A Hierachical Infrastrucutre for SOC Test Management

A Hierarchical Infrastructure for SoC Test Management

Alfredo Benso, Stefano Di Carlo, and Paolo Prinetto
Politecnico di Torino

Yervant Zorian
Virage Logic

HD²BIST—a complete hierarchical framework for BIST scheduling, data-patterns delivery, and diagnosis of complex systems—maximizes and simplifies the reuse of built-in test architectures. HD²BIST optimizes flexibility for chip designers planning an overall SoC test strategy by defining a test access method that provides direct virtual access to each core of the system.

THE CONSIDERABLE GROWTH of digital applications such as communication systems has led to strong competition in the semiconductor industry. Part of the global economy's success for at least the next decade arguably hinges not only on how fast designers can develop and manufacture new chips but also on how fast they can functionally test, diagnose, and verify them. Different test methods are possible, but the goal is always to help the industry grow and not slow it down. Complexity, performance, and density, not to mention cost, will all increase. Keeping up with these trends requires fundamental changes in IC realization methods that directly affect test methods, tools, and equipment.

Silicon area is now so cheap that a single SoC can hold all the components and functions that historically required a hardware board. In addition, each component or function is now available as a predesigned, complex functional block. Often, this block comes as an embedded IP core with an internal structure hidden from the core integrator. Moreover, each core can serve diverse scenarios and embed different test architectures (such as full or partial scan, and BIST), so are reusable in different designs. Designers can also integrate cores from different vendors. When a certain combination of cores becomes common, a system integrator or core provider can create a new core from that combination. Hence, today's SoCs could become tomorrow's IP cores in more complex SoCs.

This new design philosophy, based on the hierarchical reuse of IP cores, requires system-level test architectures that can fully support core reuse, hierarchical design, and integration of multiple test strategies. Direct accessibility to interconnections and core boundaries is impossible in SoC test, but test patterns still must travel from their source to the core, and then to a sink. An access architecture, or *test access mechanism* (TAM), can solve this problem. The TAM must activate the test functions, possibly deliver test patterns, and gather the test results for every core in the SoC hierarchy. In general, a TAM must guarantee the following:

- *Core accessibility.* TAMs should permit the control core testing through a limited set of SoC boundary signals.
- *Reusability.* Easy reconfigurability is necessary to manage the system cores' different test architectures.
- *Minimal overhead.* TAMs must have small area, routing, and performance overheads.
- *Flexible test scheduling.* TAMs should support core test scheduling that minimizes power consumption during test execution.

The proposed IEEE Standard for Embedded Core Test, P1500, will provide a plug-and-play methodology to integrate core testability into a SoC. IEEE P1500 concentrates on a standardized, configurable, and scalable core interface or wrapper that allows easy access to the core's internal-test methods. The TAM, which must manage the execution of the test for the overall chip, is out of the current scope of IEEE P1500; therefore, the test engineer must still design it.

Here, we present an innovative TAM, called hierarchical-distributed-data BIST (HD²BIST), that addresses several critical issues in SoC testing, including core accessibility and test reuse. This architecture allows smooth integration and management of cores with dif-

0740-7475/03/$17.00 © 2003 IEEE Copublished by the IEEE CS and the IEEE CASS

ferent test strategies and built-in test access ports (TAPs). The approach is fully compatible with the hierarchical design methodology of SoCs, allowing access to any core in the system regardless of its hierarchical depth. To reduce power consumption, HD²BIST supports test scheduling of cores using sophisticated control-flow mechanisms. Area overhead is very low because the HD²BIST architecture is fully customizable and adaptable to the test requirements of the cores integrated in the system. Thus, test engineers can trade off routing, area, and test length. In an IEEE-P1500-compliant design, a significant part of the HD²BIST structures would merge with the IEEE P1500 wrapper to optimize performance and minimize area overhead.

TAM architecture

For the sake of clarity, we introduce HD²BIST as a flat architecture, but it actually has a hierarchical structure, fully adaptable to the hierarchical architecture of complex SoCs. (The "State-of-the-Art TAMs" sidebar discusses other proposed TAM architectures.)

