on the core vendor. Unmodeled faults can still be detected if the compression method does not arbitrarily map all don't cares to either 1's or 0's.

We do not address the problem of output compaction in this paper. The proposed input compression method can be used with recent output compaction methods such as X-compact [2], convolutional compaction [25], and i-compact [26] to further reduce test data volume.

The rest of this paper is organized as follows. The details of the proposed approach are described in Section II. Section III presents the decompression architecture and Section IV presents compression results for industrial circuits.

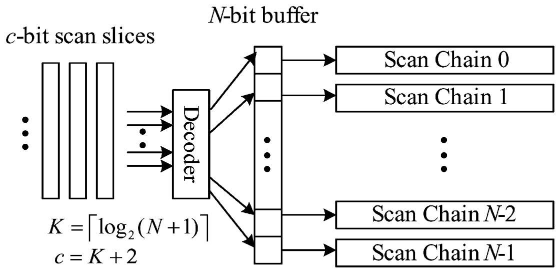


Fig. 1. Test application using the proposed approach.

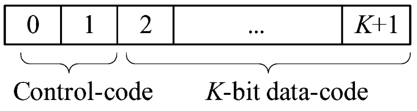


Fig. 2. Slice-code consists of a 2-bit control-code and a K-bit data code.

II. Proposed Approach

As shown in Fig. 1, the proposed approach encodes the slices of test data (scan slices) that are fed to the internal scan chains. The on-chip decoder contains an N-bit buffer, and it manipulates the contents of the buffer according to the compressed data that it receives. After all the compressed data for a single slice is received, the data in the buffer is delivered to the scan chains.

Each slice is encoded as a series of c-bit *slice-codes*, where $c = K+2, K = \lceil \log_2(N+1) \rceil$, and N is the number of internal scan chains in the CUT. The number of slice codes needed to encode a given slice depends on the distribution of 1's, 0's, and don't cares in the slice. As shown in Fig. 2, the first two bits of a slice-code form the *control-code* that determines how the following K bits, referred to as the *data-code*, are interpreted.

As described in Section I, the proposed approach only encodes a subset of the specified bits in a slice. First, the encoding procedure examines the slice and determines the number of 0- and 1-valued bits. If there are more 1's (0's) than 0's (1's), then all X's in this slice are mapped to 1 (0), and only 0's (1's) are encoded. The 0's (1's) are referred to as *target-symbols* and are encoded into data-codes in two modes: 1) *single-bit-mode* and 2) *group-copy-mode*.

In the single-bit-mode, each bit in a slice is indexed from 0 to $N-1$. A target-symbol is represented by a data-code that takes the value of its index. For example, to encode the slice XXX10000, the X's are mapped to 0 and the only target symbol of 1 at bit position 3 is encoded as 0011. In this mode, each target symbol in a slice is encoded as a single slice code. Obviously, if there are many target symbols that are adjacent or near to each other, it is inefficient to encode each of them using separate slice codes. Hence, the group-copy mode has been designed to increase compression efficiency.

In the group-copy mode, an N-bit slice is divided into $M = \lceil N/K \rceil$ groups, and each group (with the possible exception of the last group) is K-bits wide, as shown in Fig. 3. The M groups are numbered from 0 to $M-1$, with the first group (referred to as group 0) containing bits 0 to $K-1$ and so on. If a given group contains more than one target symbol, then the group-copy mode is used and the entire group is copied to a data code. Two data codes are needed to encode a group. The first data code specifies the index of the first bit of the group, and the second data code contains the actual data. In the group-

TABLE I
EXAMPLE TO ILLUSTRATE SLICE ENCODING

Case	Slice	Slice code		Description of the encoding process
		Control code	Data code	
1	XX000_0010X_00000_ 00XXX_XXXX0_XX0XX_0	00	00111	Single-bit-mode: start a new slice, map all Xs to 0, set bit 7 to 1.
2	XXXXX_XXXXX_XXXXX_ XXXXX_XXX11_XXXXX_1	01	11111	Single-bit-mode: start a new slice, map all Xs to 1, no bits are set to 0 (dummy data-code).
3	X1100_01101_XX00X_ 00XX0_00000_00XXX_1	00	00001	Single-bit-mode: start a new slice, map all Xs to 0, set bit 1 to 1.
		10	00010	Single-bit-mode: set bit 2 to 1.
		10	00110	Single-bit-mode: set bit 6 to 1.
		10	00111	Single-bit-mode: set bit 7 to 1.
		10	01001	Single-bit-mode: set bit 9 to 1.
		10	11110	Single-bit-mode: set bit 30 to 1.
4	X1100_01101_XX00X_ 00XX0_00000_00XXX_1	00	11110	Single-bit-mode: start a new slice, map all Xs to 0, set bit 30 to 1.
		11	00000	Group-copy-mode: starting from bit 0, i.e., group 0.
		11	X1100	Group-copy-mode: content of group 0.
		11	01101	Group-copy-mode: content of group 1.

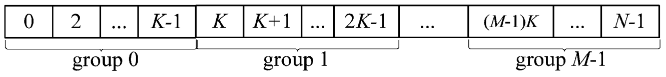


Fig. 3. N-bit scan slice is divided into $M = \lceil N/K \rceil$ groups in the group-copy mode.

copy mode, don't cares can be randomly filled instead of being mapped to 0 or 1 by the compression scheme.

For example, let $N = 31$ and $K = 5$, i.e., each slice is 31-bits wide and consists of six 5-bit groups and one 1-bit group. To encode the slice X1110_00000_XX00X_XXXX0_00000_00XXX_0 (the slice is shown separated by underscores into 5-bit groups), only the three 1's in group 0 are encoded. Since group 0 has more than one target symbol, the group-copy mode is used, yielding two data codes 00000 and X1110. The first data code refers to bit 0 (first bit of group 0), and the second data code carries the content of group 0.

The concept of a group is relevant only for the group-copy-mode. If only the single-bit mode is used, we do not need to consider groups at all. The underlying idea for the proposed method is: 1) only encode target symbols; 2) the bit-index of the target symbol is encoded in the slice code for proper decompression; 3) to further improve encoding efficiency, the group-copy mode is introduced, in which the scan slice is divided into groups.

Since data codes are used in both modes, control codes are needed to avoid ambiguity. Control codes 00, 01, and 10 indicate that the current slice code is a single-bit mode slice code, and control code 11 indicates the current slice code is a group-copy mode slice code. The first slice code for any scan slice is always a single-bit mode code, and its control code can only be 00 or 01. Hence, control codes 00 and 01 are referred to as *initial-control codes* and they indicate the start of a new slice. Control code 00 (01) indicates that all X's in the current scan slice should be mapped to 0 (1), and the target symbol for the whole slice is 1 (0). Except the first slice code, all other single-bit mode slice codes for the same scan slice can only have the control code 10, i.e., simply setting the bit specified by the associated data code to the target symbol of this slice.

Table I shows a complete example to further illustrate how scan slices are encoded into slice codes. In this example, $N = 31$ and $K = 5$. In Case 1, the slice contains only one target symbol (1), and is encoded into one slice-code using the single-bit mode. In Case 2, the slice contains no target symbols. To handle this special case, we introduce a dummy data code whose value is N. Since the N bits in the scan slice is indexed from 0 to $N - 1$, a data code with value N implies that no bit should be set to the target symbol.

In Case 3 and Case 4, the same slice is encoded with the group-copy mode turned off and on, respectively. In Case 3, since the slice contains six target symbols, six single-bit mode slice codes are required to encode it, resulting in negative compression, which is undesirable. In Case 4, however, only four slice codes are needed: three group-copy mode slice codes to encode groups 0 and 1, and one single-bit mode slice code to encode the 1 at bit 30.

