Prototype Testing Simplified by Scannable Buffers and Latches

by Andy Halliday, Greg Young, and Al Crouch

Reprinted with permission of the IEEE.

Abstract

Conventional logic devices incorporating boundary scan with the IEEE 1149.1 interface offer tremendous improvements in board testing. These benefits and improvements are contrasted against traditional approaches.

Introduction

The use of devices incorporating the IEEE 1149.1[1], hereafter referred to as 1149.1, boundary-scan architecture offers significant advantages when testing prototype systems. Prototype systems primarily are developed to allow engineers to prove a design concept or implementation before committing it to full production, where the cost to correct problems can be prohibitive. Engineers can verify such items as the basic design concept, theory of operation, board layout, parts selection, etc. Design flaws undetected during a paper design or simulation also can be corrected.

A prototype system is normally a low-cost, low-volume production of the end product. Most of the money spent during prototype development is devoted to building the prototype and verifying the design concept; very little usually is allocated for testing. Prototype testing usually is accomplished through simple functional operation and verification of the system as a whole. Functional tests of system hardware and software typically are performed on the entire system or pieces of the system, possibly with some unavailable functions being emulated externally. When failures are detected during this design verification process, a significant amount of time is spent determining the cause of the failure and correcting it. This debugging process is a manual, often time-consuming, task that does not ensure immediate success. Several hours may be spent rerunning the system after swapping boards and components that are believed to have caused the failure but, in fact, did not. Mixing design verification (concept, implementation, etc.) with fault verification can be costly in the long run if these processes do not complement one another.

Design verification and fault verification can be less painful by using devices that support the 1149.1 boundary-scan architecture and partitioning the system into small, easy-to-test functions. The use of boundary scan allows each partitioned function to be verified and tested independently, thereby reducing the time spent locating the cause of a failure. Boundary scan provides an increase in the controllability and observability of internal circuit nodes, which is mandatory when isolating system hardware faults and verifying operation of system software. Standard integrated circuits implementing the 1149.1 boundary-scan architecture are very beneficial in this situation. The use of such devices supports a hierarchical test philosophy in which the same test capabilities and test programs can be reused at each level (device, board, box, system).[2]

The following paragraphs compare traditional methods for design verification/debug/test of a prototype system with the method used for a prototype system containing boundary scan.

Traditional Test Methods

During prototype system design, testing issues are often far from the minds of the designers. If test is an issue, ad-hoc testability techniques sometimes are implemented. The design engineer primarily is concerned with how to implement a given piece of the system and have it interface correctly with the rest of the system. The software engineer is concerned with getting the system software working and verified on the system. The systems engineer is concerned with how the sys-

tem is going to function when all the pieces are put together. The complexity of the test problem comes to the surface when all the pieces are assembled and the system does not function properly. Determining why the system does not function as intended can be a major problem. Typical problems include:

- A manufacturing problem (miswire, wrong component, etc.)
- A design problem (incorrect design implementation)
- A software problem (incorrect algorithm)
- A hardware failure (bad part, etc.).

There are many approaches to identify the problem, but no real test strategy exists for debugging. The designer usually determines the process for obtaining a functional system.

The equipment used in the traditional test and debug process will vary depending on whether a whole system, a subsystem, or a few boards are being checked out. Types of equipment that are needed often include:

1. Hot mockup(s)
2. Hardware and software emulator(s)
3. Logic analyzer(s)
4. Oscilloscope(s)
5. Multimeter(s)
6. Specially designed debug box(es) (special test equipment)
7. Logic probe(s)

Depending on the design, the equipment cost may be very expensive and hard to justify for a low-volume system. User expertise and related training costs are other factors to consider.

A typical approach taken to verify/debug/test the hardware and software in a prototype system is:

1. Make a visual inspection of all the boards to check for any obvious problems; for example, wrong parts on the boards.
2. Do a continuity test of V_{CC} and GND to check for shorts. This always should be performed on each board before placing it in the system to avoid the possibility of damaging the whole system when power is applied.
3. At this point, there are many different options, depending on what the engineer decides is the best approach. One approach is:

- Run the hardware and software, making necessary patches (either hardware or software), to get around parts not currently available. If everything runs as intended, it is assumed that no problem exists. If not, there are many ways to proceed including:
  - Lower the hardware complexity by removing boards and patching around the boards.
  - Lower the software complexity by changing the software
  - Execute the software in a single-step mode and attempt to identify the problem source by using a logic analyzer and/or an oscilloscope.

