

A SmartBIST Variant with Guaranteed Encoding

by

Bernd Koenemann, Carl Barnhart, Brion Keller, Tom Snethen, Owen Farnsworth, and Donald Wheeler
IBM Microelectronics
San Jose, Endicott, Burlington
contact: berndk@us.ibm.com

Abstract

SmartBIST is a name for a family of streaming scan test pattern decoders that are suitable for on-chip integration. The Automatic Test Pattern Generation (ATPG) algorithms are modified to generate scan test stimulus vectors in a highly compacted source format that is compatible with the SmartBIST decoder hardware. The compacted stimulus vectors are streamed from Automatic Test Equipment (ATE) to the decoder, which expands the data stream in real-time into fully expanded scan test vectors. SmartBIST encoding and decoding use simple algebraic techniques similar to those used for LFSR-Coding (also known as LFSR-Reseeding). The specific SmartBIST implementation shown in this paper guarantees that all test cubes can be successfully encoded by the modified ATPG algorithm irrespective of the number and position of the care bits.

Introduction

The work described in this paper exploits two fundamental properties of automatically generated scan test patterns for the purpose of test data compression:

- The sparseness of care bits in test cubes, and
- The freedom to assign arbitrary values to don't care bits.

A practical algebraic data compression method called LFSR-Coding or LFSR-Reseeding using these two properties was introduced in [1] and [2]. LFSR-Coding translates care bit values in fault-oriented test cubes generated by Automatic Test Pattern Generation (ATPG) tools into seed values for an LFSR-based (or other linear) Pseudo-Random Pattern Generator (PRPG). The seeds, thus, become encoded representations of the care bit values in the test cubes. To apply the encoded test cubes, a new seed is downloaded from Automatic Test Equipment (ATE) into the PRPG at the beginning of each test and then the PRPG is cycled until all on-product scan cells have been filled with new test stimuli from the PRPG. The intelligently calculated seed values ensure that cycling the PRPG loads the correct test cube values into the care bit positions while all other scan

cells are filled with pseudo-random values. The fill values are generated as a by-product of the PRPG sequence and need not be stored explicitly in the ATE.

It was shown in [1] that the number of bits required to store the seeds is only slightly higher than the number of care bits in the test cubes. Since only a small portion of the test vectors consists of care bits, the total storage for the encoded seed values is much less than the storage for conventional, fully expanded stored pattern test stimuli.

The original LFSR-Coding method, thus, achieves the objective of a very significant reduction in the binary data that must be stored in ATE buffer memory. The method, however, suffers from two drawbacks.

1. The size of the LFSR determines the number of bits in a single PRPG seed and, thus, limits how many care bit values can be encoded into a single test.
2. The ATE sits idle and waits while the PRPG is cycled to completely fill the on-product scan channels with the derived expanded test vector.

The key innovation of our SmartBIST concept is to allow the ATE to continuously "stream" additional encoded care bit information to the decoder while it fills the on-product scan channels.

Reference [1] further showed that LFSR-Coding suffers from a phenomenon called test pattern lockout. Lockout is mathematically similar to the better-known phenomenon of aliasing in linear signature registers. Both phenomena are caused by algebraic dependencies: between error-bit positions in aliasing and between care bit positions in lockout. Lockout is caused by the fact that the PRPG fills each scan cell with a predictable XOR-combination of some of the seed bit values. It is possible that the XOR-combinations of several scan cells are dependent in that the XOR-sum of the XOR-combinations is a constant. That means, that only vectors with even or odd parity can be loaded from the PRPG into the dependent scan cell group. If a test cube requires a vector of opposite parity in the affected scan cells, then LFSR-Coding cannot successfully encode the required care bit values for the test cube.

This paper introduces a specific SmartBIST decode variant that a) implements the streaming concept, and b) contains an innovative provision to deal with test pattern lockout without having to resort to fully expanded vectors.

