Built-In Self-Test for Digital Integrated Circuits

Vishwani D. Agrawal
Chih-Jen Lin
Paul W. Rutkowski
Shianling Wu
Yervant Zorian

The increasing complexity of digital integrated circuits (ICs) and the requirements of system quality and availability demand a new discipline in IC testing, known as built-in self-test (BIST). This paper introduces the basic approaches of this self-contained, reusable testing technique. It also presents different attributes of BIST schemes, their tradeoffs and available tools, and demonstrates a practical application of the schemes described.

Introduction

During its lifetime, an electronic system is tested and diagnosed repeatedly. To ensure that a system performs with the fewest possible interruptions, testing and diagnosis must be quick and effective. It makes sense, therefore, to specify test as one of the functions required within a system. Today, large electronic systems are controlled by software into which self-test is often incorporated. A purely software approach to testing may suffice at the system level, but it has several disadvantages. A truly effective software test may be long, slow to run, and expensive to develop. Such a test also has poor diagnostic capability, because it must exercise hardware that is not easily testable. For these reasons, BIST is an increasingly attractive alternative.

> "Built-In Self-Test (BIST) is a design for testability technique in which testing (test generation, test application and test response analysis) is accomplished through built-in hardware features."[1]

BIST, the most promising device-level test solution:

- Enhances product quality by virtue of its enhanced testability, diagnosability, reliability, and maintainability;
- Reduces test development effort at the integrated circuit (IC), board, and system levels;
- Keeps the complexity/cost of automatic test equipment low by putting testing intelligence into the product itself;
- Provides a uniform testing methodology that fosters standardization and test re-use at the IC, board, system, and field tests;
- Simplifies processes such as test generation, test application, output data evaluation, and fault isolation, at all levels;
- Tests products thoroughly (i.e., high fault coverage) and rigorously (i.e., at the operating speed of the device) at all levels of assembly;
- Alleviates accessibility problems associated with high-density packaging/interconnect technologies;[2] and
- Increases test and diagnostic throughput and accuracy.

Motivation for BIST

BIST was developed to meet the needs for high product quality, to reduce or contain increasing circuit complexity, and to avoid physical probing within electronic systems. The latter is difficult, if not becoming impossible, because of miniaturization.[3]

Quality. The quality of a product greatly depends on the ability of its test to identify a faulty product. Tests for random-logic circuits (consisting mainly of gates and flip-flops) are calibrated according to their ability to detect single lines shorted to ground (stuck at 0) or to power supply (stuck at 1). Because physical faults depend on the type of component (chip, board, etc.) and technology (complementary metal-oxide semiconductor [CMOS], bipolar, GaAs, etc.), it is difficult to determine an exact test of quality.[4] The real goal of quality is to reduce faulty parts in the tested products to a low *reject ratio* (e.g., 1 in 10,000 parts). Based on experience, measurable test requirements are used, such as *95-percent fault coverage*, for the random-logic portion of a very large-scale integrated (VLSI) circuit chip, or 100-percent fault coverage of all interconnect faults on a printed circuit board. These requirements can only be achieved through design for testability.

Complexity. Problems related to automation of synthesis and physical design of a complex device or system can be solved using hierarchical procedures. As the complexity of an electronic system increases, there is no simple way of deriving system tests from the given tests for its components. BIST represents a departure from the conventional test methodology, permitting *divide and conquer* and *hierarchical* test strategies.

Consider a chip embedded in a board that is part of a system. A top-down hierarchy consists of the system, boards, and chips. If all levels use BIST, the system can test the chip by sending a control signal to the board, which, in turn, will activate self-test on the chip and return the results to the system. BIST efficiently tests complex embedded components and their interconnections.

Reuse. The hierarchical test strategy just described also helps to reduce the development effort and hardware cost of test circuitries at chip, multi-chip module (MCM), board, and system levels. BIST offers a superior solution. Because tests are generated internally as they are being applied, rather than being pregenerated, stored, and applied externally, the test equipment and test program become simpler, and tests can be applied *at-speed.* In addition, the same BIST hardware is reused for testing chips, MCMs, boards, and the entire system during manufacture and operation.

Economics of Electronic Test

System designers must choose from several alternatives for testability.[5] Is BIST the right solution? To answer the question, costs must be weighed against benefits. Table I shows the impact of BIST on testing cost for chips, boards, and systems. Considering the cost of design and test development, the additional cost of designing BIST hardware somewhat balances the test-generation savings.

In AT&T, it is customary for engineers to incorporate BIST using automated software tools to reduce risk and cost, and to shorten the schedule. Even though the introduction of BIST hardware increases the fabrication costs at all levels, it minimizes the cost of production test and more than pays for itself over the life of the product. At board and system levels, BIST improves troubleshooting, reduces the cost of maintenance, improves diagnosis, and reduces repair costs. Other, less effective test strategies, with lengthy or improper diagnosis, such as the infamous "no-trouble-found," are often responsible for service interruption, and loss of revenue.