Test bus

The main actor in HD²BIST is the test bus (TBus) communication link, which provides an effective solution for core accessibility and TAM reuse. The data exchanged over TBus falls into two categories:

- *Control data* configures and controls all the HD²BIST test structures.
- *Test data* carries the test vectors for testing the system-embedded cores. Test vectors can come from the on-chip BIST controllers or from outside the chip.

To reflect this logical classification, we split TBus into the test control bus (TCB) and the test data bus (TDB). By implementing these as a ring bus, we can guarantee a simple, technology-independent approach and offer high flexibility and dependability.

Because the information exchanged on the two buses differs, each uses different communication protocols. Control data on the TCB is typically easily encodable as a predefined set of commands, called test primitives. Therefore, we chose a token-based protocol for the TCB. Data on the TDB, on the other hand, often involves many test patterns and test responses. Hence, we chose a scan-chain-based protocol to transmit TDB data.

TBus structure implemented by HD²BIST provides high reliability. Thanks to the scan chain protocol, we can easily test the TDB using a standard scan-test approach. Implementing the TCB as a bidirectional link guarantees its dependability: The same information transmitted through a forward link returns via a backward link to verify correctness. If a transmission error occurs, a diagnosis procedure can locate the fault.

Test block

TBus is an efficient TAM only if a bus manager and an appropriate bus interface with the cores under test are properly defined and implemented. In HD²BIST, two special blocks—the test block (TB) and the test processor (TP)—perform these tasks. A unique address defined at design time identifies each core connected to TBus. Broadcast and group addresses add flexibility and reduce the number of primitives necessary to execute each test program.

Engineers can easily customize the TB, shown in Figure 1, to support the specific test solution implemented in the core. We optimized the TB's internal structure for

- full- and partial-scan cores, using the TDB to apply test vectors and gather test results;
- BIST-enabled cores, using the TCB to send BIST commands and read BIST results; and
- BIST-ready cores, using the TDB to exchange test vectors between the core under test and its BIST controller.

Test process

Through TBus, the TP controls all HD²BIST structures inserted in the chip and schedules the test execution for each core in the chip. To make this task as flexible as possible, we interface the TP to TBus through a lower-level interface, as Figure 2 shows. We can implement the interface as either a finite-state machine or a microprogrammed machine. It executes a sequence of test primitives, implementing a set of test programs defined by the core integrator. Each test primitive corresponds to one or more tokens exchanged over the TCB; so test primitives are either TP generated or come from outside the SoC through a Joint Test Access Group interface. The JTAG interface lets a standardized protocol access the TP and the HD²BIST structure to control execution of the test programs from an external tester.

To accommodate the SoC hierarchical design approach, HD²BIST lets you connect different TBus communication links together via the TP. In this case, a TB-like interface called the upper-level interface connects the TP of one TBus to a second TBus, as Figure 3 shows. Each TP

State-of-the-Art TAMs

The two main issues in SoC testing are core isolation and core accessibility. Zorian, Marinissen, and Dey discuss current solutions that create testable and diagnosable embedded-core-based system chips. They also present a generic conceptual architecture comprising three structural elements.[1] This architecture introduces the basic concepts of the test pattern source, test pattern sink, and test access mechanism (TAM).

Researchers have proposed several TAM architectures. These solutions propagate test data in the system in various ways, such as using the functional transparent mode of cores,[2] accessing the core under test by directly multiplexing additional wires into the IC pins,[3] and reaching the cores by having them share multiple test buses of different widths.[4]

Alternative solutions propose modifications to IEEE Std. 1149.1, Standard Test Access Port and Boundary-Scan Architecture, developed for board-level interconnects testing. Touba and Pouya present a variation of IEEE 1149.1 based on a partial-boundary-scan ring around the core.[5] Whetsel suggests providing each core with an addressable test port to directly address the core under test, and introduces a special hierarchical test access port to manage a group of cores as one.[6] To test interconnections among cores, Bhattacharya requires the insertion of test collar cells on the virtual core's I/O pins to create different connections between the cells and the system data bus.[7] Marinissen et al. propose wrapping the cores with an ad hoc interface (TestShell) and connecting them through a proper test bus (TestRail) that delivers the test data patterns and control signals.[8]

To allow flexibility in scheduling tests, Beenker, Dekkar, and Stans propose a centralized controller to activate the BIST sessions one at a time,[9] whereas Zorian suggests distributing the test management on different BIST resource controllers.[10] Finally, Benso et al. introduced the concept of hierarchical distribution of test management with the hierarchical and distributed BIST architecture, to manage different hierarchical levels of BIST-enabled blocks.[11]

Despite the novelty of these approaches, their modularity and flexibility are limited, and their support for BIST is less extensive than their support for scan-based tests of full- or partial-scan cores.

