EMBEDDED DETERMINISTIC TEST FOR LOW COST MANUFACTURING TEST

Janusz Rajski, Jerzy Tyszer[*], Mark Kassab, Nilanjan Mukherjee, Rob Thompson, Kun-Han Tsai, Andre Hertwig, Nagesh Tamarapalli, Grzegorz Mrugalski[*], Geir Eide[1], and Jun Qian[**]

Mentor Graphics Corporation
8005 S.W. Boeckman Road
Wilsonville, OR 97070, USA

[*]Poznan University of Tech.
ul. Piotrowo 3a
60-965 Poznan, Poland

[**]Cisco Systems
1790 W Tasman Drive
San Jose, CA 95129, USA

Abstract

This paper introduces Embedded Deterministic Test (EDT) technology, which reduces manufacturing test cost by providing one to two orders of magnitude reduction in scan test data volume and scan test time. The EDT architecture, the compression algorithm, design flow, experimental results, and silicon implementation are presented.

1. 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 almost complete fault coverage of several types of fault models. Typically, when an ATPG targets a fault, only a small number of scan cells get 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 over-specified.

For large circuits, the growing test data volume causes a significant increase in test cost because of much longer test time and elevated tester memory requirements. For a scan-based test, the test data volume can be approximately expressed as follows:

$$Test\ Data\ Volume \approx Scan\ Cells \times Scan\ Patterns$$

Thus, a relationship between test time and the volume of test data is given by the following equation:

$$Test\ Time \approx \frac{Scan\ Cells \times Scan\ Patterns}{Scan\ Chains \times Frequency}$$

Consider an example circuit consisting of 10M gates and 16 scan chains. Typically, the number of scan cells is proportional to the design size. Thus, assuming one scan cell per 20 gates, the total test time to apply 10,000 scan patterns at a 20 MHz scan-shift frequency will take roughly 312 millions test cycles or equivalently 15 seconds. As designs grow in size, it becomes increasingly expensive to maintain a high level of test coverage. This is due to a prohibitively large volume of test data that must be stored in the testing equipment, an increasingly long test application time, and a very high and continually increasing logic-to-pin ratio creating a test data transfer bottleneck at the chip pins.

The above problems can be overcome to some extent by adopting an embedded scheme such as logic built-in self-test (LBIST), which is a solution based on pseudo-random patterns generated and evaluated on chip. In LBIST, a high coverage of stuck-at faults can be achieved provided that test points are employed to address random pattern resistance. Other types of fault models, such as path delay faults, are not handled efficiently by pseudo-random 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 [6]. Furthermore, deterministic tests are still needed to target the remaining random pattern resistant faults. Very often the memory required to store the top-up patterns in BIST can exceed 30% of the memory used in a conventional ATPG approach. Because of increasing circuit complexity, storing an extensive set of top-up ATPG patterns targeting hard-to-test faults, either on-chip or off-chip, becomes prohibitive. Accordingly, the overall efficiency of any testing scheme strongly depends on a method employed to reduce the amount of test data [1] - [3], [8], [9], [14].

Several methods for compressing test data exploit the fact that the test cubes frequently feature a large number of unspecified positions. The original idea taking advantage of this phenomenon, known as LFSR-coding, was proposed in [11] and then refined (under the name of LFSR-

[1]Currently with Teseda Corporation, Portland, OR 97205, USA

0-7803-7542-4/02 $17.00

reseeding) in a number of approaches [5], [13], [20]. The encoding capability of most of these methods is limited by the LFSR size. Therefore, a large LFSR is 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 non-overlapping phases, resulting in inefficient utilization of the tester. Furthermore, the reseeding patterns are not applied in a similar manner to conventional ATPG patterns due to two distinct phases, which may cause problems for downstream tools and some scan testers.

An attempt to reduce both test application time and test data volume was presented in [4], [7]. This so-called Illinois scan scheme divides the scan chains into partitions and shifts in the same test vector to each scan chain through a single scan input. Clearly, a given test pattern must not contain contradictory values on corresponding cells in different chains loaded through the same input. Performance of this scheme strongly relies on the scan chain configuration, and therefore it is not easily scalable.

