Component and System Level Design-for-Testability Features Implemented in a Family of Workstation Products

Faced with testing over twenty new ASIC components going into four different workstation and multiuser computer models, designers formed a team that developed a common system-level design-for-testability (DFT) architecture so that subsystem parts could be shared without affecting the manufacturing test flow.

by Bulent I. Dervisoglu and Michael Ricchetti

Members of the latest-generation family of HP workstation and multiuser computer products use the same system architecture and differ mostly in their I/O subsystem architecture and configuration. From a system development point of view an important characteristic of these products is their use of a new high-speed system bus architecture and a large number (over 20) of new ASIC components that were developed to implement all of the various different configurations of the product line. Furthermore, all components that interface with each other via the system bus are required to operate with the same high-frequency system clock.

A further difficulty was that four different models, ranging from a single-user desktop workstation to a multiuser computer, were being developed by different design teams that were both organizationally and geographically separated from each other. This made it necessary to develop a common system-level design-for-testability (DFT) architecture to be used throughout the system and across the different computer models so that subsystem parts could be shared among the different computer models without affecting the manufacturing test flow.

To address these difficulties a DFT core team was formed at the very early stages of the project. Because of the large number of different ASIC teams involved, it was decided that all ASIC teams at the same site would be represented by a single representative on the DFT core team. This team has been instrumental in achieving goal congruence among the different design teams and manufacturing organizations. Furthermore, the presence of the DFT core team made it possible to develop and implement a DFT methodology that was used by all of the ASIC teams, although the level of adherence varied. The DFT core team also collected data and performed DFT design reviews for some of the ASICs.

ASIC DFT Design Rules and Guidelines

One of the first activities of the DFT core team was to develop a set of design rules and guidelines to be followed by the ASIC design teams to ensure that DFT features would be common among the various components. This made it possible to share efforts and results and to access the different DFT features in the ASICs during prototype system bring-up. The following is a summary of these rules.[1]

1. *All (functional) system clocks must be directly controllable from the chip pins and must not be used for any other function.* All systems use a common ASIC component (the system clock controller ASIC) to drive their clock terminals on the system board. This ASIC has control pins through which it can be programmed for different clock generation schemes as well as for starting and halting the system clocks. Thus, not only the individual ASICs but also the entire system board has directly controllable clocks.

2. *All scan and test clocks must be directly controllable from the component pins, which must not be used for any other purpose.* On the system board all test clocks are tied together and controlled from a single test point.

3. *For each ASIC there is a specific reset state which is entered when the component's* ARESET_L *signal is asserted.* On the system board, the power-on condition is detected and is used to reset the ASICs to a known starting state. Next, the memory controller ASIC generates an SRESET_L signal to all other components on the system bus. Additional reset signals are generated by other ASICs for use locally.

4. *All ASICs must implement a dedicated boundary scan register and its associated test access port (TAP) as specified in IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture.*[2] Serial scan-in and scan-out ports of all ASICs in the system (including the PA 7200 processor, which is on a separate module) are connected to form a single serial scan chain.

5. *Access to each ASIC's on-chip test functions must be provided using the IEEE 1149.1 test access port (TAP) protocol.* The same TAP controller design[3] is used or heavily leveraged in many ASICs. This way, test features implemented in this controller as an extension to the IEEE 1149.1 standard were easily leveraged across different ASICs. For example, the DRIVE_INHIBIT/DRIVE_ENABLE instructions and the OUT_OFF bit in the boundary scan register (see "TAP/SAP Controller," below) are duplicated in different ASICs in this way.

© Hewlett-Packard Company 1995

6. *All ASICs shall be designed to support I_{DDQ} testing, whenever this is not prevented by the technology used.* In most cases this requirement did not present any further design constraints or changes. In a few cases, an internal I_{DDQ} enable signal had to be used to disable active pull-up and pull-down circuits. However, because of schedule and cost considerations the PA 7200 processor chip does not support I_{DDQ} testing.