The SmartBIST Concept

Background

SmartBIST is the second phase of a technology roadmap that combines the benefits of ATPG and Logic Built-In Self-Test (Logic BIST) techniques for the cost-effective testing of 100M+ gate chips [3]. The first phase called On-Product MISR, or OPMISR, has already been implemented in the Design For Testability (DFT) and ATPG tools for selective use on very large ASIC chips [4]. OPMISR integrates a Multiple-Input Signature Register (MISR) into the products under test. The OPMISR intercepts the test response data from the on-product scan chains and compresses the responses into signatures. This eliminates the need for storing explicit responses in the ATE buffer. It also frees up ATE bandwidth normally consumed for unloading the response data. The freed-up bandwidth is used to double the number of scan chains for scan-in and, thus, helps to improve logic test throughput [5].

The use of an OPMISR essentially eliminates most of the data volume and solves some of the logic test throughput issues related to the test response data. Additional data reduction requires us to focus on improvements to the test stimulus data representations and application techniques. Reference [6] describes a very promising method for exploiting the sparseness of care bits and the freedom to assign don't care bit values with OPMISR and existing ATE software/hardware features. The idea is to match the fill algorithm for the don't care bits used in ATPG with available hardware or software features of the target test equipment such that the fill data values can be algorithmically re-generated in real-time by the ATE. One particular example described in the reference paper utilizes repeat count capabilities that are available on most ATE. This "tester-smart" fill algorithm repeats the last care bit value rather than using pseudo-random values. Results indicate a more than 5-fold reduction in total stored test data volume by the combination of OPMISR and the repeat count approach for a real-life complex ASIC chip. That compares very favorably with data reported for Logic BIST with test points [7].

While the repeat-count method with OPMISR does solve the data volume problem for even very large chips, it does not create any additional savings in logic test time beyond the roughly 2x savings obtain-

able by doubling the scan chains. The test stimuli are expanded (on-the-fly) in the ATE and, thus, must be transported in fully expanded form from the ATE to the chip under test via the bandwidth-limited external scan interface. Breaking this bandwidth barrier requires that the decoding of compacted stimulus data is moved from the tester into the product under test.

SmartBIST Hardware Overview

SmartBIST uses a hardware-based on-product decoding scheme. Figure 1 shows a high-level diagram of the generic SmartBIST decoder hardware.

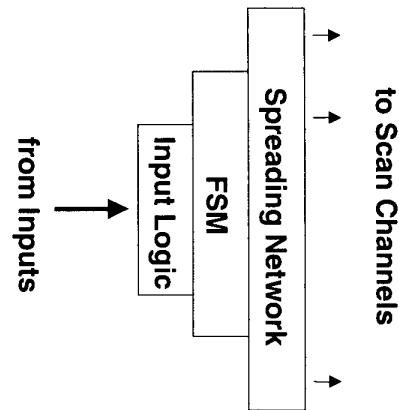


Figure 1: High-Level SmartBIST Decoder Architecture

The main blocks are an Input Logic stage, an optional Finite State Machine (FSM) stage, and a Spreading Network. Any register elements in the SmartBIST decoder are synchronized to the scan clocks used for the on-product scan channels (scan channels in our terminology are on-product scan strings without external scan-in/out connections).

The Input Logic block receives a synchronous (relative to the scan clock cycle) stream of m-bit wide encoded data words from the ATE and forwards the data words to the optional FSM or directly to the Spreading Network. The Input Logic generally does not transform the data, but rather may contain fan-out, re-powering, and possibly pipeline registers to simplify wiring and timing closure for extremely large designs. In addition, the Input Logic will have to implement pin-sharing functions (to allow for sharing test pins with functional pins) and contribute to scan multiplexing functions (to support other scan configurations and test modes as needed).

The main portion of the FSM stage typically will be LFSR-based. On very complex chips with difficult wiring constraints, the FSM stage may consist of several replicated FSM blocks each with its own

LFSR. The number of state bits in the FSM (e.g., the width of the LFSR) can be equal to or larger than the number of bits in the input data word, meaning that the output word from the FSM can be wider than the input word. The input data bits generally are combined with the FSM state bits by means of XOR gates (e.g., very much like a signature register is constructed by XOR-ing the input data into an LFSR). That is, the FSM state sequence is not only controlled by the initial seed value prior to scan, but can be continuously modified by the input data stream received from the ATE during scan.