References

1. Y. Zorian, E.J. Marinissen, and S. Dey, "Testing Embedded-Core Based System Chips," *Computer*, vol. 32, no. 6, June 1999, pp. 52-60.
2. I. Ghosh, N.K. Jha, and S. Dey, "A Low Overhead Design for Testability and Test Generation Technique for Core-Based Systems," *Proc. Int'l Test Conf.* (ITC 97), IEEE Press, 1997, pp. 50-59.
3. V. Immaneni and S. Raman, "Direct Access Test Scheme—Design of Block and Core Cells for Embedded Asics," *Proc. Int'l Test Conf.* (ITC 90), IEEE CS Press, 1990, pp. 488-492.
4. P. Varma and S. Bhatia, "A Structured Test Re-Use Methodology for Core-Based System Chips," *Proc. Int'l Test Conf.* (ITC 98), IEEE Press, 1998, pp. 294-302.
5. N.A Touba and B. Pouya, "Using Partial Isolation Rings to Test Core-Based Designs," *IEEE Design & Test of Computers*, vol. 14, no. 4, Oct.-Dec. 1997, pp. 52-59.
6. L. Whetsel, "An IEEE 1149.1 Based Test Access Architecture For ICs With Embedded Cores," *Proc. Int'l Test Conf.* (ITC 97), IEEE Press, 1997, pp. 69-78.
7. D. Bhattacharya, "Hierarchical Test Access Architecture for Embedded Cores in an Integrated Circuit," *Proc. 16th IEEE VLSI Test Symp.* (VTS 98), IEEE CS Press, 1998, pp. 8-14.
8. E.J. Marinissen et al., "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores," *Proc. Int'l Test Conf.* (ITC 98), IEEE Press, 1998, pp. 284-293.
9. F. Beenker, R. Dekker, and R. Stans, "Implementing MACRO Test in Silicon Compiler Design," *IEEE Design & Test of Computers*, vol. 7, no. 2, Apr. 1990, pp. 41-51.
10. Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices," *Proc. 11th IEEE VLSI Test Symp.* (VTS 93), IEEE CS Press, 1993, pp. 4-9.
11. A. Benso et al., "HD^{2}BIST: A Hierarchical Framework for BIST Scheduling, Data Patterns Delivering and Diagnosis in SoCs," *Proc. Int'l Test Conf.* (ITC 00), IEEE Press, 2000, pp. 893-901.

manages only the test of the cores connected to its TBus. The upper-level interface allows the management of each TP as a single unit under test: The TP of the $(i-1)$th level (where $i = 0$ indicates the chip level) considers a TP at the ith level as a standard TB, treating it like any other core under test. Nevertheless, this interface's task is to translate test primitives from the $(i-1)$th level to execute the appropriate test programs for the blocks belonging to its TBus. Using this approach, TPs support a distributed approach for system test execution. Each TP can resolve the com-

mands from the upper bus into all the operations needed on the lower bus, as Figure 3 shows.

HD²BIST treats each TBus as a distinct address domain. This approach lets TBus' addressing scheme work when the SoC is reused as an embedded core in a more complex system. Two cores belonging to different TBus communication links can share the same address without conflict. Using this approach, the only part requiring modification is the external-most TP's upper-level interface. For a stand-alone SoC, this is a JTAG interface; for a SoC used as a core, it is a TB-like interface that connects the TP to the upper-level TBus.

HD²BIST hierarchy configuration and scheduling

We can easily configure the hardware architecture in different modes to create a connection in the hierarchy between a core and its test patterns' source and sink, collect the results of a BIST procedure, diagnose faulty components, or accurately schedule test execution of the overall SoC.

Both mechanisms—for test structure configuration and test scheduling—are possible thanks to a set of test primitives that implement the test programs. These mechanisms define a software level that lets engineers partially separate the design of HD²BIST hardware from the tasks of programming and configuring it.