7. *All ASICs shall implement internal scan for testing. The percentage of internal nodes that are scannable shall be kept as high as possible without sacrificing major chip area or otherwise affecting the design methodology.* For most practical purposes all ASICs have implemented internal scan for 100% or nearly 100% of all internal flip-flops. However, because of design style and technology differences, some portions of the PA 7200 processor chip are not scannable.

8. *There shall be no asynchronous logic implemented in the ASICs.* Lack of asynchronous logic is an important requirement for many CAD tools for generating test vectors. Furthermore, this rule is intended to prevent side effects caused by changing the internal and external signals in arbitrary sequence. The only exception to this rule is granted for the reset signals, which are implemented to follow a carefully planned system reset strategy.

The following sections describe some of the DFT features that have been implemented in the ASICs. Not all features are implemented in all ASICs. Among the various ASICs, the memory controller stands out as the chip with the most extensive DFT features.

TAP/SAP Controller

Access to all on-chip DFT features is implemented through a test controller block called the *test access port/scan access port* (TAP/SAP). The test controller implements all of the required instructions for the IEEE 1149.1 TAP controller as well as an extensive set of public and private instructions which are targeted mostly for internal testing of the ASIC. Table I lists all of the TAP instructions that are implemented. Among the public instructions that have been implemented are the DRIVE_INHIBIT and DRIVE_ENABLE instructions which are used to set and clear a latch in the system logic domain (not considered part of the test logic).

System logic for all ASICs has been designed such that for normal system operation (i.e., when test logic is not controlling the I/O pins) the ASIC can drive out only if the DRIVE_INHIBIT latch is cleared. Each ASIC uses its ARESET_L input to clear the DRIVE_INHIBIT latch during power-up. Whereas ARESET_L controls the DRIVE_INHIBIT latch only if the TAP is in a reset state, explicit TAP instructions can be used at other times to set or clear this latch. This scheme allows in-circuit ATE programs to set the DRIVE_INHIBIT latch before they terminate and reset the TAP without creating possible board-level bus contention before removing electric power from the board. Whereas the DRIVE_INHIBIT latch is considered part of the on-chip system logic, it is implemented as part of the TAP controller design so that ASIC designers implementing normal system functions do not have to deal with any of the issues surrounding the DRIVE_INHIBIT and DRIVE_ENABLE operations.

Table I
TAP Instructions

Instruction	Drive I/O Pads	Scan Register
EXTEST	Boundary Register	Boundary
BYPASS	System Logic	Bypass
SAMPLE/PRELOAD	System Logic	Boundary
IDCODE	System Logic	ID Code
HI_Z	High-Impedance	Bypass
DRIVE_INHIBIT	Boundary Register	Bypass
DRIVE_ENABLE	System Logic	Bypass
SCAN_INTERNAL	System Logic	f(Mode)
CHIPTEST	High-Impedance	f(Mode)
INTEST	Boundary Register	Boundary
DR_SCAN	System Logic	f(Mode)
SELECT_MODE	Boundary Register	Mode
SET_MODE_BIT	Boundary Register	Mode
CLR_MODE_BIT	Boundary Register	Mode
ISAMPLE	System Logic	Bypass
ESAMPLE	System Logic	Bypass
DS_DRIVE	Boundary Register	Boundary
DS_RECEIVE	System Logic	Boundary

Other TAP instructions are used to set and clear bits of the mode register to provide access to additional test features such as I_{DDQ} testing, double-strobe, and so on. It is also possible to speed up internal scan operations by switching on the parallel scan bit in the mode register. This feature enables multiplexing of the chip's I/O pins to perform serial scan-in and scan-out of the internal scan register by breaking it into three independent sections which are scanned in parallel together with the boundary register, which is always scanned using the test data in and test data out pins of the TAP.

