Efficient Compression and Application of Deterministic Patterns in a Logic BIST Architecture

Peter Wohl
Synopsys Inc.
Williston VT 05495
wohl@synopsys.com

John A. Waicukauski
Synopsys Inc.
Tualatin OR 97062
johnwaic@synopsys.com

Sanjay Patel
Synopsys Inc.
Beaverton OR 97006
sanjayp@synopsys.com

Minesh B. Amin
Synopsys Inc.
Mountain View CA 94043
mamin@synopsys.com

ABSTRACT

We present a novel method to efficiently generate, compress and apply test patterns in a logic BIST architecture. Patterns are generated by a modified automatic test pattern generator (ATPG) and are encoded as linear feedback shift register (LFSR) initial values (seeds); one or more patterns can be encoded into a single LFSR seed. During test application, seeds are loaded into the LFSR with no cycle overhead. The method presented achieves reductions of at least 100x in test data and 10x in tester cycles compared to deterministic ATPG while maintaining complete fault coverage, as confirmed by experimental results on industrial designs.

Categories and Subject Descriptors: B.8.1 **[Performance and Reliability]:** Reliability, Testing and Fault-Tolerance.

General Terms: Algorithms, Design.

Keywords: self-test (BIST), test-generation (ATPG).

1. INTRODUCTION

Testing digital circuits represents a significant portion of the design, manufacture and service cost. Scan [1], [2] and logic BIST [3], [4] are the primary design-for-test (DFT) methods to control test cost. Logic BIST (hereafter referred to as BIST) is commonly implemented as originally described in [5]: values from a pseudo-random patterns generator (PRPG) are loaded into the internal scan chains of the design to be tested, and the chain outputs are unloaded into a signature analyzer that performs test response compaction. After a predetermined number of cycles, the state of the signature analyzer is compared to the known "signature" of the fault-free design. All inputs and outputs of the tested design are bounded by scan cells. The PRPG is commonly implemented as a linear-feedback shift register (LFSR) and the signature analyzer is often implemented as a multiple-input signature register (MISR) [3], [6].

To achieve a test coverage close to that achievable by deterministic ATPG patterns, the number of BIST patterns must be significantly greater and, in some cases, unreasonably so which forces a trade-off between increased test application time and reduced test coverage. Several solutions have been devised to address this problem. Using a multiplicity of parallel scan chains reduces the number of cycles per scan load, although not enough to compensate for the increased number of patterns [4], [5]. Adding test points to the design can improve fault detection by pseudo-random patterns, reducing the total number of patterns, but silicon area and propagation delay are increased. Further, pseudo-random patterns can be biased or modified so that they test random-resistant faults at the cost of adding silicon area, BIST data and BIST synthesis time [7], [8]. Deterministic ATPG patterns are often added to BIST patterns for a complete test coverage, but the total data volume stored on the tester is much greater than just the BIST data [4]. Finally, significant attention has recently been given to initializing (seeding) the PRPG so that desired scan cells are set to values that achieve target faults detection [9], [10], [11], [12], [13], [14], [15]. This solution can improve test coverage, but can result in increased pattern count and test application time.

DAC 2003, June 2-6, 2003, Anaheim, California, USA.

2. PROPOSED DBIST SOLUTION

We present an efficient solution that combines the same high fault coverage as deterministic ATPG (with full access to scan cells) with a logic BIST architecture. Compared to deterministic ATPG with highest dynamic compaction of test patterns, our method achieves significant reductions in both test data and tester cycles. Test patterns are generated by a modified deterministic ATPG, compressed into LFSR seeds and applied in a modified BIST architecture; hence we hereafter refer to our method as "deterministic BIST", or DBIST. To facilitate comparison with full-scan-cell-access deterministic ATPG, we present DBIST as a test stored on and driven by a tester; nonetheless, DBIST can also be implemented as a stand-alone selftest. The proposed solution consists of an apparatus that enables reseeding the PRPG LFSR with no cycle overhead (section 3), and a method to compress one or more deterministic ATPG patterns into a single LFSR seed (sections 4). Section 5 compares deterministic and DBIST patterns and section 6 presents experimental results on industrial designs. Section 7 concludes this paper.

All patterns can be controlled by LFSR seeds so that all "care bits" (scan cells that must be set to a certain value) are set to the desired value, while all other scan cells are set to pseudo-random values from the LFSR. The LFSR could also be used to apply "random" patterns (with no care bits determined); however, when DBIST patterns are applied from a tester, it is important to minimize the total number of patterns so that the total number of tester cycles is minimized. In this environment, it is preferable not to use random patterns because the total number of patterns would increase.