Panel 1. Abbreviations, Acronyms, and Terms

ASIC — application-specific integrated circuit
ATPG — automatic test pattern generator
ATTDFT — AT&T design for testability
AT&T-ME — AT&T-Microelectronics
BCAD — boundary-scan computer-assisted design
BIST — built-in self-test
BSDE — boundary-scan description editor
BSDL — boundary-scan description language
BSIT/Gen — boundary-scan insertion/generation tool
CAM — content-addressable memory
CKT — circuit know thyself
CMOS — complementary metal-oxide semiconductor
DFT — design for testability
DRAM — dynamic random-access memory
FIFO — first-in first-out
IC — integrated circuit
ICCAT — integrated-circuit-level computer-assisted test
LFSR — linear feedback shift register
MACLOG — a macrocell generator
MCM — multi-chip module
MHz — megahertz
ODC — output data compactor
PEST — pseudo-exhaustive self-test
PQFP — plastic-quad flat package
PSBIST — partial-scan-based built-in self-test
RAM — random-access memory
ROM — read-only memory
RSBGen — regular-structure BIST generator
SRAM — static random-access memory
TAP — test access port
TPG — test pattern generator
VLSI — very large-scale integration

BIST Principles

The basic BIST architecture adds three hardware blocks to a circuit: a test pattern generator (TPG), an output data compactor (ODC), and a test controller. The TPG block produces test stimuli for the circuit under test. To save on the hardware required to store the expected responses, the ODC circuit produces a *signature.* Ideally, a faulty signature should differ from the correct signature. Table II lists common BIST hardware structures.[1] Any combination of TPG and ODC shown can be used.

Table I. Impact of BIST on Testing Costs

Level	Design and test	Fabrication	Production test	Maintenance test	Diagnosis and repair	Service interruption
Chips	+/–	+	–			
Boards	+/–	+	–		–	
System	+/–	+	–	–	–	–

+ Cost increase – Reduction +/– Increase ≈ saving

A popular BIST structure, linear feedback shift register (LFSR), is used both as a TPG and an ODC. It sequences through almost all possible combinations of bits in the register according to an irregular ("pseudo-random") sequence.

A control block activates the test, analyzes the output data, and supports hierarchical test. Suppose a system consists of several circuit boards. A possible test architecture, shown in Figure 1, uses the technique of boundary scan,[6] discussed later in this issue of the *Journal.* The test controller at the system level simultaneously activates self-test on all boards. The test controller on a board, in turn, activates self-test on chips. An on-chip test controller executes self-test and transmits the results (fault-free or faulty) to the test controller of the board containing the chip. The board test controller accumulates test results from all its chips and transmits them to the system test controller, which then identifies faulty chips and boards. Reference 7 describes a novel distributed BIST control scheme. The effectiveness of the test procedure depends on the fault coverage of BIST. Other important issues include performance and area overhead, and its effect on chip yield.

BIST Schemes

Because faults that most commonly occur in random-logic circuits are different from those that occur in regular structures, such as random-access memory (RAM), read-only memory (ROM), first-in first-out (FIFO), etc., and because regular structures can often be tested by a set of easily generated regular patterns, these two classes of circuits require different BIST schemes to test their corresponding logic.

Random-Logic BIST. The three major types of random-logic BIST schemes are:

- Stored-pattern BIST,
- Pseudo-exhaustive BIST, and
- Pseudo-random BIST.

Table II. Hardware Structures for BIST

Test pattern generators	Output data compactors
ROM	ROM and comparison logic
Linear feedback shift register	Linear feedback shift register
	Multiple-input signature register
Cellular automata	Cellular automata
Binary counter	One's counter
	Programmable space compactor
XOR trees	XOR trees

Stored-pattern BIST. Stored-pattern BIST,[1] whose test patterns are typically generated by an automatic test pattern generator (ATPG) and stored in a ROM, uses programs or microprograms to perform functional tests on the hardware. The fault coverage of the stored patterns depends heavily on the quality of the predetermined patterns, and the applicability of this scheme is limited by its need for vast amounts of storage.

Pseudo-exhaustive BIST. Pseudo-exhaustive BIST applies all possible test patterns to each combinational logic block. Because each block is exhaustively tested, fault simulation is not needed to determine fault coverage. To test an n-input block of combinational logic, all possible 2^n input patterns are applied. However, even with high clock speed, the test's application time would make pseudo-exhaustive BIST impractical for a block with n greater than 25. Therefore, the logic is often partitioned, or segmented, into smaller, possibly overlapping, blocks with fewer than 25 inputs. One implementation of pseudo-exhaustive BIST[8] requires all circuit flip-flops to be reconfigured into an ODC. An additional TPG's register generates exhaustive patterns for application to all combinational logic blocks in the random logic, and the ODC compacts the output data into a signature. This scheme guarantees high fault coverage without the need for fault simulation. Unfortunately, because all flip-flops are