CHIPTEST Instruction

One of the major difficulties in implementing DFT in the ASICs used for this project has resulted from a common leveraged I/O pad design that contains nonscannable latches. Furthermore, the bidirectional I/O cell implements an internal bypass path to feed into the chip the same value that is being driven onto the I/O pad by that chip. In effect, I/O pads contain nonscannable pipeline stages that control both the direction and the value of data on the I/O pad. Following a recommendation from the DFT core team the basic I/O cell design was modified to allow data values received by the on-chip system logic to be set up using the dedicated boundary scan register. In addition, system logic output values can be captured into the boundary scan register using the system clock. These design changes were coupled with features provided by the CHIPTEST instruction in the TAP controller to streamline the internal testing of the ASICs. For example, all internal logic of the memory subsystem ASICs (memory controller, slave memory controller, and data multiplexer) is tested by the following sequence:

1. Load the CHIPTEST opcode into the ASIC.
2. Use test clocks to perform a parallel scan of the ASIC internal nodes and the boundary register. At the end of the scan-in process the newly scanned-in values are automatically moved from the boundary register to the nonscannable latches in the I/O-cells.

© Hewlett-Packard Company 1995

3. Apply a single system clock to capture test results in internal nodes and system logic output values in the boundary register.
4. Repeat steps 2 and 3 for each new vector, overlapping the scan-in and scan-out operations.

Since the CHIPTEST instruction drives the I/O pins to a high-impedance state it is possible (indeed it is intended) to execute these tests on a populated system board without fear of creating board-level bus clashes during such testing.

BIST Implementation

The memory controller ASIC incorporates several wide and shallow register files that are used for queueing operations within the data paths. The total number of storage elements in the register files is quite large, so it was not practical to make these storage elements scannable. Therefore, a built-in storage test (BIST) approach was chosen to test the memory controller data path register files.

The memory controller BIST implementation was developed with the following objectives:

- Provide high coverage and short test times.
- Provide at-speed testing of the register file structures to ensure that the memory controller ASIC works at the required system clock frequency.
- Provide flexibility and programmability in the BIST logic to allow alteration of the test sequence for debug and unforeseen failure modes. In particular, the system bring-up and debug plans provide a means for system-level scan access to the state within the ASICs. Providing these features allows read/write access to the nonscannable queue states for prototype system debug.
- Provide for testability of the logic surrounding the register files through added observation and control points at the inputs and outputs of the register file blocks. This is intended to support automatic test pattern generation (ATPG) tools used to generate test vectors for the memory controller and thus ensure high coverage of the standard cell control logic for the queues.

The design of the BIST logic in the memory controller data paths is based on previous work that was done for the PA 7100-based HP 9000 Model 710 workstation. For that product, a structure independent RAM BIST architecture that uses a pseudoexhaustive test algorithm and signature analysis was developed and was implemented in the I/O controller ASIC.[4] The structure independent, pseudoexhaustive test algorithm provides 99.9% fault coverage of typical RAM faults and can provide 80% to 99.9% coverage of neighborhood pattern-sensitive faults. It also allows the test time (number of read/write accesses per memory address) to be varied according to the desired fault coverage. BIST architectures for both the present memory controller ASIC and the previous I/O controller ASIC use a test algorithm similar to that described by Ritter and Schwair.[5] Using the system clock for BIST execution, the RAM structure can be tested at the normal system clock rate, thus providing at-speed testing of the RAM.

A dual-port write/single-port read register file from the present memory controller data path, with test structures that provide both BIST and ATPG support similar to the previous I/O controller BIST architecture, is shown in Fig. 1. The two write ports, A and B, can both be addressed and written independently. The single read port can also be addressed and read independently of the A and B write ports. Thus, two write operations and one read operation can all occur simultaneously for one to three register locations, depending on the A, B, and read port addresses.

Given the dual-ported design of the memory controller register files, it was necessary to extend the previous I/O controller BIST architecture to test a dual-ported RAM. This meant that the memory controller BIST implementation should be able to test not only the simultaneous dual-write operations but also the various combinations of A/B write and read operations to verify that the port interactions are working correctly. For the dual-port register files in the memory controller such interactions include an internal bypass when the read address is the same as either of the A or B write addresses and a B-port dominant write when the A and B write addresses are equal. This dual-write BIST algorithm is described in reference 6.