After verification of all the hardware and software for one complete system, this system becomes a "hot mockup" to be used for testing/debugging other boards received from manufacturing (or returned from field use). Using the hot mockup, other boards may be verified by:

1. Replacing the good board in the system with the Unit Under Test (UUT)
2. Running the hot mockup software
3. If the hot mockup software runs correctly, the UUT will be considered good. If not, the engineer(s) may attempt to isolate the problem by probing the UUT while the software is running.

The hot mockup approach is very iterative and may require significant time to be devoted to the debug/test effort. Additionally, swapping boards can induce faults into the system other than those faults associated with the unit under test.

Approaches other than the hot mockup include:

- The use of special tester boxes for testing boards, functions, etc.
- The use of software and hardware emulators
- The use of special test equipment.

The description above shows that traditional methods usually do not involve a structured design verification/testing strategy. Concurrent verification of hardware and software makes it difficult to isolate between hardware and software faults. The increasing complexity and density of today's systems may require costly equipment for verification and test, and board "real estate" must be allocated for probing.

Boundary-Scan Test Method

The poor fault isolation and lengthy, unstructured approach of traditional test methods suggest that a new method is needed. A hierarchical test methodology based on test capabilities

such as boundary scan can solve some of these problems. A hierarchical test concept benefits both the designer and the test engineer because system design verification (hardware and software) and testing are accomplished using the same methods. Such a concept stems from having test capabilities, such as boundary-scannable devices, incorporated into the design that can be used at all levels of test (device, board, box, system). Debug software and test programs using these capabilities are reusable for each level of equipment integration. This is possible by using a building-block approach to test software development and execution. Test routines developed to exercise certain equipment functions can be built upon to exercise additional equipment functions. The routines used in hardware debug can be built upon to create functional or pattern-oriented tests for the equipment.

System Description

To study the benefits of using standard devices incorporating 1149.1 boundary scan as a tool for hierarchical test, a prototype system was designed. This prototype system is shown in Figure 1.