Decrease of test data along with a 2x scan test time reduction is also accomplished in the scheme based on the On-Product MISR [10]. 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 the OPMISR architecture is proposed in SmartBIST [12], where tester channels are again used to deliver stimuli, while signatures are observed only at the end of an unload cycle. The test patterns, however, 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 very strongly depends on the volume of test data and test time, one of the key requirements for a next-generation DFT methodology is to dramatically reduce both. This new technology 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 or defect-based testing. Furthermore, the technology should have the ability to perform on-the-fly diagnostics in a manufacturing test floor flow.

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, i.e., have minimal impact on present scan design flows, design habits or styles, require no modifications to the current knowledge and skill set for scan and ATPG, and not require replacement of the existing test equipment. In other words, the user should not have to make major changes as a precondition for obtaining the benefits in reduction of test data volume and test time. Finally, long-term scalability has to be taken into account, as it is hard to expect that the industry will be retooling smoothly at a rate of one technology for every 18 months in order to keep up with Moore's law. Thus, one compression technology is required that can deliver the increasingly higher levels of compression and fit the compressed test data into the current 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 logic BIST nor conventional ATPG are ideal solutions for designs requiring high quality test and minimal impact of DFT circuitry on the design. The Embedded Deterministic Test (EDT) technology 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 earlier.

2. EDT architecture

The EDT technology consists of logic embedded on a chip and a new deterministic test pattern generation technique [17]. 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 external scan channel inputs and the internal scan chain inputs. An on-chip *selective compactor* is inserted between the internal scan chain outputs and the external scan channel outputs. Optionally, the EDT architecture may also include logic to bypass the decompressor and the compactor, thereby making the internal scan chains directly accessible from the ATE. No additional

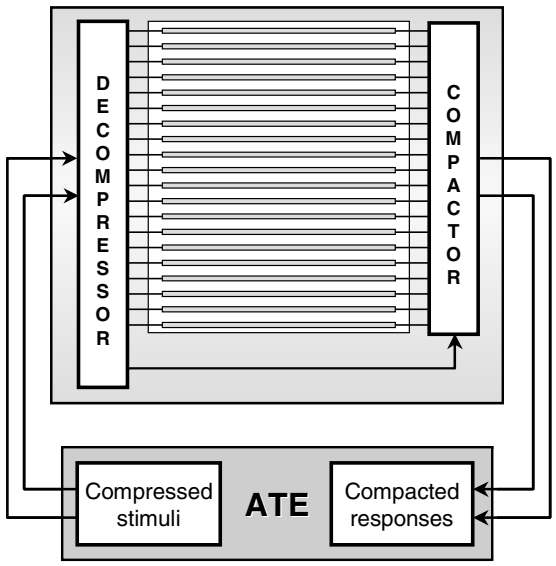

Fig. 1. EDT architecture

logic such as test points or X-bounding logic needs to be inserted into the core of the design. Therefore the EDT logic only affects scan channel inputs and outputs, and has no effect on functional paths. The primary objective of the EDT technology is to drastically reduce tester memory requirements, 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. Since the scan chains are balanced, they are 10 times shorter.

The EDT pattern generation is truly deterministic. For a given testable fault, as with conventional ATPG, a pattern is generated to satisfy the ATPG constraints and to avoid bus contention. 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 in length. 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 in such a way that only selected scan chains may stream their contents to the compactor, if necessary.

Example. To further illustrate the compression in EDT, consider a real design (see Fig. 2). In ATPG, it was configured into 16 scan chains with a maximum length of 2590. The test required 4527 patterns. In EDT, the same design was configured into 160 internal scan chains with a maximum length of 263. In total, 4540 test patterns were required to achieve the same fault coverage as that of ATPG. As can be seen, with almost the same number of patterns, the length of scan chains decreased by almost tenfold. The effective reduction of volume of scan test data and scan test time is 9.8 times. It is also worth noting how the tester memory is utilized in both cases. One ATPG scan pattern takes 2590 vectors of the tester memory. On the other hand, one EDT pattern occupies 263 vectors of the same memory. Within the memory required

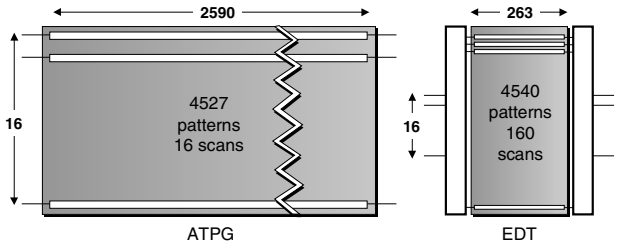

Fig.2. ATPG vs. EDT example

for one ATPG scan pattern, one can store almost 10 EDT patterns. Moreover, in the time required to apply one ATPG pattern, almost 10 EDT patterns can be applied. Serial simulation of one EDT pattern is also 10 times faster than simulation of one ATPG pattern.