3. RESEEDING WITH 0-CYCLE OVERHEAD USING SHADOWS

Figure 1 shows the basic DBIST architecture. As is common in

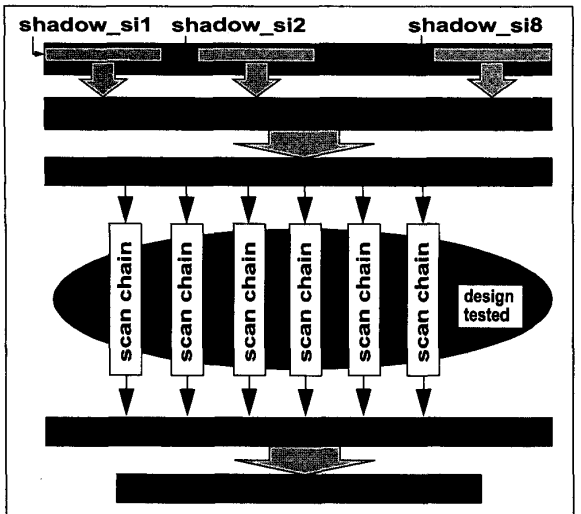

Figure 1. DBIST Architecture.

logic BIST, all primary inputs and outputs of the tested design are wrapped by scan cells. All scan cells of the design are linked in parallel, with internal scan chains placed between the pattern generator and the signature analyzer. Values from the PRPG-LFSR are loaded into the scan chains of the design to be tested, and the scan chain outputs are unloaded into a signature analyzer that performs test-response compaction. To reduce test application time, we use a large number of short parallel scan chains. A combinational phase shifter converts the uni-dimensional stream of pseudo-random values generated by the LFSR into a two-dimensional array of values to load parallel scan chains [6], [16]. The values unloaded from scan chains are fed into a MISR that must be at least as wide as there are scan chains [6]. But, a wide MISR increases BIST overhead by requiring a large area and storage of wide signatures. Therefore, we use a smaller MISR and a combinational [space-] compactor between the scan-chain outputs and the signature-analyzer inputs [17].

A PRPG shadow register, of the same length as the PRPG LFSR, has been added to the classical BIST architecture [5], [6]. The PRPG shadow is loaded through several parallel scan chains so that it can be fully loaded in the number of cycles it takes to load the internal chains. After a new PRPG seed has been loaded into the PRPG shadow, it is transferred in a single cycle to the PRPG LFSR. The architecture in Figure 1 achieves reseeding of the PRPG LFSR with 0-cycle overhead; the PRPG is continuously clocked and the scan chains are loaded every cycle with a new value. During the functional capture clock, the values loaded in the PRPG shadow can be transferred into the PRPG LFSR, thus allowing any pattern to start with a new seed if required. To use the same seed for multiple patterns, the transfer of the PRPG shadow into the PRPG LFSR is simply delayed until the prior seed has been used for all desired patterns.

Figure 2 details the PRPG shadow and the transfer mechanism. The PRPG shadow register is the same length as the PRPG LFSR; it is loaded through several parallel scan chains. In BIST mode, the scan chains of the design are configured into many, short internal chains, but no shorter than the shadow chains, so that the shadow can be completely reloaded during a single load; therefore, no additional test cycles are added to reload seeds. The PRPG shadow content is transferred to the PRPG LFSR during the capture cycle, when ‹ `shadow_transfer` ›=1, so no cycle is added even for the transfer of the new seed to the PRPG LFSR. The PRPG shadow registers are only loaded through scan; they need no unload.

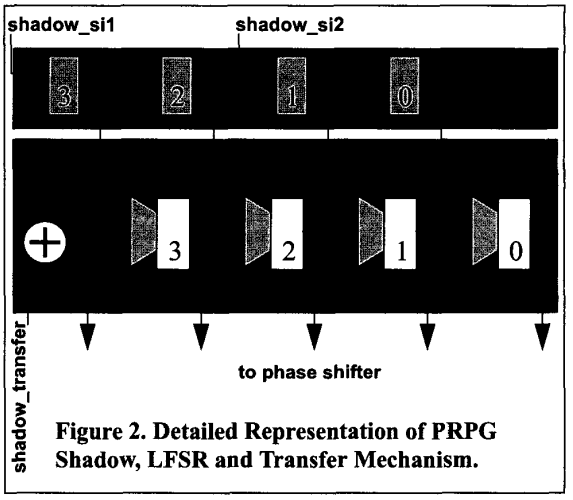

Figure 2. Detailed Representation of PRPG Shadow, LFSR and Transfer Mechanism.