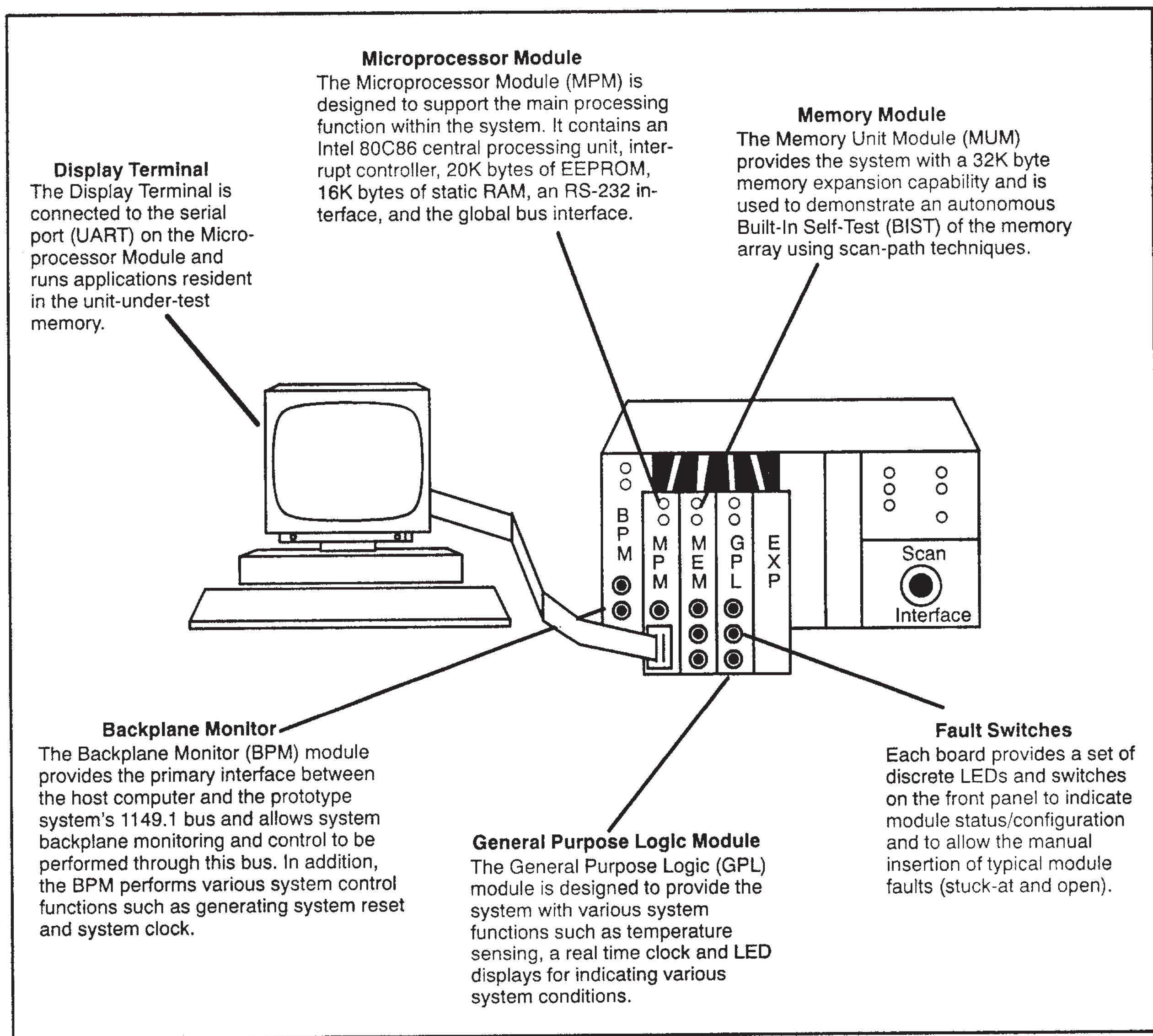

Figure 1. Prototype System

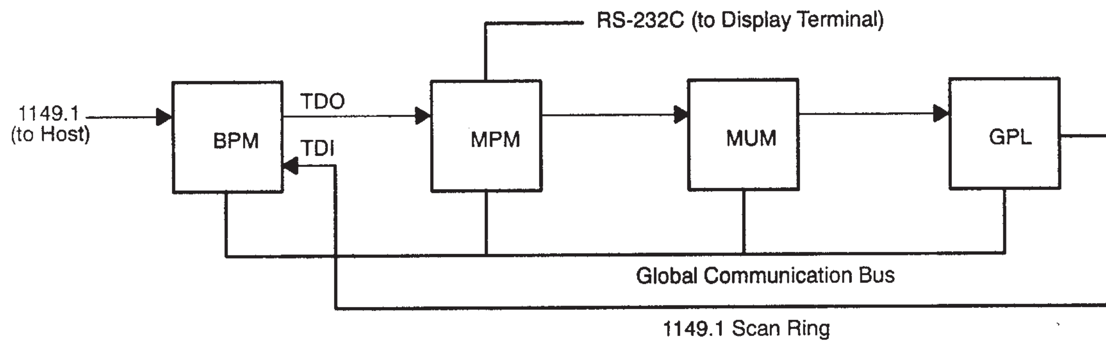

Figure 2. Prototype System Block Diagram

The system consists of a single chassis containing four printed wiring boards (PWBs) and a power supply. The PWBs are the Microprocessor Module (MPM), the Memory Unit Module (MUM), the General-Purpose Logic (GPL) module, and the Backplane Monitor (BPM) module. Two system interfaces are provided: an RS-232C interface for communication with the user via a display terminal and the four-wire 1149.1 interface for providing access to the system's internal scan paths from an external host computer. The four PWBs are interconnected via a global communications bus. This bus is a derivation of the MPM's internal address/data bus. The 1149.1 bus also interconnects the PWBs within the system in a scan-path ring configuration. Figure 2 shows a block diagram of this basic system architecture.

During the system hardware design, boundary scan was included as a key design element. Scannable buffers and latches within TI's System Controllability, Observability, and Partitioning Environment (SCOPE) product family were used to partition the design into subfunctions that could be easily verified. These devices incorporate the 1149.1 architecture and provide several additional test capabilities as summarized in Table 1.[3,4,5]

Table 1. SCOPE Device Test Functions

Device Mode	Test Function
Test or Normal	ID Register Scan
Test or Normal	Sample Boundary
Test or Normal	Bypass Scan Mode
Test or Normal	Boundary Control Register Scan
Normal	Boundary Read
Normal	Boundary Self-Test
Test	Boundary Read
Test	Boundary Toggle Outputs
Test	Boundary Scan
Test	Control Boundary to High-Impedance
Test	Control Boundary to 1/0
Test	Boundary Toggle Outputs
Test	16-Bit Pseudo-Random Pattern Generation (PRPG) on Outputs
Test	16-Bit Parallel Signature Analysis (PSA) on Inputs
Test	Simultaneous 8-Bit PSA and PRPG