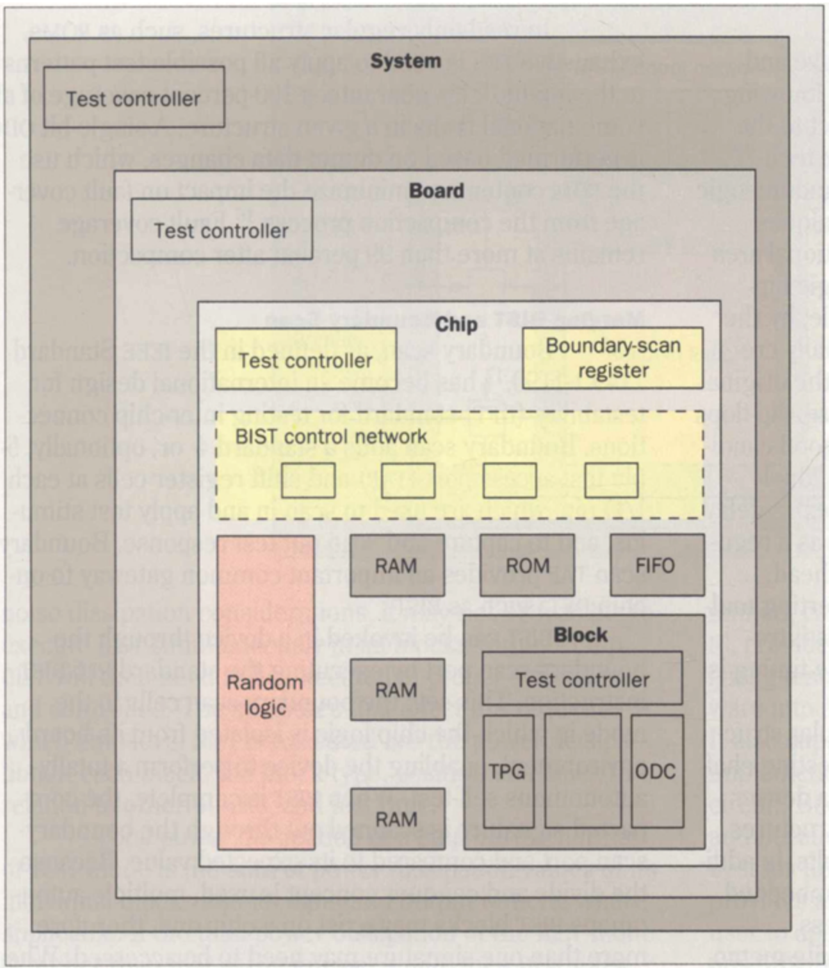


Figure 1. Hierarchical BIST architecture. In general, the circuitry on a chip is partitioned into blocks such as random logic, RAM, ROM, etc., each with its own TPC and ODC. Similarly, the blocks have separate test controllers. The dotted box contains the chip-level distributed BIST control network. The BIST/boundary-scan study provides a typical example of a chip with BIST.

modified, the hardware overhead and the performance penalty can be high.

Pseudo-random BIST. Potentially, pseudo-random BIST[9] has lower hardware and performance overheads than the schemes described earlier. Often, only a small subset of all flip-flops need to be replaced with BIST flip-flops. Because it is non-deterministic, pseudo-random BIST requires fault simulation to compute fault coverage. Test application time differs among various pseudo-random BIST schemes. For instance, a scan-based BIST technique[10] typically requires slightly longer test application time than a circular BIST technique.[11,12]

In pseudo-random BIST, each bit is assumed to have an equal probability of being a 0 or a 1. The number of pseudo-random patterns needed is typically 10^3 to 10^6, depending on the circuit's random-pattern testability and the fault coverage desired. How do we deal with random-pattern-resistant faults,[9] i.e., the residual uncovered faults? One way is to bias the probabilities of input bits, such that the probability of being 1 (0) is not necessarily 0.5. In this way, test patterns needed for the random-pattern-resistant faults are more likely to be generated.[9] Another way is to change the circuit by inserting test points to enhance controllability and observability, to render the original random-pattern-resistant faults easily detectable by random patterns.

Output data compaction for random-logic circuits is usually performed in two dimensions: space and time,[13] compacting a large number of parallel bits to smaller bits over a period of time. For example, the entire circular BIST register is considered a non-linear space compactor.[13] Time compaction is done using an

LFSR that taps into the circular BIST register.

The area overhead of pseudo-exhaustive and pseudo-random schemes can be estimated by knowing the number of flip-flops in a circuit with respect to the total device area. Typically, pseudo-exhaustive techniques require 30 to 40 percent more of the random-logic region of a circuit, while pseudo-random techniques require about 6 to 20 percent more. This additional area includes the reconfiguration of the random-logic flip-flops, and the TPG and/or ODC, when applicable, by the chosen scheme. Because a BIST flip-flop is usually created by inserting additional front-end logic in the original flip-flop, an extra delay is introduced. Therefore, flip-flops that are on a path with critical timing are not good candidates for replacement by BIST flip-flops. Using "dual-clock" BIST flip-flops can almost eliminate the extra delay. A device with non-random logic regions, such as a regular structure, incurs a much smaller area overhead.

Because flip-flops are modified by inserting multiplexers to create a BIST flip-flop, extra delay is introduced. Often, flip-flops that are on paths whose timing is critical are not candidates for BIST flip-flops.

Regular-Structure BIST. Embedded regular structures, such as RAMs, ROMs, etc., offer unique testing challenges. The regularity of their designs allows a denser packing of logic and interconnections. Such structures are, therefore, susceptible to a wider set of faults, in addition to stuck-at faults. They are often deeply embedded in random logic, leaving little or no direct access.