Test primitives

A test program is a set of test primitives issued as tokens to the TB and TP blocks connected to the TCB. Conceptually, there are atomic primitives and macroprimitives; they don't differ semantically, but their execution results depend on the target block. Atomic primitives are commands received by a TB and used to configure a core's wrapper or change the status of signals at core boundaries. Macroprimitives go to the TPs that connect different hierarchical levels in the TBus tree. Execution of these test primitives activates another test program to manage the test of the lower hierarchical levels.

Using this software level, we can treat a SoC test as a collection of test programs. Engineers don't necessarily need to choose the execution order of the test programs at design time. In fact, HD²BIST provides two ways of delivering test primitives to the blocks under test. In external mode, test instructions come from outside the system through the top-level TP, possibly using a tester connected to the TP JTAG interface. In internal mode, each TP generates the tokens on chip. The two modes are not mutually exclusive, so we can integrate

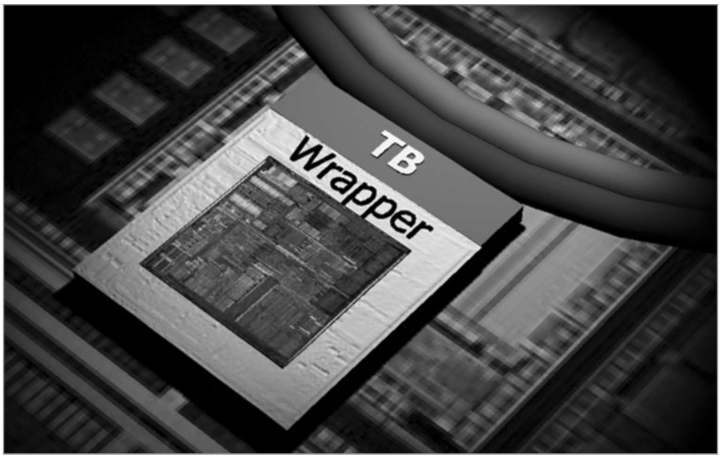

Figure 1. Test block and core wrapper.

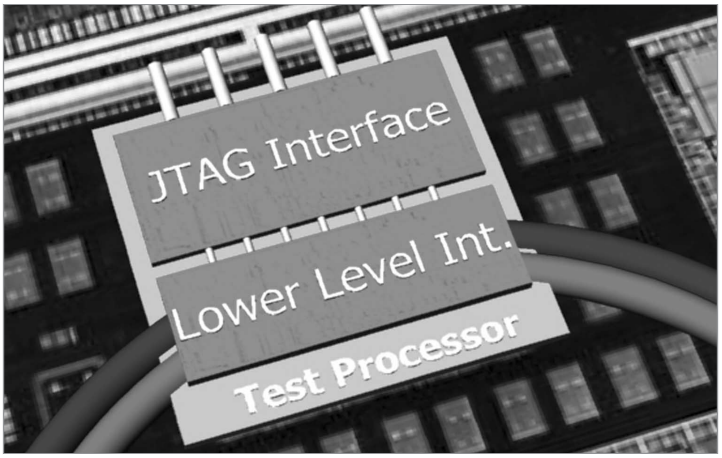

Figure 2. Test processor architecture.

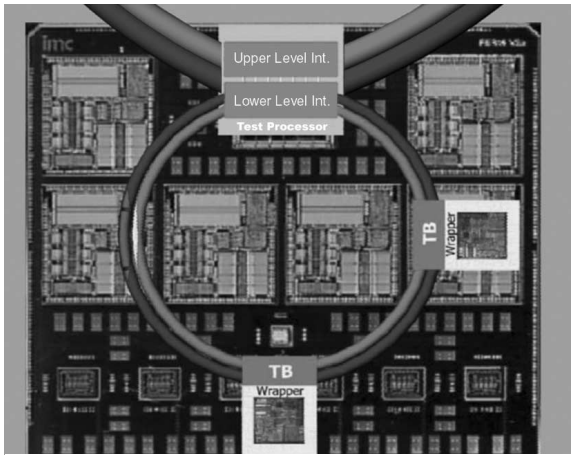

Figure 3. Connecting two rings. The upper-level interface receives data from the upper TBus and, if necessary, forwards it to the lower TBus through the lower-level interface.