The architecture in Figure 1 also supports a special scan mode to enable testing of the DBIST logic itself. In this mode, the PRPG shadow is configured as a single scan chain which is both loaded and unloaded. The PRPG LFSR, the MISR LFSR and the DBIST controller state elements (not shown in Figure 1) are also configured as scan chains. The internal scan chains of the design are bypassed with multiplexors. In this mode, all DBIST logic (including the aforementioned scan chains and the phase shifter, compactor and DBIST controller) can be fully tested with a handful of scan ATPG patterns.

4. MULTIPLE PATTERNS PER SEED

Testing a single fault typically requires only a small number (around 20) of scan cells to be set at desired values; these values are referred to as "care bits" and, collectively, as a "test cube". ATPG exploits the sparsity of care bits by generating patterns that each test as many target faults as possible, a process termed "dynamic compaction." Even so, in the final test cube, only a small percentage of the total number of scan cells are set; attempts to find further faults that can be tested by the same pattern are likely to fail because of conflicting values in some of the specified bit positions. For deterministic patterns the remaining bits are filled in randomly with the hope of fortuitously detecting additional faults. The fully specified pattern is then fault-simulated and all detected faults are taken off the active fault list. In DBIST, the test cube is encoded as an LFSR seed computed so that the desired bits are set when the

LFSR is loading the scan chains; don't-care bits are filled in randomly from the LFSR. At the beginning of an ATPG run, a few test patterns can be created to each set thousands of care bits and test hundreds of target faults. To supply thousands of care bits from an LFSR seed would require an LFSR with at least as many bits, certainly an impractical approach. Another approach continuously streams "seed" bits into a smaller LFSR [15], but during the fill-up cycles too few care bits can be controlled, while later too many bits are available to control the remaining few care bits. We chose to use an LFSR of reasonable size (250 - 500 bits) and limit DBIST ATPG to a smaller number of care bits per test pattern, about 19 bits less than the LFSR size [9].

For large designs that require thousands of highly compacted deterministic patterns to fully test, the number of care bits per pattern decreases very rapidly. After a few hundred patterns, test cubes are only about one hundred bits, and the last hundreds of patterns have test cubes of around 20-40 bits each. Several small test cubes can be encoded into a single seed [10]. After loading a seed, the LFSR is clocked during each internal chain shift and provides all needed care bits for the loads of several consecutive patterns.

To generate DBIST test patterns and encode them as seeds, test generation, encoding into seeds, and fault simulation are alternated to obtain an optimal set of seed-encoded patterns. Unlike previous methods [10], [11], a variable-sized set of patterns is first created by ATPG, and then encoded into a single seed; the set encoded into a single seed may contain any number of patterns and need not be augmented with dummy cubes or random patterns. If the system of equations to satisfy all care bits in a pattern set has no solution (no seed exists), the care bits associated with the last targeted fault are removed and the resulting reduced system of equations is solved. We found that, in 98% of cases, a seed could be found to satisfy all care bits; for the remaining cases, a seed was found after removing a small number of care bits.

The test-pattern generator [18] was modified to generate and add test cubes to each pattern until the total number of scan cells set for all patterns reaches a user-selectable limit, **total_cells**, or the number of patterns reaches its limit, **pats_per_set**; **pats_per_set** is also user-selectable, but is set to a high enough value (at least 20) to not limit the generation of an optimally encoded DBIST pattern set. The test generator targets as many faults as possible in a single pattern but, at the same time, limits the number of scan cells set to a selectable limit, **cells_per_pattern**. Unlike previous methods [11], DBIST ATPG uses the same powerful fault dynamic compaction algorithm as deterministic ATPG and need not be biased toward patterns with the least number of specified bits.

Controlling the set of patterns encoded into a single seed by the two selectable limits, **cells_per_pattern** and **total_cells**, minimizes total data volume and number of patterns at the same time, without the added complexity of changing the LFSR polynomial [10]. Computing LFSR seeds for multiple patterns results in full utilization of each seed, particularly towards the tail end of the pattern set where each compacted pattern has only few care bits. Minimizing the total number of seeds directly reduces total test data because MISR signatures, the other component of test data, represent only a very small part. Minimizing test data is most important for self-test implementation of DBIST if all data is to be stored on-chip.

