Embedded Deterministic Test

Janusz Rajski, *Associate Member, IEEE*, Jerzy Tyszer, *Senior Member, IEEE*, Mark Kassab, *Member, IEEE*, and Nilanjan Mukherjee, *Member, IEEE*

***Abstract*—This paper presents a novel test-data volume-compression methodology called the embedded deterministic test (EDT), which reduces manufacturing test cost by providing one to two orders of magnitude reduction in scan test data volume and scan test time. The presented scheme is widely applicable and easy to deploy because it is based on the standard scan/ATPG methodology and adopts a very simple flow. It is nonintrusive as it does not require any modifications to the core logic such as the insertion of test points or logic bounding unknown states. The EDT scheme consists of logic embedded on a chip and a new deterministic test-pattern generation technique. The main contributions of the paper are test-stimuli compression schemes that allow us to deliver test data to the on-chip continuous-flow decompressor. In particular, it can be done by repeating certain patterns at the rates, which are adjusted to the requirements of the test cubes. Experimental results show that for industrial circuits with test cubes with very low fill rates, ranging from 3% to 0.2%, these schemes result in compression ratios of 30 to 500 times. A comprehensive analysis of the encoding efficiency of the proposed compression schemes is also provided.**

***Index Terms*—Design for testability (DFT), linear finite state machines, manufacturing test, test application time reduction, test data volume compression.**

I. Introduction

DESIGN for testability (DFT) based on scan and automatic test pattern generation (ATPG) has been adopted as a reliable and broadly acceptable methodology that provides very high test coverage. The process of scan insertion and test generation is automated and guarantees very high predictability and quality of results. Conventional ATPG systems can generate test sets that guarantee very high fault coverage of several types of fault models. Typically, when an ATPG targets multiple faults, only a small number of scan cells gets specified. The remaining positions are filled with random values. Such a fully specified pattern is more likely to detect additional faults, and can be stored on a tester. As a result of random fill, however, the test patterns are grossly overspecified.

For large circuits, the growing test-data volume causes a significant increase in test cost because of much longer test time and large tester-memory requirements. For a scan-based test, the test data volume is proportional to the number of scan patterns and the number of scan cells, while test application time can be easily obtained as a ratio of the volume of test data to the number of scan chains divided by the scan-shift frequency. Consider a circuit consisting of 10 M gates and 16 scan chains. Assuming one scan cell per 20 gates (that is, 31 250 cells in each scan chain), applying 10 000 scan patterns at a 20 MHz scan-shift frequency will take roughly 312 million test cycles or equivalently 15 s of test time. As designs grow in size, it becomes increasingly expensive to maintain a high level of test coverage. This is due to a large volume of test data that must be stored in the testing equipment, an increasingly long test application time, and a very high and still increasing logic-to-pin ratio creating a test data transfer bottleneck at the chip pins.

If one accepts certain levels of fault coverage, the above problems can be resolved by using logic built-in self-test (LBIST), which is a solution based on pseudorandom patterns generated and evaluated on a chip. In LBIST, typically up to 95% coverage of stuck-at faults can be achieved provided that test points are employed to address random pattern resistance [12]. Other types of fault models, such as transition or path-delay faults, are not handled efficiently by pseudorandom patterns. In LBIST, all test responses have to be known. Unknown values corrupt the signature and, therefore, have to be bounded by additional test logic. Also, deterministic tests are almost always needed to target the remaining random pattern resistant faults. Very often the memory required to store the top-up patterns in LBIST can exceed 30% of the memory used in a conventional ATPG approach [12]. Because of the increasing circuit complexity, storing an extensive set of ATPG patterns, either on-chip or off-chip, becomes prohibitive. Thus, reduction of test-data volume is recognized as an extremely important problem and, consequently, is receiving a lot of attention [38].

Many methods for compressing test data exploit the fact that test cubes frequently feature a large number of unspecified positions. The original idea, known as LFSR-coding, takes advantage of this phenomenon [19]. It was refined under the name of LFSR-reseeding in a number of approaches [10], [22]. The encoding capability of most of these methods is limited by the LFSR size. Usually, because of linear dependencies, large LFSRs are required, incurring a significant area overhead. Another drawback is that the loading of the seed and loading/unloading of the scan chains are done in two separate nonoverlapping phases, resulting in inefficient utilization of the tester. This can be alleviated by employing shadow registers. Unfortunately, such an approach doubles the number of required memory elements. Finally, as the number of specified bits may vary significantly in successive test cubes, several techniques were also proposed to improve the encoding capability of the conventional reseeding scheme [24], [35].

Another group of methods uses on-chip devices to handle various forms of test pattern compression based on run-length

Manuscript received May 6, 2003; revised July 29, 2003. This paper was recommended by Associate Editor S. Hellebrand.

J. Rajski, M. Kassab, and N. Mukherjee are with the Mentor Graphics Corporation, Wilsonville, OR 97070 USA (e-mail: janusz_rajski@mentor.com; mark_kassab@mentor.com; nilanjan_mukherjee@mentor.com).

J. Tyszer is with the Institute of Electronics and Telecommunications, Poznan University of Technology, 60-965 Poznan, Poland (e-mail: tyszer@et.put.poznan.pl).

Digital Object Identifier 10.1109/TCAD.2004.826558

codes [3], [7], [14], [15], [17], statistical codes [7], [14]–[16], [22], Golomb codes [4], XOR networks [2], [36], hybrid patterns [5], folding counters [11], [24], and reuse of scan chains [6]. Both test application time and test data volume can be reduced by shifting in the same test vector to several scan chains through a single input (a broadcast mode) as presented in [23], and then further refined under the name of Illinois scan [8], [9], [13]. In these schemes a given test pattern must not contain contradictory values on corresponding cells in different chains loaded through the same input. Thus, performance of the schemes strongly relies on the scan chain partitions, and therefore they are not easily scalable.

Decrease of test data along with a $2\times$ scan test-time reduction is also accomplished in the scheme based on the on-product MISR [1], [18]. In this technique, the test responses in conventional scan data are replaced with much more compact signatures, and a tester's repeat capability is used to shift the same values into the scan chains for several consecutive cycles. Further refinement of this approach is proposed in SmartBIST [20], where tester channels are again used to deliver stimuli, while signatures are observed only at the end of an unload cycle. The test patterns are delivered in a compressed form, and an on-chip decoder expands them into the actual data loaded into scan chains.

Since manufacturing test cost depends very strongly on the volume of test data and test time, one of the key requirements for next-generation DFT schemes is to reduce them dramatically. These new approaches should provide a way to achieve very high-quality test, thus allowing one to generate and apply patterns, in a cost-effective manner, for any fault model used today, as well as for any new fault models that might be employed in the future. Furthermore, the solutions should have the ability to perform on-the-fly diagnostics in a manufacturing test floor.

With staggering complexity of designs, it is very important to accommodate test reuse. It should be possible to test an embedded complex block with many internal scan chains by accessing it through a simple and narrow interface. The solution should work in an existing environment and infrastructure with minimal impact on present scan design flows, design habits or styles, no modifications to the current knowledge and skill set for scan and ATPG, and no replacement of the existing test equipment. In other words, the user should not be compelled to make major changes as a precondition for reducing test data volume and test time. Finally, long-term scalability has to be taken into account because of the exponential growth of test data volume that follows the trend of increasing design complexity defined by the Moore's law. Thus, one compression method is required that can deliver the increasingly higher levels of compression necessary to fit the compressed test data into the existing tester memory for at least one decade, with no upgrades.

As can be seen, when test data volume and test time must be reduced, neither LBIST nor conventional ATPG are ideal solutions for designs requiring high-quality test and minimal impact of DFT on the design. The Embedded Deterministic Test (EDT) scheme presented in this paper offers a high-quality test, ease of use, and broad applicability with no design impact. It builds on the solid foundation of scan and ATPG while, at the same time, fulfilling all the requirements stated in the previous paragraphs. The paper is divided into six parts. The basic principles of the proposed scheme are presented in Section II. Next, properties of an on-chip decompressor of test patterns are briefly reviewed. The compression algorithms used by EDT are developed in Sections IV and V, while their encoding efficiency is examined in Section VI. Experimental results obtained by the EDT scheme when run on industrial designs and other benchmark circuits are presented in Sections VII and VIII. The paper concludes with Section IX.

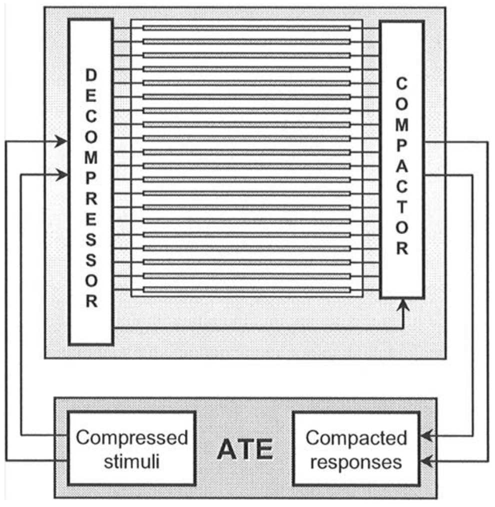

Fig. 1. EDT architecture.

II. EDT OVERVIEW

The EDT scheme consists of logic embedded on a chip and a new deterministic test-pattern generation technique. The EDT logic is inserted along the scan path but outside the design core, and consists of two main blocks as shown in Fig. 1. An on-chip *decompressor* is located between the tester scan channel inputs and the internal scan chain inputs. An on-chip *selective compactor* is inserted between the internal scan chain outputs and the tester scan channel outputs. No additional logic such as test points or X-bounding logic needs to be inserted into the core of the design. Therefore, the EDT logic affects only scan channel inputs and outputs, and has no effect on functional paths. The primary objective of the EDT approach is to reduce tester memory requirements drastically, rather than eliminate it altogether. Minimizing memory usage leads to test time reduction and increases throughput of a tester while maintaining the same test quality.

The ratio of internal scan chains to tester scan channels usually sets the maximum compression level. The design in Fig. 1 has two scan channels and 20 short internal scan chains. From the tester's point of view, the design appears to have two short scan chains. In every clock cycle, 2 bits are applied to the decompressor inputs (one bit on each input channel), while the decompressor outputs load 20 scan chains. This EDT design has the same number of scan channels interfacing with the tester when compared to ATPG, but up to 10 times more scan chains. Assuming that the scan chains are balanced, they are 10 times shorter.

The patterns that are generated and stored on the tester are the stimuli that are applied to the decompressor and the responses observed on the outputs of the compactor. The application of one pattern involves sending a compressed stimulus from the ATE to the decompressor. The continuous flow decompressor receives the data on its inputs every clock cycle, and it produces the necessary values in the scan chains to guarantee fault detection. The random fill is achieved as a side effect of decompression once the compressed pattern goes through the decompressor. The functional input and output pins are directly controlled and observed by the tester, as with conventional test. In fact, to the tester an EDT pattern looks exactly the same as an ATPG pattern except that it is much shorter. The responses captured in the scan cells are compacted by the selective compactor, shifted out, and compared to their golden references on the tester. In order to ensure that there is no aliasing and unknown states do not mask the fault effects, the decompressor controls the compactor such that only selected scan chains may stream their contents to the compactor, if necessary.

Example: To further illustrate the compression in EDT, consider an industrial design. In ATPG, it was configured into 16 scan chains with a maximum length of 11 292. The test required 1600 patterns. In EDT the same design was configured into 2400 internal scan chains with a maximum length of 76. In total, 2238 test patterns were required to achieve the same fault coverage as that of ATPG. As can be seen, the length of scan chains decreased almost 150 times. The effective reduction of volume of scan test data and scan test time is 101 ×. It is also worth noting how the tester memory is utilized in both cases. One ATPG scan pattern takes 11 292 locations of the tester memory, while one EDT pattern occupies 76 locations of the same memory. Within the memory required for one ATPG scan pattern, one can store 148 EDT patterns. Moreover, in the time required to apply one ATPG pattern, almost 148 EDT patterns can be applied. However, since EDT requires more patterns then ATPG to achieve the same fault coverage, the total time to apply all EDT vectors is effectively 101 times shorter than ATPG.