As discussed before, two group-copy mode slice codes are needed to encode a given group, one to indicate the starting bit of the group, and the other to carry the content of the group. However, to further improve compression efficiency, if $n \geq 2$ adjacent groups are to be encoded using the group-copy mode, they can be merged together. This optimization procedure is referred to as *group merging*. As shown in Case 4, since groups 0 and 1 are adjacent, instead of using four slice codes 1100000, 11X1100, 1100101, and 1101101 to encode the two groups, only three slice codes are needed. The code that indicates the starting bit of group 1 (1100101) is omitted. Therefore, to encode $n \geq 2$ adjacent groups, $n + 1$ group-copy mode slice codes are needed, with the first slice code indicating the starting bit of the first group, and the other n group-copy mode slice codes carrying the contents of the n groups. With group merging, the number of consecutive group-copy mode slice codes is no longer fixed. To avoid ambiguity, two series of consecutive group-copy mode slice codes that encode two series of adjacent groups must be interleaved by one single-bit mode slice code.

Cases 3 and 4 also show that the group-copy mode should be used whenever possible. Since two group-copy mode slice codes are sufficient to encode any K-bit group, if a group contains more than one target symbol, it should be encoded using the group-copy mode.

The encoding procedure is summarized in Fig. 4. In Step 1), each test vector is divided into a series of slices. We then encode each slice as a series of slice codes. In Steps 3)–7), the numbers of 0's and 1's are calculated, and the target symbol as well as

```
Encoding procedure:
 (1) Format the given test vectors into slices;
 (2) for each slice
 (3)    Determines the number of 0s ($k_0$) and 1s ($k_1$) in the
     slice;
 (4)    If $k_0 > k_1$ then
 (5)      target-symbol := 1, 1st control-code := 00;
 (6)    else
 (7)      target-symbol := 0; 1st control-code := 01;
 (8)    for each group of the slice
 (9)      calculate the number of target-symbols;
(10)      if number-of-target-symbols > 1 then
(11)        encode the group using the group-copy-mode;
(12)      else
(13)        encode the group using the single-bit-mode;
(14)    end for (group);
(15)    generate slice-codes for the current slice.
(16) end for (slice);
```

Fig. 4. Encoding procedure.

the control code of the first slice code are set. The first slice code of each slice must contain an initial-control code (00 or 01).

Steps 8)–14) encode all the groups of a slice. For each group of a slice, if it contains more than one target symbol, it is encoded using the group-copy mode; otherwise, it is encoded using the single-bit mode.

Once all groups have been encoded, the slice code generation step (Step 15) becomes straightforward. It first merges group-copy mode slice codes that correspond to adjacent groups, and then appropriately interleaves group-copy mode slice codes with single-bit mode slice codes. It also ensures that the first slice code is a single-bit mode code.

As can be seen from Fig. 4, the encoding procedure consists of two nested loops: the outer loop is for scan slices, and the inner loop is for groups of a given slice. Hence, its time complexity is $O(S \cdot N) = O(V)$, where V is the total data volume of the test set in bits.

A. Upper and Lower Bounds on Compression

By deriving the upper and lower bounds on compression, this section provides more insights into the proposed compression method. The maximum compression is achieved when each slice is encoded as a single slice code. Hence, the upper bound on compression is simply N/c, where N is the number of scan chains and c is the number of ATE channels.

The upper bound can only be achieved if every slice is encoded as a single slice code. This, in turn, is only possible if every scan slice contains either zero or one target symbol. However, the maximum compression factor of N/c can be achieved even if the test set contains a large fraction of care bits. Test set relaxation methods as in [27] can be used to help approach the upper bound. Such methods, however, require structural information about the circuit.

To derive the lower bound, we do not consider the group-copy mode. Note that the outcome of the group-copy mode depends on the actual distribution of the target symbols in scan slices. It is not possible to derive a bound that considers the group-copy mode, but is independent of the distribution of the target symbols. Since the target symbol distribution is difficult to model and it usually changes from slice-to-slice, such an analysis is rather difficult. Moreover, the optimization step that merges consecutive group-copy mode slice codes further makes the analysis even more difficult.

The lower bound is reached when each target symbol is encoded into a single-bit mode slice code, i.e., each group contains at most one target symbol such that the group-copy mode is not used for the entire test set. The lower bound can be expressed in terms of care-bit density. Let the fraction of 1's, 0's, and don't cares in any scan slice be p_0, p_1, and p_x, respectively. The probability p_t that any bit in the scan slice is a target symbol is given by $p_t = \min\{p_0, p_1\}$. The number of target symbols in any scan slice can therefore be viewed as a random variable X that follows a binomial distribution, i.e.,

$$P[X = i] = \binom{N}{i} p_t^i (1 - p_t)^{N-i}.$$

The expected number of target symbols $E[X]$ is simply Np_t. Each target symbol in a scan slice must be mapped to one slice code, hence, the average number of slice codes for a given scan slice is Np_t. If we ignore the additional compression that can be obtained using the group-copy mode, and let S be the number of scan slices, we get the following expression for the compression factor f:

$$f \geq \frac{NS}{Np_t(K+2)S} \geq \frac{2}{(p_0 + p_1)(K+2)} = \frac{2}{p(K+2)}. \quad (1)$$

Note that even if we do not assume any knowledge of p_1 and p_0, a lower bound can be obtained from the knowledge of the care-bit density, i.e., $p = p_0 + p_1$. If the care-bit density for each scan slice is different, i.e., the care-bit densities are $q_0, q_2 \cdots q_{s-1}$ for the S scan slices, respectively, we get

$$f \geq \frac{NS}{\sum_{i=0}^{S-1} Nq_i(K+2)} = \frac{S}{(K+2)\sum_{i=0}^{S-1} q_i}. \quad (2)$$

Equation (2) provides a more accurate bound because the first few pattern in a (ordered) test set contain many more care bits than the latter patterns. However, computing this lower bound requires nearly as much effort as compressing the test set.

Fig. 5 shows the probabilistic lower bound on the compression factor f for different values of N and p. The lower bound decreases with increasing N, since K increases with N. In practice, due to the use of the group-copy mode, we often achieve a larger compression factor. Thus, while the upper bound is overly optimistic for larger values of p, the lower bound is overly pessimistic for larger values of N.

B. ATE Pattern Repeat

A property of the proposed compression method is that consecutive c-bit compressed slices fed by the ATE are often identical or compatible. Therefore, ATE pattern repeat can be used to further reduce test data volume after selective encoding of scan slices. In the uncompressed data sets, especially among the test vectors that lie near the end of a test set, there are a large number of consecutive slices that contain no target symbols. These slices are encoded as identical single slice codes that have only dummy data codes. With ATE pattern repeat, these slice codes can be further compacted. Additionally, consecutive group-copy mode slice codes can also be compacted if they are compatible. Fig. 6 shows how a set of scan slices are encoded. The example shows