3. Decompressor structure

Since the decompressor plays a crucial role in determining the effectiveness of EDT test data compression, it has to satisfy several requirements, including:

- very low linear dependency in its outputs,
- very high speed of operation,
- very low silicon area,
- high modularity of the design.

Although it is possible to build a decompressor based on LFSRs or CAs, a completely new and original architecture, called a ring generator, was developed for this application. The ring generator is a distinct form of a linear finite state machine and, to perform on-chip decompression, it is further connected to a linear phase shifter (Fig. 3). The phase shifter is necessary to drive a relatively large number of scan chains and reduce linear dependen-

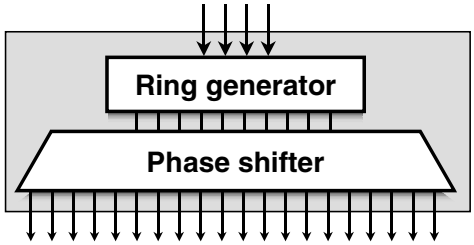

Fig. 3. On-chip decompressor

cies between sequences entering the scan chains. In addition, the phase shifter is designed to guarantee a balanced usage of all memory elements in the ring generator, and it

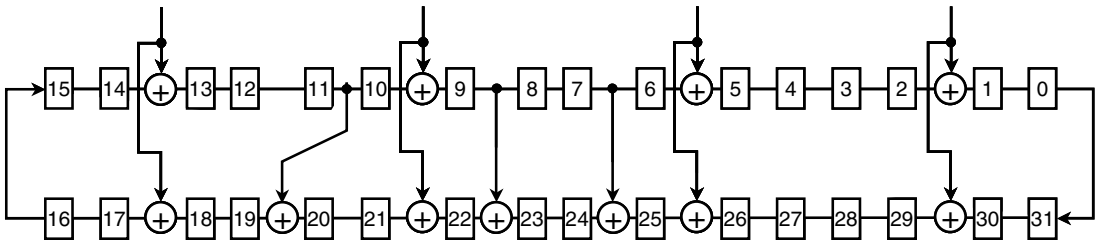

Fig. 4. The ring generator implementing polynomial $x^{32} + x^{18} + x^{14} + x^9 + 1$

introduces a minimal propagation delay between the outputs of the ring generator and the serial inputs of 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. 4. It has been obtained by applying a number of *m*-sequence preserving transformations to the canonical form of the type I LFSR featuring the same feedback polynomial. Details of the ring generator synthesis process can be found in [18]. The proposed structure has three main benefits as compared with conventional LFSRs or CAs. 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 fan-out is limited to only two devices fed by any stem in the ring generator. Furthermore, the total length of feedback lines is drastically reduced. The circuit in Fig. 4, for instance, features only three very short connections. In general, a typical ring generator has fewer levels of logic than a corresponding type I LFSR and a smaller fan-out than the original type II LFSR. Hence, it can operate at higher speeds than those of conventional solutions and is layout- and timing-friendly. It is therefore well positioned to achieve higher performance than any other pseudo-random test pattern generator used so far.

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. Those 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. 4, each having two injectors. Although there are several schemes that can be employed to configure injectors, the pattern shown in Fig. 4 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 linear phase shifter, which is added to the outputs of the ring generator memory elements in the form of an XOR network, allows the ring generator to mutually displace the produced sequences in various scan paths [16], [18]. As demonstrated in [15], 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 [5] once the inter-chain separation reaches the length of the scan chains. The EDT phase shifter is made of XOR gates with a limited number of inputs (called XOR taps) to reduce propagation delays. In this class of circuits, the effective inter-chain separation is much larger than the required one [16]. As a result, the probabilities of linear dependency become practically independent of the minimal required separation; they closely follow the limiting bounds and allow one to maximize the likelihood of compression of test patterns. All results presented in this paper were obtained assuming that the number of XOR taps is equal to 3.