III. On-Chip Decompressor Structure

Since a decompressor is crucial to the effectiveness of EDT test data compression, it has to satisfy several requirements, including very low linear dependency in its output sequences, very high speed of operation, very low silicon area, and high modularity of the design. A new structure, called a ring generator, was designed and deployed for this application. The ring generator is a distinct form of a linear finite state machine. In order to perform on-chip decompression, it is further connected to a phase shifter (Fig. 2). The phase shifter is necessary to drive a relatively large number of scan chains, and to reduce linear dependencies between sequences entering the scan chains.

Ring Generator: An example 32-bit ring generator implementing the primitive polynomial $x^{32} + x^{18} + x^{14} + x^9 + 1$ is shown in Fig. 3. It has been obtained by applying a number of m-sequence preserving transformations to one of the canonical forms of the LFSR featuring the same feedback polynomial. Details of the ring generator synthesis process can be found in [27] and [32]. The proposed structure has three main benefits

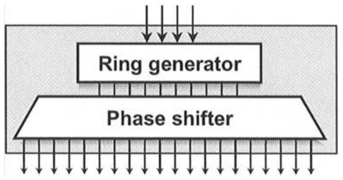

Fig. 2. On-chip decompressor.

as compared with conventional LFSRs. First, the propagation delay introduced by the feedback logic is significantly reduced. In fact, in the worst case, only one 2-input XOR gate is placed between any pair of memory elements. Second, the maximum internal fanout is confined to only two branches. Furthermore, the total length of feedback lines can be drastically reduced. The circuit in Fig. 3, for instance, features only three short connections that do not cause any frequency degradation. In general, a ring generator has fewer levels of logic than a corresponding external feedback LFSR and a smaller fanout than the internal feedback LFSR. Hence, it can operate at higher speeds than conventional solutions and is layout- and timing-friendly.

Injectors: The compressed test data are provided to the decompressor through input channels connected to taps of the ring generator by means of additional XOR gates placed between the memory elements. These connections are referred to as injectors. In fact, each input channel is usually split into a number of internal injectors providing the same test data to several memory elements at the same time. For instance, four external channels feed the ring generator of Fig. 3, each having two injectors. Although there are several schemes that can be employed to configure injectors, the pattern shown in Fig. 3 appears to assure the best performance of the whole decompressor. Due to this arrangement, test data can be quickly distributed to the entire ring generator.

Phase Shifter: The phase shifter, which is added to the outputs of the ring-generator memory elements, allows the ring generator to mutually displace the produced sequences in various scan paths [29]. As demonstrated in [26], given any linear finite state machine with a phase shifter, the probabilities of linear dependency in multiple scans are virtually equal to the theoretical bounds presented in [10] provided the inter-chain separation is greater than the length of the scan chains. The phase shifter is made of XOR gates with a limited number of inputs (called XOR taps) to reduce propagation delays. If the number of XOR taps is relatively small, then the effective inter-chain separation becomes much larger than the size of scan chains used in real designs [29]. As a result, the probabilities of linear dependency follow the limiting bounds closely and allow one to maximize the likelihood of compression of test patterns. The majority of results presented in this paper were obtained assuming that the number of XOR taps is equal to three.

Decompression: The concept of continuous flow decompression rests on the fact that deterministic test vectors have typically only between 1% and 5% of bits specified, while the remaining are randomly filled with 0s and 1s. The test vectors with partially specified bit positions are defined as test cubes. These test cubes are compressed so that the volume of test data stored on the tester is significantly reduced. The fewer the number of specified bits, the better the ability to encode the

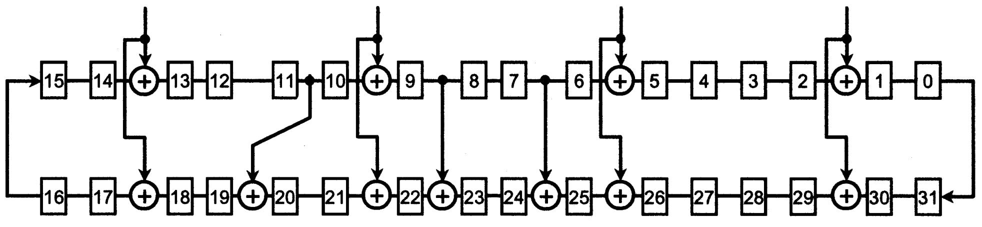

Fig. 3. The decompressor based on a ring generator implementing polynomial $x^{32} + x^{18} + x^{14} + x^9 + 1$.

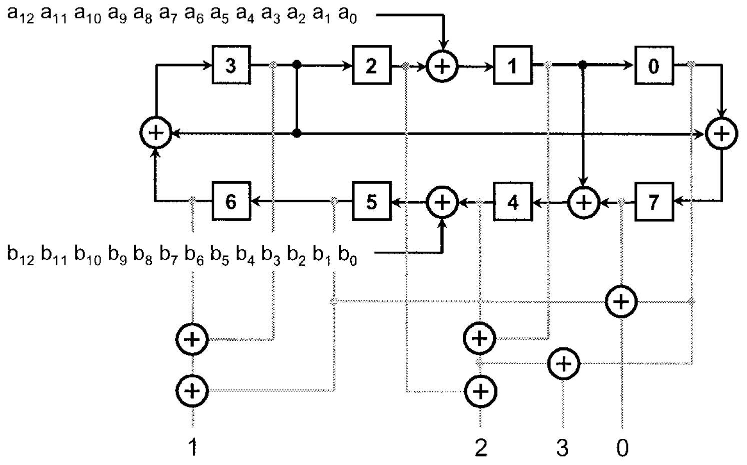
a12 a11 a10 a9 a8 a7 a6 a5 a4 a3 a2 a1 a0
b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0
3
2
1
0
6
5
4
7
1
2
3
0

Fig. 4. Example of four-output 8-bit decompressor.

information into a compressed form. As shown in Section II, this ability is exploited by having a few inputs driving the circuit, whereas the actual circuit has its memory elements connected to a large number of short scan chains. Under these circumstances, a tester that has few scan channels and a small memory for storing test data could still drive a fairly large circuit.

The decompressor operates as follows. At the beginning of every pattern, the first group of data is shifted into the ring generator. These data are referred to as initial variables, and their quantity will be denoted by V_0. Note that data shifted out of the decompressor into the scan chains during those cycles will not be used as a part of the decompressed test pattern. Next, a group of V variables is scanned for decompression. Loading the scan chains is carried out in parallel with continuous injections of new variables into the ring generator. The total number of shift cycles is equal to the initial cycles in which the V_0 variables are injected, in addition to the length of the longest scan chain (during which the V variables are injected). Comprehensive experiments indicate that the probability of successful decompression can be maximized by choosing the number V_0 of initial variables to be equal to at least $0.75D$, where D is the decompressor size.

IV. Compression of Test Stimulation

The compression of test cubes is performed by treating the external test data as Boolean variables. All scan cells are conceptually filled with symbolic expressions that are linear functions of input variables injected into the decompressor. Given the following information: 1) a feedback polynomial implemented by the ring generator; 2) structure of the associated phase shifter; 3) location of injection sites; as well as 4) the number of shift cycles, one can obtain a set of linear equations corresponding to scan cells whose values are specified. Subsequently, a compressed pattern can be determined by solving the system of equations in a manner similar to techniques presented in [10], [19], [20], [22], and [35]. If the compressed pattern is then scanned in through the decompressor, all the bits that were specified by an ATPG algorithm will match. Other unspecified bits are set to pseudorandom values based on the decompressor architecture. In the remaining part of the paper the compressor and the corresponding compression technique will be referred to as the solver and the standard EDT (EDT-S) compression scheme, respectively.

Example: Consider a decompressor and a corresponding phase shifter shown in Fig. 4. The decompressor consists of an 8-bit ring generator implementing primitive polynomial $x^8 + x^6 + x^5 + x + 1$. The four-output phase shifter is comprised of XOR gates connected as shown in the figure. Note that a feedback line corresponding to coefficient x^5 has been relocated in order to avoid undesired expansion of the XOR logic at the input of flip-flop 5. The input variables $a_0, b_0, \ldots, a_{12}, b_{12}$ are provided in pairs through two external input channels connected to the inputs of memory elements 1 and 5, respectively. Continuous operation of the decompressor yields the linear expressions shown in Table I for first 7 clock cycles. It is assumed that expressions associated with scan cells are not produced until all ring generator stages are filled with at least one input variable. This particular state occurs after four consecutive clock cycles (see Table I). Expressions appearing

TABLE I
EQUATIONS ASSOCIATED WITH DECOMPRESSOR CELLS FOR FIRST SEVEN CYCLES

0	1	2	3	4	5	6	7
0	a_0	0	0	0	b_0	0	0
a_0	a_1	0	0	b_0	b_1	a_0	0
a_1	a_2	0	b_0	b_1	$a_0 \oplus b_2$	a_1	a_0
a_2	a_3	b_0	$b_1 \oplus b_0$	$a_0 \oplus b_2$	$a_1 \oplus b_3$	$a_2 \oplus a_0$	$a_1 \oplus b_0$
a_3	$a_4 \oplus b_0$	$b_1 \oplus b_0$	$a_0 \oplus b_2 \oplus b_1 \oplus b_0$	$a_1 \oplus b_3$	$a_2 \oplus a_0 \oplus b_4$	$a_3 \oplus a_1 \oplus b_0$	$a_2 \oplus b_1 \oplus b_0$
$a_4 \oplus b_0$	$a_5 \oplus b_1 \oplus b_0$	$a_0 \oplus b_2 \oplus b_1 \oplus b_0$	$a_1 \oplus a_0 \oplus b_3 \oplus b_2 \oplus b_1 \oplus b_0$	$a_2 \oplus a_0 \oplus b_4$	$a_3 \oplus a_1 \oplus b_5 \oplus b_0$	$a_4 \oplus a_2 \oplus b_1$	$a_3 \oplus a_0 \oplus b_2 \oplus b_1 \oplus b_0$
$a_5 \oplus b_1 \oplus b_0$	$a_6 \oplus a_0 \oplus b_2 \oplus b_1 \oplus b_0$	$a_1 \oplus a_0 \oplus b_3 \oplus b_2 \oplus b_1 \oplus b_0$	$a_2 \oplus a_1 \oplus b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus b_0$	$a_3 \oplus a_1 \oplus b_5 \oplus b_0$	$a_4 \oplus a_2 \oplus b_6 \oplus b_1$	$a_5 \oplus a_3 \oplus a_0 \oplus b_2$	$a_4 \oplus a_1 \oplus a_0 \oplus b_3 \oplus b_2 \oplus b_1$