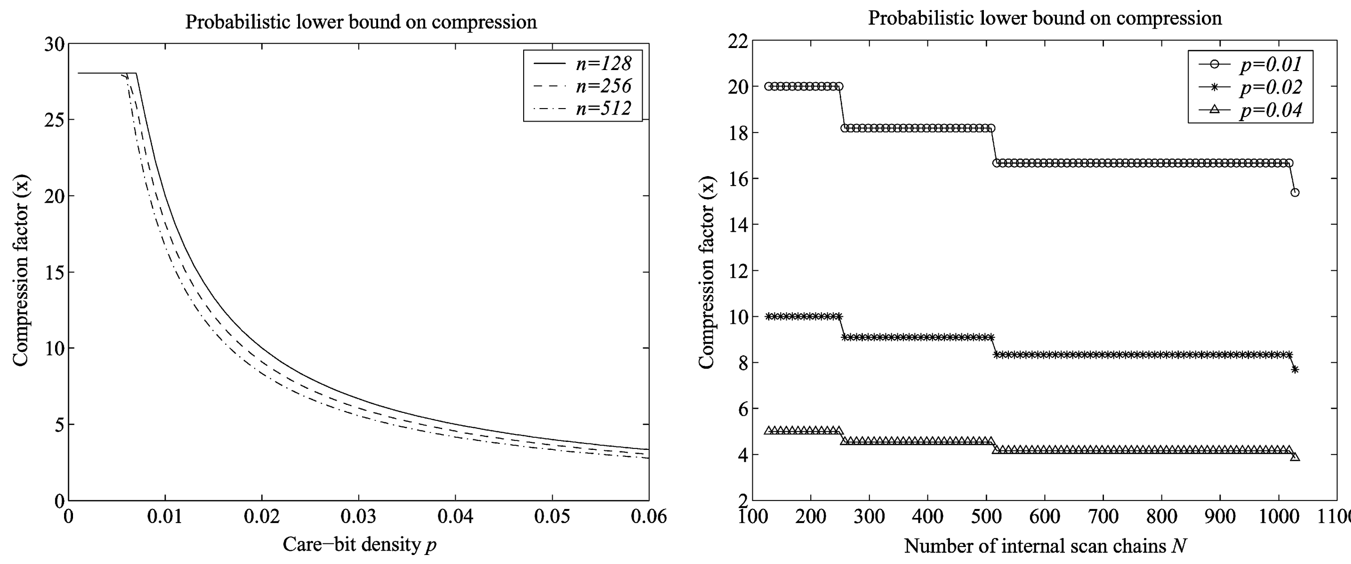


Fig. 5. Lower bounds on the compression factor for different values of N and p.

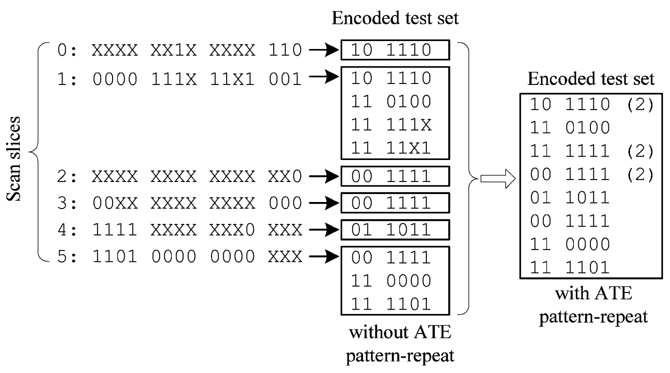


Fig. 6. Encoding example.

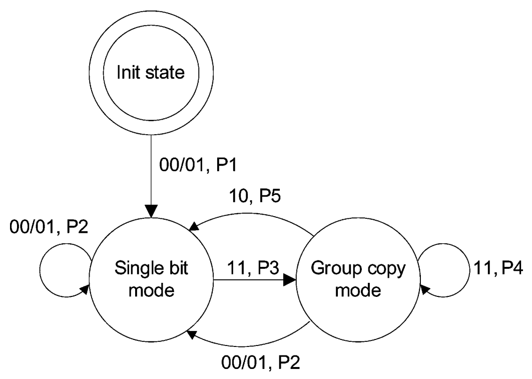


Fig. 7. State transition diagram of the decoder.

that some slice codes, e.g., the first two in the encoded test set, can be combined and applied using ATE pattern repeat.

III. Decompression Architecture

Fig. 7 shows the state transition diagram of the decoder. The decoder enters designated states and performs different operations as specified by the control codes that it receives. Initially, the decoder is in the init state; when it receives an initial-control code, it enters the single-bit mode and performs a series of operations referred to as P1. Table II explains the five groups of operations (P1–P5) in Fig. 7.

Fig. 8 shows the block diagram of the decoder. The finite-state machine (FSM) generates control signals for the other components. The K-bit address register is used in the group-copy mode to store the index of the first bit of the target group. This register can be incremented by K to address a series of adjacent groups. The K-to-N address decoder generates selection signals to address a single bit of the buffer. The N-bit buffer contains combinational logic that provides the following functionalities: 1) each bit in the buffer can be individually addressed and 2) all bits in the same group can be addressed in parallel. These two functions are used in the single-bit mode and the group-copy mode, respectively.

The 2-bit input signal $control$ is the control-code from the tester. The signal rst, when asserted to 0, resets the FSM to its initial state. The signal v is set to high when the decoding process for a slice is finished and the content of the buffer is shifted to the internal scan chains.

If the signal is_grp is asserted, the decoder works in the group-copy mode. The inc_grp is used to increment the address register by K. In the group-copy mode, the K-to-N address decoder receives input from the address register; in the single-bit-mode, it receives input from the data-code input. The N-bit selection (sel) signal is used to address a single bit in the buffer. At any given time only one of the N wires is asserted.

The second bit of the control code $(\mathrm{control})$, i.e., the target symbol, is latched to ts. In the single-bit mode, the specified bit is set to ts. If the control code is 00 or 01, the signal set_buf is asserted, and all other bits except the specified bit are set to the complement of ts $(\overline{ts})$. The signal den is asserted whenever the buffer contents are to be changed. The signal den is set to 0 only during the first clock cycle of the group-copy mode; at the same time the address of the first bit of the group is loaded to the address register.

The sel signal from the address decoder can only address a single bit of the buffer. However, in the group-copy mode, all the K bits in the target group should be addressed (the last group

TABLE II
FIVE GROUPS OF OPERATIONS FOR THE DECODER

Group	Description
P1	If the control-code is 00 (01), the target-symbol is 1 (0). Set the bit specified by the associated data-code to the target-symbol and all other bits to the complemented value.
P2	Shift the current content of the buffer to the internal scan chains, then perform P1.
P3	Save the value of the data-code to an address register, which is the index of the first bit of the group.
P4	Copy the value of the data-code to the group specified by the address register, then increment the address register by K.
P5	Set the bit specified by the data-code to the target-symbol.

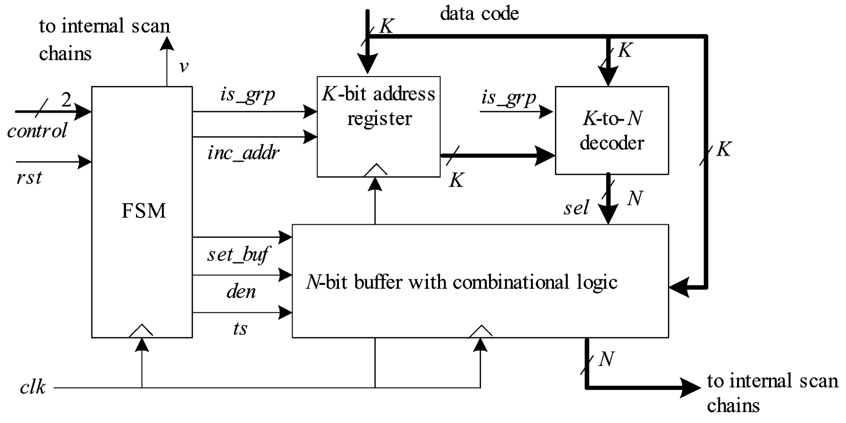


Fig. 8. Block diagram of the decoder.

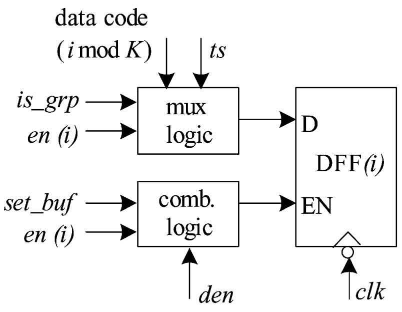


Fig. 9. One bit of the buffer.