The Spreading Network is a space expansion network that in the linear implementations consists of an XOR-network very similar to the networks used for "linear sums" in [8] and for Phase Shifting in LFSR-based Logic BIST PRPGs [9].

A Simple Decoder Example

The FSM in the SmartBIST decoder architecture is optional. For many applications it is expected that a purely combinational implementation consisting of the input logic and a spreading network is sufficient [e.g., 10]. In the extreme case, the spreading network can be reduced to a fan-out network as proposed in [11].

A simple combinational decoder example that expands 6-bit input words into 12-bit words for 12 parallel scan channels is shown in Figure 2. The combinational implementation has no memory, and the input data values (SI1,...,SI6) provided for each scan cycle only are available for that one respective scan cycle. Each input vector, thus, has to create all care bit values associated with the current cycle of the scan-load operation.

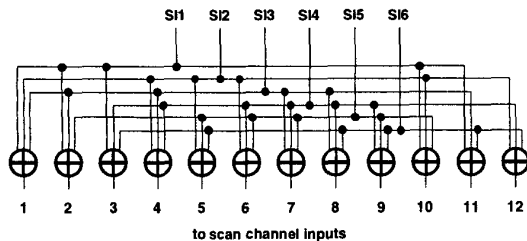


Figure 2: Combinational SmartBIST Decoder Example

The spreading network creates a different Boolean combination of input bit values for each scan channel input. The associated expressions are shown in Figure 3.

Scan Chain 1	=	SI1 + SI2 + SI3
Scan Chain 2	=	SI1 + SI3 + SI5
Scan Chain 3	=	SI1 + SI4 + SI6
Scan Chain 4	=	SI2 + SI3 + SI4
Scan Chain 5	=	SI2 + SI5 + SI6
Scan Chain 6	=	SI2 + SI4 + SI5
Scan Chain 7	=	SI3 + SI4 + SI5
Scan Chain 8	=	SI3 + SI4 + SI6
Scan Chain 9	=	SI4 + SI5 + SI6
Scan Chain 10	=	SI1 + SI2 + SI5
Scan Chain 11	=	SI1 + SI3 + SI6
Scan Chain 12	=	SI2 + SI4 + SI6

Figure 3: Boolean Expressions for the Scan Channel Inputs from Figure 2

Figure 4 depicts hypothetical care bit values for a test vector shifted into 12 scan channels that must be derived from the input data words received in 8 successive scan cycles.

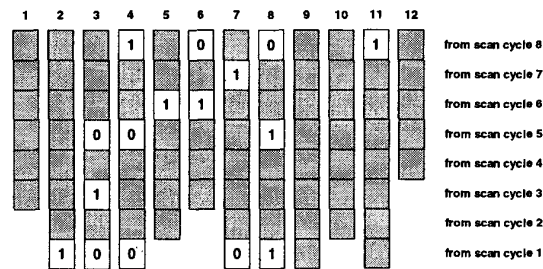


Figure 4: Test Vector Example (Note: gray cells without data are don't-care-bits)

The data from the first scan cycle are shifted down the furthest into the scan channels, while the data from the last scan cycle end up in the scan cells closest to the scan channel inputs. As described in the LFSR-Coding paper [1], each care bit value creates an equation for associated input values according to the expressions shown in Figure 2. The derived care bit equations for scan cycle 8 are as follows:

Scan Chain 4:	$SI2 + SI3 + SI4 = 1$
Scan Chain 6:	$SI2 + SI4 + SI5 = 0$
Scan Chain 8:	$SI3 + SI4 + SI6 = 0$
Scan Chain 11:	$SI1 + SI3 + SI6 = 1$

Figure 5: Care Bit Equations for Scan Cycle 8

Any combination of input values that simultaneously solves these four equations above will generate the correct care bit values in chains 4, 6, 8, and 11. The input vector {SI1, SI2, SI3, SI4, SI5, SI6}={011100} is one such solution, and the input vector {110010} would be another solution.