5. COMPARING DBIST WITH DETERMINISTIC PATTERNS

DBIST encodes load data into LFSR seeds and unload data into MISR signatures; most data is taken by LFSR seeds. The data reduction vs. deterministic ATPG can be computed as:

$$\frac{Datadet}{Databist} \approx \frac{Patdet}{Patbist}\frac{PatsperSeed}{PRPGlen}(3Cells). \qquad (1)$$

assuming that 2 bits are required to store unload data and only 1 bit is required to store load values; *Patdet* and *Patbist* are the number of deterministic ATPG and, respectively, DBIST patterns; *Cells* is the number of scan cells; *PatsperSeed* is the average number of DBIST patterns that use the same seed, typically 1.5 - 3 (section 6); *PRPGlen*, the number of bits of the PRPG LFSR, is from 250-500. For very large designs (last rows in Table 1) larger or additional PRPGs and MISRs can be added to support as many internal scan chains as desired.

The number of scan chains in deterministic ATPG is limited by available pins. By contrast, DBIST can use a large number of shorter, internal chains, achieving a tester cycles reduction:

$$\frac{Cdet}{Cbist} \approx \frac{Patdet}{Patbist}\frac{Clendet}{Clenbist}. \qquad (2)$$

Clendet and *Clenbist* are the length of scan chains when deterministic ATPG and, respectively, DBIST patterns are applied. The number of scan chains in DBIST mode can be larger than the number of scan chains in deterministic ATPG mode, on the average, by a factor of 20 - 30, because DBIST chains do not require tester pins. Consequently, *Clenbist* is, on the average, 20 - 30 times smaller than *Clendet*.

6. RESULTS

This section provides experimental confirmation of the assumptions made when comparing DBIST with deterministic patterns (section 5). The most critical assumption is that $\frac{Patdet}{Patbist} \geq \frac{1}{2}$; this ratio directly affects the reduction of both data (equation (1)) and cycles (equation (2)). The data reduction further depends on *PatsperSeed*, the average number of DBIST patterns that use the same seed; the cycle reduction depends on the length *(Clendet)* of scan chains available in deterministic ATPG mode. Results measured on several industrial designs are shown in Table 1. The total design sizes are estimated as number of equivalent NAND-gates. The area overhead of DBIST logic is 1-2%; larger or multiple PRPGs and MISRs support more internal chains, thereby reducing chain length and total cycle count, but increasing the area overhead. The number of scan cells and the total number of faults are also shown as measures of design size.

For all designs, the fault coverage obtained from deterministic ATPG and from DBIST patterns were within 0.2% (differences were due to different fault accounting), ranging from 98% to 99+%. The number of seeds for DBIST patterns increases with design size, but the ratio *PatsperSeed* remains within a small range (1.5 to

Table 1. DBIST results.

Design size						ATPG results			DBIST effectiveness		
#gates (1000s)	#scan cells (1000s)	#faults (1000s)	#chains (det)	#chains (bist)	*PRPGlen* (total)	#seeds	*Patdet*/ *Patbist*	CPUdet/ CPUbist	*Patsper Seed*	$\frac{Datadet}{Databist}$	$\frac{Cdet}{Cbist}$
150	6.4	216	16	512	257	458	0.5	**2.6**	1.4	**55**	**11.8**
600	5.1	1,330	20	512	257	2,010	0.7	**1.2**	1.6	**71**	**16.9**
600	13.9	1,055	22	512	257	1,490	0.7	**1.6**	1.4	**164**	**16.3**
800	17.6	1,554	24	512	257	1,764	0.5	**2.9**	1.5	**151**	**11.1**
3,500	120.0	7,596	32	512	479	5,579	0.9	**3.2**	1.7	**756**	**15.0**
17,500	508.5	24,620	196	4592	4311	8,925	0.8	**2.9**	3.2	**879**	**18.2**

3). More importantly, $\frac{Patdet}{Patbist} \geq \frac{1}{2}$ as expected. Interestingly, the total CPU time for DBIST is less than the CPU time for deterministic patterns even though DBIST includes additional operations (such as seed encoding) and generates more patterns; this is due to the fact that deterministic ATPG tries much harder to merge faults into a pattern whereas DBIST ATPG is limited by **cells_per_pattern**, which limits the fault-merging effort. As expected, the DBIST data reduction $\frac{Datadet}{Databist}$ increases with the number of scan cells; the DBIST cycles reduction $\frac{Cdet}{Cbist}$ is about constant and larger than 10 (Table 1).

