A Tutorial on Built-In Self-Test

Part 2: Applications

Vishwani D. Agrawal
AT&T Bell Laboratories

Charles R. Kime
Kewal K. Saluja
University of Wisconsin, Madison

Concluding an overview of built-in self-test (BIST) concepts and practices, Part 2 covers BIST hardware structures, applications, and tools. The authors describe testing approaches for general and structured logic. They illustrate BIST techniques with real-world examples and present hardware structures and software tools supporting BIST design.

Having introduced the BIST pattern generation and response analysis concepts in Part 1, we now focus on their implementation. First we will describe the testing of general logic; then we will briefly discuss testing approaches for structured logic such as ROMs, RAMs, and PLAs. We refer the reader back to Table 2 in Part 1,which contains typical pattern generator and response analyzer structures. Here, we concentrate almost entirely on the LFSR and MISR structures. For these structures, scan is an integral part of the design. Note also that ROMs and counters can be useful as supplements or alternatives to a number of the structures described.

BIST structures for general logic

First of all, most BIST techniques for general logic involve a fundamental trade-off between time and hardware. We can characterize this trade-off most easily by classifying BIST techniques into two categories: *test-per-clock* and *test per-scan.* In test per-clock BIST, we apply a test vector and capture a response each clock period. In test-per-scan BIST, we use scan capability to apply a test vector and capture a response each scan cycle. A scan cycle is the number of clock cycles required to shift the vector into a serial scan path or to shift the response out of the serial scan path (whichever is larger) plus one or more normal mode clocks. For example, if a chip containing 200 edge-triggered flip-flops has a single, full scan path, test-per-scan requires 201 clock periods to apply a test vector and simultaneously observe the response to the previous vector—roughly 200 times slower than the test-per-clock approach. The two approaches involve distinct hardware structures and trade-offs.

Figure la (next page) shows a simple test-per-clock configuration using an LFSR as a pattern generator and an MISR as a response compactor. Figure lb shows a test-per-clock configuration using an LFSR, an MISR, and a shift register. The first configuration is suitable for exhaustive or pseudorandom testing and the second for pseudoexhaustive or pseudorandom testing. Notice that the pattern generator LFSR costs little more than a serial scan register because we can ordinarily implement it with only two additional XOR gates.[1] But using the MISR on the circuit's outputs requires XOR structures on every out-put stage, plus the LFSR hardware.

Originally published in *IEEE Design and Test of Computers*, June 1993, pp. 69–77.

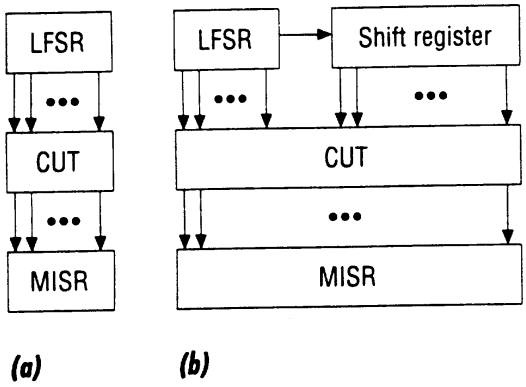


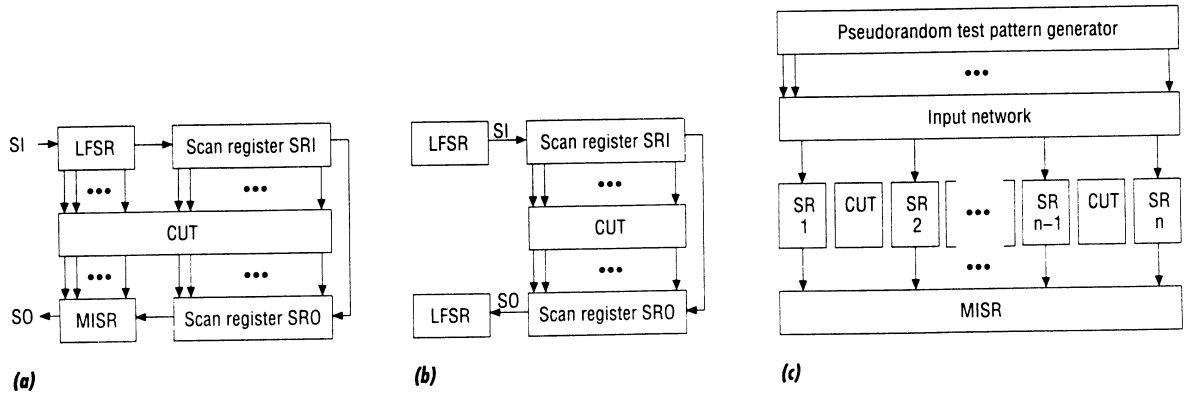
Figure 1. Test-per-clock configurations.

In contrast, when we examine the test-per-scan-configuration in Figure 2a, we find that the LFSR and the scan register SRI on the circuit's inputs are identical to those in Figure lb. But instead of the MISR logic on the outputs, a scan register SRO drives a shortened MISR on only a portion of the outputs. This configuration saves the MISR hardware on the remaining outputs, but it must shift all the outputs captured in SRO into the output MISR between each normal clock. Thus, it is much slower in applying tests. For a circuit that uses a single serial scan path, one can use the configuration in Figure 2b, with the pattern generator LFSR and the response LFSR separated totally from the circuit under test (CUT).