For the register file shown in Fig. 1 each of the BIST structures—LFSR (linear feedback shift register), SHIFT, COUNT, and MISR (multi-input signature register)—is dedicated to BIST. Each register file also has its own dedicated programmable BIST control queue for sequencing the BIST algorithm. The BIST_MODE signal enables the BIST functions and can be

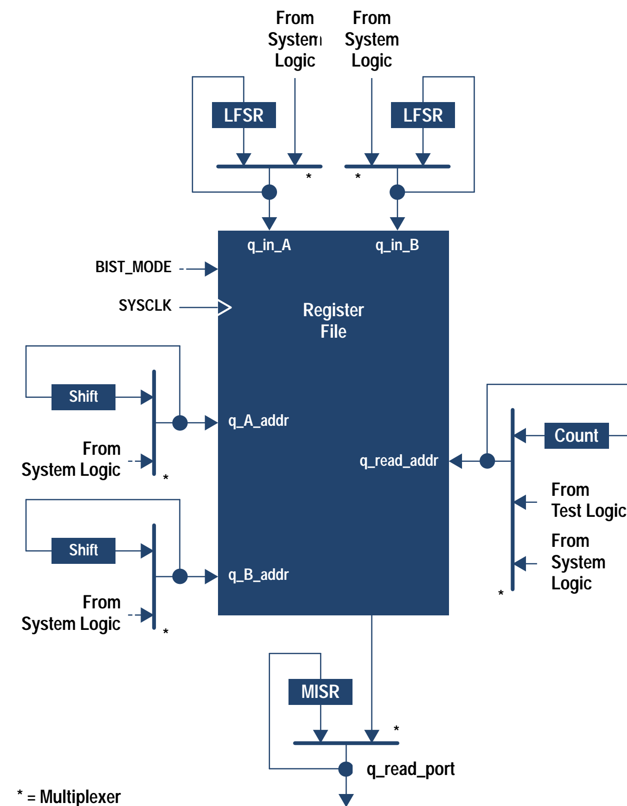

Fig. 1. Dual-port register file with built-in storage test (BIST) and automatic test pattern generation (ATPG) features. The inputs to the central, embedded RAM structure are provided by multiplexing between the normal system value and a BIST register, which is implemented as a linear feedback shift register (LFSR). The output multiplexer makes it possible to capture the outputs into a multi-input signature register (MISR) and to send either the RAM outputs or the MISR contents to the rest of the system.

© Hewlett-Packard Company 1995

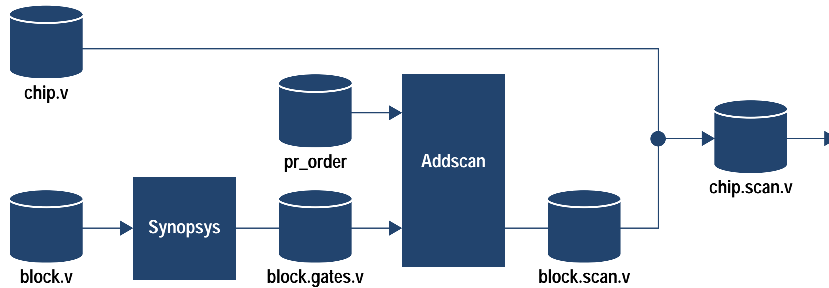

Fig. 2. Addscan tool flow. Addscan is an in-house software tool for scan insertion. Synopsys is an automatic design synthesis tool from Synopsys, Inc.

controlled either by a pin on the chip or through other test access logic such as an IEEE 1149.1 TAP controller or P1149.2 SAP controller.[2,7] All of the BIST registers are implemented in standard cell blocks separate from the data path register files. A detailed description of the memory controller BIST implementation and operation, along with hardware overhead and test coverage, can be found in reference 6.

Test Tools

The following sections describe the tools and tests that were used and developed to test the three memory subsystem ASICs: the memory controller, the slave memory controller, and the data multiplexer.

Addscan. Fig. 2 shows the flow for scan synthesis. All three memory subsystem ASICs were designed using a structured-custom design method.[8] Each standard cell block used the in-house Addscan tool[9] for scan insertion and the scan links between blocks were connected by hand in the top-level netlist of the chip. The internal scan order of each block is based on post place and route information.