In regular structures, such as read/write memories — for example, static random-access memory (SRAM), Register File, FIFO, or content-addressable memory (CAM) — deterministic BIST algorithms (i.e., those whose test patterns are determined, as opposed to random) are generally adopted. These algorithms provide high fault coverages (more than 99 percent) specific to the structure's fault types, function, and physical layout.[14,15] The BIST circuitry needed to execute these algorithms has low area overhead (typically less than 5 percent more than the regular-structure block). This BIST circuitry generates and applies the algorithmic patterns on the regular structures and collects a single-bit (Go/No-Go) response as a signature. The compaction to obtain a single-bit signature is performed by a special space-compaction technique that does not degrade fault coverage,[13] maintaining the overall coverage at more than 99 percent after compaction.

In read-only regular structures, such as ROMs, exhaustive TPG is used to apply all possible test patterns to the inputs. This guarantees 100-percent coverage of all combinational faults in a given structure. A single-bit ODC is performed based on output data changes, which use the ROM contents to minimize the impact on fault coverage from the compaction process.[16] Fault coverage remains at more than 99 percent after compaction.

Merging BIST and Boundary Scan

Boundary scan, as defined in the IEEE Standard 1149.1-1190,[17] has become an international design for testability (DFT) standard for testing inter-chip connections. Boundary scan adds a standard 4- or, optionally, 5-pin test access port (TAP) and shift register cells at each I/O pin, which are used to scan in and apply test stimulus, and to capture and scan out test response. Boundary scan TAP provides an important common gateway to on-chip DFT, such as BIST.

BIST can be invoked in a device through the boundary-scan port by executing the standard RUNBIST instruction. This sets the boundary-scan cells to the mode in which the chip logic is isolated from its board environment, enabling the device to perform a totally autonomous self-test. When BIST is complete, the compacted signature is scanned out through the boundary-scan port and compared to its expected value. Because the divide and conquer concept is used, multiple autonomous BIST blocks may exist on a chip and, therefore, more than one signature may need to be accessed. When BIST and boundary scan are combined, the same test can be performed during chip test, after the chip is mounted on a board, after the board is inserted into a system, and after the system is installed in the field. The high test quality provided by BIST is the same at all these levels, greatly reducing the complexity and cost of board and system-level DFT.

As Figure 2 shows, by merging BIST and boundary scan and reconfiguring portions of the boundary-scan register to serve BIST control functions, BIST hardware overhead can be significantly reduced. This also provides a natural interface between the boundary-scan port and the BIST controller.

BIST Scheduling and Control

The built-in nature of self-test allows it to be run at system speed. However, because of chip power and

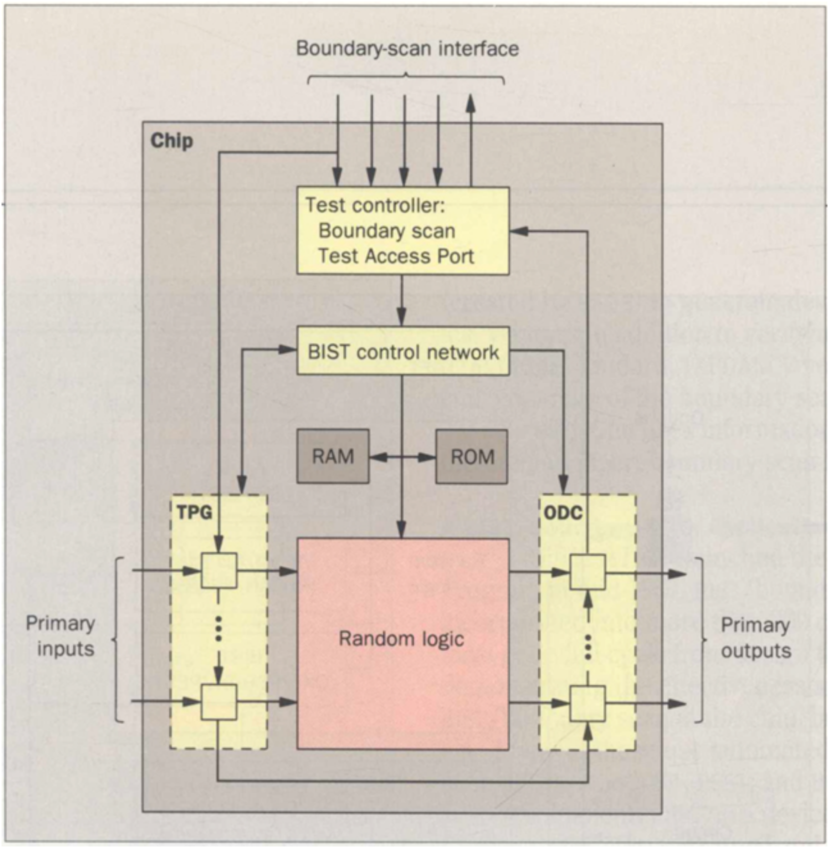


Figure 2. Sharing boundary-scan logic for BIST. The dotted box on the left (right) contains input (output) boundary-scan cells. During BIST mode, it acts as the TPG (ODC).

noise dissipation considerations, it may not be feasible to execute BIST simultaneously in all blocks within a chip. BIST will then need to be executed in proper sequence and controlled. The attributes that affect the sequence in which chip-level BIST is executed are the power dissipation of each block, the block type, position of a block in relation to other blocks, and test time.[7]

Total power dissipation of a chip during normal or BIST mode is the sum of power dissipation values of its individual blocks and its input and output buffers, where applicable. If the total power dissipation of the BIST mode is higher than that of the normal mode, then BIST must be executed in multiple stages. This multiple-stage BIST execution can be represented by a BIST sequencing profile.[7] The hardware implementation of a BIST sequencing profile is called a BIST control network. A distributed architecture for such a BIST control network is optimal at the overall chip level.