STUMPS[2] (self-test using an MISR and parallel shift register sequence generator) is a test-per-scan approach to pseudorandom testing of circuits with multiple serial scan paths. In this approach an LFSR used as a pseudorandom test pattern generator (PRTPG) feeds the inputs directly to a set of serial scan paths, as in Figure 2c, with the input network consisting of simple connections from the LFSR to the scan path. The outputs from the serial scan paths feed MISR inputs. Suppose that the LFSR is of the external-XOR type (described in "LFSR theory" section of Part 1) and that the serial scan paths are fed directly from adjacent bits of the LFSR. Then, value in bit*i* of a serial scan path will be identical to the value of bit*i* – l of the serial scan path to its right.

Thus, these bits are 100% correlated, and if they both feed the same combinational network, the patterns seen by the network are certainly not pseudorandom and may yield reduced fault cover age. In general, a BIST designer must be very mindful of such correlation and design to avoid it. In the case of STUMPS, the input network in Figure 2c is actually a phase shift network constructed of XOR trees specifically designed to avoid correlation between inputs to the serial scan chains and hence to the logic under test.[1] The BIST hardware overhead for STUMPS beyond full scan consists of the PRTPG, the input network, and the MISR. This approach can simultaneously test all the synchronous internal logic in a chip or set of chips, except embedded structures such as RAMs and ROMs.

Obviously, in a given circuit, combinational logic typically not only drives but is driven by storage elements. Thus, BIST structures that can generate or deliver test patterns to driven logic as well as compact outputs from driving logic are useful. This capability is quite natural for test-per-scan using the STUMPS configuration but is more difficult for test-per-clock. One of the earliest structures designed for test-per-clock is the built-in logic block observer (BILBO).[2, p. 304] The original BILBO provided normal operation, reset, serial scan, and MISR functions. By using the feedback associated with the MISR, the BILBO can also provide pattern generation as a variation of serial scan.

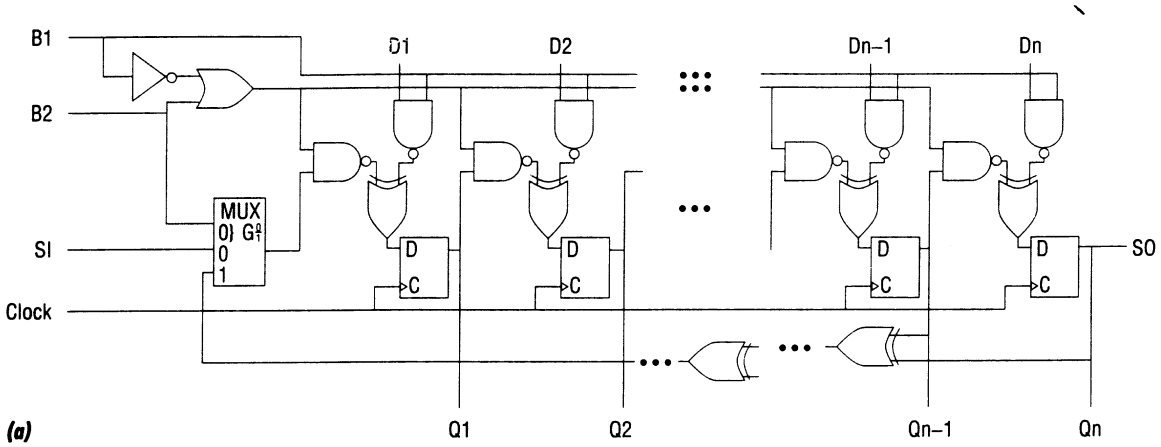


Originally published in *IEEE Design and Test of Computers*, June 1993, pp. 69–77.

Figure 2. Test-per-scan configurations.

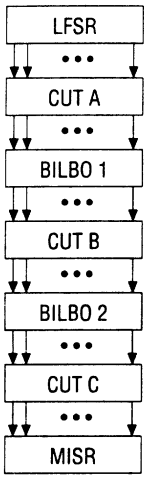
Figure 3 shows a modified version of the BILBO structure, its modes of operation, and an application to a circuit with a pipeline structure. In Figure 3c, CUTs A and C can be tested simultaneously, and CUT B individually, with the modes specified in Figure 3d. The BILBO modes are defined on the premise that the outputs of an MISR are not adequate to serve as pseudorandom test inputs.

A final test-per-clock approach, called *circular self-test* or *circular BIST*, eliminates the linear feedback circuits for both pattern generation and response compaction.[3 p. 496, 4] The circular approach converts each flip-flop to an MISR stage, and the feedback is the natural nonlinear feedback provided by the CUT itself. Figure 4 shows a full circular BIST structure. Here, we initialize the BIST path by scan, allow the circuit to operate for some number of clock cycles, and scan out all or part of the signature left in the path and compare it to the correct signature. Problems can arise with this structure because the MISR's outputs, with the circuit as feedback, serve as test patterns. Thus, fault simulation and possibly redesign of the path or the use of multiple testing cycles are necessary to ensure adequate coverage



B1	B2	Mode
0	0	Serial scan
0	1	LFSR
1	0	Normal
1	1	MISR

(b)



CUT	Mode BILBO 1	Mode BILBO 2
A,C	MISR	LFSR
B	LFSR	MISR

(d)

Figure 3. A modified BILBO configuration: hardware (a), operating modes (b), application structure (c), and application modes (d).