TABLE II
EQUATIONS ASSOCIATED WITH FIRST THREE CELLS OF EACH SCAN CHAIN

S.0	S.1	S.2	S.3
$a_3 \oplus a_0 \oplus b_4 \oplus b_1 \oplus b_0$	$a_3 \oplus a_2 \oplus a_1 \oplus b_4 \oplus b_2 \oplus b_1$	$a_4 \oplus a_1 \oplus b_3 \oplus b_1$	$a_4 \oplus a_3 \oplus a_1 \oplus b_3 \oplus b_0$
$a_4 \oplus a_1 \oplus a_0 \oplus b_5 \oplus b_2 \oplus b_1 \oplus b_0$	$a_4 \oplus a_3 \oplus a_2 \oplus a_0 \oplus b_5 \oplus b_3 \oplus b_2$	$a_5 \oplus a_2 \oplus b_4 \oplus b_2$	$a_5 \oplus a_4 \oplus a_2 \oplus a_0 \oplus b_4 \oplus b_1$
$a_5 \oplus a_2 \oplus a_1 \oplus a_0 \oplus b_6 \oplus b_3 \oplus b_2 \oplus b_1$	$a_5 \oplus a_4 \oplus a_3 \oplus a_1 \oplus a_0 \oplus b_6 \oplus b_4 \oplus b_3 \oplus b_0$	$a_6 \oplus a_3 \oplus b_5 \oplus b_3 \oplus b_0$	$a_6 \oplus a_5 \oplus a_3 \oplus a_1 \oplus a_0 \oplus b_5 \oplus b_2 \oplus b_0$

on the outputs of the phase shifter since then (for the next three clock cycles) are presented in Table II. Notice that new variables are being injected into the decompressor in parallel to these operations

Once the expressions are determined, a system of linear equations is formed for each test cube. These equations are obtained by selecting expressions corresponding to specified positions in the test cubes and assigning values to them. Suppose the decompressor needs to generate a test pattern based on the following test cube (scan shifting is from left to right):

$$\begin{array}{ccccccccc} x & x & x & x & x & x & 1 & x & 1 \\ 1 & x & x & 0 & x & 1 & 1 & x & x \\ x & x & x & x & x & x & x & x & x \\ 0 & 0 & x & x & 1 & x & x & 0 & x \end{array}$$

where x denotes a "don't care" condition. The corresponding compressed pattern can be determined by solving the following system of equations shown at the bottom of the page (the equations are ordered with respect to successive clock cycles—the first four expressions, therefore, occur in Table II as well): It can be verified that variables a_2, a_1, b_6, b_3, and b_1 are equal to one, while the remaining variables assume the value of zero. This data will subsequently produce a fully specified test pattern having the form shown at the bottom of the page (the initial specified positions are now underlined).

In order to achieve a high degree of compression, the associated ATPG algorithm works as follows. Whenever the algorithm generates a test cube (without random fill), the solver is invoked to generate the compressed pattern. When dynamic compaction is enabled, the solver works iteratively with ATPG to maximize

$$\begin{aligned}
a_3 \oplus a_0 \oplus b_4 \oplus b_1 \oplus b_0 &= 1\\
a_5 \oplus a_4 \oplus a_2 \oplus a_0 \oplus b_4 \oplus b_1 &= 0\\
a_5 \oplus a_2 \oplus a_1 \oplus a_0 \oplus b_6 \oplus b_3 \oplus b_2 \oplus b_1 \oplus b_0 &= 1\\
a_5 \oplus a_4 \oplus a_3 \oplus a_1 \oplus a_0 \oplus b_6 \oplus b_4 \oplus b_3 \oplus b_0 &= 1\\
a_8 \oplus a_7 \oplus a_5 \oplus a_3 \oplus a_2 \oplus b_7 \oplus b_4 \oplus b_2 &= 1\\
a_6 \oplus a_5 \oplus a_4 \oplus a_2 \oplus a_1 \oplus a_0 \oplus b_7 \oplus b_5 \oplus b_4 \oplus b_4 \oplus b_1 \oplus b_1 \oplus b_0 &= 1\\
a_8 \oplus a_7 \oplus a_6 \oplus a_4 \oplus a_3 \oplus a_2 \oplus a_1 \oplus a_0 \oplus b_9 \oplus b_7 \oplus b_6 \oplus b_3 \oplus b_2 \oplus b_0 &= 0\\
a_{11} \oplus a_{10} \oplus a_8 \oplus a_6 \oplus a_5 \oplus a_1 \oplus b_{10} \oplus b_7 \oplus b_5 \oplus b_1 &= 0\\
a_{11} \oplus a_{10} \oplus a_9 \oplus a_7 \oplus a_6 \oplus a_5 \oplus a_4 \oplus a_3 \oplus a_2 \oplus b_{12} \oplus b_{10} \oplus b_9 \oplus b_6 \oplus b_5 \oplus b_3 &= 1\\
a_{12} \oplus a_{11} \oplus a_9 \oplus a_7 \oplus a_6 \oplus a_2 \oplus a_0 \oplus b_{11} \oplus b_8 \oplus b_6 \oplus b_2 &= 0.
\end{aligned}$$

```
while (not all faults targeted)
    select fault and generate test cube
    try to compress cube
    if (compression failed)
        declare fault as EDT aborted
        continue with new fault
    loop                                        // dynamic compaction loop
        select fault and generate incremental  test cube
        if (dynamic compaction failed)
            if (attempts < limit) continue with new fault
            else exit loop
        try to compress cube incrementally
        if (compression failed)
            if (attempts < limit) continue with new fault
            else exit loop
    end loop                            // one compressed pattern generated
    decompress compressed pattern
    perform fault simulation                    //group of 32 or 64 patterns
end while
```

Fig. 5. Generation of compressed patterns.

compression. According to this scenario, as long as the solver can compress a test cube, ATPG attempts to pack more specified bits into that test by targeting subsequent faults. The ATPG flow and interactions with the solver are shown in Fig. 5. It is worth noting that the solver operates in an incremental mode, i.e., it is invoked several times for a gradually increasing system of linear equations.

There are two benefits of using an incremental solver. First, the newly specified bits are appended to the earlier processed equations so that the computations are not repeated for the previously specified bits. This ensures that the running time of the solver remains significantly lower than that of ATPG. Second, it ensures that if dynamic compaction fails in the nth iteration, the specified bits in the test cube up to iteration $n-1$ will appear when the pattern is decompressed.

Note that if a test generated for the first fault cannot be compressed, that fault is not retargeted. The solver may fail to compress the test cube due to a large number of specified bits, or due to linear dependency between the positions with specified values. However, if the test cube for any subsequent fault in the dynamic compaction flow cannot be compressed, that fault will become a target again, as its test cube may be compressible with tests for other faults or when targeted separately.

Detailed analysis of test cube profiles shows that in many industrial designs, even if they are generated with sophisticated dynamic compression algorithms targeting multiple faults and performing multiple clock compression, the fill rate is 1% to 5% only at the beginning of the process. After the first couple of vectors the fill rate may drop well below 1%. Fig. 6 shows an example of a test cube profile for an industrial design. This design has 7 million gates and 330 thousand scan cells. In the first 64 patterns, the fill rate averages 11 318 positions. After 320 patterns, the fill rate drops below 1%. The average is then 675 (0.2%). Following this observation, the amount of test data can be reduced beyond the compression levels offered by the EDT-S. An approach presented in this section is based on the analysis that allows one to eliminate, in an adaptive fashion, several Boolean variables so that test patterns could be further compressed. The scheme can be subsequently implemented by taking advantage of a tester's repeat capability [39].

As can be easily observed, the ratio of internal scan chains to tester scan channels sets the maximum EDT compression level

$$M = \frac{S \times L}{V_0 + V} = \frac{S \times L}{V_0 + C \times L} \approx \frac{S}{C}$$

where S is the number of scan chains, L represents the maximum scan length, C is the number of external channels, and the product $S \times L$ amounts to a volume of uncompressed test data, while $C \times L$ is equal to the number of variables shifted in when loading the scan chains. The above relationship remains valid as long as a new set of C variables is injected every shift (clock) cycle. In many practical situations, however, the number of specified positions may vary significantly from test cube to test cube, and a quite large fraction of test cubes may feature extremely sparse specified bits (i.e., very low scan fill rates—see Fig. 6). To be successfully compressed, these test cubes do not require all variables, which would be injected within the framework of EDT-S scheme. Thus, one may further elevate the compression ratio if the volume of test data is reduced by injecting new variables only when needed. Through the rest of this section, we will assume that variables are injected in indivisible portions of C bits, that is, for each shift cycle either new C variables are provided to the decompressor or no new variables are injected at all.

In order to achieve higher degrees of compression, two additional problems have to be resolved. First, a guiding rule is required to decide when to inject a new portion of variables into the decompressor to ensure successful encoding of a given test cube. Second, a proper replacement for eliminated variables has to be chosen such that it neither hinders operation of the continuous flow decompressor nor the resulting volume of test data compromises targeted compression rates.

The first problem is solved by using an auxiliary vector test cube profile (TCP). It is employed to characterize every test cube. Given a scan architecture, an entry TCP_k is equal to the number of specified bits associated with those scan cells that are loaded during kth scan shift cycle, $k = 1, 2 \ldots, L$, where L is the scan length. Furthermore, a variable pool (VP) item is maintained to monitor the number of variables involved in creating

$$\begin{array}{cccccccccccccc|ccccccccc}
 & & & & & & & & & & & & & & 0 & 1 & 1 & 1 & 1 & 0 & \underline{1} & 0 & \underline{1} \\
a: & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & \rightarrow & \underline{1} & 0 & 0 & \underline{0} & 0 & \underline{1} & \underline{1} & 0 & 1 \\
b: & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & \rightarrow & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
 & & & & & & & & & & & & & & \underline{0} & \underline{0} & 1 & 1 & \underline{1} & 1 & 1 & \underline{0} & 0
\end{array}$$

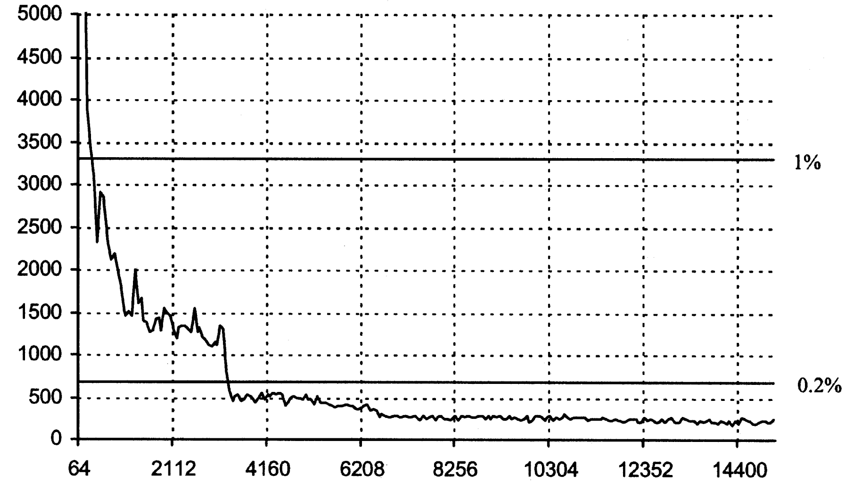

Fig. 6. Example of test cube profile.

successive equations. Initially, VP is set to V_0—the number of variables loaded into the decompressor during the initialization period.

Assuming that a set of B equations can be solved provided it comprises $B + R$ variables, where R is an arbitrarily chosen integer, a new compression algorithm, called adaptive variable elimination (AVE), works as follows. For each scan shift cycle k, it subtracts TCP_k from the current value of VP. Next, depending on the outcomes of this operation, different sequences of actions are performed. If a new value of VP is smaller than R, then new C variables are injected into the decompressor, and VP is increased by C, accordingly. Otherwise, no further steps are required besides those necessary to indicate that certain variables have to be eliminated. As shown in [10], [19], R should equal at least 20, but as proved by experiments, R smaller than 20 suffices in many cases.

Whenever the AVE algorithm generates a modified set of equations (as shown above), the solver is invoked to generate the compressed pattern. However, given a particular set of equations and corresponding variables, a solution may not always exist. Therefore, as long as the solver cannot compress a test cube, the whole algorithm works by repeatedly increasing the value of R, creating a new set of equations, and invoking the solver. It is worth noting that by employing such an approach, test cubes can be compressed for different values of R, thus preserving the best encoding efficiency in each individual case.