Decompression. 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 compression can be maximized provided the number V_0 of initial variables is chosen such that it is at least equal to $0.75D$, where D is the decompressor size (the number of memory elements comprising the ring generator).

4. Compression of test stimuli

The concept of continuous flow decompression rests on the fact that deterministic test vectors have typically only between 1 to 5 percent 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 information into a compressed form. This ability is exploited by having a number of inputs driving the circuit, whereas the actual circuit has its memory elements configured into a relatively large number of short scan chains. Consequently, a tester that has a few scan channels and a small memory for storing test data could still drive a fairly large circuit.

Solver of linear equations. 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 [5], [11], [12], [13], [20]. Therefore, in the remaining part of the paper, the compressor will be referred to as the solver.

In order to achieve a high degree of compression, the ATPG algorithm works as follows. Whenever the algorithm generates a test cube (without random fill), the solver is invoked to generate the compressed pattern. If the compressed pattern is scanned in through the decompressor, all the bits that were specified by the ATPG algorithm should match. Other unspecified bits are set to pseudo-random values (based on the decompressor architecture). When dynamic compaction is enabled, the solver works iteratively with ATPG to maximize 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 the incremental mode; i.e., it is invoked several times for a gradually increasing system of linear equations.

There are two main advantages 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 computational complexity of the solver remains significantly less 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 be satisfied 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.

```
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

Encoding efficiency. Performance of the compression scheme was evaluated by means of comprehensive characterization experiments. The primary objective was to measure encoding efficiency for various test setups. The encoding efficiency E is defined as a ratio of successfully encoded specified bits B to the total number of deployed variables, i.e.,

$$E = B / (0.75D + C \times L),$$

where D is the size of the decompressor, C is the number of external channels, and L represents the scan length. Recall that the component $0.75D$ 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 fraction F 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 $S / C \approx E / F$.

In order to determine the encoding efficiency, the following experimental setup was adopted. 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.

Table 1. Encoding efficiency for C = 16, L = 128

	Number of scan chains			
D	128	256	512	1024
32	77.08	77.19	76.21	85.88
40	88.46	88.21	87.46	90.23
48	91.08	91.70	91.16	93.94
56	94.93	94.86	94.00	95.42
64	96.36	95.90	95.92	96.05
72	96.87	96.67	96.00	96.81
80	96.70	96.52	97.02	96.88
88	97.52	97.08	96.75	97.31
96	97.94	97.74	97.83	97.73
104	98.06	97.97	97.79	97.46
112	97.73	97.61	97.51	97.99
120	98.25	98.14	97.89	98.11
128	98.20	98.06	97.95	97.49
136	97.88	97.40	97.41	97.87
144	98.22	97.91	97.99	97.73
152	97.89	97.78	97.80	97.93
160	98.14	98.14	98.00	97.75
168	97.89	97.90	97.66	98.03

As can be seen, this experimental platform mimics the functional behavior of the EDT technology operating in the incremental mode. The main difference is that if EDT 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 of successfully encoding b specified positions (but not b + 1). This information allows one to easily calculate the number of specified bits that can be encoded on average, and then to obtain the final encoding efficiency. Note that in all experiments, each individual test cube was created by randomly determining specified positions. Both the location of the particular scan cell and the value assigned to it were generated by sampling a uniform distribution in the range of $[0, S \times L - 1]$ and [0,1], respectively.