Originally published in *IEEE Design and Test of Computers*, June 1993, pp. 69–77.

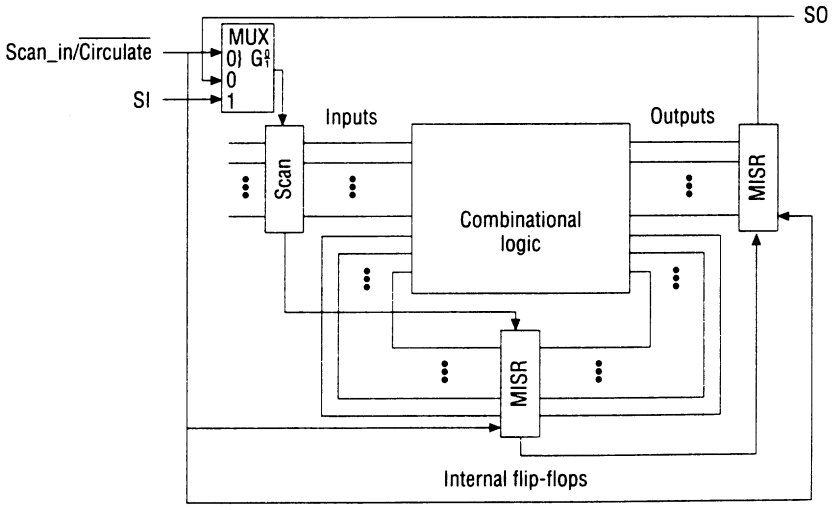


Figure 4. General full-circular BIST configuration.

Thus far, all the methods we have considered assume that BIST or serial scan structures have made the CUT combinational. But that is not essential; BIST can be successful on sequential circuits as well. Usually, however, MISR or scan capabilities must be provided to at least a portion of the internal storage elements. This approach, which we call partially sequential BIST, can be applied to the circular BIST configuration.[4] Partially sequential BIST can also employ techniques that change the sequential circuit, for testing purposes, either into one that has no cycles between storage elements or into one that roughly resembles a pipeline.[5]

The fundamental BIST structures and methods given so far may not detect all faults, so we must consider how to deal with the remaining hard-to-detect faults. If we simply use deterministic patterns to detect such faults, we will need a serial scan path to scan in the patterns and scan out the responses, a ROM to store the patterns. and associated test manager hardware. If we attack hard-to-detect faults with weighted pseudorandom tests, we will need a weighted pseudorandom generator.[6, 7] Another possibility is the use of cellular automata.[8]

BIST for structured logic

The basic structured logic block types to which BIST is applied separately from general logic blocks are PLAs, ROMs, and RAMs. PLAs are almost always embedded in a larger circuit, whereas ROMs and RAMs may be embedded or stand-alone structures for application of BIST. In either case, the BIST method for embedded structures uses extra logic to isolate these blocks from the remaining logic. That is, BIST treats these blocks as stand-alone structures. Thus, if the structures are embedded, BIST integration can take place at the architectural and chip organization levels.

Figure 5 diagrams a generic BIST PLA structure. The figure shows a number of extra logic blocks around a PLA. Not all of these blocks are necessary to self-test a PLA. For example, if the PLA has only a small number of inputs (less than about 25), the best strategy may be exhaustive testing. In that case, all we need is a pattern generator at the input, a response analyzer such as an MISR at the output, and a BIST control block to isolate the PLA from the rest of the logic. The control block initiates the self-test and at its completion evaluates the compacted results to determine the status of the PLA.

A host of other methods, using either deterministic or random vectors for maximum fault coverage with minimum silicon area overhead due to extra logic, have been proposed. The structure shown in Figure 5 is generic, and all BIST architectures for PLAs fit into it. The most prevalent fault model used for PLAs is cross-point faults, as opposed to conventional stuck-at faults. Research has shown that tests that detect cross-point faults also detect most other classical and nonclassical faults in PLAs.[9] We refer the reader to two works that describe almost all these methods and compare their performance impact (delay), silicon area overhead, and fault coverage.[3, 10] Our general assessment is that until some novel technique appears, BIST will be practical only for small- to medium-size PLAs that can be tested exhaustively or for cases where multiple-fault detection

Originally published in *IEEE Design and Test of Computers*, June 1993, pp. 69–77.

is so important that the silicon area overhead is a secondary consideration.