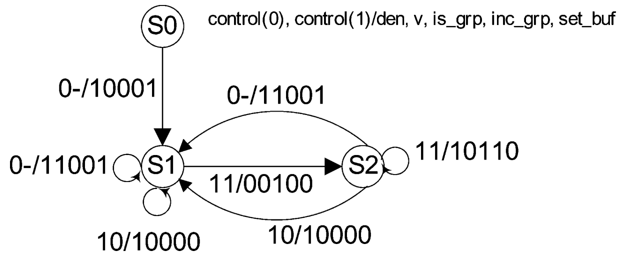


Fig. 10. Decode FSM state diagram.

may contain less bits). Therefore, in the group-copy mode, additional combinational logic is needed to address the other bits together with the first bit. For each bit i of the buffer, an N-bit signal $en(i)$ is defined, where

$$en(i) = \begin{cases} \text{sel}(i), & \text{if } i \bmod K = 0 \\ \text{sel}(i) \vee (is\_grp \wedge \\ \text{sel}\left(i - (i \bmod K)\right), & \text{if } i \bmod K \neq 0 \end{cases}$$

where $i = 0, 1, \ldots, N-1$.

Fig. 9 shows the structure of a single bit of the buffer. Each bit is represented by a falling edge-triggered D-flipflop (DFF) with enable input (EN). The DFF receives input data from a multiplexer and receives the enable signal from the combinational logic. In the group-copy mode, the input is from the data code from the tester. Bit i of the buffer is connected to bit $i \bmod K$ of the data code. In the single-bit mode, the input is either ts or $\overline{ts}$, as determined by $en(i)$. The multiplexer implements the following function:

$$D = \begin{cases} \text{data code}(i \bmod K), & \text{if } is\_grp = 1 \\ \overline{ts}, & \text{if } is\_grp = 0, en(i) = 1 \\ ts, & \text{if } is\_grp = 0, set\_buf = 1. \end{cases}$$

The DFF is changed only when EN is high. The combinational logic implements the boolean function

$$EN = den \wedge (set\_buf \vee en(i)).$$

The DFF's are falling edge-triggered and the FSM is rising edge-triggered. Therefore, upon the rising edge of the clk signal, all control signals are generated and become stable before the falling edge of clk. The buffer is updated at the falling edge of clk.

The state diagram of the FSM is shown in Fig. 10. The state $S0$ is the initial state. States $S1$ and $S2$ correspond to the single-bit mode and the group-copy mode, respectively.

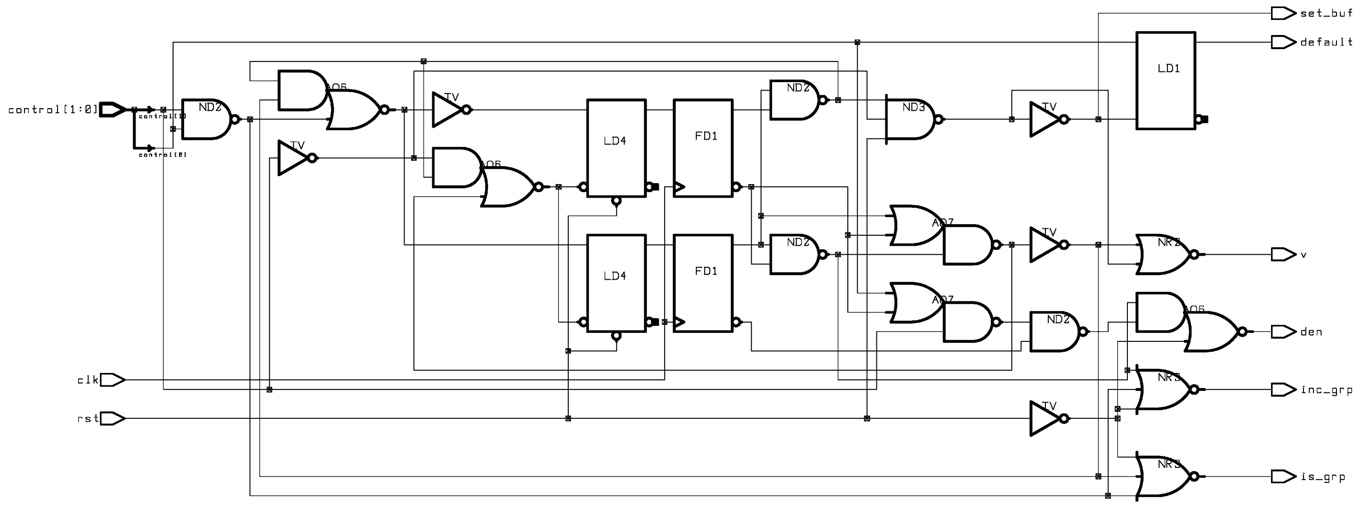


Fig. 11. Synthesized FSM in the decoder.

TABLE III
DESCRIPTION OF INDUSTRIAL CIRCUITS AND TEST SETS

Circuit	No. of gates	No. of scan cells	No. of test cubes	Size of T_D (bits)	No. of faults	Fault coverage	Percentage of specified bits
ckt-1	51,082	12,256	3,768	46,180,608	NA	99.80%	2.18
ckt-2	94,340	22,216	2,636	58,561,376	NA	99.76%	4.27
ckt-3-2000	121,470	9,628	2,191	21,094,948	NA	98.66%	1.96
ckt-3-1000			2,409	23,193,852	NA	98.51%	2.04
ckt-3-500			3,072	29,577,216	NA	98.45%	1.83
ckt-3-200			4,927	47,437,156	NA	98.27%	1.32
ckt-4	302,714	43,414	1,528	66,336,592	1,950,356	99.42%	1.58
ckt-5	404,860	26,970	4,899	132,126,030	1,254,760	98.85%	1.31
ckt-6	1.18M	$\sim$ 80,000	2,859	231,601,872	NA	97.86%	2.58
ckt-7	1.21M	$\sim$ 20,000	18,027	400,289,535	NA	99.16%	1.76
ckt-8	1.41M	$\sim$ 110,000	18,142	1,974,992,546	NA	95.07%	0.92

NA: Not Available.

We simulated the decoder using VHDL and Synopsys tools to ensure its correct operation. We also synthesized the decoder using Synopsys Design Compiler to assess the hardware overhead. The synthesized FSM contains only 5 flip-flops and 23 combinational gates. For the lsi_10k library, the reported area is 55 units. The other parts of the decoder are synthesized separately since they depend on K and N. For $N = 64$ and $K = 7$, the synthesized circuit contains 536 gates and 71 flip-flops, and the area is 1341 units. If $N = 1024$ and $K = 11$, the synthesized circuit contains 6409 gates and 1035 flip-flops, and the area is 18 877 units. For the larger than million-gate designs considered in our experiments, this corresponds to an area overhead of only 1%. The schematic of the FSM is shown in Fig. 11.

IV. Experimental Results

In this section, we apply the proposed approach to eight representative industrial circuits. These circuits vary in size from approximately 50-K gates to over 1.4-M gates. For each circuit, we compress test sets with high fault coverage that were provided to us by industrial partners. These test sets are generated by commercial ATPG tools with dynamic compaction turned on and random fill turned off. The percentages of specified bits in these test sets are approximately in the range 1%–4%. During the ATPG flow, fault grading is performed to drop accidentally detected faults once a test pattern is obtained after dynamic compaction. However, as required by any other compression method, the ATPG tool cannot randomly fill all unspecified bits before fault grading. This restriction increases the number of test patterns but such a tradeoff is generally considered to be inevitable for test compression methods that exploit don't care bits in test cubes.