BIST controls each chip using a three-level hierarchical control architecture.[7] The three levels, shown in Figure 1, are the chip-level test controller, which includes the external BIST/boundary-scan access port; the BIST control network; and the individual block-level BIST controllers.

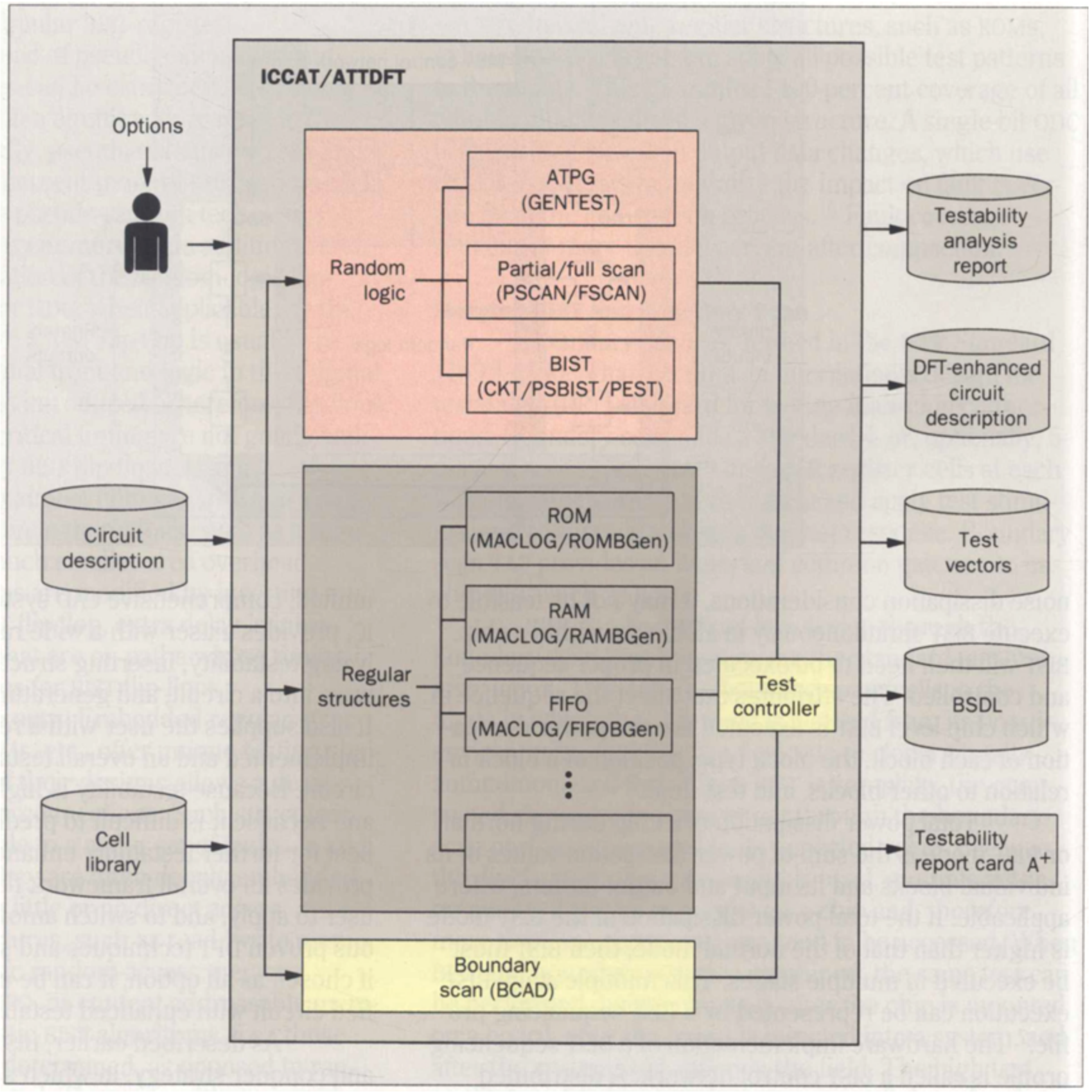
Tools

In AT&T, BIST is inserted automatically using IC-level computer-assisted test (ICCAT) AT&T design for testability (ATTDFT) tools. The ICCAT/ATTDFT tool suite, a unified, comprehensive CAD system for an *entire* digital IC, provides a user with a wide range of options for analyzing testability, inserting structured testability hardware into a circuit, and generating and simulating tests. It also supplies the user with a report on the solution implemented and an overall testability report card of the circuit. Because testability is highly circuit-dependent, and because it is difficult to predict which solution is the best for further testability enhancements, ICCAT/ATTDFT provides an overall framework flexible enough for the user to apply, and to switch among (when needed), various proven DFT techniques and schemes with ease. Also, if chosen as an option, it can be used to produce a modified circuit with enhanced testability.