BIST glossary

Aliasing: condition in which a faulty circuit with erroneous response produces the same signature as a good circuit

Automatic test pattern generator (ATPG): a system (typically a computer program) that generates a sequence of test patterns to test faults in a circuit

Boundary scan: a method of providing serial scan access to all the inputs and outputs of a device

Built-in logic block observer (BILBO): an LFSR-based BIST circuit with four modes of operation: normal storage, reset, serial scan, and multiple-input signature analysis

Cellular automaton: a circuit consisting of an array of sequential cells having a repetitive local interconnection structure (the most common form is a one-dimensional cell array consisting of a single bit of storage dependent on its own value and the value stored in its nearest neighbor)

Circular BIST: a technique in which pattern generation and response compaction are carried out simultaneously by the same set of flip-flops, which form a circular chain

Compaction: reduction of a response sequence into a much shorter sequence with possible loss of information (that is, the response sequence may not be recoverable from the shorter sequence)

Compression: reduction of a response sequence into a much shorter sequence without loss of information (that is, the response sequence can be recovered from the shorter sequence)

Design for testability (DFT): any process applied to a circuit design that facilitates testing the circuit

Exhaustive testing: a testing technique that applies all possible input combinations to the circuit under test

Fault coverage: ratio, expressed as a fraction or percentage, of all faults detected by a test sequence to the modeled faults in the circuit under test

In-circuit testing (ICT): a method that uses direct access to the chip pins to test chips or interconnections on a board

Level-sensitive scan design (LSSD): a variant of the serial scan design concept defined by IBM

Linear feedback ship register (LFSR): a circuit made up of Flip-Flops and XOR gates interconnected in certain configurations (typically used for BIST pattern generation and response analysis)

Multiple-input signature register (MISR): an LFSR-based BIST circuit that simultaneously compacts multiple response sequences for response analysis

Partial scan: a DFT technique in which only a subset of all circuit flip-flops is scannable

Pattern: an ordered set of binary values that is applied simultaneously to the inputs of a circuit or that appears simultaneously on the outputs of a circuit

Programmable logic array (PLA): usually refers to a structured implementation of a digital function based on a two-level AND/OR description of the combinational part of the function

Pseudoexhaustive testing: a testing method in which the subcircuits in a covering set for the circuit under test are tested exhaustively

Pseudorandom patterns: sequences of patterns having properties similar to sequences of random patterns

Pseudorandom testing: a testing method that applies pseudorandom patterns to the circuit under test

Pseudorandom test pattern generator (PRTPG): a BIST circuit, usually based on an LFSR or a cellular automaton, that produces a pseudorandom sequence of patterns

Reject ratio: percentage (or fraction) of faulty devices among all devices passing a set of tests

Scan design: a DFT technique in which all storage elements, except those in storage arrays, can be controlled and observed without using the functional logic (the most common form is serial scan, in which storage elements are electronically reconnected to form one or more ship registers for testing)

Signature: contents of a set of storage elements containing the compacted or compressed responses of a circuit under test

Stuck fault a fault model in which a line in the circuit remains at a constant logic value irrespective of the signal value at the line

Surface mount technology (SMT): a method of assembly in which electronic components are mounted on both sides of a printed circuit board

Vector: another term for pattern

Originally published in *IEEE Design and Test of Computers*, June 1993, pp. 69–77.

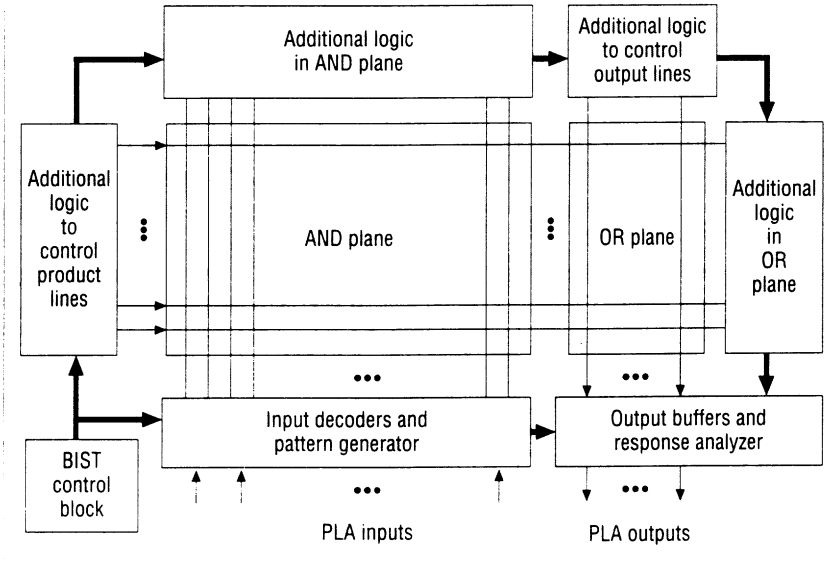


Figure 5. A generic BIST architecture of PLAs.

ROM, embedded or stand-alone, is relatively easy to test. The basic method is to read the ROM contents and compact the outputs by means of an MISR. Both the pattern generator and the response analyzer are simple structures. Aliasing is a problem only if the probability of multiple bit errors in the ROM outputs is high. Typical multiple bit errors are caused by the failure of an output buffer or a decoder circuit. Such faults either are not masked by the MISR or can be detected relatively easily by a few extra deterministic test vectors. Also, faults in the control part of the ROM are often catastrophic and hence are detected by a test that simply reads the ROM contents.