Scan cycle 7 only generates a single equation, because only one care bit value is needed in scan chain

7 from that scan cycle. The equation for scan chain 7 only involves the input variables SI3, SI4, and SI6 (see Table 1). The other three input variables do not contribute to the single care bit value and, hence, can be chosen freely. The input vector {SI1, SI2, SI3, SI4, SI5, SI6}={xx111x}, where x denotes an unspecified value, is one of several solutions for the single care bit equation associated with scan cycle 7.

The care bit values for the other scan cycles generate similar sets of equations that must be solved for each scan cycle. Figure 6 illustrates a particular set of solutions for all 8 scan cycles that results in proper values for all care bit positions.

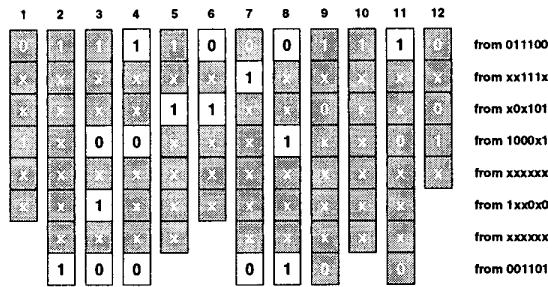


Figure 6: Scan Channels with Decoded Data (corresponding encoded data shown on the right)

Figure 6 shows that all care-bit values are covered properly in the decoded data even if some bit values in the encoded data are left unspecified. In other words, the encoding algorithm still has a degree of freedom in assigning values to the unspecified bits in the encoded data. In the example, only 27 out of 48 encoded data bits have specified values that generate the correct values for 16 care bits embedded within 87 bits of decoded data. Furthermore, the set of care-bit equations for a particular scan cycle may have multiple solutions with different specified values. It has been found advantageous in traditional ATPG algorithms to use pseudo-random values for the don't-care-bits. These pseudo-random values help to detect other non-targeted faults and non-modeled defects. It therefore is suggested that the encoding algorithm also should use pseudo-random values whenever there is a freedom to choose a value. This makes it more likely that the don't-care-bits in the decoded patterns are filled in a pseudo-random manner to increase the effectiveness of the tests.

Test Pattern Lockout

The original paper on LFSR-coding introduced the issue of test pattern lockout [1]. Test pattern lockout occurs if the Boolean expressions for a set of care bit values are interdependent such that it becomes impossible to create the all care bits. Scan channels

1,2,4, and 7 from Figure 2, for example, are interdependent as illustrated in Figure 7.

Scan Chain 1	=	SI1 + SI2 + SI3
Scan Chain 2	=	SI1 + SI3 + SI5
Scan Chain 4	=	SI2 + SI3 + SI4
Scan Chain 7	=	SI3 + SI4 + SI5
Total Sum		= \emptyset

Figure 7: Example of Dependent Expressions

The total XOR sum of the Boolean expressions for the four scan channels always is 0 irrespective of the actual input data values. That is, any decoded vector will always have even parity in these four bits. The decoder, hence, is unable to generate any care bit combination with odd parity in the four bit positions.

Reference [1] contains an approximate formula to estimate the likelihood of test pattern lockout as a function of the number of care bits and the number of input bits. The new decoder architecture described in the following includes features that allow for, if needed, overcoming the test pattern lockout problem.

Sequential SmartBIST Decoder with Guaranteed Encoding

One disadvantage of the simple combinational implementation is that the input data received from the tester are only useful for the current scan cycle. Scan cycle 4 in Figure 4, for example, has no care bits. Thus, the data supplied from the tester for that scan cycle cannot contribute to satisfying any care bit demand in subsequent cycles. An optional FSM inserted between the Input Logic and the Spreading Network can "remember" data received in previous cycles such that the data can be used for resolving care bit equations in subsequent cycles.