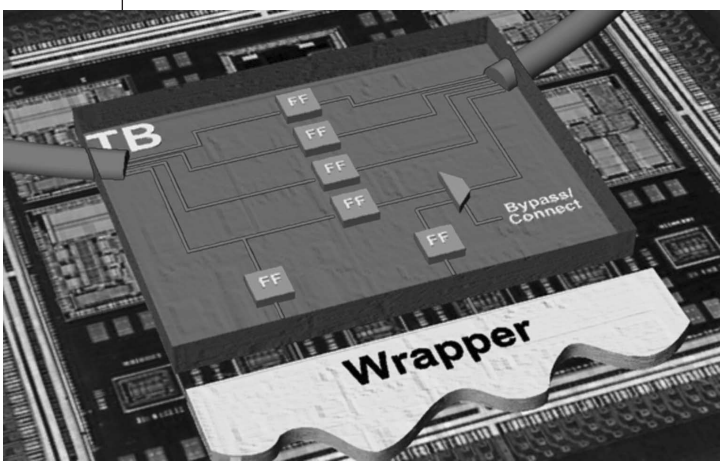

Figure 4. Test data bus (TDB) connections.

them to add flexibility to the overall test strategy. Internal mode is most suitable for activating BIST procedures and reading their results, or for connecting a BIST-ready core to its BIST controller. External mode is suitable for creating a direct data path from outside the core under test, to diagnose problems in or apply test patterns to full- or partial-scan cores.

For external mode, we use the SETENV and UNSETENV macroprimitives to configure the TCB and TDB lines, allowing direct access from outside the chip to any core of the system regardless of its hierarchical depth. SETENV goes to the TP blocks to create a bypass connection between two adjacent hierarchical levels. After receiving a SETENV primitive, a TP begins forwarding all test instructions from the ith level to the $(i+1)$th level. A sequence of SETENV primitives, therefore, makes it possible to reach any level of the hierarchy from outside. The UNSETENV primitive restores normal functionality to TBus.

Test structure configuration

Each test program requires a different configuration of the HD²BIST. Here, a configuration is a connection scheme between the cores under test and the TDB lines for transmitting test vectors. We call each connection scheme a configuration mode, and the set of configuration modes is fully customizable. The only constraint is that each TB must implement at least bypass mode, which disables each TDB line and automatically forwards all the data coming into a TB to the next block on the bus, as Figure 4 shows. The multiplexer lets you select between data coming from the bus (bypass mode) and data coming from the core. The two flip-flops shown in the lower part of Figure 4 perform the routing of the information from the bus to the core and vice versa.

The user can set the configuration mode using the CONF test primitive. The possibility of setting different configuration modes allows *width sharing* (sharing the TDB lines between different blocks in a single test session) or *time sharing* (reusing the same data line to test different cores in different test sessions).

Test scheduling

The last issue solved by the bus-based approach is the test scheduling problem. Four test primitives manage the test session of BIST-enabled and BIST-ready blocks. We can translate the definition of a scheduling algorithm for BIST-enabled and BIST-ready cores into an appropriate sequence of configurations, activations, and collections of a BIST routine's test results.

The START command starts the execution of a BIST routine; the COLLECT command collects the results. These two primitives are useful for defining simple sequential tests but do not allow complex scheduling algorithms for decisions that depend on certain test results. To overcome this problem, we define two additional test primitives:

- WAIT suspends the execution of a test program until the completion of the BIST procedure or the test program execution of one or more blocks. This instruction lets test engineers address possible power consumption issues with concurrent testing of multiple blocks in the system.
- JUMP improves flexibility in test scheduling, letting test engineers make decisions on the fly—for example, to skip testing additional parts of an already-revealed faulty component. Depending on the result of the BIST (or test program) for one or more blocks, the test program execution jumps to a certain label.

Case study

To demonstrate the proposed TAM's effectiveness, we implemented the HD²BIST architecture in LSI Logic's DacTOPplus circuit for transmission devices. We chose this example to highlight our approach's flexibility.