In deriving tests for RAMs, engineers have used different fault models for different parts of a RAM. For example, the fault model for decoders includes not only stuck-at faults but also arbitrary address-mapping faults. For the memory array, on the other hand, the fault model assumes stuck-at, coupling, and pattern-sensitive faults.[11] RAMs require such diverse and complex fault models because of their very high density.

Both deterministic tests[12] and random tests[2, p. 242] have been used for embedded RAMs. For the application of BIST to embedded RAMs, the pattern generator contains an address generator and a data generator. The address generator consists of an LFSR or a counter, and the data generator is often an LFSR or a finite-state machine that produces the data to be written into the RAM. During reading from the RAM, the address generator and the data generator can generate the same sequence of addresses and data again, but responses are usually compacted to protect against failure of the generators. The present trend is to use deterministic test patterns for embedded RAMs because fault coverage for the deterministic method can be computed exactly for possible impacts on reliability.

Although neither deterministic nor random testing is a high-overhead method, the sharing of test logic between multiple RAM blocks on the chip potentially reduces overhead even further. However, such sharing raises a concern about interconnection overhead. To keep the interconnection area low, we can integrate internal scan or boundary scan with the BIST logic.[13]

Thus far, most practical applications of BIST to embedded RAMs have used only algorithms that test for stuck-at faults. Stand-alone RAMs have even higher integration density and therefore must be tested for more complex fault types. The test sequence length and hence the time necessary to test a RAM are proportional to the number of bits in the RAM. With the continual increase in RAM size and the increasing cost of tester-based testing, self-test of large RAM chips has become imperative.

Originally published in *IEEE Design and Test of Computers*, June 1993, pp. 69–77.

Boundary scan

Boundary scan provides a method of testing board interconnections as well as isolating the interior of chips for BIST. Thus, boundary scan supports the integration of BIST into the hierarchical structure proposed in this article. A number of manufacturers have developed their own boundary-scan and test access methods for chips. For designers planning to use vendor parts on their boards, the approach used by those parts is given in IEEE Std 1149.1-1990. The standard defines a four-to-five-signal test access port, requirements for boundary scan, an instruction register and a standard sub-set of instructions, a bypass register, and an optional device identification register. It also provides for optional, user-defined, scannable registers. These registers plus the instruction register constitute a convenient means of BIST implementation and control. Support for boundary scan in hardware and CAD tools is growing rapidly.

Bibliography

Bleeker, H., P. van den Eijnden, and F. de Jong, *Boundary-Scan Test: A Practical Approach*, Kluwer Academic, Boston, 1993.

IEEE Std 1149.1-1990 *Test Access Port and Boundary Scan Architecture (ANSI/IEEE, IEEE)*, Pisataway, N.J., 1990.

Journal of Electronic Testing: Theory and Applications, Special Issue on Boundary Scan, Vol. 2, No. 1, 1991.

Maunder, C.M. and R.E. Tulloss, eds., *The Test Access Port and Boundary-Scan Architecture*, IEEE Computer Society Press, Los Alamitos, Calif., 1990.

Parker, K., *The Boundary-Scan Handbook*, Kluwer Academic, Boston, 1992.

Among the earliest BIST RAM architectures were two proposed by Kinoshita and Saluja.[14] These architectures use transition count as the compaction function; data are generated in such a way that no aliasing can take place even though the compactor consists of a 1-bit transition counter. One of the proposed test architectures is based on random logic, the other on microcode. The overhead for these architectures is negligibly small (less than 1%) for very large RAMs (4 Mbits or more). Both architectures have been used in practice. A list describing the special features of BIST RAM designs and implementations appears in an article by Franklin and Saluja.[15] All these RAMs use deterministic test patterns.

Example BIST applications

Manufacturers are increasingly employing BIST in real products. Here we offer three examples of such applications, selected to illustrate the use of BIST in the semiconductor, communications, and computer industries.

Exhaustive test in the Intel 80386.[16] In the 80386, BIST logic exhaustively tests three control PLAs and the control ROM. All have MISRs on their outputs. Exhaustive stimuli are provided for the PLAs by LFSRs embedded in the input storage elements and for the ROM by the microprogram counter that is part of the normal logic. The largest structure is a PLA with 19 inputs, so the test length is 512K clock cycles. The information gathered by the MISRs is continuously shifted out to additional LFSRs tied to the data path. An internal comparison at the end of the test yields an output of zero in an observable register if it detects no error.

Circular BIST in AT&T ASICs.[4] AT&T has employed a partial sequential approach using circular BIST in seven application-specific integrated circuits. Four of the devices contain embedded RAM, and for all but one device, the goal was complete self-test except for I/O buffers and portions of the multiplexer logic on the inputs. The implementation uses a cell similar to that shown in Figure 3 but with the logical functions of the original BILBO. The signature read-out occurs only from an LFSR or MISR embedded in the circular chain. In addition, BIST is provided for the embedded RAMs. For the six devices with full BIST, the gate count, including BIST logic, averages 14,150, the logic overhead averages 20%, and the active area overhead averages 13%. The average fault coverage for the portion of the circuit covered by BIST for four fault-simulated circuits is 92%. Much of the hardware overhead is in the BIST cells, and a trade off is apparent between the number of storage cells to which BIST is applied and the fault coverage. The company has automated the BIST design for standard cell VLSI implementation or PLD-based circuit packs.