Although the concept of variable elimination is sound, it has to be accompanied by an efficient technique to form linear equations for successive test cubes and often different values of R, if the solver fails to compress the test cubes in the first place. In the proposed method, all equations associated with successive scan cells are created initially by assuming that variables are injected into the decompressor every clock cycle. Clearly, this is the most general case, and this step needs only be done once. Subsequently, a given subset of equations (corresponding to a particular test cube) is further modified by adopting one of the following two scenarios.

Replacement: If certain variables should be eliminated, they are replaced with variables injected in the previous scan shift cycle. We will denote this approach as AVE-R. Suppose a decompressor is driven by 8 tester channels. Consider then the following equation:

$$a_0 \oplus b_0 \oplus d_0 \oplus e_0 \oplus f_0 \oplus g_0 \oplus h_0 \oplus c_1 \oplus d_1 \oplus f_1 \oplus h_1 = 1.$$

If AVE algorithm decides to exclude variables injected during the second step (i.e., variables $a_1, b_1, c_1, d_1, e_1, f_1, g_1$, and h_1) and to replace them with variables injected in the former step (i.e., $a_0, b_0, c_0, d_0, e_0, f_0, g_0$, and h_0), then the resulting equation will assume the following form:

$$\begin{aligned} &a_0 \oplus b_0 \oplus d_0 \oplus e_0 \oplus f_0 \oplus g_0 \oplus h_0 \oplus c_0 \oplus d_0 \oplus f_0 \oplus h_0 \\ &\qquad = a_0 \oplus b_0 \oplus c_0 \oplus e_0 \oplus g_0 = 1. \end{aligned}$$

To accomplish this action in a time-efficient manner, we can use an array of bits to represent variables. Then, the part of each equation corresponding to omitted variables can be added modulo 2, in a bit-wise fashion, to that part which is used as their replacement. If we continue this way until all variables are examined, the resulting equation will have exactly the same form as the one obtained by symbolic simulation of the decompressor with certain variables repeatedly injected several times. The eliminated variables are eventually replaced with 0s.

When applying AVE-R method, test-data volume reduction depends primarily on the number v of resulting C-bit patterns. Thus, once all patterns are determined, an additional post-processing step is carried out in order to identify those adjacent patterns which might be identical. If found, they are replaced with a single copy of that pattern. This is a commonly observed case for all-zero seed vectors. Finally, each pattern requires a counter indicating how many times this particular vector should be applied. In the experiments described in Sections V–VIII, this quantity is measured in the number of bits needed to encode the counter value. Hence, the total volume of compressed test data is equal to v C-bit patterns and associated v binary counters. As can be seen, the ability of test equipment to repeat a given pattern multiple times allows one to implement the AVE-R technique easily.

All-Zero Patterns: The second approach used to handle excluded variables assumes that these variables are replaced with the logic value of 0. Thus, the equation presented earlier would have the following form:

$$a_0 \oplus b_0 \oplus d_0 \oplus e_0 \oplus f_0 \oplus g_0 \oplus h_0 = 1.$$

This technique will be referred to as AVE-0. If it can be paired with a custom ATE memory management scheme, then all-zero seed patterns do not have to be stored on a tester. As a result, the total test data volume will consist, in this case, of two components: 1) the nonzero C-bit patterns and 2) L-bit vector Z whose position $Z_k = 1$ indicates that a nonzero seed pattern is to be loaded into the decompressor during kth shift cycle, while entry $Z_k = 0$ enables injecting the all-zero vector. The latter one can be provided, for instance, through properly configured multiplexers. Note that the content of the vector Z can be also used to advance the address counter of the ATE memory.

V. Nonadaptive Variable Elimination (NAVE)

In certain situations performance of the adaptive schemes presented in Section IV can be further enhanced by assuming that injections of new variables occur regularly in predetermined scan shift cycles only. Although this nonadaptive variable elimination technique NAVE may not offer flexibility of the previous schemes, we will show that its simplicity paired with further data volume savings can be advantageous.

Since data reduction in a practical NAVE approach is achieved by deploying only one counter per test cube, the most important decision to be made is the choice of the value of U (called an under-sampling rate), which is the number of repeated injections of the same C-bit pattern. For experiments reported in Sections VI–VIII, we have used the following formula:

$$U = \left\{ \frac{V_0 + V}{B} \right\} = \left\{ \frac{V_0 + C \times L}{B} \right\}$$

where $\{x\}$ denotes rounding to the nearest integer. As can be seen, the injection rate is obtained by dividing the total number of variables that might be injected by the number of specified positions.

Similarly, as for AVE schemes, the nonadaptive approach may either use the same variables many times or inject the all-zero patterns instead. By contrast, however, the compression ratios for NAVE technique will remain virtually the same for both implementations. This is because nonzero as well as all-zero seed patterns are treated in the same manner: they cannot be further merged or replaced with single bits indicated a type of the pattern. Nevertheless, an implementation of NAVE-R adheres more conveniently to the ATE ability of repeating the same patterns multiple times. Indeed, injecting the all-zero vectors would require additional scan operational codes to differentiate between the first pattern of each U-step cycle and the remaining ones. For this reason, experimental results presented in this paper were derived for NAVE-R scheme. When computing the compression rates, it is assumed that test data are comprised, for each test cube, of L/U C-bit patterns, and one additional counter of size $\log_2 U$ bits.

It should be noted that as long as the solver cannot compress a test cube, the whole algorithm works by repeatedly decreasing U (until it reaches the value of 1), creating a new set of equations, and invoking the solver again. Due to this more flexible strategy, each test cube is compressed separately, though for some test cubes, once $U = 1$, the scheme becomes equivalent to EDT-S approach.

Monte Carlo simulations were run to determine compression rates offered by schemes AVE and NAVE. The compression rate is defined here as a ratio of the number of scan cells and the actual number of test data bits necessary to compress test cubes. Assuming a 96-bit decompressor, the following experimental setup was adopted. For given values of C, L, and S, each experiment comprises generating 1000 test cubes with three different fill rates (2%, 1%, and 0.5%, respectively) and applying three compression schemes presented earlier, i.e., NAVE-R, AVE-R, and AVE-0. Tables III and IV summarize compression numbers obtained for test setups with 16 384 and 32 768 scan cells, respectively. The third columns in both tables show the compression rates achievable by using the EDT-S scheme. These numbers are used as reference points. Note that they remain independent of a fill rate provided a given test cube can be compressed. Entries denoted by "-" indicate cases where, given values of C, S and a fill rate, it becomes impossible to encode test cubes.

Clearly, all schemes with reduced variable injection rates have the desirable property of vastly reducing the test data volume. Not surprisingly, the decrease in the fill rate causes increase in the compression rates. However, the compression grows as well with the decreasing size of the scan chains provided the adaptive schemes are employed (the same applies to EDT-S, but the effective gain is usually much lesser). For NAVE approach we observe an opposite trend: the compression slightly decreases with the decreasing size of the scan. Hence, the choice of the compression scheme depends on the scan length: the shorter the scan chain, the better chance to opt for the adaptive schemes. In particular, AVE-0 approach seems to outperform other schemes if relatively large numbers of external channels are used, as these numbers guarantee the most visible savings when the all-zero patterns are to be injected. With the increasing number of scan cells, however, the AVE-R scheme offers better average performance.

It is worth noting that the results presented in Tables III and IV do not pronounce the adaptive nature of the proposed schemes in their full capacity. This is because there is virtually no variance in the randomly generated test cubes as far as the number of specified positions is concerned. Therefore, all advantages of the proposed schemes become even more evident when applied to designs with sparsely distributed specified bits, as shown in Section VIII.

VI. Encoding Efficiency

Another figure of merit—the encoding efficiency E is defined as a ratio of successfully encoded specified bits B to the total number of deployed variables, i.e.,

$$E = \frac{B}{0.75D + C \times L}$$

where D is the size of the decompressor, C is the number of external channels, and L represents the scan length. Recall that

TABLE III
COMPRESSION RATES ($\times$), D-96, SCAN CELLS-16 384

Scan	C	**EDT-S**	**NAVE**	**AVE-R**	**AVE-0**	**NAVE**	**AVE-R**	**AVE-0**	**NAVE**	**AVE-R**	**AVE-0**
			fill rate – 2%			**fill rate – 1%**			**fill rate – 0.5%**		
32 × 512	4	7.73	34.20	29.76	19.47	68.27	51.42	23.87	107.1	90.63	26.98
64 × 256	4	14.95	28.15	34.57	27.91	68.55	58.66	37.96	105.0	101.4	46.58
128 × 128	4	28.05	28.20	41.05	35.69	68.84	67.82	53.90	105.7	115.1	73.10
256 × 64	4	49.95	-	-	-	50.41	66.73	68.30	106.4	132.9	102.0
512 × 32	4	81.92	-	-	-	83.17	81.57	79.95	83.17	130.7	127.8
32 × 512	8	3.93	40.55	32.19	19.11	71.55	59.16	23.82	109.2	106.4	27.14
64 × 256	8	7.73	34.49	34.99	27.16	69.42	63.79	37.75	110.0	113.9	46.88
128 × 128	8	14.95	28.35	38.46	34.49	69.72	68.87	53.12	110.7	122.7	73.82
256 × 64	8	28.05	28.40	42.10	39.00	70.02	75.08	66.87	111.5	132.9	103.8
512 × 32	8	49.95	-	-	-	51.04	81.76	75.37	112.2	143.3	128.5
32 × 512	16	1.98	40.45	36.57	19.21	71.23	66.92	23.75	107.8	118.3	27.38
64 × 256	16	3.92	40.55	38.22	27.45	71.55	69.85	37.57	108.5	123.9	47.38
128 × 128	16	7.73	35.08	40.02	34.94	71.86	73.10	53.10	109.2	129.3	74.56
256 × 64	16	14.95	28.35	41.80	40.47	72.18	76.51	65.99	110.0	135.3	103.4
512 × 32	16	28.05	28.40	43.90	43.93	72.50	78.93	76.13	110.7	142.2	128.6
32 × 512	32	1.00	42.01	39.20	18.29	70.62	70.02	24.27	118.7	102.7	26.82
64 × 256	32	1.98	42.12	40.26	26.86	70.93	71.86	39.10	119.6	105.5	46.22
128 × 128	32	3.93	42.23	41.37	33.95	71.23	73.80	76.24	120.5	108.3	72.35
256 × 64	32	7.73	36.33	42.56	39.13	71.55	75.86	106.9	121.4	111.0	100.8
512 × 32	32	14.95	28.35	43.81	43.62	71.86	78.03	136.1	122.3	114.2	125.6