As described earlier, BIST is based on a divide and conquer strategy, in which a chip is divided into blocks of regular structures, random logic, input/output regions, and control. Each block is applied its dedicated testability scheme and the hardware needed to ensure an optimal overall chip-level testability. Experience has shown that one scheme does not fit all in enhancing the testability of an entire chip.

ICCAT/ATTDFT modules precisely support this strategy. They include, but are not limited to, tools for random logic, regular structures, boundary scan, and the control logic that links all on-chip test circuitries. ICCAT/ATTDFT links those software modules with powerful, easy-to-use graphical user interface (GUI) front-end software. This system accepts pertinent IC-specific DFT

Figure 3. The ICCAT/ATTDFT tool suite provides a comprehensive CAD system for an entire digital IC. Its software uses object-oriented technology and is written in C++ programming language to maximize software reuse, expandability, and maintainability.



information and requirements from the user, performs the requested features for each block of the circuit, and links the blocks with a scheme-specific test controller (see Figure 3). ICCAT/ATTDFT is written in C++ programming language using object-oriented technology.

Random-Logic BIST. BIST schemes for random-logic include pseudo-random circular BIST,[11] pseudo-random partial-scan BIST,[10] and pseudo-exhaustive BIST.[8] The tools for implementing these are circuit know thyself (CKT), PSBIST, and PEST, respectively. PEST replaces all circuit flip-flops with BIST flip-flops; CKT replaces a subset; and PSBIST replaces an even smaller subset. Sometimes, to enhance fault coverage, these BIST tools add test points.

Not all blocks are considered suitable for BIST. Asynchronous and very high-speed blocks can be marked to be isolated (bypassed) during automatic BIST incorporation by the tools.

Regular-Structure BIST. For memories and other regular structures (such as multipliers and different data path blocks), AT&T BIST tools use algorithms that cover an extensive set of faults and guarantee more than 99-percent fault coverage for the structure under test.[14,16] Ad-hoc BIST for regular structures is discouraged because it is error-prone and the quality is unknown.

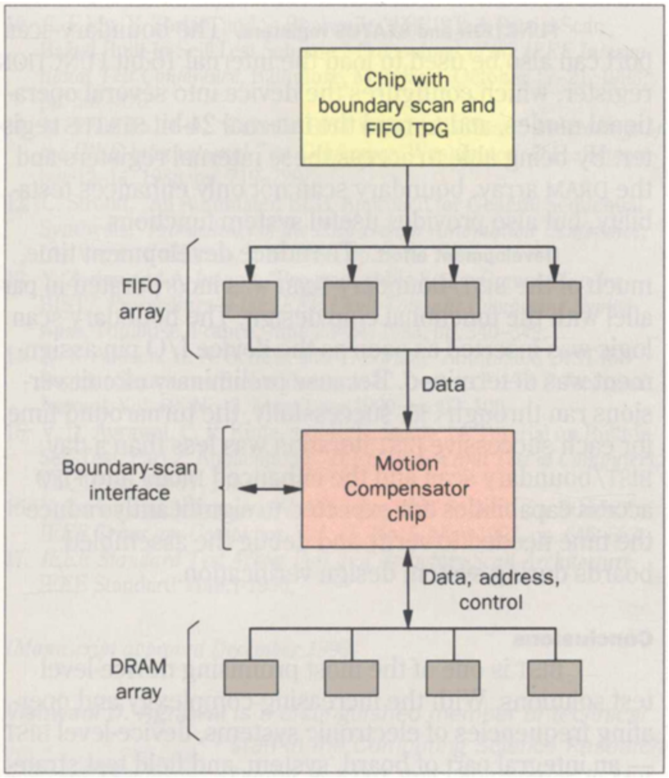


Figure 4. A BIST/boundary-scan application: the Motion Compensator chip. It tests not only itself, but also the surrounding DRAM and FIFO arrays.

One such tool is MACLOG, a macrocell generator available from AT&T-Microelectronics (AT&T-ME). For AT&T-ME devices, MACLOG can generate BIST versions of regular structures, such as RAMs, at the push of a button. Where MACLOG is not applicable, RSBGen, under ICCAT/ATTDFT, uses generic algorithms to synthesize logic for self-testing regular structures.

Boundary Scan. The BCAD tool has three parts:

- The boundary-scan description editor (BSDE),
- TAPDANCE™, and
- The boundary-scan insertion/generation (BSIT/Gen) tool.

The BSDE is an easy-to-use GUI-based editor for creating and editing a boundary-scan description language (BSDL) file. Required for all devices with boundary scan, the BSDL file captures boundary-scan design intent in a standard format.

TAPDANCE uses information from the BSDL file (created by BSDE) to generate device-level conformance test vectors. In addition to verifying conformance to the IEEE 1149.1 standard, TAPDANCE vectors also provide high fault coverage of the boundary-scan logic.

BSIT/Gen uses information from the BSDL to synthesize and insert boundary-scan logic into the circuit.