Pseudorandom test in the IBM RISC/6000.[17] The RISC/6000 uses an extensive BIST structure that covers the entire system. Its BIST techniques include full serial scan and pseudorandom pattern use in the form of STUMPS. In addition, the system uses embedded RAM self-test and performs delay testing.

The RISC/6000's BIST system is hierarchically structured, with a common on-chip processor (COP) on each chip. The COP, which occupies less than 3%, of

Originally published in *IEEE Design and Test of Computers*, June 1993, pp. 69–77.

the chip area, contains an LFSR for pattern generation, an MISR for response compaction, and a counter for addressing during RAM testing. In addition, the COP contains a control finite-state machine, attached by a few dozen control lines to the normal on-chip and BIST logic. Each chip also has boundary scan.

For delay tests, two normal clock phases are applied in sequence at normal operating speed between scan-in and scan-out operations. For RAM tests, RAM data inputs and outputs are attached directly to scan paths by means of DFT techniques, and RAM address and read/write controls are attached to the COP.

BIST tools

Most designers of BIST circuits use general-purpose tools such as logic and fault simulators, but some specialized tools have also been developed. Table 1 presents a list of commercially available CAD tools suitable for BIST circuit design. These are largely analysis tools, and some have their origin in the DFT environment. Such tools can be useful because DFT is often a precursor to BIST. Next we describe some hardware structures and software design systems developed specifically for BIST implementations.

BIST support of VISI design. Mucha, Daehn, and Gross[18] have designed building blocks to facilitate BIST. Their catalog of three CMOS chips supports printed circuit board design. Two of the chips implement BILBO structures, and the third contains the necessary hardware for testing memories. They have also designed a BIST multiplier chip and a BIST arithmetic logic chip. The authors give the area overhead and performance degradation due to BIST in their article.

Table 1. CAD tools for BIST/DFT.*

Vendor/Tool	Capabilities
CrossCheck AIDA Testability Tools	TRDC: test rules design checker; Scangen: automatic scan insertion(single/multiple chain), combinational test generator; Fltsim: fault simulator; boundary-scan support available
Siemens Cerberus, Socrates	Cerberus: design rule checker, automatic scan register insertion, overhead optimization; Socrates: combinational test generator
Racal-Redac Intelligen 2	Full or partial scan, testability analyzers, sequential circuit test generator, CADAT fault simulator, boundary-scan support available
Synopsys Test Compiler	Redundancy removal from automatically synthesized circuit, automatic scan insertion and test generation
AT&T Titus, Testpilot	Alert: design rule checker, combinational test generator, fault simulators, scan overhead optimizer for standard-cell design; Gentest/Pascant: partial scan and sequential circuit test generator; CKT: automatic circular BIST; PEST: pseudoexhaustive self-test; Maclog: BISTmacrocell generator
Cadence TestScan	Design rule audit (random and serial scan, manual scan register implementation, combinational test generator, fault simulator
Philips Panther Tool Kit	DFT rule checker, testability hardware insertion, automatic full-scan design and scan chain routing, test control block generator, combinational test pattern generator, boundary-scan insertion
Sunrise TestGen/Test Syn	Partial scan and sequential circuit test generator

**Due to rapid growth in this field, vendor offerings may have changed and new products may be available.*

Originally published in *IEEE Design and Test of Computers*, June 1993, pp. 69–77.

Many VLSI chips contain memory blocks. The conventional methods of testing these blocks require special access to the I/O pins. Such access requires extra routing area. BIST design of embedded memory blocks keeps the routing overhead contained and is widely used in ASICs and custom chips. [12, 19] A variety of tests can be generated from the BIST circuitry—for example, tests for cell-stuck faults and data retention faults. The overhead is about 5% for 16-Kbit static RAM and decreases for larger memories. The Cathedral-II silicon compiler system also uses a BIST memory approach.[20]

General Electric's A/VLSI design svstem[21] partitions a chip into macrocells and supports the implementation with a standard cell library. BIST implementation starts from the chip definition, which includes a plan for implementing the test features. The system follows this plan in selecting appropriate macrocells for BIST.

AT&T's macrocell generation tool Maclog provides automated design of ROMs, register files, and content-addressable memories with BIST features.[22] Among other tools in use at AT&T, PEST[23] supports the design of pseudoexhaustive BIST, and CKT[24] inserts hardware necessary for circular BIST in random-logic circuits.

BIST support of synthesis. Incorporating DFT in automatic synthesis is a current trend. The Silc silicon compiler,[25] developed at GTE and supported by Racal-Redac, performs rule-based synthesis. The following is a partial list of the rules Silc uses:

1. Use PLA-type control logic with clocked feedbacks only.
2. All internal clocks must be controllable from primary inputs.
3. All flip-flops should be master-slave D-type.
4. All reset lines should be controllable from primary inputs.
5. Allow a test pin.