TABLE IV
COMPRESSION RATES ($\times$), D-96, SCAN CELLS-32 768

Scan	C	EDT-S	NAVE	AVE-R	AVE-0	NAVE	AVE-R	AVE-0	NAVE	AVE-R	AVE-0
			fill rate – 2%			fill rate – 1%			fill rate – 0.5%		
32 × 1024	4	7.86	36.94	30.01	19.70	79.53	52.20	24.22	140.6	91.80	27.36
64 × 512	4	15.46	29.95	34.96	28.45	68.41	59.52	38.91	136.5	103.0	47.73
128 × 256	4	29.90	29.98	41.69	36.61	56.30	69.00	55.73	137.1	117.3	75.95
256 × 128	4	56.11	-	-	-	56.40	82.19	71.36	137.7	135.7	107.7
512 × 64	4	99.90	-	-	-	-	-	-	100.8	133.5	136.5
1024 × 32	4	163.8	-	-	-	-	-	-	166.3	162.9	160.0
32 × 1024	8	3.97	43.81	32.73	19.31	80.91	59.54	23.98	142.5	110.2	27.33
64 × 512	8	7.86	37.11	35.71	27.61	81.11	64.40	38.21	143.1	118.2	47.64
128 × 256	8	15.46	30.06	39.20	35.14	68.99	70.10	54.32	138.8	127.7	75.44
256 × 128	8	29.90	30.09	43.18	40.75	56.69	76.85	68.63	139.4	138.1	105.6
512 × 64	8	56.11	-	-	-	56.79	84.09	79.35	140.0	150.5	131.7
1024 × 32	8	99.90	-	-	-	-	-	-	102.1	163.2	150.8
32 × 1024	16	1.99	46.22	36.74	19.29	84.02	69.89	24.02	141.8	127.9	27.38
64 × 512	16	3.97	47.35	38.48	27.63	84.24	73.04	38.43	142.5	133.5	47.48
128 × 256	16	7.86	44.34	40.43	35.07	88.09	76.50	54.20	153.1	140.0	75.20
256 × 128	16	15.46	44.40	42.51	39.38	81.31	80.09	68.05	153.8	146.4	106.5
512 × 64	16	29.90	29.84	44.87	44.34	81.51	83.57	77.26	154.6	153.2	132.1
1024 × 32	16	56.11	-	-	-	56.01	87.73	85.60	168.0	158.3	148.4
32 × 1024	32	1.00	46.15	39.31	19.29	83.81	76.38	24.33	140.6	136.5	27.61
64 × 512	32	1.99	46.22	40.34	27.61	84.02	78.39	37.83	141.2	140.0	48.55
128 × 256	32	3.97	48.47	41.46	35.20	84.24	80.51	53.68	141.8	143.7	73.89
256 × 128	32	7.86	44.34	42.59	41.12	91.79	82.75	67.88	165.5	147.6	103.4
512 × 64	32	15.46	44.40	43.82	44.55	84.67	85.11	78.27	166.3	151.7	129.3
1024 × 32	32	29.90	29.84	45.14	46.35	84.89	87.61	87.16	167.2	156.0	145.8

$0.75D$ corresponds to the initial variables, while the second product is equal to the number of remaining variables shifted in when loading the scan chains. It is worth noting that the maximum compression can be approximately expressed in terms of the encoding efficiency E and a scan fill rate F regarded as a fraction of scan cells which are specified. Let $E \approx B/(C \times L)$ and $F = B/(S \times L)$, where S is the number of scan chains. Then the maximum compression is given by the formula $S/C \approx E/F$. As can be seen, if the fill rate decreases, then the compression level will increase provided one can maintain a given

TABLE V
ENCODING EFFICIENCY FOR $L = 128$

	C = 8				C = 16					C = 32			
	Number of scan chains				Number of scan chains					Number of scan chains			
D	128	256	512	1024	128	256	512	1024	D	128	256	512	1024
32	90.25	87.95	88.77	88.10	77.08	77.19	76.21	85.88	64	86.83	86.59	86.55	86.34
40	92.04	92.20	91.44	92.60	88.46	88.21	87.46	90.23	72	95.79	94.95	94.03	94.10
48	94.02	94.07	94.00	93.62	91.08	91.70	91.16	93.94	80	94.82	95.33	94.66	94.92
56	96.36	95.04	95.85	95.47	94.93	94.86	94.00	95.42	88	96.44	95.76	94.67	94.77
64	96.71	96.66	96.15	96.39	96.36	95.90	95.92	96.05	96	96.14	95.52	95.37	94.92
72	96.20	96.58	96.08	96.60	96.87	96.67	96.00	96.81	104	97.51	96.87	96.51	96.14
80	96.49	96.75	96.91	96.63	96.70	96.52	97.02	96.88	112	97.41	96.04	96.79	96.86
88	97.34	96.97	96.82	97.03	97.52	97.08	96.75	97.31	120	97.54	97.07	96.83	96.40
96	97.19	97.14	96.91	96.99	97.94	97.74	97.83	97.73	128	98.80	98.57	98.33	98.28
104	97.24	96.72	97.14	97.26	98.06	97.97	97.79	97.46	136	98.49	98.17	98.16	98.07
112	97.09	96.92	96.97	96.90	97.73	97.61	97.51	97.99	144	98.61	98.33	98.21	97.90
120	97.31	97.12	97.10	97.17	98.25	98.14	97.89	98.11	152	98.82	98.46	98.54	98.19
128	97.10	97.01	97.03	97.01	98.20	98.06	97.95	97.49	160	98.84	98.60	98.53	98.21
136	96.89	96.84	97.08	97.12	97.88	97.40	97.41	97.87	168	98.88	98.71	98.60	98.61
144	97.01	96.86	96.89	96.97	98.22	97.91	97.99	97.73	176	98.64	98.38	98.34	98.17
152	96.81	96.91	96.94	96.93	97.89	97.78	97.80	97.93	184	98.74	98.25	98.46	98.32
160	96.98	96.92	96.80	96.77	98.14	98.14	98.00	97.75	192	98.92	98.72	98.68	98.63
168	97.23	96.79	96.72	96.77	97.89	97.90	97.66	98.03	200	98.83	98.62	98.59	98.13

encoding efficiency. However, the smaller fill rate F leads to the smaller value of B, which in turn, reduces also the encoding efficiency.

Comprehensive characterization experiments were run to measure the encoding efficiency. An experimental setup that was adopted here follows the functional behavior of the EDT-S compression procedure working in the incremental mode. For given values of D, C, L, and S, every experiment is comprised of a number of successive steps. In step k it is verified (by solving the corresponding linear equations) whether B_k specified bits can be encoded. The specified bits subjected to compression in step k are obtained by adding a new small quantity of specified bits Δ to those bits that have already been used in step $k-1$. Thus, $B_k = B_{k-1} + \Delta$. Also $B_0 = \Delta$, and it is assumed that $\Delta = 5$. The process continues until the first failure. In such a case, the number of positions that the system was able to encode, i.e., those used in the previous step, is recorded by incrementing a corresponding entry of a histogram. Subsequently, a new test pattern becomes the subject of compression.

As can be seen, the main difference between the experimental setup and the actual EDT is that if the latter fails to compress a cube in the incremental dynamic compaction flow, it can backtrack and try to merge the test for a different fault, allowing a higher encoding efficiency to be achieved. After analyzing a sufficiently large number of test patterns, entry b to the resulting histogram can be regarded as being proportional to the probability P_b of successful encoding of b specified positions. This information allows one to easily calculate the number of specified bits that can be encoded on average. The average value $\Sigma b P_b$ is then employed to determine the final encoding efficiency as shown earlier. Note that in all the experiments, each individual test cube was created by randomly determining specified positions. Both, locations of scan cells and values assigned to them were generated by sampling a uniform distribution in the range of $[0, S \times L - 1]$ and $[0, 1]$, respectively.

Tables V and VI detail the encoding efficiency numbers for various decompressor sizes assuming that EDT test setups feature 8, 16, and 32 external channels (each having two injectors), as well as 128 and 256 cells in each scan chain. The corresponding phase shifters consist of three-input XOR gates. A close examination of the presented data and many similar results not reported here indicates that the encoding efficiency is relatively independent of the number of scan chains and the scan length, and it increases with the increasing size of the decompressor. The latter trend continues to a certain point, beyond which there is no noticeable improvement. For instance, for 16 channels and 256 scan chains, the diminishing returns point is located around 120. For sufficiently large decompressors, the number of bits that can be encoded approaches the number of employed variables. Results obtained in a manner similar to that of Tables V and VI can be used to select the most appropriate decompressor size.

It appears that the choice of the number of XOR taps (used by phase shifters—see Section III) and the number of injectors has certain impact on the encoding efficiency as well. Some illustrative examples are provided in Table VII. As can be seen, either the number of XOR taps increased to five, or the number of injectors per channel increased to four, or both, result in the improved encoding efficiency as compared to the results shown in Table V. This property allows designers to tradeoff between

TABLE VI
ENCODING EFFICIENCY FOR $L = 256$

	C = 8				C = 16					C = 32			
	Number of scan chains				Number of scan chains					Number of scan chains			
D	128	256	512	1024	128	256	512	1024	D	128	256	512	1024
32	85.45	86.89	86.27	-	72.84	72.13	69.13	-	64	83.51	84.92	83.75	85.64
40	91.37	90.82	90.55	90.23	84.75	83.29	83.04	83.49	72	95.13	92.33	91.56	93.14
48	93.94	94.03	93.90	93.58	89.75	89.35	88.22	88.26	80	95.16	93.61	92.92	94.33
56	95.09	94.88	94.51	94.69	92.94	93.31	92.98	92.98	88	95.56	95.07	94.24	94.50
64	96.18	96.02	95.18	95.48	95.42	94.59	94.95	94.71	96	95.89	94.81	94.84	94.21
72	96.31	96.01	95.60	95.90	96.49	96.05	95.22	95.48	104	97.23	96.41	95.97	95.19
80	96.66	96.79	96.39	96.84	96.25	96.21	96.25	96.34	112	97.18	95.20	96.39	96.09
88	97.25	96.88	97.22	97.00	97.28	96.78	96.78	96.57	120	97.26	97.12	95.98	96.37
96	97.56	97.55	97.63	97.16	97.84	97.70	97.32	97.43	128	98.58	98.39	97.97	97.73
104	96.95	96.80	96.64	96.71	97.95	97.59	97.81	97.32	136	98.51	98.17	97.96	98.04
112	97.71	97.76	97.58	97.29	97.86	97.56	97.47	97.02	144	98.61	98.21	98.06	97.74
120	97.68	97.43	97.44	97.51	98.35	98.33	97.57	97.99	152	98.84	98.48	98.53	98.32
128	97.97	97.98	97.83	97.76	98.37	98.27	98.27	98.15	160	98.92	98.53	98.23	98.30
136	97.86	97.77	97.88	97.89	98.23	97.97	97.78	98.00	168	98.92	98.79	98.55	98.46
144	97.95	97.83	97.83	97.84	98.43	98.33	98.23	98.17	176	98.76	98.52	98.47	98.29
152	97.84	97.87	97.82	97.83	98.37	98.01	98.23	97.96	184	98.92	98.57	98.52	98.44
160	97.89	98.03	97.78	97.91	98.55	98.43	98.43	98.28	192	99.06	98.56	98.64	98.69
168	98.10	97.85	97.88	97.85	98.35	98.22	98.35	98.17	200	99.10	98.86	98.72	98.55

TABLE VII
ENCODING EFFICIENCY FOR $C = 8$, $L = 256$

	XOR = 5, I = 2				XOR = 3, I = 4				XOR = 5, I = 4			
	Number of scan chains				Number of scan chains				Number of scan chains			
D	128	256	512	1024	128	256	512	1024	128	256	512	1024
32	93.22	93.01	92.86	92.74	93.00	92.55	92.56	-	93.91	93.71	93.56	93.44
40	95.41	95.28	95.26	95.14	95.20	95.14	95.32	94.80	96.14	96.01	95.80	95.80
48	96.37	96.37	96.35	96.25	96.69	96.65	96.45	96.42	97.12	97.02	97.03	96.78
56	96.90	96.97	96.80	96.87	97.20	97.17	97.13	97.08	97.64	97.62	97.63	97.65
64	97.74	97.62	97.53	97.47	97.91	97.84	97.60	97.71	98.16	98.07	98.13	98.06
72	97.64	97.66	97.66	97.50	98.02	97.98	97.92	97.90	98.39	98.38	98.28	98.33
80	98.00	97.97	97.99	97.99	98.19	98.17	98.15	98.12	98.50	98.46	98.46	98.47
88	98.14	98.06	97.97	97.97	98.37	98.30	98.27	98.19	98.59	98.56	98.75	98.52
96	98.20	98.16	98.15	98.11	98.36	98.34	98.28	98.33	98.66	98.67	98.69	98.65
104	98.24	98.19	98.17	98.24	98.42	98.35	98.38	98.31	98.78	98.80	98.72	98.73
112	98.26	98.25	98.20	98.18	98.39	98.43	98.36	98.39	98.79	98.74	98.72	98.77
120	98.29	98.24	98.18	98.23	98.45	98.46	98.49	98.46	98.78	98.77	98.80	98.75
128	98.02	98.01	97.96	97.97	98.32	98.31	98.34	98.36	98.76	98.75	98.77	98.74

the number of ring generator memory elements and the number of XOR gates used to implement a phase shifter and injector sites.