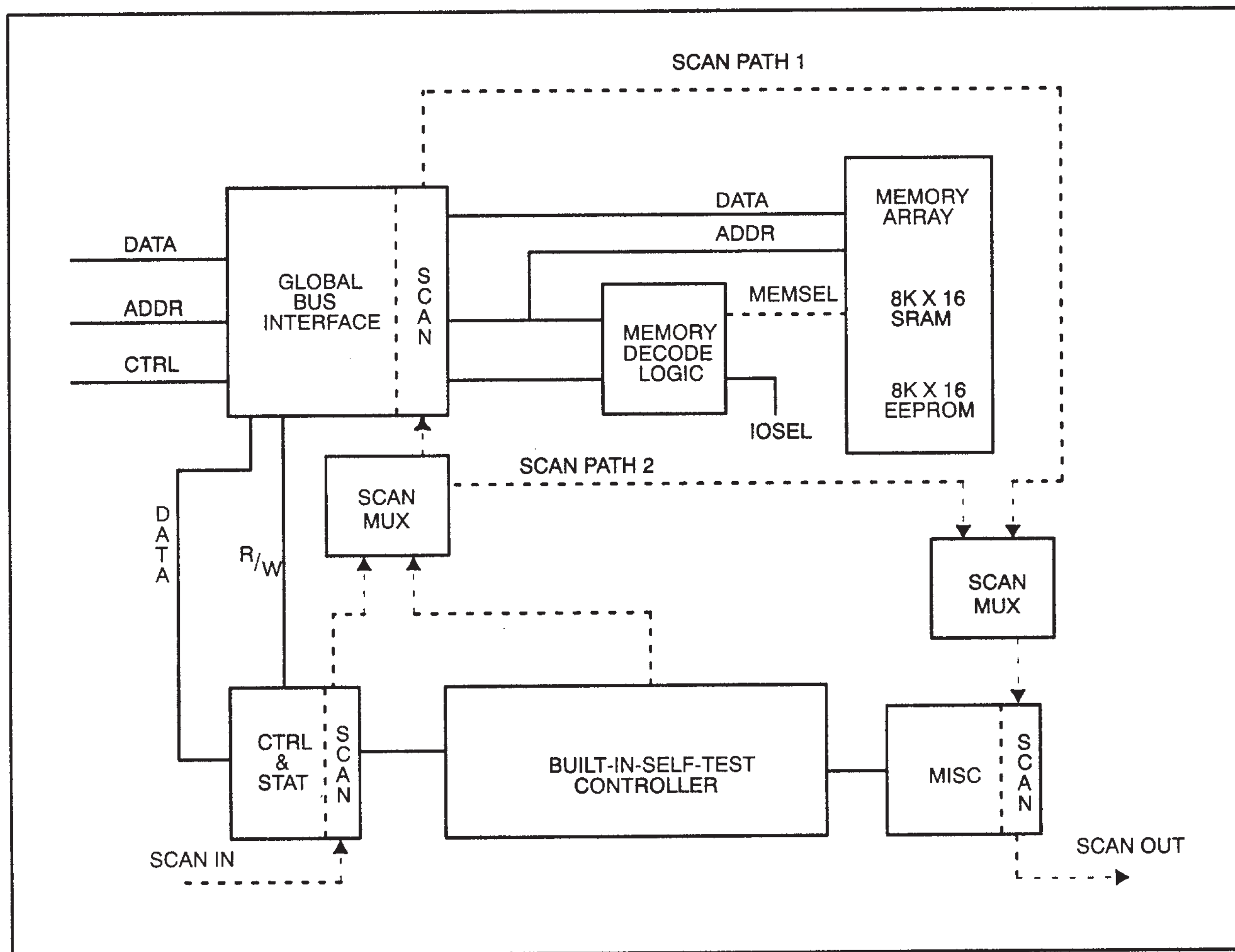

Figure 3. MUM Block Diagram

The SCOPE octal implementation in the MUM is shown in Figure 3. SCOPE octal devices are substituted for conventional octal devices such as buffers and latches and were placed strategically on data, address, and control signals. Thus, the MUM is divided into four distinct subfunctions, each individually accessible through the module's 1149.1 interface. These subfunctions are the global bus interface, the Built-In Self-Test (BIST) controller, the memory decoding logic, and the memory array [Static Random-Access Memory (SRAM) and Electrically Erasable Programmable Read-Only Memory (EEPROM)].