The Silc system uses a SCOAP-like (Sandia Controllability Observability Analysis Program) testability measure to identify testing problems.[2, p. 49] Such analysis identifies circuit areas where signals are difficult to control or observe. The system then uses an expert-system approach to implement testability. Although the final design is heavily biased toward BIST, Silc uses other techniques when necessary. It can implement BIST PLAs and modify other structures to form pattern generators and signature analyzers. While adding the BIST hardware, the system considers design parameters such as required fault coverage, acceptable overhead, cost, test time, and the capability of available CAD tools.

TDES (Testable Design Expert System),[26] a CAD system developed at the University of Southern California, contains information about various testable design methodologies (TDMs) in its knowledge base. The designer sets the design goals and constraints on area overhead, speed degradation, fault coverage, I/O pins for test, test time, and the use of external test equipment. TDES then analyzes the block-level design consisting of PLAs, registers, buses, RAMs, and ROMs. Recognizing these basic structures, it modifies the circuit to incorporate the appropriate TDMs. An interesting feature of this system is that it presents alternatives and allows the user to make choices.

Somewhat similar features are implemented in Bides (BIST Design Expert System),[27] developed at the University of Virginia, and Tiger (Testability Insertion Guidance Expert System),[28] at Microelectronics and Computer Technology Corporation.

Our discussion of BIST structures and their applications in real-world designs demonstrates the applicability of BIST in today's electronic systems. BIST offers the unique opportunity to use hierarchy in the testing of complex systems. It also provides an economical answer to the test problems of future technologies in which physical access to parts embedded in very large systems may be restricted. Although most commercially available CAD tools have advanced only to the level of supporting DFT within captive tool environments, BIST tools have appeared. As these tools proliferate, the essential components for broader adoption of hierarchical BIST will be in place.

We have presented the principles of pattern generation and response analysis and described the structures for their implementation. In addition, we have outlined real BIST applications and indicated the state of development of CAD tools for BIST. With this foundation established, we encourage readers to further explore the world of BIST to meet present and future needs in the testing of their products.

Acknowledgment

The National Science Foundation, Division of Microelectronic Information Processing Systems, under grants MIP-9003292 and MIP-9111886, partially supported this work.

Originally published in *IEEE Design and Test of Computers*, June 1993, pp. 69–77.

References

1. P.H. Bardell, "Design Considerations for Parallel Pseudorandom Pattern Generators," *J. Electronic Testing: Theory and Applications,* Vol. 1, Feb.1990, pp. 73–87.
2. P.H. Bardell, W.H. McAnney, and J. Savir, *Built-In Test for VLSl :Pseudorandom Techniques,* John Wiley & Sons, New York, 1987.
3. M. Abramovici, M. Breuer, and A. Friedman, *Digital Systems Testing and Testable Design,* Computer Science Press, New York, 1991.
4. C.E Stroud, "Automated BIST for Sequential Logic Synthesis," *IEEE Design & Test of* Computers, Vol. 5, No. 6, Dec. 1988, pp. 22–32.
5. H.-J. Wunderlich, "The Design of Random-Testable Sequential Circuits," *Proc. Int'l Symp. Fault-Tolerant Computing,* IEEE Computer Society Press, Los Alamitos, Calif., 1989, pp. 110–117.
6. J. Waicukauski et al., "A Method for Generating Weighted Random Test Patterns," *IBM J. Research and Development,* Vol. 33, No. 2, May 1989, pp. 149–161.
7. H.-J. Wunderlich, "Self-Test Using Unequiprobable Random Pattems," *Proc. Int'l Symp. Fault-Tolerant Computing,* IEEE Computer Society Press, Los Alamitos, Calif., 1987, pp. 258–263.
8. P.D. Hortensius, R.D. McLeod, and B.W. Podaima, "Cellular Automata Circuits for Built-In Self-Test," *IBM J. Research and Development,* Vol. 34, No. 2/3, Mar./May 1990, pp. 389–405.
9. M.M. Ligthart and RJ. Stans, "A Fault Model for PLAs," *IEEE Trans. Computer Aided Design,* Vol. 10, No. 2, Feb. 1991, pp. 265–270.
10. C.Y. Liu and K.K. Salula. "Built-in Self-Test Techniques for Programmable Logic Arrays," in *VLSl Fault Modeling and Testing Techniques,* G.W. Zobrist, ed., Ablex Pub lishing, Norwood, NJ.,1993.
11. A.J. van de Goor and C.A. Verruijt, "An Overview of Deterministic Functional RAM Chip Testing," *ACM Computing Surveys,* Vol. 22, Mar. 1990, pp. 5–33.
12. S. Jain and C. Stroud, "Built-In Self-Testing of Embedded Memories," *IEEE Design & Test of Computers,* Vol. 3, No. 5, Oct. 1986, pp. 52–63.
13. B. Nadeau-Dostie, A. Silburt, and V.K. Agarwal, "Serial Interfacing for Embedded Memory Testing," *IEEE Design & Test of Computers,* Vol. 7, No. 2, Apr. 1990, pp. 52–63.
14. K. Kinoshita and K.K. Salula, "Built-In Testing of Memory Using an On-Chip Compact Testing Scheme," *IEEE Trans. Computers,* Vol. C-35, No. 10, Oct. 1986, pp. 862–870
15. M.Franklin and K.K.Saluja, "Built-In Self-Test of Random-Access Memories," *Computer,* Vol. 23, No. 10, Oct. 1990, pp. 45–56.
16. P. Gelsinger, "Design and Test of the 80386," *IEEE Design & Test of Computers,* Vol. 4, No. 3, June 1987, pp.42–50.
17. I.M. Ratiu and H.B. Bakoglu, "Pseudorandom Built-In Self-Test Methodology and Implementation for the IBM RISC System/ 6000 Processor," *IBM J. Research and Development,* Vol. 34, Jan. 1990, pp. 78–84.
18. J.P. Mucha, W. Daehn, and J. Gross, "Self-Test in a Standard Cell Environment," *IEEE Design & Test of Computers,* Vol. 3, No. 6, Dec. 1986, pp. 35–41.
19. R. Dekker, F. Beenker, and L. Thijssen, "Realistic Built-In Self-Test for Static RAMs," *IEEE Design & Test of Computers,* Vol. 6, No.1, Feb. 1989, pp. 26-34.
20. F. Catthoor, et al., "A Testability Strategy for Multiprocessor Architecture," *IEEE Design & Test of Computers,* Vol. 6, No. 2, Apr. 1989, pp. 18–34.
21. R.C. Kroeger, "Testability Emphasis in the General Electric A/VLSI Program," *IEEE Design & Test of Computers,* Vol. 1, No. 2, May 1984, pp. 61–65.
22. Y. Zorian, "A Structured Approach to Macrocell Testing Using Built-In Self-Test," *Proc. IEEE Custom Integrated Circuits Conf,* 1990, pp. 28.3.1–28.3.4.
23. E. Wu, "PEST: A Tool for Implementing Pseudo-Exhaustive Self-Test," *AT&T Technical J.,* Vol. 70, No. l, Jan./Feb.1991, pp. 87–100.
24. M.M. Pradhan et al., "Circular BIST with Partial Scan," *Proc Int'l Test Conf,* IEEE Computer Society Press, Los Alamitos, Calif., 1988, pp. 719–729.
25. H.S. Fung and S. Hirschhom, "An Automatic DFT System for the Silc Silicon Compiler," *IEEE Design & Test of Computers,* Vol. 3, No. 1, Feb. 1986, pp. 45–57.
26. M.S. Abadir and M.A. Breuer, "A Knowledge-Based System for Designing Testable VLSI Chips," *IEEE Design & Test of Computers,* Vol. 2, No. 4, Aug. 1985, pp. 56–68.
27. K. Kim, J.G. Tront, and D.S. Ha, "BIDES: A BIST Design Expert System," *J. Electronic Testing: Theory and Applications,* Vol. 2, No.2, June 1991, pp. 165–179.
28. M.S. Abadir, "TIGER: Testability Insertion Guidance Expert System," *Proc. Int'l Conf Computer-Aided Design,* IEEE Computer Society Press, Los Alamitos, Calif., 1989, pp. 562-565.