Table III describes these circuits and the corresponding test sets. The test sets for ckt-1 and ckt-2 are obtained after applying 64 000 random vectors. For ckt-3, we were provided with four different sets of test cubes. The differences in these test sets lie in the maximum number of care bits in each test pattern. For example, in ckt-3-2000, each test pattern is constrained to have no more than 2000 care bits.

We do not report compression results for the ISCAS'89 benchmark circuits because they are too small to be representative of today's designs. Moreover, their test sets contain too many care bits, due in part to their large logic depth.

Table IV shows the compression results obtained using the proposed method for the different test cases. Column N refers to the number of internal scan chains and column c denotes the number of ATE channels. We consider a varying number of internal scan chains N to show how an appropriate value of N can be determined for designs with flexible scan chains.

TABLE IV
COMPRESSION RESULTS

Circuit	N	c	Without ATE pattern-repeat				With ATE pattern-repeat		
			TAT (cycles)	$\|T_E\|$ (bits)	Υ_V (x)	Lower bound estimate on Υ_V	$\|T_E\|$ (bits)	Υ_V (x)	Υ_{TAT} (x)
ckt-1	255	10	400,960	3,971,920	11.63	9.17	3,322,690	13.90	11.53
	511	11	354,260	3,855,412	11.98	8.34	3,325,267	13.89	11.87
	800	12	306,482	3,632,568	12.71	7.65	3,153,672	14.64	12.58
	1,023	12	338,049	4,011,372	11.51	7.65	3,421,764	13.50	11.40
ckt-2	255	10	802,354	7,997,180	7.32	4.68	6,514,040	8.99	7.30
	511	11	739,859	8,109,453	7.22	4.26	6,626,180	8.84	7.20
	700	12	628,795	7,513,908	7.79	3.90	6,108,024	9.59	7.77
	1,023	12	692,295	8,275,908	7.08	3.90	6,540,048	8.95	7.06
ckt-3-2000	255	10	208,556	2,063,650	10.22	10.20	1,424,740	14.81	10.13
	511	11	180,526	1,961,685	10.75	9.28	1,448,392	14.56	10.64
	1,023	12	164,771	1,950,960	10.81	8.50	1,440,396	14.65	10.69
ckt-3-1000	255	10	232,104	2,296,950	10.10	9.80	1,784,390	13.00	10.01
	511	11	208,640	2,268,541	10.22	8.91	1,909,611	12.15	10.13
	1,023	12	198,263	2,350,248	9.87	8.17	1,942,968	11.94	9.77
ckt-3-500	255	10	256,100	2,530,280	11.69	10.93	2,042,330	14.48	11.56
	511	11	234,888	2,549,976	11.60	9.94	2,277,814	12.98	11.47
	1,023	12	228,603	2,706,372	10.93	9.11	2,423,400	12.20	10.80
ckt-3-200	255	10	279,450	2,745,230	17.28	15.15	2,215,140	21.42	17.00
	511	11	253,568	2,735,051	17.34	13.77	2,506,174	18.93	17.04
	1,023	12	253,761	2,986,008	15.89	12.63	2,798,136	16.95	15.61
ckt-4	255	10	508,588	5,070,600	13.08	12.66	3,264,850	20.32	13.05
	511	11	411,454	4,509,186	14.71	11.51	3,391,454	19.56	14.66
	1,023	12	369,497	4,415,628	15.02	10.55	3,505,908	18.92	14.97
	2,047	13	347,795	4,501,471	14.74	9.74	3,574,064	18.56	14.68
ckt-5	255	10	816,757	8,118,580	16.27	15.27	6,079,410	21.73	16.18
	511	11	677,339	7,396,840	17.86	13.88	6,462,555	20.44	17.74
	1,023	12	662,548	7,891,788	16.74	12.72	7,178,544	18.41	16.63
	2,047	13	669,840	8,644,233	15.28	11.74	7,831,798	16.87	15.18
ckt-6	255	10	2,543,116	25,402,570	9.12	7.75	22,531,820	10.28	9.11
	511	11	2,510,478	27,583,809	8.40	7.05	25,517,118	9.08	8.39
	1,023	12	2,535,011	30,385,824	7.62	6.46	27,914,232	8.30	7.61
	2,047	13	2,540,256	32,986,161	7.02	5.96	29,083,392	7.96	7.02
ckt-7	255	10	2,939,692	29,216,650	13.70	11.36	25,847,430	15.49	13.63
	511	11	2,845,279	31,099,772	12.87	10.33	29,058,689	13.78	12.80
	1,023	12	2,915,940	34,774,956	11.51	9.47	32,587,644	12.28	11.45
	2,047	13	2,992,613	38,669,618	10.35	8.74	35,185,618	11.38	10.30
ckt-8	255	10	9,835,025	98,168,830	20.12	21.74	68,528,200	28.82	20.08
	511	11	7,839,684	86,036,962	22.96	19.76	74,718,468	26.43	22.91
	1,023	12	7,607,971	91,077,948	21.68	18.12	86,462,016	22.84	21.64
	2,047	13	7,809,912	101,293,010	19.50	16.72	98,412,002	20.07	19.46

TABLE V
EFFECTIVENESS OF THE GROUP-COPY MODE

Circuit	# target_ symbols	Without group-copy		With group-copy	
		$\|T_E\|$ (bits)	Υ_V (x)	$\|T_E\|$ (bits)	Υ_V (x)
ckt-6	3,409,136	40,909,632	5.66	30,385,824	7.62
ckt-7	4,135,288	49,623,456	8.06	34,774,956	11.51
ckt-8	11,203,091	134,437,092	14.69	91,077,948	21.68

The test application time and the size of compressed data are shown in columns TAT and T_E, respectively. The parameter $\Upsilon_V = |T_D|/|T_E|$ refers to the data volume reduction factor. The parameter Υ_{TAT} is the test application time reduction factor over standard scan testing based on c ATE channels. Without ATE pattern repeat, Υ_V and Υ_{TAT} are as high as 22.96$\times$ and 22.91$\times$, respectively. With ATE pattern repeat, Υ_V is as high as 28.82$\times$. For ckt-8 with 255 internal scan chains, the compression of 20.12$\times$ is close to the theoretical maximum of 25.5$\times$ (without ATE pattern repeat). The CPU time for generating the compressed data is at most two minutes, even for the largest circuits.

The estimate of lower bounds obtained using the probabilistic method described in Section II-A on the compression factor are also listed. These lower bounds are close to the compression factor determined experimentally; hence, the easily computed lower bound serves as a good estimate for the compression factor. In the case of ckt-8 ($N = 255$), the estimated lower bound is slightly larger than the actual compression factor. This is because we use (2) to compute the lower bound, which assumes that the care-bit density is the same for every scan slice. It is for this reason that the lower bound claim is valid only in a probabilistic sense, and the lower bound is presented as an estimate. Since the test set for ckt-8 has a very low care-bit

TABLE VI
COMPARION WITH 2-D COMPRESSION [21]