In Table 1, the encoding efficiency numbers for various decompressor sizes are presented assuming that EDT test setups feature 16 external channels (each having 2 injectors) and 128 cells in each scan chain. A close examination of the presented data and many other 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 saturation point is located around 120. For a sufficiently large decompressor size, the number of bits that can be encoded approaches the number of variables that are provided. Results similar to those of Table I can be used to select the most appropriate decompressor size.

5. Selective compaction of test responses

Undoubtedly, MISRs are the most popular test response compaction devices used in a parallel scan chain environment. However, they do not handle unknown states and provide only limited support for fault diagnosis, even if they are reset for every test pattern. Therefore, the EDT compaction scheme is designed in a distinct manner so that it does not compromise test coverage [19]. In particular, it provides the ability to:

- handle X states propagating to scan cells,
- eliminate aliasing effects completely,
- support diagnosis.

As shown in Fig. 6, the scheme comprises a number of linear spatial compactors driven by outputs of selected scan chains. The spatial compactors are used to reduce the outputs of multiple scan chains into a significantly smaller number of test data outputs which is equal to the number of scan channel inputs. The scan chains are partitioned into multiple groups where each group is connected to a separate spatial compactor feeding an output scan channel. It is worth noting that while spatial compactors are essentially XOR trees, they are not necessarily combinational circuits. If the propagation delay through the XOR tree becomes unacceptable with respect to the shift frequency, the XOR tree can be pipelined (see Fig. 6) to allow faster operation.

Scan chain masking. A prominent feature of the EDT test response compaction is its ability to selectively mask scan chains to protect the captured fault effects. Masking test data stored in a scan chain before it is unloaded through the spatial compactor consists of adding logic between the output of the scan chain and the compactor. This logic may force one or more scan chain outputs to 0 before going into the compactor. 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 (X). Indeed, the fault effect will remain visible as it is XOR-ed with the logic value of 0 instead of X.

Aliasing. A similar technique is used to handle those faults

that would escape detection due to a phenomenon known as aliasing. It occurs when the fault effects appear only on corresponding positions in an even number of scan chains and therefore they are XOR-ed in the spatial compactor at the same time. In this scenario, aliasing effects are completely eliminated by masking test data in certain scan chains in such a way that an odd number of fault effects enter the compactor.

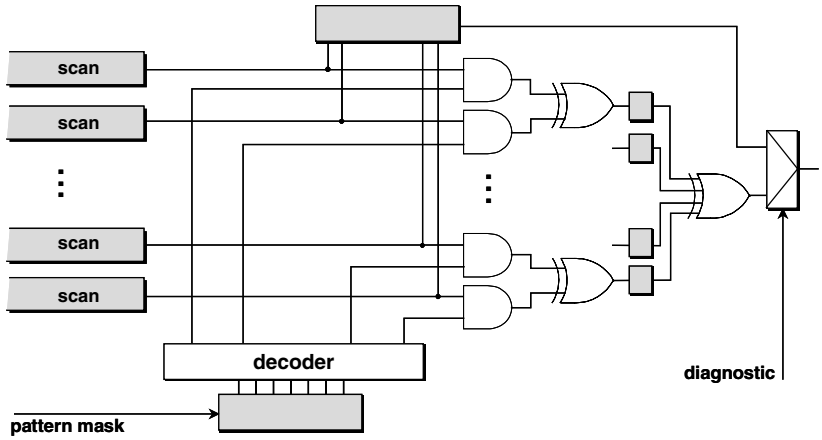

Fig. 6. Test response compaction

Selection logic. In order to facilitate masking of test data captured in scan chains, gating circuitry is added to every spatial compactor. It consists of logic gates and decoders with shift registers. These components facilitate selective control of scan chains in such a way that the required subset of scan chains (connected to a given spatial compactor) propagates their values to the output of the compactor, whereas the remaining ones are blocked. The process of compaction begins by loading the *pattern mask* into the shift register that drives the selection logic. The mask is part of the corresponding test pattern, and is loaded concurrently with the internal scan chains. The content of the pattern mask is compressed in the same manner as done for the test cubes. Thus, 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. Note that the data for the pattern mask can also be provided directly from the ATE during the loading of the scan chains via the decompressor. The scan chain masks decoded from the pattern mask are static for each pattern and are applied throughout the unloading of the captured values. Another masking scheme may be used where the static pattern masks may be disabled on a cycle-by-cycle basis, allowing all scan chains to propagate to the compactor for specific clock cycles.