A BIST/Boundary-Scan Application

Since AT&T launched the BIST/Boundary-Scan Program in mid-1987, BIST/boundary scan has been incorporated into more than 200 chips. Most of these have gone full-cycle from design through manufacturing, demonstrating the effectiveness and benefits of BIST/boundary scan at the chip, board, and system levels. Many of these use automated DFT-insertion tools, such as MACLOG, CKT, PEST, and BCAD. BIST/boundary scan was implemented on a device called the Motion Compensator chip (designed and fully simulated, but not yet fabricated), described in the next section. This chip demonstrates the range of testability that can be inserted into a device.

BIST/Boundary Scan in the Motion Compensator Chip. BIST/boundary scan was used on the Motion Compensator video processing chip for the Advanced Video Technology Group in AT&T Bell Laboratories. This device, which was designed to serve as a main component for video decompression, is a standard-cell application-specific integrated circuit (ASIC) using AT&T-ME's 0.9-μm CMOS technology, packaged in a 132-pin PQFP with a maximum clock speed of 27 megahertz (MHz). Because the Motion Compensator was designed to reside on a board containing a dynamic random-access memory (DRAM) array and FIFO chips, as shown in Figure 4, a full range of testability was inserted into the device to test not only the device itself, but also its surrounding devices.

Boundary scan. Boundary scan was added to test the board interconnections and to serve as a gateway to the BIST and other special on-chip and off-chip functions. Boundary scan was implemented in the Motion Compensator chip using BCAD, an ICCAT/ATTDFT software tool. BCAD-BSIT was used to automatically insert the boundary-scan register and a controller that supports the three mandatory IEEE 1149.1 instructions (BYPASS, EXTEST, and SAMPLE/PRELOAD).[17] The logic needed for ten other instructions (IDCODE, HIGHZ, LOGBIST, MEMBIST, FIFBIST, and the DRAM and internal-register access instructions) was added manually. To make sure that the logic design was correct, simulations were run using IEEE 1149.1 conformance vectors generated by BCAD–TAPDANCE.

Random-Logic BIST. CKT, AT&T's tool for pseudorandom BIST, was used to insert BIST into the Motion Compensator. More than 96-percent fault coverage is achieved after 33,066 BIST cycles. These good results spare the designers the time-consuming task of writing fault-coverage vectors. The BIST circuitry increased the chip gate count by about 14.3 percent over the random-logic portion of the chip. Making the boundary-scan register reconfigurable, to serve as the BIST vector counter and signature analyzer, helped to minimize the BIST control logic overhead.

DRAM BIST. The Motion Compensator chip was designed to be connected to four 64K × 16 DRAM chips at the board level. Special BIST logic, tailor-made for this DRAM, was included on the Motion Compensator to provide 100-percent coverage of an extended-memory fault model to cover faults that occur in the DRAM array and the address decoder. The functionality of the DRAM BIST logic can be verified during a chip test using a special quick-test mode. As in all BIST, the DRAM BIST can be used, simultaneously, at more than one level of testing (i.e., from chip and board test through system integration). As a bonus, the DRAM BIST control logic is also made self-testable by the random-logic BIST tool, CKT.

FIFO BIST. The Motion Compensator chip was designed to receive data from a bank of FIFO chips residing on the same board. The outputs of the FIFO array go to an ODC within the Motion Compensator chip. When the FIFOs are supplied by boundary-scan-controllable ASICs that contain TPG capabilities, the Motion Compensator chip can be used as the data compactor for the FIFO's BIST. The FIFO BIST can be coordinated using boundary scan, as shown in Figure 4.

DRAM write/read. Boundary scan enables the Motion Compensator chip to write into (and read from) user-specified locations in the off-chip DRAM by scanning a 17-bit seed address into the internal address register. Next, the MEMWRITE (MEMREAD) instruction is loaded and data is scanned. As each 32-bit word is scanned in (out), it is written to (read from) the DRAM, and the address register is auto-incremented to point to the next location in the memory. This write/read capability enables the system to initialize the DRAM contents as a background process.

FUNCTION and STATUS registers. The boundary-scan port can also be used to load the internal 16-bit FUNCTION register, which configures the device into several operational modes, and to read the internal 24-bit STATUS register. By being able to access these internal registers and the DRAM array, boundary scan not only enhances testability, but also provides useful system functions.

Development effort. To reduce development time, much of the BIST/boundary scan was incorporated in parallel with the functional chip design. The boundary-scan logic was inserted as soon as the device I/O pin assignment was determined. Because preliminary circuit versions ran through CKT successfully, the turnaround time for each successive BIST iteration was less than a day. BIST/boundary scan and the enhanced DRAM and FIFO access capabilities are expected to significantly reduce the time needed to verify and debug the assembled boards during system design verification.

Conclusions

BIST is one of the most promising device-level test solutions. With the increasing complexity and operating frequencies of electronic systems, device-level BIST — an integral part of board, system, and field test strategies — is becoming the cornerstone of an integrated test strategy.