For reasons of simplicity, it is desired to preserve the simple linear nature of the care bit equations. The FSM, thus, should be linear in nature as well. One particular FSM with that property is an LFSR connected to the external scan inputs by XOR gates. The resulting structure, in essence, looks very much like a Single-Input Signature Register (SISR) or a Multiple-Input Signature Register (MISR). The SISR/MISR inputs are used to continuously modify the LFSR sequence as needed for achieving the correct care bit values in the scan channels. Other researchers ([12], [13]) have suggested similar structures.

The proposed new SmartBIST decoder variant illustrated in Figure 8 combines a SISR/MISR-like FSM with clock gating provisions similar to those introduced in [14].

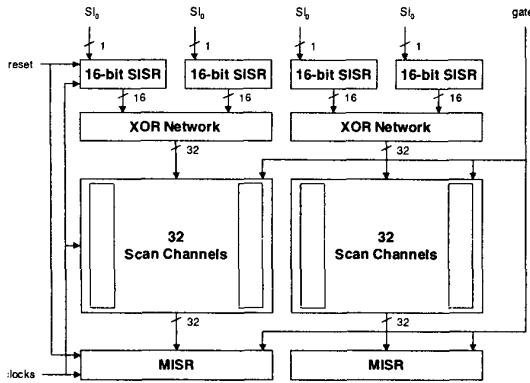


Figure 8: Proposed New Decoder Architecture

The proposed design is very modular in that it can be built up in slices of some bit width, which should simplify the physical design of the structure for very large circuits with 500 or more scan channels. The example uses 16-bit SISR slices combined with 32-bit spreading networks and MISR slices plus a single clock gate signal for each 64-bit slice. The numbers are meant to be illustrative and may be different in actual implementations. The 64-bit slice shown expands 5 input bit values (4 data plus 1 gate) into 64 scan channel bit values.

A global reset signal is used to reset the all SISR and MISR states at the beginning of each test. That helps to make the tests independent and simplifies diagnostic data collection [e.g., 5].

The optional spreading networks are designed such that the XOR-expressions for all 32 scan channel inputs are independent (that is possible because there are 32 input values from the SISRs). The global gating signal selectively de-gates the scan clocks for all scan channels and MISRs. When the de-gating condition is asserted, the scan channel and MISR states are frozen, while the SISR clocks are still running. Temporarily freezing the scan chains makes it possible to advance the SISRs to any arbitrary state as needed to overcome potential test pattern lock-out, and then continue with the channel load. For the example in Figure 8 it takes at most 16 clock cycles to completely update all SISR states to any arbitrary value (in some cases, the SISR may already be partially in the correct state such that less than 16 cycles could be sufficient to arrive at the desired state).

There are 4 new input bits from the SISR inputs available to each 64-bit slice for each scan cycle, while 64 bits are loaded into the scan channels. In other words, there is a 16:1 ratio between the data scanned into the channels and the independent variables provided at the SISR inputs. The total number

of independent variables provided in a full channel load, thus, should be roughly sufficient for a care bit density of up to $1/16=6.25\%$ in the expanded data. Besides this average number, the success of encoding also depends on how many independent variables are "consumed" by care bits in each individual scan cycle, how the "demand" for independent variables fluctuates over the scan cycles, and how the "supply" matches up with the fluctuating demand.

Let us assume a worst-case scenario wherein a particular scan cycle requires 32 care bit values in one of the 32-bit slices shown in Figure 8. That means all 32 bit values stored in the two 16-bit SISRs associated with the 32-bit channel slice must be at a specific value to satisfy the 32 care bits. In the subsequent scan cycle the tester provides 2 new bits, but there is no more freedom to re-adjust the previous SISR state to satisfy a larger number of care bits in this subsequent scan cycle. In other words, the momentary supply of unused independent variables for the upcoming scan cycle is very low. Without the gate signal it could be impossible to satisfy the care bit requirements for this scan cycle. With the gate signal, the channel load can be interrupted temporarily, and the SISRs can be updated with additional new independent variables as needed for the subsequent scan cycles.