Fault diagnosis. Finally, the EDT technology supports two schemes for fault diagnosis. In the first scheme, an additional shift register is placed at the outputs of the scan chains as shown in Fig. 6. If a tester recognizes an incorrect test response, it can switch the output multiplexer so that the last captured response can be shifted out bypassing the spatial compactors. The second scheme makes use of the optional bypass mode, where test patterns can be loaded into fewer concatenated scan chains directly from an ATE. Whenever there is a failing pattern, an uncompressed version of the pattern can be loaded directly into the scan chains from the ATE and the corresponding response is observed. Consequently, in both cases, individual test responses can be observed for any test pattern and every scan cell allowing complete reconstruction of circuit responses before compaction is carried out.

6. Design flow

One of the main factors that allows for easy adoption of the EDT technology is its simple flow that closely adheres to the conventional ATPG flow. In the beginning, standard DFT design rule checks are applied to analyze the testability of the circuit. The EDT logic comprising mainly the decompressor and the compactor is then generated for the netlist with pre-inserted scan chains. Preliminary test pattern generation can be performed at this stage to determine the test coverage and scan volume compression. Note that using the EDT technology, it is possible to target any fault model, such as stuck-at, transition, path delay, multiple detects, etc., that are handled by conventional ATPG tools. The technology 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.

Synthesis of the EDT logic. 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. In addition to the main components, the EDT logic may also include circuitry to facilitate the bypass mode of operation. This bypass circuitry allows the internal scan chains to be concatenated into fewer and longer scan chains that are directly accessible from the ATE. In addition to diagnosis, the bypass mode can be used for conventional ATPG as well as for system debug.

The insertion of the EDT logic requires some control pins that are necessary to drive the controller, as well as scan channel pins that are now the new scan I/Os for the design. The scan channel pins and the EDT control pins can be shared with functional pins such that no additional pins are required for EDT. Fig. 7 illustrates a complete block diagram of the EDT logic along with the original core. The output sharing logic consists of multiplexors that

facilitate the aforementioned pin sharing.

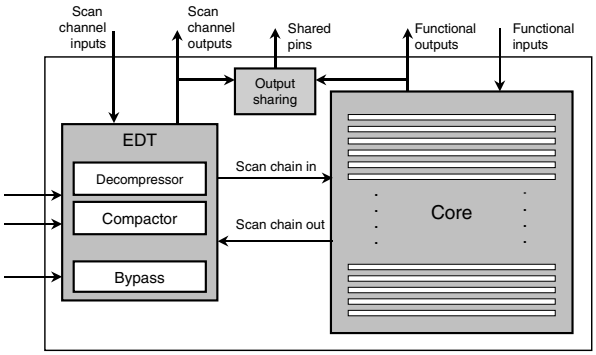

Fig. 7. EDT logic for a given core

Given a design, the architecture of the EDT logic depends primarily on the number of internal scan chains and the number of external scan channels. 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. Once the EDT components are instantiated, the boundary scan logic and I/O pads can be inserted. On the other hand, if a netlist contains pre-inserted I/O pad cells, the EDT logic can be placed between the I/O pad cells and the original core. Subsequently, the EDT logic is synthesized at the gate level along with the boundary scan logic and I/O pad cells.

Pattern generation and EDT logic test. The next step in the flow is the generation of the final set of EDT test patterns. The final test pattern set also contains a few scan patterns (typically around 20 patterns for 10x compression) that test the EDT logic and the integrity of the scan chains. In addition to guaranteeing very high test coverage of the EDT logic, those patterns also help in debugging simulation mismatches. The ring generator is thoroughly exercised and observed as all the patterns are shifted through it into the scan chains. The phase shifter is tested pseudo-exhaustively. All masking modes of the compactor, including the selection logic and the spatial compactors, are also tested thoroughly. The optional bypass logic, if synthesized, is tested in the EDT mode by a sequential pattern, where the bypass mode of operation is simulated for a number of clock cycles. In essence, these additional test patterns guarantee over 99% stuck-at coverage for the entire EDT logic. Finally, before test pattern sign-off, the generated patterns may be verified using a timing-based simulator.

7. Experimental results