Test Vector Generation. The test vector generation flow is shown in Fig. 3. The ATPG tools from Crosscheck, Inc. were used to generate most scan-based vectors. Some vectors were hand-generated. A gate-level netlist of the chip, prior to Addscan scan insertion, is used to create an ATPG database for vector generation. A similar data base is used by designers to do Timver static timing analysis. Timver, a timing analysis tool from Aida, Inc., is used as a part of the test methodology for two purposes. First, it allows the design to be checked for hold violations on all paths to guarantee that there will be no timing violations even if ATPG vectors exercise nonfunctional paths in the design. Secondly, Timver critical paths can be fed back into ATPG in the form of a vector.tcp file to generate double-strobe path delay vectors.

The following test vector sets were created for each of the memory subsystem ASICs:

- Continuity. Checks for opens and shorts among the ESD protection diodes. Prepared manually.
- Ringtest. Uses serial "flush" speed (total scan path delay) through the boundary scan register as a measure of the IC process and verifies that the part is within the six-sigma range. Generated manually in the form of a Cadence Verilog body file.
- Dc. These tests use the boundary scan ring to drive out all ones or zeros for dc parametric testing. Generated manually in the form of a Verilog body file.
- Leakage and tristate testing. Places the ASIC into a high-impedance state to allow testing the I/O pads for leakage. Generated manually in the form of a Verilog body file.
- I_{DDQ}. These vectors are generated by ATPG and are used to perform static I_{DDQ} test and measurement.
- TAP Tests. These are tests targeted at functional testing of the TAP controller. Generated manually in the form of a Verilog body file.
- Chiptest. These vectors are generated by ATPG to test the core chip logic in from and out to the boundary scan ring using the TAP CHIPTEST instruction. I/O pad logic is not fully tested by chiptest vectors.
- Pintest: These vectors are generated by ATPG and will test the remaining faults (primarily in the I/O pad logic) that are not covered by the chiptest.
- Bus Holder. Further testing of the electrical characteristics of the bidirectional I/O cells. Generated manually in the form of a Verilog body file.
- BIST. BIST vectors are only generated on the memory controller. These tests require only two scan vectors, one each to set up the initialization and test passes for BIST. After that, a burst of system clocks is applied to test the target

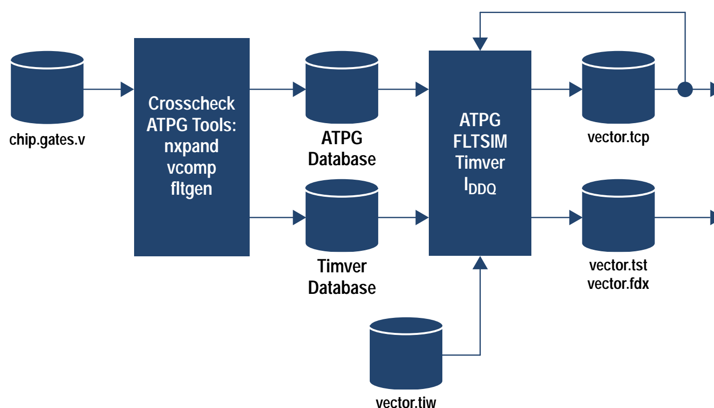

Fig. 3. ATPG (automatic test pattern generation) and timing tools flow. Timver is a timing analysis tool from Aida, Inc. FLTSIM is an in-house fault simulator for test verification. The I_{DDQ} vectors are generated by ATPG from Crosscheck Corp.