DacTOPplus architecture

DacTOPplus contains four identical macrocores (DacTOPs) and two BIST-enabled 8192x8 RAMs, as Figure 5 shows. Each DacTOP macrocore has four submodules: one transmission macrocell (NDS_TX), one receiving macrocell (NDS_RX), and two identical NDS macrocells. The NDS_RX and NDS_TX macrocells are

full-scan modules with seven scan chains. The HD²BIST implementation treats the two NDS modules as glue logic, and all their flip-flops connect through a single scan chain. The circuit uses the G11 LSI Logic library. Table 1 gives the area that DacTOPplus, shown in Figure 5, occupies in Synopsys equivalent gates.

DacTOP test structure

The test structure implemented in each DacTOP macro consists of a single HD²BIST chain controlled by the TP. Two TBs control the NDS_TX and the NDS_RX macros, packaged by a P1500-like wrapper. The TP directly controls or tests the NDS modules. Therefore, as Figure 6 shows, the HD²BIST structure inserted in each DacTOP macro consists of one TBus, two TBs, and one TP. TBus splits into one 1-bit-wide TCB and one 9-bit-wide TDB. Because each module has seven scan chains, and the TDB must transmit the scan-enable and reset signals driven by the ATPG patterns, we decided to drive all scan chains in parallel; thus, we set the TDB width at 9 bits.

Each TB can implement three connection modes:

- *Bypass.* The TDB merely forwards information to the next level's TB, as described earlier.
- *Connect.* The TDB connects to the scan chains and delivers the scan patterns to the module.
- *Glue.* The core wrapper isolates the core and, through the TDB, applies test patterns to the glue logic at the core boundary.

The TP implements three test programs, PROG[1-3], to connect NDS_RX, NDS_TX, and the two NDS macros to the TDB. Each program sets a different target block in connect mode, and the other TBs in bypass mode. The TP implements three connection modes:

- *Bypass.* Unlike in the TBs, a TP's bypass mode controls only the uppermost TDB.
- *Connect.* The upper-level TDB connects to the lower one.
- *Glue.* The TP creates a direct path from outside the chip to the scan chain connecting the glue logic.

Tables 2 and 3 report the area obtained in synthesizing the DacTOP test case and the HD²BIST architecture using the G11 LSI Logic library.

The area overhead of the HD²BIST structure is 7.03% of the original DacTOP area, and 2.97% of the area of DacTOP with wrappers. We single out the wrapped version of the DacTOPplus because we consider this test requirement to be independent of the HD²BIST structure.

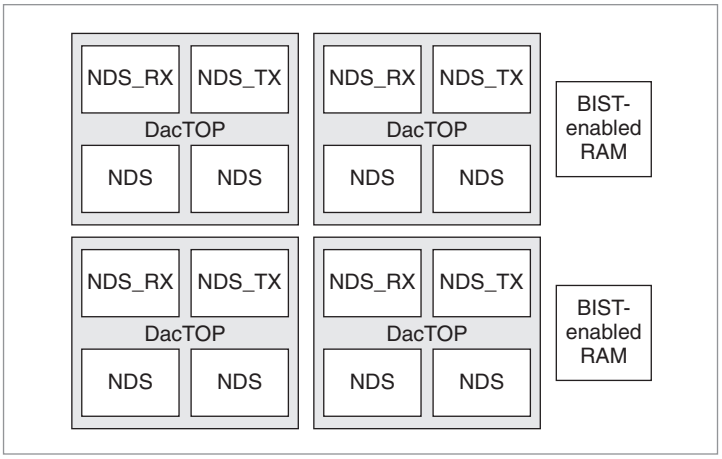

Figure 5. DacTOPplus scheme.

Table 1. Area for the DacTOPplus circuit.

Core	No. of equivalent gates
NDS	99,801
NDS_RX	102,688
NDS_TX	102,802
DacTOP	430,356
BIST-enabled RAM	163,694
DacTOPplus	2,048,814

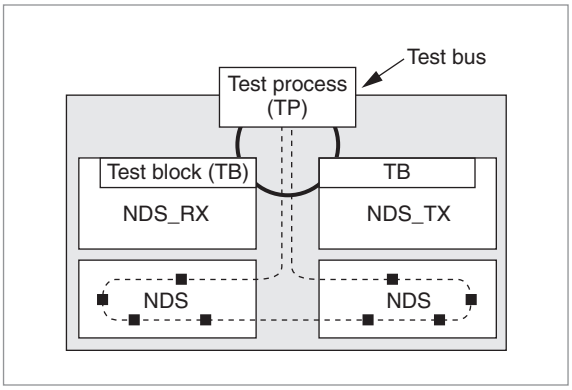

Figure 6. DacTOP HD²BIST scheme.