The EDT technology was tested on a large number of industrial designs. In this section, 9 representative designs ranging in size from 367K gates to 10.9 million gates are presented. For all the designs (except C9), conventional ATPG with 16 scan chains as well as EDT with three different compression levels, i.e., 5x, 10x and 25x, were performed by fixing the number of scan channels to 16 and utilizing 80, 160, and 400 internal scan chains, respectively. In case of C9, the number of scan chains was fixed, and the number of scan channels was varied to get approximately 5x, 10x, and 25x compression levels. Note that all these designs have several hundred to several thousand sources of observable unknown states. They are also random-pattern resistant. Hence none of them, without substantial modifications, would work with LBIST or any other scheme that uses a MISR for response compaction. The circuits represent different design styles and scan methodologies. The experiments were performed using every 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 EDT pattern set. The results of the experiments are summarized in Table 2. For each circuit the following information is presented:

- the number of gates and memory elements (MEs),
- the test coverage (TC - the percentage of testable faults that are detected) for a conventional ATPG tool,
- the test coverage after generating compressed scan patterns using the EDT technology,
- 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; this also represents the ratio of the number of scan cycles between the two pattern sets, and therefore, 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 contribute 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,
- faults whose test cubes cannot be compressed; this usually occurs only at very high compression levels.

The average effective compression for the targeted compression of 5x, 10x and 25x is 4.87, 9.36 and 21.56, respectively. Note that in some instances such as in C5, the effective compression can be greater than the target (5x) 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 levels of compression. This is typically observed on designs with

Table 2. Experimental results

		Circuits								
		C1	C2	C3	C4	C5	C6	C7	C8	C9
Circuit parameters	Gates	367K	498K	543K	577K	1.2M	1.2M	1.5M	2.1M	10.9M
	MEs	19K	30K	45K	41K	71K	63K	86K	181K	321K
ATPG	TC (%)	99.41	95.63	98.88	98.05	97.04	99.89	98.98	98.84	94.37
EDT – 5x	TC (%)	99.40	95.63	98.87	98.05	97.04	99.88	98.98	98.86	94.35
	CPU time	0.99	1.15	1.13	1.18	1.01	1.16	1.40	1.08	1.41
	Area (%)	0.71	0.54	0.48	0.46	0.22	0.22	0.17	0.12	0.03
	EC (×)	4.93	4.96	4.97	4.67	5.16	4.97	4.65	5.01	4.55
EDT – 10x	TC (%)	99.39	95.61	98.86	98.04	97.03	99.88	98.97	98.90	94.35
	CPU time	1.01	1.15	0.98	1.18	1.01	1.05	1.23	1.06	1.19
	Area (%)	1.02	0.77	0.69	0.66	0.31	0.31	0.24	0.17	0.03
	EC (×)	9.50	8.95	9.87	9.38	10.16	9.86	8.29	9.92	8.28
EDT – 25x	TC (%)	99.35	94.84	98.83	98.01	97.02	99.86	98.96	98.87	94.32
	CPU time	0.98	1.34	1.01	1.04	1.01	0.90	1.26	0.97	1.03
	Area (%)	1.72	1.30	1.16	1.11	0.53	0.54	0.41	0.29	0.02
	EC (×)	21.71	17.50	23.13	22.4	26.15	22.97	17.15	25.11	17.92

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 3 extends the targeted compression to 50x and 100x for designs that have relatively fewer sources of unknown states (hundreds instead of thousands). The results demonstrate the scalability of the technology for relatively "clean" designs. For example, C5, which has around 400 sources of unknown states, achieves 85x effective compression for a targeted compression of 100x. Another factor affecting the compression is obviously the number of bits typically specified 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 achieves remarkable compression with test logic that requires only about 20 gates per internal scan chain (including bypass logic but not the diagnostic register). For example, in order to achieve 25x compression, the average area overhead for the EDT logic is around 0.88%. The average CPU run time to generate test patterns for EDT is about 1.09 times the ATPG run time.

Fig. 8 illustrates a microphotograph of a design incorporating the EDT logic. This design contains about 500K gates and 31K scan cells. The number of channels and chains are 5 and 65, respectively. The resulting effective compression that was achieved is 11x.