© Hewlett-Packard Company 1995

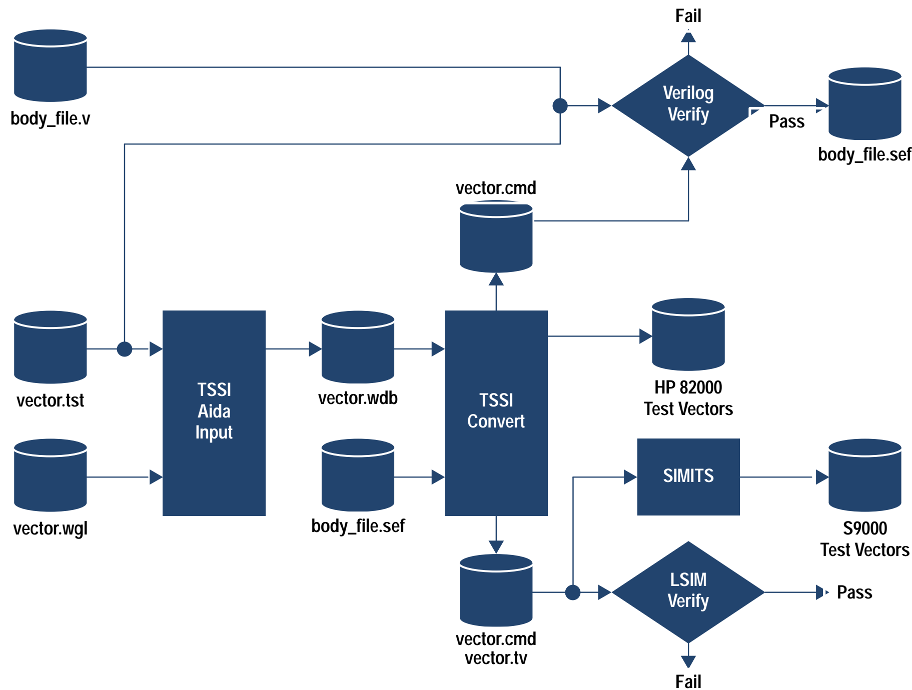

Fig. 4. Vector verification and tester translation tools flow. TSSI is a test program generation tool from TSSI, Inc. Aida represents a suite of test tools from Aida, Inc. LSIM is a special FET-level simulator. SIMITS is a format conversion tool from Schlumberger.

blocks at speed. These vectors are generated using a Perl script to produce a .tst vector file.

- Double-Strobe. These vectors are generated by ATPG based on Timver critical paths and are used to provide at-speed testing of the ASIC.[10,11]
- Ac testing of I/O Paths. These are functional tests that test the speed characteristics of critical I/O paths. Generated manually only if testing, design review, and chip characterization results indicate a concern.
- Process, Voltage, and Temperature (PVT) Block Test. Generated manually, this group of tests applies only to the slave memory controller chip which uses a unique PVT block to compensate for process, voltage, and temperature variations in a particular I/O cell.

Vector and Test Logic Verification. Fig. 4 shows the flow for test vector verification using Verilog or LSIM (a special FET-level simulator). Test vectors from ATPG can be directly converted using TSSI (a tool for test program generation from TSSI, Inc.) into a command file format and verified against a gate-level netlist in Verilog or a FET-level netlist using LSIM. Alternatively, test vectors can be simulated using a Verilog body file. A body file is a wrapper or test jig that can either be a test vector set itself (hand-generated functional tests) or can run scan-based ATPG vectors using a scan and clock sequence.

The AT&T Tapdance tool was used for further verification of the TAP logic before tape release of the ASICs. Tapdance generates a set of IEEE 1149.1 compliance tests to verify standard TAP functionality. The Tapdance vectors were converted using Perl scripts† into a Verilog force file and simulated on a gate-level netlist.

Tester Format Translation. Fig. 4 also shows the flow for translation of vectors into a tester format. Using TSSI, vectors were formatted directly to HP 82000 tester format. To get to the Schlumberger S9000 tester, vectors were first formatted to LSIM and then passed through SIMITS, a format converter from Schlumberger.

† Perl is a high-level programming language.

Using a Verilog programming language interface that outputs a TSSI simulation event format file dump, vectors can also be translated from body files to one of the testers.

System DFT Features

The new systems have been designed to provide a method to access ASIC scan paths, both boundary and internal, at the system level. This has two major purposes. First, it provides a means of accessing the internal state of complex VLSI components. This provides additional hardware state information to designers that would typically be inaccessible and can aid traditional prototype bring-up and debug methods. Second, it provides the ability to do scan-based testing of board and system interconnect and internal scan testing of ASICs.