DacTOPplus test structure

The test structure inserted in the DacTOPplus test case contains one HD²BIST chain at the top level and one HD²BIST chain for each DacTOP module. The test

Table 2. DacTOP area with wrapped modules.

Core	No. of equivalent gates
Glue Logic	199,602
Wrapped NDS_RX	112,049
Wrapped NDS_TX	110,976
DacTOP with wrappers	447,891

Table 3. HD²BIST-enabled DacTOP area.

Core	No. of equivalent gates
TB of NDS_RX	3,695
TB of NDS_TX	3,701
TP	6,145
HD²BIST-enabled DacTOP	461,434

architecture in each top-level DacTOP macro is reusable without modification. As Figure 7 shows, the top-level chain includes one test bus, two TBs, four TPs (one for each DacTOP macro), and one top-level processor with a JTAG interface. The test bus splits into one 1-line-wide TCB and one 9-line-wide TDB (each DacTOP module needs nine lines, whereas the BIST-enabled RAMs do not need any data lines).

In the TP, we implemented 13 different test programs. In the top-level processor, the following programs can execute in any desired order:

- PROG[1] starts the BIST of the two RAMs, waits for BIST to end, and reads the test results.
- PROG[2-4] start PROG[1-3] of the first DacTOP and wait for their end. They then connect the TAP interface's scan-in with the first DacTOP to scan out the test results.
- PROG[5-7], PROG[8-10], and PROG[11-13] perform the same function as PROG[2-4] but with the second, third, and fourth DacTOP modules.

Table 4 reports the area obtained in synthesizing the DacTOPplus using the G11 LSI Logic library.

The area overhead of the HD²BIST structure is 6.61% of the original DacTOPplus area, and 3.06% of the area of the DacTOPplus with wrappers.

Running a test program

To demonstrate how HD²BIST runs a system test, Figure 8 gives the test program, which activates the BIST procedures of the two BIST-enabled RAMs and then starts polling the two TBs until the end of the BIST procedure. This test program allows flexible implementation of any test scheduling; here, it executes BIST for the two memories in parallel, but by simply exchanging instructions 3 and 4, it can execute BIST sequentially.