Circuit	N	[21]					Proposed method				
		c	TAT (cycles)	$\|T_E\|$ (bits)	Υ_V (x)	Υ_{TAT} (x)	c	TAT (cycles)	$\|T_E\|$ (bits)	Υ_V (x)	Υ_{TAT} (x)
ckt-1	255	20	195,799	3,840,620	12.02	11.82	10	400,960	3,322,690	13.90	11.53
	511	30	126,840	3,692,160	12.51	12.18	11	354,260	3,325,267	13.89	11.87
	1023	45	65,712	2,787,480	16.57	15.71	12	338,049	3,421,764	13.50	11.40
ckt-3-2000	255	191	36,771	6,604,780	3.19	3.10	10	208,556	1,424,740	14.81	10.13
	511	311	19,671	5,436,280	3.88	3.56	11	180,526	1,448,392	14.56	10.64
	1023	476	11,321	4,345,880	4.85	4.26	12	164,771	1,440,396	14.65	10.69
ckt-3-1000	255	147	46,451	6,474,174	3.58	3.47	10	232,104	1,784,390	13.00	10.01
	511	228	24,487	5,033,784	4.61	4.33	11	208,640	1,909,611	12.15	10.13
	1023	349	14,149	4,097,260	5.66	4.94	12	198,263	1,942,968	11.94	9.77
ckt-3-500	255	106	61,744	6,219,232	4.76	4.58	10	256,100	2,042,330	14.48	11.56
	511	166	32,674	4,913,932	6.02	5.55	11	234,888	2,277,814	12.98	11.47
	1023	248	18,802	3,901,040	7.58	6.54	12	228,603	2,423,400	12.20	10.80
ckt-3-200	255	67	116,191	7,454,688	6.36	6.15	10	279,450	2,215,140	21.42	17.00
	511	103	62,326	5,912,097	8.02	7.51	11	253,568	2,506,174	18.93	17.04
	1023	149	36,207	4,660,720	10.18	8.98	12	253,761	2,798,136	16.95	15.61
ckt-4	255	235	178,513	41,591,475	1.59	1.59	10	508,588	3,264,850	20.32	13.05
	511	466	89,673	41,075,570	1.61	1.62	11	411,454	3,391,454	19.56	14.66
	1023	898	46,033	39,965,490	1.66	1.66	12	369,497	3,505,908	18.92	14.97
ckt-5	255	241	400,809	95,414,310	1.38	1.38	10	816,757	6,079,410	21.73	16.18
	511	453	202,801	89,649,606	1.47	1.47	11	677,339	6,462,555	20.44	17.74
	1023	798	105,744	80,474,310	1.64	1.62	12	662,548	7,178,544	18.41	16.63

density, the distribution of care bits among scan slices has a greater impact on the compression factor.

To demonstrate the effectiveness of the group-copy mode, Table V lists the compression results obtained with and without the group-copy-mode. The experiments were conducted on the three largest circuits and with 1023 internal scan chains. Table V clearly shows that the group-copy-mode results in a significant reduction in the test data volume.

We next compare the proposed compression method to four other recent compression methods that have been proposed for IP cores. These methods also do not require fault simulation or test generation. To ensure fairness of comparison, we do not consider compression methods that require structural information about the core under test. For some of these compression methods, we do not report the comparison results for ckt 68 because the CPU time was excessive and we ran out of memory.

Table VI presents comparative data for 2-D compression [21]. The compression method in [21] was implemented and applied to a number of industrial test cases. In every case considered, the number of ATE channels required is much less for the proposed method compared to [21]. Out of the 21 cases considered, $|T_E|$ for [21] is higher in 20 cases. The value of Υ_{TAT} in [21] is smaller in 18 cases.

The test sets described in Table III were obtained using dynamic compaction during ATPG. As is the case of other compression methods, these test sets were not compressed further using static compaction after ATPG. In some cases, e.g., ckt-3, a commercial ATPG tool was given a maximum number of care bits per vector as a constraint. It was reported in [21] that the compressed test sets for ckt-1 and ckt-2 are an order of magnitude smaller than the compacted test sets used during production testing for these circuits. Therefore, the proposed method can achieve significant reduction in data volume over ATPG-compacted test sets.

Table VII compares the proposed method to the recent compression technique based on dictionaries with corrections [18]. We implemented the procedures from [18] and applied this technique to several industrial test cases. Table VII is similar to Table VI, with an additional column mem that shows the size of the on-chip memory. Out of the 24 cases considered, $|T_E|$ in [18] is higher in 15 cases. Note that for the nine cases where $|T_E|$ in [18] is less, an excessive amount of on-chip storage (as high as 8 Mbits) is needed for [18]. Hence, it is difficult to use [18] in practice for these cases. TAT is lower in [18] in most cases, but it also requires a much larger number of ATE channels. If the number of ATE channels and the amount of on-chip storage are limited (or constrained), the proposed method outperforms [18] both in terms of T_E and TAT.

We also compare the proposed method to the adjustable-width linear combinational decompression method [28]. We implemented the compression procedures from [28] and applied this technique to several industrial test cases. Since [28] does not report the structure of the XOR network, we generated random XOR networks that satisfied the linear independence requirements. We compared the two methods for the same values of c, the number of ATE channels. As can be seen from Table VIII, the comparison results are mixed. For ckt-1 and ckt-6, [28] outperforms the proposed method in most scan chain configurations. For ckt-2, ckt-3-2000, and ckt-4, the proposed method achieves better results for all configurations. However, if the ATE pattern repeat feature is turned-off for the proposed method, then [28] yield better results in most cases.

We next compare the proposed method to the test-data mutation method proposed in [29]. There is some similarity between the two techniques, since they both maintain a scan slice buffer and obtain a new scan slice by flipping some bits in the buffer. In [29], a new scan slice is obtained from the previous slice by flipping the conflicting bits. Instead of directly encoding the

TABLE VII
COMPARISON WITH RECENT DICTIONARY-BASED COMPRESSION METHOD [18]

Circuit	N	[18]						Proposed method				
		c	mem (bits)	TAT (cycles)	$\|T_E\|$ (bits)	Υ_V (x)	Υ_{TAT} (x)	c	TAT (cycles)	$\|T_E\|$ (bits)	Υ_V (x)	Υ_{TAT} (x)
ckt-1	255	19	295,176	188,400	3,508,008	12.14	12.94	10	400,960	3,322,690	13.90	11.53
	511	21	1,126,244	94,200	1,899,072	15.26	23.40	11	354,260	3,325,267	13.89	11.87
	1023	22	3,433,920	48,984	994,752	10.43	43.00	12	338,049	3,421,764	13.50	11.40
ckt-2	255	20	946,726	234,604	4,639,360	10.48	12.49	10	802,354	6,514,040	8.99	7.30
	511	22	3,350,170	118,620	2,551,648	9.92	22.47	11	739,859	6,626,180	8.84	7.20
	1023	24	8,589,040	60,628	1,391,808	5.87	40.30	12	692,295	6,540,048	8.95	7.06
ckt-3-2000	255	19	401,574	85,449	1,581,902	10.64	13.03	10	208,556	1,424,740	14.81	10.13
	511	20	834,015	43,820	832,580	12.66	24.15	11	180,526	1,448,392	14.56	10.64
	1023	21	1,523,466	24,101	460,110	10.63	41.82	12	164,771	1,440,396	14.65	10.69
ckt-3-1000	255	19	354,584	93,951	1,739,298	11.08	13.03	10	232,104	1,784,390	13.00	10.01
	511	20	919,698	48,180	915,420	12.64	24.15	11	208,640	1,909,611	12.15	10.13
	1023	21	1,909,629	26,499	505,890	9.60	41.82	12	198,263	1,942,968	11.94	9.77
ckt-3-500	255	18	152,146	119,808	2,101,248	13.13	13.74	10	256,100	2,042,330	14.48	11.56
	511	20	614,484	61,440	1,167,360	16.60	24.15	11	234,888	2,277,814	12.98	11.47
	1023	21	1,685,250	33,792	645,120	12.69	41.82	12	228,603	2,423,400	12.20	10.80
ckt-3-200	255	15	23,622	192,153	2,808,390	16.75	16.49	10	279,450	2,215,140	21.42	17.00
	511	18	146,523	98,540	1,685,034	25.90	26.80	11	253,568	2,506,174	18.93	17.04
	1023	20	648,099	54,197	985,400	29.04	43.91	12	253,761	2,798,136	16.95	15.61
ckt-4	255	20	871,474	262,816	5,225,760	10.88	12.63	10	508,588	3,264,850	20.32	13.05
	511	21	1,568,770	131,408	2,727,480	15.44	24.06	11	411,454	3,391,454	19.56	14.66
	1023	22	2,880,520	67,232	1,445,488	15.33	44.89	12	369,497	3,505,908	18.92	14.97
ckt-5	255	20	574,260	524,193	10,385,880	12.06	12.62	10	816,757	6,079,410	21.73	16.18
	511	21	1,363,323	264,546	5,452,587	19.39	23.81	11	677,339	6,462,555	20.44	17.74
	1023	22	3,347,556	137,172	2,910,006	21.11	43.82	12	662,548	7,178,544	18.41	16.63