The following test and debug features are provided by system scan access:

- Ability to halt the system clocks and interrogate the internal scan state of the ASICs.
- Single-cycle debug of the system core by halting the system clocks, interactive scanning of the internal state, and then starting or cycling the system clocks.
- Board-level and system-level interconnect testing and interactive debug using boundary scan. This includes testing connectors between two boards where boundary scannable buses cross the connector.
- Ability to test an ASIC while it is on the board using boundary and internal scan. This may include double-strobe tests and running on-chip BIST, if supported by the ASIC under test.

As part of the overall DFT requirements, all ASICs implement the IEEE 1149.1 Standard Test Access Port and Boundary-Scan Architecture. This provides support for system-level scan access. In addition, key debug support features are

© Hewlett-Packard Company 1995

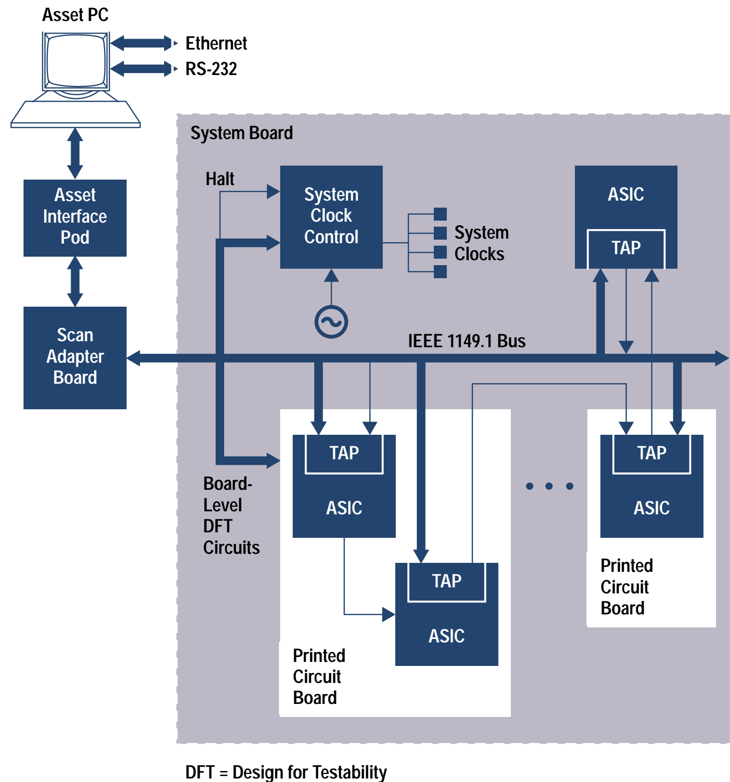

Fig. 5. System-level scan access. Asset is a set of scan tools from Texas Instruments, Inc.

incorporated into the system clock controller chip to allow for halting and controlling the system clocks. Further information on system clock controller features can be found in reference 12.

Fig. 5 shows a diagram of the system-level scan access hardware. The Texas Instruments PC-based Asset toolset is used as the interface to system scan. The Asset PC is connected to a scan adapter board via the Asset interface pod. The scan adapter board then plugs onto the system board and provides control of the system clock controller features, the TAP controller and system logic reset, the clock halt triggers, and the I/O device clock halt from the Asset software. The scan paths in the system are configured as a single serial scan chain with optional system boards implemented as dynamic scan paths that can be configured in Asset.

The Asset scan tools provide the following capabilities for system scan access:

- Interactive control of scan path data and TAP controller instructions with scan-bit name mapping and packing and unpacking of scan data.
- Macro scripting capabilities for combining several interactive operations into a single macro command. Asset also accepts serial vector format scan vectors for user-developed tests.
- Specification of system scan path configuration for dynamic scan paths and optional boards, such as CPUs, memory extender boards, and I/O.
- Scan path integrity testing and boundary scan interconnect testing of intraboard and interboard nets.
- Control of system clock halt, single-cycle stepping, and system and TAP reset.