Vishwani D. Agrawal is a distinguished member of the technical staff at AT&T Bell Laboratories and a visiting professor of electrical and computer engineering at Rutgers University. His interests include VLSI testing and neural network algorithms. He has co-authored three books. A former editor-in-chief of *IEEE Design & Test of Computers,* Agrawal presently serves as editor-in-chief of *The Journal of Electronic Testing: Theory and Applications.* He received his PhD from the University of Illinois at Urbana-Champaign. He is a fellow of the IEEE and a member of the IEEE Computer Society, the ACM, and, the VLSI Society of India. He has served on the IEEE CS Board of Governors.

Charles R. Kime is a professor in the Department of Electrical and Computer Engineering at the University of Wisconsin-Madison, where he has developed and taught a broad range of computer engineering courses. His research interests include testing, design for testability, BIST, and fault-tolerant computing. He has served as general chair of the 1979 International Symposium on Fault-Tolerant Computing, as associate editor of *IEEE Transactions on Computers* and *IEEE Transactions on Computer-Aided Design,* and on the program committees of IEEE conferences.

Originally published in *IEEE Design and Test of Computers,* June 1993, pp. 69–77.

Kime is a fellow of the IEEE and a member of the IEEE Computer Society.

Kewal K. Sanka is a professor in the Department of Electrical and Computer Engineering at the University of Wisconsin Madison, where he teaches logic design, computer architecture, microprocessor-based systems, and VLSI design and testing. Previously, he worked at the University of Newcastle, Australia. He has also held visiting and consulting positions at the University of Southern California, the University of Iowa, and Hiroshima University. His research interests include design for testability, fault-tolerant computing, VLSI design, and computer architecture. He is an associate editor of the *Joumal of Electronic Testing: Theory and Applications.* Sanka received the BE from the University of Roorkee, India, and the MS and the PhD in electrical and computer engineering from the University of Iowa. He is a member of the IEEE Computer Society.

Send correspondence about this article to Vishwani D. Agrawal, AT&T Bell Laboratories, 600 Mountain Ave., Room 2C476, Murray Hill, NJ 07974; va@research.att.com.

Originally published in *IEEE Design and Test of Computers*, June 1993, pp. 69–77.