TABLE VIII
COMPARISON WITH THE ADJUSTABLE WIDTH LINEAR COMBINATIONAL DECOMPRESSION [28]

Circuit	c	N	$\|T_E\|$ (bits) [28]	c	N	$\|T_E\|$ (bits) (proposed)
ckt-1	10	256	**2,870,750**	10	255	3,322,690
	11	512	**2,691,051**	11	511	3,325,267
	12	765	**3,175,524**	12	765	3,655,584
	12	1017	4,616,184	12	1017	**3,543,552**
ckt-2	10	256	7,918,130	10	255	**6,514,040**
	11	512	8,965,979	11	511	**6,626,180**
	12	765	10,905,648	12	765	**6,757,632**
	12	1017	13,235,052	12	1017	**6,540,048**
ckt-3-2000	10	256	1,760,420	10	255	**1,424,740**
	11	512	1,719,135	11	511	**1,448,392**
	12	765	1,972,728	12	765	**1,477,644**
	12	1017	2,291,940	12	1017	**1,440,396**
ckt-4	10	256	4,296,650	10	255	**3,264,850**
	11	512	3,663,759	11	511	**3,391,454**
	12	1017	4,428,960	12	1017	**3,696,984**
	13	2000	6,411,119	13	2000	**3,781,258**
ckt-6	11	512	**13,583,339**	11	511	25,517,118
	12	1017	**23,936,424**	12	1017	27,914,232
	13	1800	51,557,207	13	1800	**29,417,596**
	13	2000	63,092,003	13	2000	**29,257,267**

positions of the conflicting bits, an FSM is used in [29] to determine which bit should be flipped. The FSM receives a 1-bit data stream and makes a transition between its states whenever it receives a 1-bit data. Each state of the FSM corresponds to a bit in the buffer. Another control data stream is required to specify whether the bit corresponding to the current state of the FSM should be flipped. We implemented the compression procedures from [29] and applied this technique to the five largest industrial test cases. The authors of [29] did not specify how the control data stream is generated, and the corresponding control overhead was not reported. In our implementation, we assume that each data bit is accompanied by a control bit. Table IX shows that the proposed method leads to lower test data volume than [29] for all circuits. The test application time for [29] is significantly higher because it relies on a very narrow ATE interface, e.g., single ATE channel. It appears that the main goal of [29] is to reduce test data volume using a very small number of ATE channels, hence, a direct comparison with the proposed method is difficult.

Finally, as discussed in Table IV, and in more detail in [30], the number of internal scan chains N that leads to the highest data volume reduction does not always equal the value of N that leads to the maximum TAT reduction for IP cores with flexible scan chains. This information can be used to determine appropriate scan chain configurations.

TABLE IX
COMPARISON WITH THE TEST DATA MUTATION METHOD [29]

Circuit	N	$\lvert T_E \rvert$ (bits) [29]	Υ_V (x) [29]	N	$\lvert T_E \rvert$ (bits) (proposed)	Υ_V (x) (proposed)
ckt-4	256	4,432,084	15.52	255	**3,264,850**	20.32
	512	4,593,348	14.98	511	**3,391,454**	19.56
	1024	4,817,416	14.29	1023	**3,505,908**	18.92
ckt-5	256	8,058,590	16.58	255	**6,079,410**	21.73
	512	8,515,072	15.69	511	**6,462,555**	20.44
	1024	9,135,990	14.62	1023	**7,178,544**	18.41
ckt-6	256	23,552,456	9.83	255	**22,531,820**	10.28
	512	**25,078,472**	9.24	511	25,517,118	9.08
	1024	**23,936,424**	12	1023	27,914,232	8.30
ckt-7	256	40,541,964	9.87	255	**25,847,430**	15.49
	512	42,455,206	9.43	511	**29,058,689**	13.78
	1024	43,467,486	9.21	1023	**32,587,644**	12.28
ckt-8	256	90,489,328	21.83	255	**68,528,200**	28.82
	512	96,713,420	20.42	511	**74,718,468**	26.43
	1024	100,958,308	19.56	511	**86,462,016**	22.84

V. CONCLUSION

We have presented a test data compression technique for designs with multiple scan chains. This method does not require detailed structural information about the CUT, and utilizes a generic on-chip decoder that is independent of the CUT and the test set. While the hardware overhead depends on the number of internal scan chains, we have seen that for an industrial circuit with over 1-M gates, the overhead is only 1% for as many as 1024 internal scan chains. If a small amount of circuit redesign is permitted, we can reduce the hardware overhead by modifying the first scan cells of each scan chain such that they can be used as the N-bit on-chip buffer. The clock inputs of these scan cells need to be appropriately gated so that they can be triggered separately from other cells in the same scan chain. The area overhead we report in Section III is for the decoder that contains the scan slice buffer and does not manipulate the scan clock.

Experimental results for eight industrial circuits show that compared to dynamically compacted test sets, up to 28× reduction in test data volume and 20× reduction in test application time can be obtained.

ACKNOWLEDGMENT

The authors would like to thank S. Mitra and A. Chandra for their helpful suggestions. They are grateful to a number of colleagues from the industry who provided test sets (ckt-1 to ckt-10) for their experiments. These contributors wish to remain anonymous.

REFERENCES

[1] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 5, pp. 776–792, May 2004.

[2] S. Mitra and K. S. Kim, "X-Compact: An efficient response compaction technique," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 3, pp. 421–432, Mar. 2004.

[3] B. Koenemann, C. Banhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, "A SmartBIST variant with guaranteed encoding," in *Proc. Asia Test Symp.*, 2001, pp. 325–330.

[4] S. Mitra and K. S. Kim, "XMAX: X-tolerant architecture for maximal test compression," in *Proc. IEEE Int. Conf. Comput. Des.*, 2003, pp. 326–330.

[5] Semiconductor Industry Association, International Technology Roadmap for Semiconductors (ITRS), San Jose, CA, 2003. [Online]. Available: http://public.itrs.net/Files/2003ITRS/Home2003.htm

[6] Semiconductor Industry Association, International Technology Roadmap for Semiconductors (ITRS), San Jose, CA, 2005 [Online]. Available: http://www.itrs.net/Common/2005ITRS/Home2005.htm

[7] H. Vranken, F. Hapke, S. Rogge, D. Chindamo, and E. Volkerink, "ATPG padding and ATE vector repeat per port for reducing test data volume," in *Proc. Int. Test Conf.*, 2003, pp. 1069–1076.

[8] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic BIST for large industrial designs: Real issues and case studies," in *Proc. Int. Test Conf.*, 1999, pp. 358–367.

[9] B. Koenemann, "LFSR-coded test patterns for scan design," in *Proc. Eur. Test Conf.*, 1991, pp. 237–242.

[10] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-in test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers," *IEEE Trans. Computers*, vol. 44, no. 2, pp. 223–233, Feb. 1995.

[11] C. Krishna and N. A. Touba, "Reducing test data volume using LFSR reseeding with seed compression," in *Proc. Int. Test Conf.*, 2002.