Results and Conclusions

The DFT techniques described above, which were championed by the DFT core team, were implemented in several different ASICs with varying degrees of adherence to the DFT rules and methodology. In general, results obtained during prototype chip debug have shown a direct correlation between the level of DFT implementation and the rapidity of test development, chip characterization, and root-cause analysis. For example, while the three memory subsystem ASICs were the last to reach tape release, these chips were the first to reach the operational test release (OTR) and release to manufacturing test (RTPT) checkpoints. The availability of high-quality and comprehensive test sets for these chips enabled chip characterization efforts to be started right away. Furthermore, success in reaching the OTR checkpoint made it possible to transfer the task of testing prototype chips (which are used in the prototype systems) to the manufacturing engineers. This had a very positive effect on resources available to perform chip characterization. In turn, successful completion of this step coupled with efforts of the R&D engineers to improve test coverage enabled the team to reach the RTPT milestone well before any of the other ASICs had reached their OTR checkpoints.

The Asset tool and its customized extensions provided a low-cost system scan access solution with flexible functionality and ease of use. As a commercial tool solution it cut down on development and maintenance costs compared to developing a proprietary toolset and can be reused for future projects.

© Hewlett-Packard Company 1995

Acknowledgments

The authors gratefully acknowledge contributions from many members of the Chelmsford DFT group, the DFT core team, and several ASIC designers. In particular, contributions from Ruya Atac, Steven Zedeck, Tom Dickson, Shawn Clayton, Joy Han, Jim Williams, Matt Harline, Dave Malicoat, Ken Pomaranski, and Mick Tegethoff have been key to the successful implementation of DFT in this project.

References

1. B. Dervisoglu, *ASIC DFT Design Rules*, Hewlett-Packard ASD/CSL, November 1992.

2. IEEE Test Technology Technical Committee, *Standard Test Access Port and Boundary Scan Architecture*, IEEE Standard 1149.1/1990, Institute of Electrical and Electronic Engineers, Inc.

3. B. Dervisoglu, *PA 7200 Memory Subsystem TAP/SAP Design*, Hewlett-Packard ASD/CSL, January 1994.

4. M. Ricchetti, "A Structure Independent Built-In Test Architecture for Random Access Memories," *Proceedings of the 1992 Hewlett-Packard Design Technology Conference*, pp. 417-424.

5. H.C. Ritter and T.M. Schwair, "A Universal Test Algorithm for the Self-Test of Parametrizable Random Access Memories," *Proceedings of the Second European Test Conference*, April 1991, pp. 53-59.

6. M. Ricchetti, "Built-In Self-Test of the Memory Controller Data Paths," *Proceedings of the 1993 Hewlett-Packard Design Technology Conference*, pp. 303-312.

7. IEEE Test Technology Technical Committee, *Extended Digital Serial Subset*, IEEE Draft Standard P1149.2/D0.2, Institute of Electrical and Electronic Engineers, Inc.

8. S. Clayton, D. Chou, T. Dickson, D. Montrone, R. Ryan, and D. Schumacher, "The Automation of Standard Cell Block Design for High-Performance Structured Custom Integrated Circuits," *Proceedings of the 1993 Hewlett-Packard Design Technology Conference*, pp. 349-356.

9. S.A. Markinson, "The Application of Scan Synthesis in System ASIC Design," *Proceedings of the 1992 Hewlett-Packard Design Technology Conference*, pp. 449-458.

10. B.I. Dervisoglu and G.E. Stong, "Test and Design-for-Test Standards for the Kodiak Project," *Proceedings of the 1990 Hewlett-Packard Design Technology Conference*, pp. 564-571.

11. B.I. Dervisoglu and G.E. Stong, "Design for Testability: Using Scanpath Techniques for Path-Delay Test and Measurement," *Proceedings of the International Test Conference*, October 1991, pp. 364-374.

12. D. Malicoat, *System Clock Controller ASIC: External Reference Specification, Version 0.91*, Hewlett-Packard Systems Technology Division, November 30, 1992.

© Hewlett-Packard Company 1995