References

1. V. D. Agrawal, C. R. Kime, and K. K. Saluja, "A Tutorial on Built-In Self-Test, Part 1: Principles," *IEEE Design and Test of Computers*, Vol. 10, March 1993, pp. 73–82.
2. R. P. Prasad, *Surface Mount Technology Principles and Practice*, Van Nostrand Reinhold, New York, 1989.
3. J. Bateson, *In-Circuit Testing*, Van Nostrand Reinhold, New York, 1985.
4. S. C. Seth and V. D. Agrawal, "Characterizing the LSI Yield Equation from Wafer Test Data," *IEEE Transactions on Computer-Aided Devices*, Vol. CAD-3, April 1984, pp. 123–126.
5. I. D. Dear, et al., "Economic Effects in Design and Test," *IEEE Design and Test of Computers*, Vol. 8, December 1991, pp. 64–77.
6. O. F. Haberl and T. Kropf, "A Chip Solution to Hierarchical and Boundary-Scan Compatible Board Level BIST," *Proceeding of the 2nd Great Lakes Symposium on VLSI*, February 28–29, 1992, pp. 16–21.
7. Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices," *Proceedings of the 11th IEEE VLSI Test Symposium*, Atlantic City, New Jersey, April 6–8, 1993, pp. 4–9.
8. E. Wu, "PEST: A Tool for Implementing Pseudo-Exhaustive Self-Test," *AT&T Technical Journal*, Vol. 70, No. 1, January/February 1991, pp. 87–100.
9. P. H. Bardel, W. H. McAnney, and J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques*, John Wiley & Sons, New York, 1987.

10. C. J. Lin, Y. Zorian, and S. Bhawmik, "PSBIST: A Partial-Scan Based Built-In Self-Test Scheme," *Proceedings of the IEEE International Test Conference*, Baltimore, Maryland, October 17–21, 1993, pp. 507–516.
11. M. Pradhan, et al., "Circular BIST with Partial Scan," *Proceedings of the IEEE International Test Conference*, Washington, D.C., September 12–14, 1988, pp. 719–729.
12. C. Stroud, "An Automated BIST Approach for General Sequential Synthesis," *Proceedings of the 25th Design Automation Conference*, Las Vegas, California, June 1988, pp. 3–7.
13. Y. Zorian and A. Ivanov, "Programmable Space Compaction for BIST," *Proceedings of the IEEE Fault-Tolerant Computing Symposium*, Toulouse, France, June 22–24, 1993, pp. 340–349.
14. D. R. Aadsen, H. N. Scholz, and Y. Zorian, "Automated BIST for Regular Structures Embedded in ASIC Devices," *AT&T Technical Journal*, Vol. 69, No. 3, May/June 1990, pp. 97–109.
15. V. D. Agrawal, C. R. Kime, and K. K. Saluja, "A Tutorial on Built-In Self-Test, Part 2: Applications," *IEEE Design and Test of Computers*, Vol. 10, pp. 69–77, June 1993.
16. Y. Zorian and A. Ivanov, "An Effective BIST Scheme for ROMs," *IEEE Trans. on Computers*, Vol. 41, No. 5, May 1992, pp. 646–653.
17. *IEEE Standard Test Access Port and Boundary-Scan Architecture*, IEEE Standard 1149.1-1990.

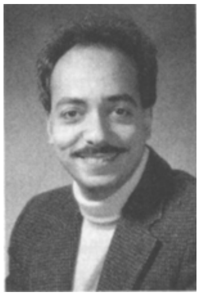
(Manuscript approved December 1993)

Vishwani D. Agrawal *is a distinguished member of technical staff in the Computing Science Research Center at AT&T Bell Laboratories in Murray Hill, New Jersey. His research centers on distributed computing, neural network algorithms, and electronic testing. Mr. Agrawal received a Ph.D. in electrical engineering from the University of Illinois, Urbana-Champaign. He joined AT&T in 1978. Mr. Agrawal is also the Editor-in Chief of the* Journal of Electronic Testing: Theory and Application (JETTA).

Chih-Jen Lin *is a member of technical staff in the Test and Reliability Center of AT&T Bell Laboratories, located in Princeton, New Jersey. His research activities are in the areas of design for testability, especially built-in self-test (BIST), as well as delay fault testing. Mr. Lin received a Ph.D. in electrical and computer engineering from the University of Iowa, Iowa City, and joined AT&T in 1988.*

Paul W. Rutkowski *is a member of technical staff in the Test and Reliability Center of AT&T Bell Laboratories in Princeton, New Jersey. He works on development and application of IC-level built-in self-test (BIST) and boundary-scan tools and methodologies. Mr. Rutkowski received a B.S.E.E. from Rutgers College of Engineering, Piscataway, New Jersey, and an M.S.E.E. from Stevens Institute of Technology, Hoboken, New Jersey. He joined AT&T in 1980.*

Shianling Wu *is a technical manager in the Test and Reliability Center of the Engineering Research Center at AT&T Bell Laboratories in Princeton, New Jersey. She is responsible for research and development of computer-assisted test technology and tools, as well as their deployment and applications. Ms. Wu received a B.S. and an M.S. from Columbia University, New York, New York, both in computer science. She joined AT&T in 1979.*



Yervant Zorian *is a distinguished member of technical staff in the Test and Reliability Center at AT&T Bell Laboratories in Princeton, New Jersey. His research activities are in the area of new BIST methodologies for integrated circuits and multi-chip modules. Mr. Zorian received an Ms.C. in computer engineering from the University of Southern California, Los Angeles, and a Ph.D. in electrical engineering from McGill University, Montreal, Canada. He joined AT&T in 1987.*