The empirical data shown in [14] suggest that the situation described in the previous paragraph should be quite unlikely for realistic large chip designs. We, therefore, anticipate that the channel load operations need to be interrupted only very rarely and, if so, only for a few scan cycles per load. Thus, the proposed new architecture allows for a graceful adaptation for tests with a mismatch between care bits and independent variables. The adjustments can be made "locally" at individual cycles in the scan operation rather than having to fall back on a full load with uncompressed data.

Future Work

SmartBIST is intended for use with very large and complex designs. We, therefore, have no plans to apply the techniques to small test cases. The encoding algorithm is more or less identical to the one used for LFSR-coding techniques. Thus, on average, the encoding results are similar to results reported for previous LFSR-Coding techniques.

Comprehensive experimentation with SmartBIST will follow after the current work on the OPMIS implementation and qualification has been completed. The local density of care bits depends not only on the design under test and the ATPG algorithm, but also on the logical and physical scan chain

assignments. Realistic experiments, thus, have to incorporate realistic scan chain optimization techniques. We intend to report experimental results once the scan chain optimization tools in the logical and physical design system have been upgraded for SmartBIST.

Overall, a full, practical implementation of SmartBIST for ASICs requires automation support for

- The construction and integration of the SmartBIST hardware components,
- The balancing and optimization of a very large number of scan channels,
- Design rules checking and verification for SmartBIST,
- Integration of the encoding algorithms into ATPG,
- Updating the Test Data Supply System (TDS) to optimize ATE programming and buffer allocation, and
- Updating the diagnostic data collection procedures to deal with On-Product signatures.

Thus far, the last item has been addressed with the release of the OPMISR support. The remaining items will be addressed after the qualification of OPMISR is complete.

Summary

A new SmartBIST test pattern decoder architecture has been introduced. The architecture is very modular in nature helping to facilitate logical and physical design. The proposed structure uses SISR-like elements as linear Finite State Machines. Clock gating is used to temporarily interrupt the scan channel load and to update the SISRs to any arbitrary state if needed. The clock gating feature, thus, guarantees that any arbitrary care bit combinations can be successfully encoded into the derived test vectors.

References

[1] B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs", ETC '91, pp.237-242, 1991

[2] U. Diebold et al., European Patent, EP EP00481097B1, 1995

[3] B. Koenemann et al., "Logic DFT and Test Resource Partitioning for 100M Gate Asverbal ICs", presentation at the Test Resource Partitioning Workshop (TRP), 2000

[4] <http://www.chips.ibm.com/products/asics/products/cu-11.html>

[5] B. Koenemann et al., "OPMISR: Accelerated Scan with On-Product Signatures", submitted to TECS '01 and ETW '01

[6] C. Barnhart et al., "OPMISR: The Foundation for Compressed ATPG Vectors", submitted to ITC '01

[7] G. Hetherington et al., "Logic BIST for Large Industrial Designs: Real Issues and Case Studies", Proc. International Test Conference, pp. 358-367, 1999

[8] S. B. Akers, "On the Use of Linear Sums in Exhaustive Testing", Proc. FTCS, pp. 148-153, 1985

[9] P.H. Bardell et al., "Built-In Test for VLSI: Pseudorandom Techniques", John Wiley, New York, 1987

[10] Bayraktaroglu and A.Orailoglu, "Test Volume and Application Time Reduction Through Scan Chain Concealment" submitted to DAC '01

[11] Ilker Hamzaoglu and Janak H. Patel, "Reducing Test Application Time for Full Scan Embedded Cores", Proceedings FTCS, pp. 260-267, 1999

[12] D. Kay and S. Mourad, "Controllable LFSR for BIST", Proceedings of the 17th IEEE Instrumentation and Measurement Technology Conference, IMTC 2000., Page(s): 223 -228 vol.1, 2000

[13] N. Touba, private communication

[14] O. Farnsworth et al., "Extending Self-Testing Clock-Gates for LFSR-Based Test Pattern Encoding", submitted to ITC '01