Further experiments were also performed to demonstrate the performance of EDT in addressing at-speed defects. The EDT technology can handle embedded PLLs by using its legal clock sequences to generate at-speed patterns. Typically, 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 [21]. For a circuit with 1.5 million gates, conventional ATPG required 14.2M 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 4x compression compared to ATPG targeted for just stuck-at faults.

Table 3. 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

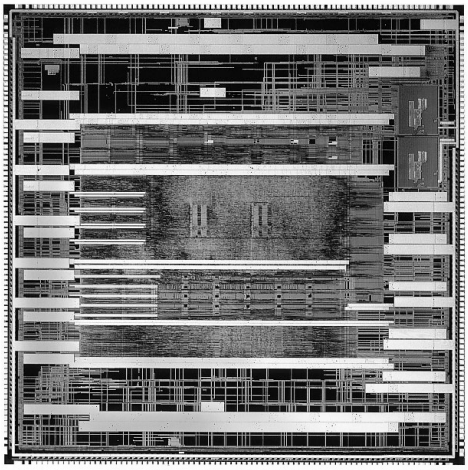

Fig. 8. First silicon with EDT
Courtesy of CISCO Systems

8. Conclusions

In this paper, the EDT technology, which significantly reduces manufacturing test cost by providing a dramatic reduction in scan test data volume and scan test time, is introduced. The technology has been shown to consistently achieve compression in excess of one order of magnitude. EDT 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 non-intrusive, as it does not require any modifications to the core logic, such as the insertion of test points or X-bounding logic. High test quality is guaranteed as the technology can work with all the traditional fault models. The EDT technology also 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 sizes 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 sub-micron devices.

References

1. I. Bayraktaroglu and A. Orailoglu, "Test volume and application time reduction," *Proc. DAC*, pp. 151-155, 2001.
2. D. Das and N.A. Touba, "Reducing test data volume using external/LBIST hybrid test patterns," *Proc. ITC*, pp. 115-122, 2000.
3. R. Dorsch and H.-J. Wunderlich, "Tailoring ATPG for embedded testing," *Proc. ITC*, pp. 530-537, 2001.
4. I. Hamzaoglu and J. Patel, "Reducing test application time for built-in self-test pattern generators," *Proc. VLSI Test Symp.*, pp. 369-376, 2000.
5. 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. on Comput.*, vol. 44, pp. 223-233, 1995.
6. G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic BIST for large industrial designs: real issues and case studies," *Proc. ITC*, pp. 358-367, 1999.
7. F. Hsu, K. Butler, and J. Patel, "A case study on the implementation of the Illinois scan architecture," *Proc. ITC*, pp. 538-547, 2001.
8. A. Jas, J. Ghosh-Dastidar, and N.A. Touba, "Scan vector compression/decompression using statistical coding," *Proc. VLSI Test Symp.*, pp. 114-120, 1999.
9. A. Jas and N.A. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based designs," *Proc. ITC*, pp. 458-464, 1998.
10. B. Keller, C. Barnhart, V. Brunkhorst, F. Distler, A. Ferko. O. Farnsworth, and B. Koenemann, "OPMISR: The foundation for compressed ATPG vectors," *Proc. ITC*, pp. 748-757, 2001.
11. B. Koenemann, "LFSR-coded test patterns for scan designs," *Proc. European Test Conference*, pp. 237-242, 1991.
12. B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and, D. Wheater, "A SmartBIST variant with guaranteed encoding," *Proc. ATS*, pp. 325-330, 2001.
13. C. Krishna, A. Jas, and N.A. Touba, "Test vector encoding using partial LFSR reseeding," *Proc. ITC*, pp. 885-893, 2001.
14. H-G. Liang, S. Hellebrand, and H.-J. Wunderlich, "Two-dimensional test data compression for scan-based deterministic BIST," *Proc. ITC*, pp. 894-902, 2001.
15. G. Mrugalski, J.Rajski, and J. Tyszer, "Linear independence as evaluation criterion for two-dimensional test pattern generators," *Proc. VLSI Test Symp.*, pp. 377-386, 2000.
16. J. Rajski, N. Tamarapalli, and J. Tyszer, "Automated synthesis of phase shifters for built-in self-test applications," *IEEE Trans. CAD*, vol. 19, No. 10, October 2000, pp. 1175-1188.
17. J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Test pattern compression for an integrated circuit test environment," USA patent, Serial No.6,327,687, December 4, 2001.
18. J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Method for synthesizing linear finite state machines," USA patent, Serial No. 6,353,842, March 5, 2002.
19. J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Selective linear compactor of test responses with unknown values," USA pending patent application.
20. J. Rajski, J. Tyszer, and N. Zacharia, "Test data decompression for multiple scan designs with boundary scan", *IEEE Trans. on Comput.*, vol. 47, pp. 1188-1200, 1998.
21. N. Tendolkar et al, "Novel techniques for achieving high at-speed transition fault test coverage for Motorola's microprocessor based on PowerPC instruction set architecture", *VLSI Test Symp.*, pp. 3-8, 2002.