[12] A. A. Al-Yamani and E. J. McCluskey, "Built-in reseeding for serial BIST," in *Proc. IEEE VLSI Test Symp.*, 2003, pp. 63–68.

[13] I. Bayraktaroglu and A. Orailoglu, "Test volume and application time reduction through scan chain concealment," in *Proc. Des. Autom. Conf.*, 2001, pp. 151–155.

[14] S. Samaranayake, E. Gizdarski, N. Sitchinava, F. Neuveux, R. Kapur, and T. W. Williams, "A reconfigurable shared scan-in architecture," in *Proc. IEEE VLSI Test Symp.*, 2003, pp. 9–14.

[15] I. Bayraktaroglu and A. Orailoglu, "Decompression hardware determination for test volume and time reduction through unified test pattern compaction and compression," in *Proc. IEEE VLSI Test Symp.*, 2003, pp. 113–118.

[16] M. Chandramouli, "How to implement deterministic logic built-in self-test (BIST)," *Compiler: A Monthly Magazine for Technologies World-wide, Synopsys*, Jan. 2003.

[17] P. T. Gonciari, B. 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.

[18] A. Wrtenberger, C. S. Tautermann, and S. Hellebrand, "Data compression for multiple scan chains using dictionaries with corrections," in *Proc. Int. Test Conf.*, 2004, pp. 926–934.

[19] 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, pp. 91–115, Jan. 2005.

[20] A. Jas, J. Gosh-Dastidar, and N. A. Touba, "Scan vector compression/decompression using statistical coding," in *Proc. IEEE VLSI Test Symp.*, 1999, pp. 114–120.

[21] 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. IEEE Int. Conf. VLSI Des.*, 2005, pp. 53–58.

[22] F. G. Wolff, C. Papachristou, and D. R. McIntyre, “Test compression and hardware decompression for scan-based SoCs,” in *Proc. DATE Conf.*, 2004, pp. 716–717.

[23] V. De and S. Borkar, “Technology and design challenges for low power and high performance,” in *Proc. Int. Symp. Low Power Electron. Des.*, 1999, pp. 163–168.

[24] T. Hiraide, K. O. Boateng, H. Konishi, K. Itaya, M. Emori, H. Yamanaka, and T. Mochiyama, “BIST-aided scan testA new method for test cost reduction,” in *Proc. IEEE VLSI Test Symp.*, 2003, pp. 359–364.

[25] J. Rajski, J. Tyszer, C. Wang, and S. M. Reddy, “Convolutional compaction of test responses,” in *Proc. Int. Test Conf.*, 2003, pp. 745–754.

[26] J. H. Patel, S. S. Lumetta, and S. M. Reddy, “Application of Saluja-Karpovsky compactors to test responses with many unknowns,” in *Proc. IEEE VLSI Test Symp.*, 2003, pp. 107–112.

[27] K. Miyase and S. Kajihara, “XID: Don’t care identification of test patterns for combinational circuits,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 2, pp. 321–326, Feb. 2004.

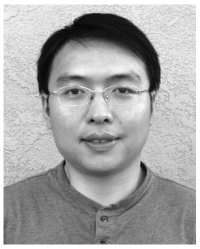
[28] C. V. Krishna and N. A. Touba, “Adjustable width linear combinational scan vector decompression,” in *Proc. Int. Conf. Comput.-Aided Des.*, 2003, p. 863.

[29] S. Reda and A. Orailoglu, “Reducing test application time through test data mutation encoding,” in *Proc. DATE Conf.*, 2002, pp. 387–393.

[30] Z. Wang and K. Chakrabarty, “Test data compression for IP embedded cores using selective encoding of scan slices,” in *Proc. Int. Test Conf.*, 2005, pp. 477–486.

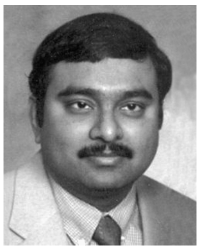
[31] M. Naruse, I. Pomeranz, S. M. Reddy, and S. Kundu, “On-chip compression of output responses with unknown values using LFSR reseeding,” in *Proc. Int. Test Conf.*, 2003, pp. 1060–1068.

[32] S. M. Reddy, K. Miyase, S. Kajihara, and I. Pomeranz, “On test data volume reduction for multiple scan chain designs,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 8, pp. 460–469, Oct. 2003.



Zhanglei Wang (M’08) received the B.Eng. degree from Tsinghua University, Beijing, China, in 2001, and the M.S.E. and Ph.D. degrees from Duke University, Durham, NC, in 2004 and 2007, respectively, all in computer and electrical engineering.

He is currently a Hardware Design Engineer with Cisco Systems, Inc., San Jose, CA. His research interests include test compression, test pattern grading, and test generation.



Krishnendu Chakrabarty (S’92–M’96–SM’01–F’08) received the B.Tech. degree from the Indian Institute of Technology, Kharagpur, India, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively, all in computer science and engineering.

He is currently a Professor with the Electrical and Computer Engineering Department, Duke University, Durham, NC. His current research projects include: testing and testing and design-for-testability of system-on-chip integrated circuits; digital microfluidic biochips; logic circuits based on DNA self-assembly; delay-tolerant wireless networks. He has authored five books *Microelectrofluidic Systems: Modeling and Simulation* (CRC Press, 2002), *Test Resource Partitioning for System-on-a-Chip* (Kluwer, 2002), *Scalable Infrastructure for Distributed Sensor Networks* (Springer, 2005), *Digital Microfluidics Biochips: Synthesis, Testing, and Reconfigutaion Techniques* (CRC Press, 2006), and *Adaptive Cooling of Integrated Circuits using Digital Microfluidics* (Artech House, 2007) and edited the book volumes *SOC (System-on-a-Chip) Testing for Plug and Play Test Automation* (Kluwer, 2002) and *Design Automation Methods and Tools for Microfluidics-Based Biochips* (Springer, 2006). He has contributed over a dozen invited chapters to book volumes, published 280 papers in archival journals and refereed conference proceedings, and delivered over 110 keynote, plenary, and invited talks. He holds one U.S. patent in built-in self-test and is a coinventor of a pending U.S. patent on sensor networks

Prof. Chakrabarty was a recipient of the National Science Foundation Early Faculty (CAREER) Award, the Office of Naval Research Young Investigator Award, Best Paper Awards from the *2007 IEEE International Conference on VLSI Design*, the *2005 IEEE International Conference on Computer Design*, and the *2001 IEEE Design, Automation and Test in Europe (DATE) Conference*, the Humboldt Research Fellowship, awarded by the Alexander von Humboldt Foundation, Germany, and the Mercator Visiting Professorship, awarded by the Deutsche Forschungsgemeinschaft, Germany. He served as a Distinguished Visitor of the IEEE Computer Society for 20052007 and a Distinguished Lecturer of the IEEE Circuits and Systems Society for 20062007. Currently, he serves as an ACM Distinguished Speaker. He is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS, *ACM Journal on Emerging Technologies in Computing Systems*, an Editor of the *IEEE Design & Test of Computers*, and an Editor of the *Journal of Electronic Testing: Theory and Applications (JETTA)*. He served as Program Chair for the IEEE Asian Test Symposium in 2005 and the CAD, Design, and Test Conference for the 2007 IEEE Symposium on Design, Integration, Test, and Packaging of MEMS/MOEMS. Prof. Chakrabarty is a senior member of the ACM and a member of Sigma Xi. He is a recipient of the IEEE Computer Society Meritorious Service Award and Duke University’s 2008 Dean’s Award for Excellence in Mentoring (http://www.gradschool.duke.edu/our_faculty/mentoring_awards/index.html).