7. CONCLUSIONS

We presented DBIST, a deterministic BIST method that combines test-generation, LFSR-seed encoding and fault simulation to achieve the same high fault coverage as deterministic ATPG while applying patterns in a logic BIST architecture. The number of patterns encoded into a single LFSR seed varies continuously to accommodate the most efficient encoding. LFSR seeds control all care bits in all patterns. We also presented a mechanism that allows seeds to be loaded and the LFSR reseeded with no cycle overhead. The resulting patterns are fully compatible with scan testers. DBIST can be applied to all fault models that deterministic test generation supports. DBIST can also be implemented as self-contained BIST, storing the seeds on-chip. Both test data volume and tester cycles are significantly reduced over highly compacted deterministic patterns while obtaining the same test coverage. The method presented is fully integrated into an automated flow that performs all design modifications, rules checking and DBIST pattern generation.

ACKNOWLEDGMENTS

We are very grateful to T.W. Williams, G. Maston, M. Funcell, S. Kaushik and B. Lloyd for their contributions to the development of this work.

REFERENCES

[1] M. Abramovici, M.A. Breuer, A.D. Friedman, *Digital Systems Testing and Testable Design*, IEEE Press, 1990.

[2] E.B. Eichelberger, E. Lindbloom, J.A. Waicukauski, T.W. Williams, *Structured Logic Testing*, Prentice-Hall, 1991.

[3] V.D. Agrawal, C.R. Kime, K.K. Saluja, "A Tutorial on Built-In Self-Test, Part 1: Principles", *IEEE Design & Test* 1993, Vol. 10, No.1, pp. 73-82.

[4] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, J. Rajski, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies", *International Test Conference* 1999, pp.358-367.

[5] P.H. Bardell, W.H. McAnney, "Self-Testing of Multichip Logic Modules", *International Test Conference* 1982, pp.200-204.

[6] P.H. Bardell, W.H. McAnney, J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques*, John Wiley & Sons, 1987.

[7] H.-J. Wunderlich, G. Kiefer, "Bit-Flipping BIST", *International Conference on Computer-Aided Design*, 1996.

[8] A. Irion, G. Kiefer, H. Vranken, H.-J. Wunderlich, "Circuit Partitioning for Efficient Logic BIST Synthesis", *Design and Test Europe*, 2001.

[9] B. Könemann, "LFSR-Coded Test Patterns for Scan Designs", *European Test Conference*, Munich, 1991.

[10] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, B. Courtois, "Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers", *IEEE Transactions on Computers*, Vol. 44, No. 2, Feb. 1995.

[11] S. Hellebrand, B. Reeb, S. Tarnick, H.-J. Wunderlich, "Pattern Generation for a Deterministic BIST Scheme", *International Conference on Computer-Aided Design*, 1995.

[12] S. Chiusano, S. DiCarlo, P. Prinetto, H.-J. Wunderlich, "On Applying the Set Covering Model to Reseeding", *Design and Test Europe*, 2001.

[13] N.C. Lai, S.J. Wang, "A Reseeding Technique for LFSR-Based BIST Applications", *Asian Test Symposium* 2002, pp. 200-205.

[14] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, D. Wheater, "A SmartBIST Variant with Guaranteed Encoding", *Asian Test Symposium* 2001, pp. 325-330.

[15] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K.H. Tsai, A. Hertwig, N. Tamarapalli. G. Mrugalski, G. Eide, J. Qian, "Embedded Deterministic Test for Low Cost Manufacturing Test", *International Test Conference* 2002, pp. 301-310.

[16] J. Rajski, N. Tamarapalli, J. Tyszer, "Automated Synthesis of Phase Shifters for Built-In Self-Test Applications", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 2000, Vol. 19 No. 10, pp. 1175-1188.

[17] P. Wohl, J.A. Waicukauski, T.W. Williams, "Design of Compactors for Signature-Analyzers in Built-In Self Test", *International Test Conference* 2001, pp.54-63.

[18] "TetraMAX ATPG", *http://www.synopsys.com/products/test/tetramax_ds.html*