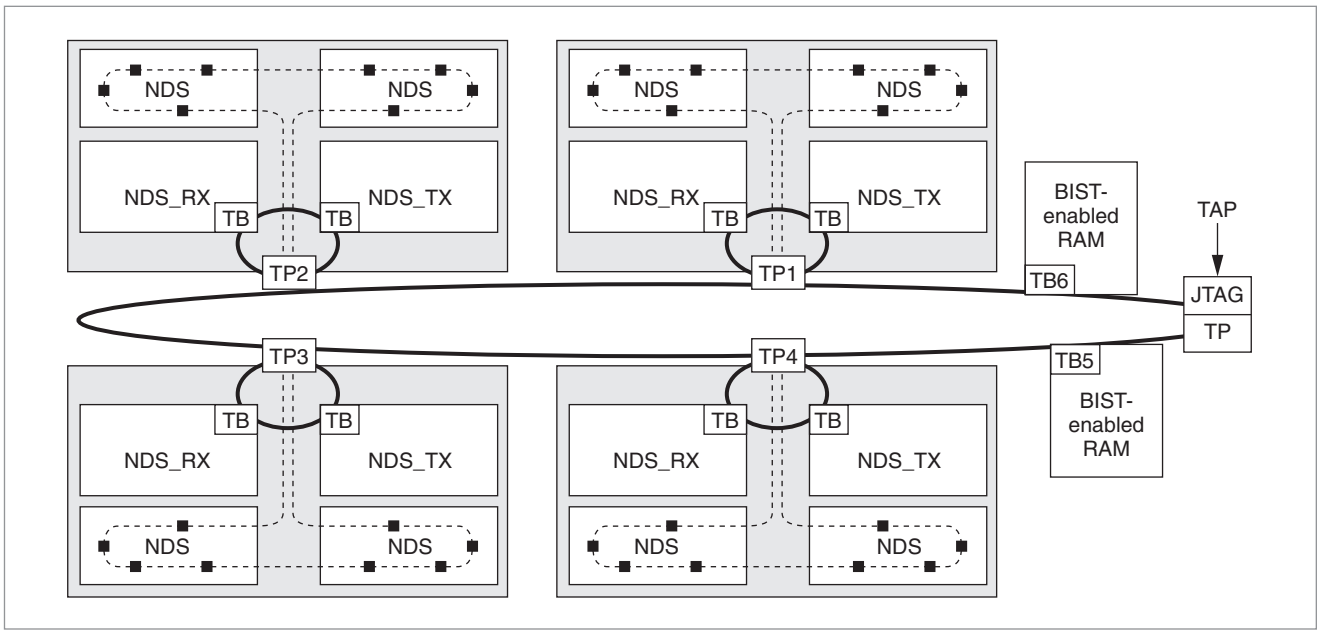

Figure 7. DacTOPplus with HD²BIST.

FUTURE WORK will continue to apply and refine the design methodology presented here to achieve higher levels of testability and dependability. In particular, more work is necessary to better integrate the presented methodologies with existing standards. We especially need to investigate the possibility of automatically integrating the HD²BIST structures with test structures produced by commercial tools for BIST insertion. ■

Acknowledgments

This work is partially supported by Istituto Superiore per le ICT Mario Boella under contract Test DOC: Quality and Reliability of Complex SoC.

Alfredo Benso is a researcher in the Department of Automation and Information Technology at Politecnico di Torino, Italy. His research interests include DFT techniques, BIST for complex digital systems, dependability analysis of computer-based systems, and software-implemented hardware fault tolerance. Benso has an MS in computer engineering and a PhD in information technologies, both from Politecnico di Torino. He chairs the IEEE Computer Society Test Technology Technical Council (TTTC) Web-Based Activities Group.

Stefano Di Carlo is a research assistant in the Department of Automation and Information Technology at Politecnico di Torino. His research interests include DFT techniques, SoC testing, BIST, and FPGA testing. Di Carlo has an MS in computer engineering and a PhD in information technologies, both from Politecnico di Torino. He chairs the IEEE Computer Society Test Technology Technical Council (TTTC) Electronic Submissions Committee.

Paolo Prinetto is a full professor of computer engineering at Politecnico di Torino and a joint professor at the University of Illinois at Chicago. His research interests include testing, test generation, BIST, and dependability. Prinetto has an MS in electronic engineering from Politecnico di Torino. He is a Golden Core Member of the IEEE Computer Society and is the TTTC's chair-elect.

Table 4. Area result of DacTOPplus.

Core	No. of equivalent gates
HD²BIST-enabled DacTOP	461,434
TB of RAM	2,956
TP_TAP	5,958
HD²BIST-enabled DacTOPplus	2,184,997

```
Program PROG[1]
{
    Conf ALL,BYPASS   // Configure all TPs and TBs in bypass mode
                         (the TDB isn't used during the BIST phase).
    Start TB5         // Start the first RAM BIST by sending a start
                         primitive to TB5.
    Start TB6         // Start the second RAM BIST.
    Wait ALL          // Wait for the end of all BISTs. A polling
                         mechanism implements this primitive.
    Collect ALL       // Read the BIST results contained in the RAM
                         TBs, storing them in the top-level TP. If
                         there's a fault, external mode gives direct
                         access to the TBs and BIST controllers to
                         locate the faulty block.
}
```

Figure 8. Test program PROG[1] of the top-level test process.

Yervant Zorian is editor in chief emeritus of *IEEE Design & Test*, vice president and chief scientist of Virage Logic, and chief technology advisor of LogicVision. His research interests include developing embedded test and repair strategies for IP cores, chips, and systems. Zorian has an MSc in computer engineering from the University of Southern California and a PhD in electrical engineering from McGill University. He is a Golden Core Member of the IEEE Computer Society, an honorary doctor of the National Academy of Sciences of Armenia, and a Fellow of the IEEE.

■ Direct questions and comments about this article to Stefano Di Carlo, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Turin, Italy; dicarlo@polito.it.