VII. EXPERIMENTAL RESULTS—PART I

The EDT-S compression scheme was tested on over 30 industrial benchmarks. In this section, eight representative designs ranging in size from 367 K gates to 2.2 million gates are presented. For each design, conventional ATPG with 16 scan chains as well as EDT with three different compression levels, i.e., 5, 10, and 25 ×, were performed by fixing the number of scan channels to 16 and utilizing 80, 160, and 400 internal scan chains, respectively. The circuits represent different design styles and scan methodologies. They are also random-pattern resistant. The experiments were performed using every

TABLE VIII
EXPERIMENTAL RESULTS

		Circuits							
		C1	C2	C3	C4	C5	C6	C7	C8
Circuit Parameters	Gates	370K	500K	545K	580K	1210K	1220K	1540K	2180K
	Scan cells	19K	29K	46K	42K	71K	63K	87K	181K
ATPG	TC (%)	99.41	95.63	98.88	98.05	97.04	99.89	98.98	98.84
EDT – 5x	TC (%)	99.40	95.63	98.87	98.05	97.04	99.88	98.98	98.86
	CPU time	0.99	1.15	1.13	1.18	1.01	1.16	1.40	1.08
	Area (%)	0.71	0.54	0.48	0.46	0.22	0.22	0.17	0.12
	EC (×)	4.93	4.96	4.97	4.67	5.16	4.97	4.65	5.01
EDT – 10x	TC (%)	99.39	95.61	98.86	98.04	97.03	99.88	98.97	98.90
	CPU time	1.01	1.15	0.98	1.18	1.01	1.05	1.23	1.06
	Area (%)	1.02	0.77	0.69	0.66	0.31	0.31	0.24	0.17
	EC (×)	9.50	8.95	9.87	9.38	10.16	9.86	8.29	9.92
EDT – 25x	TC (%)	99.35	94.84	98.83	98.01	97.02	99.86	98.96	98.87
	CPU time	0.98	1.34	1.01	1.04	1.01	0.90	1.26	0.97
	Area (%)	1.72	1.30	1.16	1.11	0.53	0.54	0.41	0.29
	EC (×)	21.71	17.50	23.13	22.4	26.15	22.97	17.15	25.11

available software pattern compression techniques for both ATPG and EDT, including dynamic and static compaction, such that EDT is compared to the best that conventional ATPG technology can produce. The patterns that test the EDT logic were also included in the pattern set.

Note that all these designs have several hundred to several thousand sources of observable unknown states. Hence, none of them, without substantial modifications, would work with LBIST or any other scheme that uses a MISR for output response compaction. In particular, as MISRs do not handle unknown states and provide only limited support for fault diagnosis even if they are reset for every test pattern, the EDT compaction scheme is designed in a distinct manner so that it does not compromise test coverage [33], [34]. It provides the ability to handle X states propagating to scan cells, eliminate aliasing effects completely, and support diagnosis.

The scheme comprises a number of linear spatial compactors driven by outputs of selected scan chains. The scan chains are partitioned into multiple groups where each group is connected to a separate spatial compactor feeding an output scan channel. The EDT test response compaction has the ability to mask scan chains selectively to protect the captured fault effects. Consequently, it allows detecting faults even when they are captured on positions (scan cells) for which there exists at least one corresponding counterpart in another scan chain that captures an unknown state. A similar technique is used to handle those faults that would escape detection due to aliasing. The spatial compactor control signals are treated as an integral part of the test data and are loaded through the decompressor as shown in Fig. 1.

Table VIII summarizes the results of the experiments. For each circuit the table gives the following information:

- the number of gates and scan cells;
- the test coverage (TC), i.e., the percentage of testable faults detected by a conventional ATPG;
- the test coverage after generating compressed scan patterns using the EDT scheme;
- relative CPU time computed as a ratio of the EDT and ATPG run times;
- the area associated with the on-chip decompressor and the selective test response compactor;
- the effective compression (EC), i.e., the ratio of scan data volume for the EDT and ATPG pattern sets. Since all experiments use 16 scan channels, this also represents the ratio of the number of scan cycles between the two pattern sets, and thus the reduction in test-application time.

As can be seen in all cases, virtually the same test coverage as that of ATPG was achieved. Several factors may have contributed to the minor differences in coverage, such as differences in random fill, reduction in fortuitous fault detection due to scan chain masking in designs with many sources of X states, and faults whose test cubes cannot be compressed; this usually only occurs at very high compression levels.

The average effective compression for the targeted compression of 5, 10, and 25 $\times$ is 4.96, 9.44 and 21.57, respectively. Note that in some instances such as in C5 the effective compression can be greater than the target (5 $\times$) due to scan chain imbalances as well as EDT generating slightly fewer patterns than ATPG due to differences in random fill. In other cases such as C7, the effective compression starts to deviate from the target for higher

TABLE IX
EDT TARGETING HIGH LEVELS OF COMPRESSION

	Circuits				
	C3	C4	C5	C6	C8
EDT – 50x					
TC (%)	98.78	97.97	97.00	99.82	98.83
CPU Time	0.93	0.85	0.96	1.02	1.05
EC (x)	37.13	40.63	49.62	39.86	45.85
EDT – 100x					
TC (%)	98.69	97.88	96.90	99.76	98.01
CPU Time	0.82	0.94	0.78	1.17	1.12
EC (x)	54.27	57.74	85.22	59.18	57.79

levels of compression. This is typically observed on designs with a large number of observable X sources. C7 has around 2000 sources of unknown states. So while the EDT compaction technique can tolerate X states, they can have an impact on the effective compression. In designs with relatively few X sources, much higher levels of compression are possible. Table IX extends the targeted compression to $50\times$ and $100\times$ for designs that have relatively fewer sources of unknown states (hundreds instead of thousands). The results demonstrate the scalability of the approach for relatively "clean" designs. For example, C5, which has around 400 sources of unknown states, achieves $85\times$ effective compression for a targeted compression of $100\times$. Another factor affecting the compression is obviously the number of bits specified typically in the test cubes. A large number of ATPG constraints usually results in more bits being specified and therefore lower levels of compression.

As can be seen from the table, EDT-S achieves desired compression with test logic that requires only about 20 gates per internal scan chain. For example, for C8 with over 2 million gates, the EDT logic targeting $25\times$ compression accounted for an overhead of only 0.29%.

Further experiments were also performed to demonstrate the performance of EDT in addressing at-speed defects. The EDT approach can handle embedded PLLs by using its legal clock sequences to generate at-speed patterns. For example, a sequence of four clocks (a slow clock, followed by two at-speed clocks, and a final slow clock) per pattern is required for at-speed transition fault and path delay fault tests in LSSD designs [37]. For a circuit with 1.5 million gates, conventional ATPG required 14.2 M test cycles to obtain 99.2% stuck-at test coverage. For the same circuit, EDT required 3.5M test cycles to get 99.2% test coverage for stuck-at faults, 93.6% for transition faults, and generated 1500 path-delay patterns for testing the critical paths. In other words, EDT enabled the usage of two additional fault models, and still provided $4\times$ compression compared to ATPG targeted for just stuck-at faults.

Finally, by using the volume of compressed test data as a figure of merit, performance of the EDT scheme was compared with other compression schemes. Experiments were run on the largest ISCAS'89 benchmark circuits, and the results are presented in Table X. For all the circuits, EDT produced superior results. There are, however, a few important issues that need to be clarified. First, unlike the other schemes, the presented EDT results assume a compaction scheme at the output, where a fault is marked as detected only if it propagates to the compactor outputs. Second, as the circuits are very small in size, the lowest percentage of specified bits varied between 2% and 5%—thereby imposing a theoretical limit on the compression that the scheme can achieve. Third, since the scan chains were very short, the effective compression was skewed due to the initialization cycles required by the decompressor.

VIII. EXPERIMENTAL RESULTS—PART II

The AVE and NAVE compression schemes were tested on even larger industrial designs. Here, four representative circuits ranging in size from 543 K gates to 10.5 million gates are presented. For each design, conventional ATPG as well as EDT with the highest achievable compression levels were performed by fixing the number of external channels to certain values, utilizing as many internal scan chains as required to guarantee the desired compression level, and selecting the decompressor size in such a way that all test cubes were successfully compressed. Subsequently, using the same test setups, experiments were repeated for the compression schemes using the reduced injection rates. As these experiments focus on input data compression only, results reported here are particularly meaningful for cases in which the number of unknown states is very small, and a test response compaction can handle these states without reducing the effective compression ratios. Examples of such compactors have been recently presented in [25], [28], and [30].

The results of the experiments are summarized in Tables XI (compression rates) and XII (the corresponding encoding efficiencies). Both tables are partitioned into a number of sections corresponding to successive designs whose characteristics are shown in the headers. For each circuit, the table gives the number of gates, the number of scan cells, the number of test patterns, the maximum number of specified bits, as well as the reference compression level obtained by using the EDT-S scheme. Moreover, the scan architecture (the number of scan chains and their length), the number of tester channels, and the decompressor size are provided in each case.

To clarify further how scan architectures were chosen for successive experiments, consider, as an example, circuit D1 consisting of 543 304 gates and 45 044 scan cells. A test set generated for this circuit features a high variance in the number of specified bits: from a few positions in certain test cubes to the maximum value of 2916. Assuming that a desired encoding efficiency is expected to be around 0.9, it follows that in the worst case the required number of variables, which should be injected into the decompressor to assure successful compression, is equal to $2916/0.9 = 3240$. Thus, the compression ratio that can be achieved using the EDT-S scheme equals approximately $45\,044/3240 \approx 13.9\times$. Given the compression ratio and the number of scan cells, one can easily configure the parallel scan chains provided the number of external channels is known. Indeed, in order to inject 3240 variables using C channels, we need roughly $L = 3240/C$ cycles. Consequently, the number of scan chains is given by $45\,044/L$ (in fact, this number can

TABLE X
COMPARISON OF DIFFERENT COMPRESSION SCHEMES—VOLUME OF TEST DATA

Circuit	Scan cells	Illinois scan [9]	FDR codes [3]	Linear decompression [2]	Statistical coding [16]	Seed compression [22]	EDT scheme
s5378	214	14,572	12,346	NA	15,417	6,180	5,676
s9234	247	27,111	22,152	NA	19,912	12,112	9,534
s13207	700	109,772	20,368	2,096	52,741	11,285	10,585
s15850	611	32,758	21,590	16,302	49,163	12,438	9,805
s38417	1664	96,269	57,066	63,936	172,216	34,767	31,458
s38584	1464	96,056	70,328	34,944	128,046	29,397	18,568

TABLE XI
COMPRESSION RATES ($\times$)

D1: gates – 543K, scan cells – 46K, max spec. - 2916, E – 90%, pat: 2447, $\boldsymbol{M}$ – **13.9x**								
Scan	115 × 392	235 × 192	352 × 128	470 × 96	704 × 64	939 × 48	1408 × 32	1877 × 24
C - D	8 -192	16 – 128	24 - 128	32 - 128	48 - 128	64 - 128	96 - 192	128 - 256
NAVE	53.94	54.54	58.15	46.08	60.81	59.44	62.41	61.01
AVE-R	52.93	60.80	63.99	65.20	66.80	66.11	64.81	65.36
AVE-0	45.86	58.08	64.28	67.99	73.22	75.70	80.54	83.86
D2: gates - 2,676K, scan cells – 139K, max spec. - 2707, E – 90%, pat: 11079, $\boldsymbol{M}$ – **45.0x**								
Scan	192 × 720	384 × 360	768 × 180	1151×120	1535 × 90	2302 × 60	3070 × 45	4606 × 30
C - D	4 - 128	8 - 128	16 - 128	24 – 160	32 - 160	48 - 160	64 - 160	96 - 192
NAVE	306.72	311.20	322.39	321.51	325.90	328.81	358.48	344.00
AVE-R	218.58	272.10	319.37	334.33	342.73	343.58	347.96	327.54
AVE-0	131.45	196.32	265.73	301.48	329.77	364.21	394.78	435.62
D2 (0.25% fill rate): max spec. - 664, E – 96%, pat: 11829, $\boldsymbol{M}$ – **196.0x**								
Scan	231 × 598	478 × 289	953 × 145	1919 × 72	3837 × 36	5755 × 24	8125 × 17	9866 × 14
C - D	1 – 144	2 - 144	4 - 160	8 - 160	16 - 160	24 - 192	32 - 192	40 - 192
NAVE	381.77	385.70	385.63	387.99	388.83	386.83	402.56	395.23
AVE-R	257.25	340.33	358.93	395.66	426.77	429.78	431.55	435.61
AVE-0	165.70	257.11	326.89	387.01	436.01	451.62	470.48	495.75
D3: gates - 10,532K, scan cells – 309K, max spec. - 887, pat: 21015, E – 90%, $\boldsymbol{M}$ – **286x**								
Scan	322 × 960	643 × 480	1286×240	2571×120	5141 × 60	7711 × 40	10282×30	12852×24
C - D	1 - 160	2 - 160	4 - 160	8 - 160	16 – 160	24 - 160	32 - 160	40 - 192
NAVE	689.83	704.12	694.91	703.53	711.54	720.05	723.06	718.72
AVE-R	456.86	485.27	639.35	713.65	771.38	793.30	804.06	800.33
AVE-0	248.93	392.67	529.46	651.69	747.18	795.31	837.06	850.57
D4: gates - 1,560K, scan cells – 45K, max spec. - 1249, pat: 8389, E – 90%, $\boldsymbol{M}$ – **32x**								
Scan	71 × 640	141 × 320	282 × 160	563 × 80	833 × 54	1125×40	1406×32	1666×27
C - D	2 - 256	4 - 192	8 - 192	16 - 160	24 - 160	32 - 160	40 -160	48 - 160
NAVE	31.58	31.79	31.88	32.37	32.33	32.72	33.10	32.33
AVE-R	32.86	41.73	43.52	45.66	46.53	45.75	47.26	47.17
AVE-0	32.21	37.35	41.37	44.83	46.39	48.33	48.37	49.07

be slightly bigger as the last formulas do not account for the initialization period).

As indicated by the data in Table XI, application of the examined schemes leads to remarkable compressions. In all the simulated cases, these algorithms yield significantly higher compression levels than those of the EDT-S scheme. As observed earlier, the NAVE-R approach appears to be a superior solution for designs with relatively long scan chains and small number of external channels. With the decreasing size of scan chains and increasing number of ATE channels, the adaptive schemes gain a momentum visibly. Finally, for test setups with very short scan chains, the AVE-0 approach becomes a dominant compression technique. For example, in the case of circuit D2 with almost 140 K scan cells and the EDT-S scheme targeting 45 $\times$ compression, the NAVE-R approach provides a substantial test data volume reduction for first three setups (up to 322 $\times$), then additional savings can be gained due to AVE-R scheme, and eventually AVE-0 technique achieves an order of magnitude im-

TABLE XII
ENCODING EFFICIENCY (%)

D1: gates – 543K, scan cells – 46K, max spec. - 2916, E – 90%, ***M* - 13.9x**								
Scan	115× 392	235× 192	352 × 128	470 × 96	704 × 64	939 × 48	1408 × 32	1877 × 24
C - D	8 - 192	16 - 128	24 - 128	32 -128	48 - 128	64 - 128	96 - 192	128 - 256
NAVE	71.39	72.11	77.00	61.03	80.64	78.80	82.77	80.93
AVE-R	70.05	80.38	84.73	86.35	88.58	87.64	85.95	86.70
AVE-0	60.69	76.79	85.11	89.99	97.10	100.35	106.81	111.23
D2: gates - 2,676K, scan cells – 139K, maxspec. - 2707, E – 90%, ***M* – 45.0x**								
Scan	192× 720	384× 360	768 × 180	1151×120	1535× 90	2302 × 60	3070 × 45	4606 × 30
C - D	4 - 128	8 - 128	16 -128	24 – 160	32 – 160	48 - 160	64 - 160	96 - 192
NAVE	71.71	72.76	75.37	75.23	76.24	76.94	83.87	80.50
AVE-R	51.10	63.62	74.67	78.23	80.18	80.40	81.41	76.65
AVE-0	30.73	45.90	61.13	70.55	77.15	85.23	92.36	101.94
D2 (0.25% fill rate): max spec. - 664, E – 86%, ***M* – 180.0x**								
Scan	231× 598	478× 289	953 × 145	1919 × 72	3837 × 36	5755 × 24	8125 × 17	9866 × 14
C - D	1 – 144	2 - 144	4 – 160	8 - 160	16 - 160	24 - 192	32 - 192	40 - 192
NAVE	71.21	71.96	71.93	72.37	72.55	72.17	75.10	73.73
AVE-R	47.98	63.50	66.95	73.80	79.63	80.18	80.50	81.26
AVE-0	30.91	47.97	60.97	72.19	81.35	84.27	87.77	92.48
D3: gates - 10,532K, scan cells – 309K, max spec. - 887, E – 90%, ***M* – 286x**								
Scan	322× 960	643× 480	1286×240	2571×120	5141 × 60	7711 × 40	10282×30	12852×24
C - D	1 - 160	2 - 160	4 - 160	8 - 160	16 - 160	24 - 160	32 - 160	40 – 192
NAVE	72.74	74.36	73.39	74.33	75.19	76.09	76.41	75.95
AVE-R	48.17	51.25	67.52	75.40	81.51	83.84	84.97	84.58
AVE-0	26.25	41.47	55.92	68.85	78.96	84.05	88.46	89.89
D4: gates - 1,560K, scan cells – 45K, max spec. - 1249, E – 90%, ***M* – 32x**								
Scan	71 × 640	141× 320	282 × 160	563 × 80	833 × 54	1125×40	1406×32	1666×27
C - D	2 - 256	4 - 192	8 - 160	16 - 160	24 - 160	32 - 160	40 -160	48 - 160
NAVE	61.74	62.60	62.78	63.86	63.86	64.60	65.37	63.86
AVE-R	64.25	82.18	85.70	90.08	91.92	93.04	93.33	93.17
AVE-0	62.97	73.54	81.46	88.44	91.64	94.17	95.52	96.94

provement over the reference level by elevating compression to 423 ×. It is worth noting that the results presented in Table XI do not account for additional test data compression, which can be achieved by using various forms of test response compaction. Therefore, it is justified to expect that the actual reduction in test data volume can be significantly higher than indicated by the numbers gathered in the table once time or space compactors are employed.

Note that in one instance (circuit D4), there is no visible improvement as far as the nonadaptive approach is concerned. In this case, there is no significant variance in the number of specified bits, so the amount of variables that could be eliminated is relatively small (compare results obtained by using the AVE schemes). Consequently, the number U of repeated injections in the NAVE scheme was, for most of the test cubes, equal to 1.

The encoding efficiency numbers (Table XII) follow closely the results presented in Table XI. Interestingly enough, however, in one case (circuit D1), the encoding efficiency is greater than 100%. This is typically observed on designs with a large number of variables assuming the value of 0. In such a case, several valid seed patterns appear to be all-zero vectors. They can be eliminated during the second iteration of the algorithm and replaced with single bits indicating injection of all-zero patterns, as shown earlier for the AVE-0 scheme.

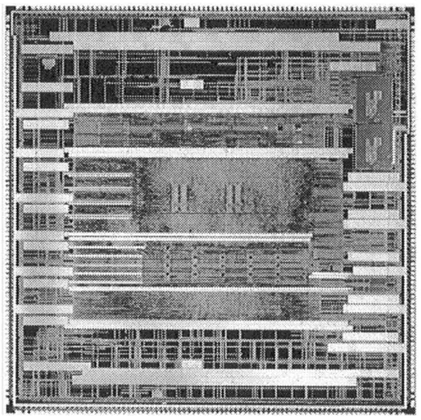
Fig. 7. Silicon with EDT (courtesy of CISCO systems).

IX. Conclusion

In this paper, the EDT method, which reduces manufacturing test cost significantly by providing a dramatic reduction in scan test data volume and scan test time, is presented. The approach has been shown to achieve one order of magnitude compression consistently, and up to two orders of magnitude compression for some designs. EDT is widely applicable and easy to deploy because it is based on the standard scan/ATPG methodology and adopts a very simple flow that adheres closely to the conventional ATPG flow. Note that using the EDT approach, high test quality is guaranteed as it is possible to target any fault model, such as stuck-at, transition, path delay, multiple detects, etc., that are handled by commercial ATPG tools. The methodology allows the generation of a compressed set of test patterns for these fault models that would provide similar test coverage when compared to any deterministic method.

Fig. 7 is a microphotograph of a design incorporating the EDT logic. This design contains about 500 K gates and 31 K scan cells. The number of channels and chains are 5 and 65, respectively. The resulting effective compression that was achieved is 11 ×. The EDT logic can be placed either as a wrapper surrounding the original design, or it can be instantiated within the design. In both cases, however, it does not affect the functional paths. Given a design, the architecture of the EDT logic depends primarily on the number of internal scan chains, the number of external channels, the number of specified positions and a target compression ratio, as well as the number of captured unknown states. In particular, the decompressor size is usually chosen as a small multiplicity of the number of test pins, large enough, however, to maintain high encoding efficiency. Only logic that lies at the interface of the EDT hardware and the scan chains depends on the clocking of the first and last scan cells in every scan chain. The EDT logic is therefore pattern independent and in most cases does not have to be regenerated if a design changes.

The EDT scheme is nonintrusive as it does not require any modifications to the core logic, such as the insertion of test points or X-bounding logic [31]. Finally, the EDT fits into the core-based design paradigm, where only a few external pins can be employed to drive a large number of internal scan chains in the cores. As design size doubles every 18 months, the compression ratio can be scaled accordingly so that all the scan data can fit on existing testers for at least a decade. Consequently, EDT is ideally suited as the next generation DFT technology for low-cost high-quality manufacturing test targeting deep submicron devices.

Acknowledgment

The authors would like to thank N. Tamarapalli, K.-H. Tsai, C. Wang, and G. Mrugalski of Mentor Graphics, as well as A. Hertwig, R. Thompson, and G. Eide, who were with Mentor Graphics when EDT was developed and deployed, for their valuable contributions. They would also like to thank J. Qian of Cisco Systems for numerous discussions, constructive comments, and help in the preparation of this manuscript.

References

[1] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, A. Ferko, B. Keller, D. Scott, B. Koenemann, and T. Onodera, “Extending OPMISR beyond 10 × scan test efficiency,” *IEEE Design Test Comput.*, vol. 19, pp. 65–73, Sept./Oct. 2002.

[2] I. Bayraktaroglu and A. Orailoglu, “Test volume and application time reduction through scan chain concealment,” in *Proc. Design Automation Conf.*, 2001, pp. 151–155.

[3] A. Chandra and K. Chakrabarty, “Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression,” in *Proc. VLSI Test Symp.*, 2001, pp. 42–47.

[4] ——, “Test data compression for system-on-a-chip using Golomb codes,” in *Proc. VLSI Test Symp.*, 2000, pp. 113–120.

[5] D. Das and N. A. Touba, “Reducing test data volume using external/LBIST hybrid test patterns,” in *Proc. ITC*, 2000, pp. 115–122.

[6] R. Dorsch and H.-J. Wunderlich, “Tailoring ATPG for embedded testing,” in *Proc. ITC*, 2001, pp. 530–537.

[7] P. T. Gonciari, B. M. Al-Hashimi, and N. Nicolici, “Variable length input Huffman coding for system-on-a-chip test,” *IEEE Trans. Computer-Aided Design*, vol. 22, pp. 783–796, June 2003.

[8] I. Hamzaoglu and J. Patel, “Reducing test application time for built-in self-test pattern generators,” in *Proc. VLSI Test Symp.*, 2000, pp. 369–376.

[9] ——, “Reducing test application time for full scan embedded cores,” in *Proc. Int. Symp. Fault Tolerant Comput.*, 1999, pp. 260–267.

[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. Comput.*, vol. C-44, pp. 223–233, Feb. 1995.

[11] S. Hellebrand, H.-G. Liang, and H.-J. Wunderlich, “A mixed mode BIST scheme based on reseeding of folding counters,” in *Proc. ITC*, 2000, pp. 778–784.

[12] 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. ITC*, 1999, pp. 358–367.

[13] F. Hsu, K. Butler, and J. Patel, “A case study on the implementation of the Illinois scan architecture,” in *Proc. ITC*, 2001, pp. 538–547.

[14] H. Ichihara, A. Ogawa, T. Inoue, and A. Tamura, “Dynamic test compression using statistical coding,” in *Proc. ATS*, 2001, pp. 143–148.

[15] H. Ichihara, K. Kinoshita, I. Pomeranz, and S. Reddy, “Test transformation to improve compaction by statistical encoding,” in *Proc. Int. Conf. VLSI Design*, 2000, pp. 294–299.

[16] A. Jas, J. Ghosh-Dastidar, and N. A. Touba, “Scan vector compression/decompression using statistical coding,” in *Proc. VLSI Test Symp.*, 1999, pp. 114–120.

[17] A. Jas and N. A. Touba, “Test vector decompression via cyclical scan chains and its application to testing core-based designs,” in *Proc. ITC*, 1998, pp. 458–464.

[18] B. Keller, C. Barnhart, V. Brunkhorst, F. Distler, A. Ferko, O. Farnsworth, and B. Koenemann, “OPMISR: The foundation for compressed ATPG vectors,” in *Proc. ITC*, 2001, pp. 748–757.

[19] B. Koenemann, “LFSR-coded test patterns for scan designs,” in *Proc. Eur. Test Conf.*, 1991, pp. 237–242.

[20] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, “A smartBIST variant with guaranteed encoding,” in *Proc. ATS*, 2001, pp. 325–330.

[21] C. Krishna, A. Jas, and N. A. Touba, “Test vector encoding using partial LFSR reseeding,” in *Proc. ITC*, 2001, pp. 885–893.

[22] C. V. Krishna and N. A. Touba, “Reducing test data volume using LFSR reseeding with seed compression,” in *Proc. ITC*, 2002, pp. 321–330.

[23] K.-J. Lee, J.-J. Chen, and C.-H. Huang, “Using a single input to support multiple scan chains,” in *Proc Int. Conf. Computer-Aided Design*, 1998, pp. 74–78.

[24] H.-G. Liang, S. Hellebrand, and H.-J. Wunderlich, “Two-dimensional test data compression for scan-based deterministic BIST,” in *Proc. ITC*, 2001, pp. 894–902.

[25] S. Mitra and K. S. Kim, “X-Compact: An efficient response compaction technique for test cost reduction,” in *Proc. ITC*, 2002, pp. 311–320.

[26] G. Mrugalski, J. Rajski, and J. Tyszer, “Linear independence as evaluation criterion for two-dimensional test pattern generators,” in *Proc. VLSI Test Symp.*, 2000, pp. 377–386.

[27] ——, “High speed ring generators and compactors of test data,” in *Proc. VLSI Test Symp.*, 2003, pp. 57–62.

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

[29] J. Rajski, N. Tamarapalli, and J. Tyszer, "Automated synthesis of phase shifters for built-in self-test applications," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 1175–1188, Oct. 2000.

[30] J. Rajski, J. Tyszer, C. Wang, and S. M. Reddy, "Convolutional compaction of test responses," in *Proc. ITC*, 2003, pp. 745–754.

[31] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Test Pattern Compression for an Integrated Circuit Test Environment," US Patent Number 327687, December 4, 2001.

[32] ——, "Method for Synthesizing Linear Finite State Machines," US Patent Number 353842, Mar. 5, 2002.

[33] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide, and J. Qian, "Embedded deterministic test for low cost manufacturing test," in *Proc. ITC*, 2002, pp. 301–310.

[34] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Method and Apparatus for Selectively Compacting Test Responses," US Patent Number 557129, Apr. 29, 2003.

[35] J. Rajski, J. Tyszer, and N. Zacharia, "Test data decompression for multiple scan designs with boundary scan," *IEEE Trans. Comput.*, vol. 47, pp. 1188–1200, Nov. 1998.

[36] S. M. Reddy, K. Miyase, S. Kalihara, and I. Pomeranz, "On test data volume reduction for multiple scan chain design," in *Proc. VLSI Test Symp.*, 2002, pp. 103–108.

[37] N. Tendolkar, R. Raina, R. Woltenberg, X. Lin, B. Swanson, and G. Aldrich, "Novel techniques for achieving high at-speed transition fault test coverage for motorola's microprocessor based on PowerPC instruction set architecture," in *Proc. VLSI Test Symp.*, 2002, pp. 3–8.

[38] "Test data compression—ITC 2002 roundtable," *IEEE Design Test Comput.*, vol. 20, pp. 76–87, Mar./Apr. 2003.

[39] E. H. Volkerink and S. Mitra, "Efficient seed utilization for reseeding based compression," in *Proc. VLSI Test Symp.*, 2003, pp. 232–237.

Janusz Rajski (A'87) received the M.Eng. degree in electrical engineering from the Technical University of Gdansk, Gdansk, Poland, in 1973 and the Ph.D. degree in electrical engineering from the Poznan University of Technology, Poznan, Poland, in 1982.

From 1973 to 1984, he was a member of the faculty at the Poznan University of Technology. In June 1984, he joined McGill University, Montreal, Canada, where he became an Associate Professor in 1989. In January 1995, he became a Chief Scientist at Mentor Graphics Corporation, Wilsonville, OR. He has performed contract work and has been a consultant to a number of companies in the area of testing. He has published more than 100 research papers. He is the coauthor of *Arithmetic Built-In Self-Test for Embedded Systems* (Englewood Cliffs, NJ: Prentice Hall, 1997). His main research interests include design automation and testing of VLSI systems, design for testability, built-in self-test, and logic synthesis.

Dr. Rajski was a guest Co-editor of the special issue of the IEEE COMMUNICATIONS MAGAZINE devoted to testing telecommunication hardware, and an Associate Editor for IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON COMPUTERS, and the IEEE DESIGN AND TEST OF COMPUTERS MAGAZINE. He is a Member of the editorial board of the *Journal of Electronic Testing* (JETTA). He was a co-recipient of the 1993 Best Paper Award for a paper on synthesis of testable circuits published in the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the 1995 and 1998 Best Paper Awards at the IEEE VLSI Test Symposium, and the 1999 Honorable Mention Award at the IEEE International Test Conference. He was Guest Co-editor of the June 1990 and January 1992 special issues of *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* devoted to the 1987 and 1989 International Test Conferences, respectively. He has served on technical program committees of various conferences and coordinated the topic of automatic test pattern generation and delay testing for the International Test Conference. He is a Co-founder of the International Test Synthesis Workshop.

Jerzy Tyszer (M'91–SM'96) received the M.Eng. (honors) and Ph.D. degrees in electrical engineering from the Poznan University of Technology, Poznan, Poland, in 1981 and 1987, respectively, and the Dr. Habilis degree in telecommunications from the Technical University of Gdansk, Gdansk, Poland, in 1994.

From 1982 to 1990 he was a member of the faculty at the Poznan University of Technology, Poland. In January 1990, he joined McGill University, Montreal, Canada, where he was a Research Associate and an Adjunct Professor. In 1996, he became a Professor at the Institute of Electronics and Telecommunications, Poznan University of Technology, where he carries on his work in the areas of testing and synthesis of self-testable circuits. He has done contract work and has been a consultant in the area of testing to a number of companies. He has published seven books, more than 70 research papers in the following areas and jointly holds 16 patents. He is a coauthor of *Arithmetic Built-In Self-Test for Embedded Systems* (Englewood Cliffs, NJ: Prentice Hall, 1997) and the author of *Object-Oriented Computer Simulation of Discrete Event Systems* (Norwell, MA: Kluwer, 1999). His main research interests include design automation and testing of VLSI systems, design for testability, built-in self-test, digital switching, and computer simulation of discrete event systems.

He was a co-recipient of the 1995 and 1998 Best Paper Awards at the IEEE VLSI Test Symposium. In 1999, he was a Guest Co-editor of the special issue of the IEEE COMMUNICATIONS MAGAZINE devoted to telecommunication hardware testing. He has served on the technical program committees of various conferences.

Mark Kassab (S'91–M'93) received the M.Eng. and Ph.D. degrees in electrical engineering from McGill University, Montreal, Canada, in 1994 and 1996, respectively.

In July 1996, he joined the Design-for-Test Division, Mentor Graphics Corporation, Wilsonville, OR, where he currently manages the ATPG and Embedded Deterministic Test (EDT) Development Group. He is a co-inventor of the EDT technology, and the Lead Engineer and Architect for TestKompress. He has published more than ten research papers in the following areas and jointly holds six patents. His main research interests include design for testability, improving the quality of test, test-cost reduction, design automation, and design of high-performance computer-aided design tools.

He was a co-recipient of the 1995 Best Paper Award at the IEEE VLSI Test Symposium, and the 1999 Honorable Mention Award at the IEEE International Test Conference for papers on built-in self-test.

Nilanjan Mukherjee (S'87–M'89) received the B.Tech. (honors) degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, the M.Sc. degree in computing and information sciences from University of Guelph, Guelph, Canada, and the Ph.D. degree from McGill University, Montreal, Canada, in 1996.

He is currently managing a technical group in the Design, Verification, and Test division at Mentor Graphics Corporation. His research focuses on developing next-generation test methodologies for deep submicron designs, test data compression, test synthesis, memory testing, and fault diagnosis. Prior to joining Mentor Graphics, he worked at Lucent Bell Laboratories, Princeton, NJ, where he primarily contributed in the areas of logic BIST, RTL testability analysis, path-delay testing, and online testing. He has authored or coauthored more than 30 technical articles in various IEEE journals and conferences. He was an invited author for the special issue of the IEEE COMMUNICATIONS MAGAZINE, June 1999. He jointly holds nine US patents.

Dr. Mukherjee was the recipient of the Best Paper Award at the 1995 IEEE VLSI Test Symposium and coauthored a paper that received the Best Student Paper Award at the Asian Test Symposium in November 2001.