The target verification/test/debug environment consisted of TI's Advanced Support System for Emulation and Test (ASSET). ASSET is a test software development/execution environment hosted on an IBM PC-AT™ compatible computer. A Scan Controller Module (SCM) plugs into an expansion slot of the host computer and provides the ASSET software with an interface to the target system's 1149.1 bus. ASSET software is an extension of the C programming language to support boundary scan. In addition, ASSET software provides a library of low-level functions for controlling 1149.1-based boundary-scan devices and an interactive debugging utility that allows the user to manipulate each of those devices. The library functions are used to control the SCOPE device test capabilities described in Table 1.[6]

Methodology Employed

The goal of the prototype system development was to demonstrate the test and debug capabilities provided by the SCOPE octals when supported in the ASSET test environment. Each board design engineer generated their own board debug routines using ASSET and use the boundary-scan interface as much as possible during board debug. Traditional test equipment was to be limited to an ohmmeter for performing continuity checks before applying power to the board. The in-

tention was to generate standard debug/test routines that could be used repeatedly to verify/test subsequent boards as they were received from manufacturing or when a fault was detected during normal operation.

Both board designers and test engineers were involved in the definition of ASSET debug routines for the boards. These debug routines were written to support the development of either functional tests or pattern-oriented tests for the hardware. Examples of some debug routines developed for the MUM are shown in Table 2.

As a board came in, visual inspection and continuity checks of the board were performed. The board then was placed in the system chassis for debugging. Using ASSET, it was possible to debug each board independently without requiring other boards in the chassis for support. The ASSET interactive debug routines were executed allowing various board functions to be controlled and observed. Functional and pattern oriented tests were developed by building upon certain basic debug routines to allow larger board functions to be verified. This building block approach was used throughout the debugging process to develop tests for the hardware. Once a series of functional and/or pattern-oriented tests had been developed and solidified, the tests were compiled using ASSET to form a production test that could be used to verify incoming boards. One such example for the MUM board is shown in Table 3.

Table 2. Memory Module Interactive Debug Routines

Routine
Initialize Memory With a Pattern
Backplane Read/Write to MUM via BPM
Read MUM Global Bus Interface Boundary
Read/Write a Memory Location
Read/Write a Block of Memory Locatiaons
Copy One Memory Location to Another
Swap the Contents of Two Memory Locations
Read/Write Any I/O Register
Read Memory Control Signals

Table 3. Sequence of Memory Module Routines Used to Form Board-Level Test

Routine
Scan-Path Continuity Test
BPM to MUM I/O Continuity Test
MUM to BPM I/O Continuity Test
Board Identification Test
Chip Select/Decode Logic Test
Memory Tests
Address Uniqueness Test
March II Algorithm Test

Lessons Learned

Through development of the prototype system, we were able to discover the major benefits of using boundary scan versus using traditional methods for design verification, debug, and test. We also learned several design and test practices that were beneficial to our effort. Most of the lessons learned fall into one of the following categories: fault isolation, design partitioning and test access, ease of use, scan-path design, and structured process. The following paragraphs discuss these categories and specific lessons, design rules, and observations that were made.

- Fault Isolation (Traditional Versus Boundary Scan)

  As the first set of boards was being debugged, ASSET was still under development. Code syntax and format changes, along with compiler changes, forced traditional methods to be used in some cases. Because of this, the MPM designer decided to use traditional methods in debugging his board, which slowed the effort considerably. While using traditional methods, the designer was unable to isolate a code execution problem. He eventually resorted to using ASSET after several days of tracking the problem without success. Using the boundary scan within the module, the memory was loaded and verified both directly (without using decode logic) and indirectly (using decode logic). This allowed the problem to be isolated to a design error within the memory decode logic.

  Another example of how ASSET and boundary scan were successful in isolating various system problems was in debugging of the MUM board. A miswired bus-enable signal was creating bus contention on data lines. ASSET helped isolate the problem quickly, without requiring any manual probing of the board.

  In addition to the design errors that were isolated, several component and signal failures also were isolated. Switches purposely were designed into each module to allow stuck-at and open faults to be simulated. The types of faults easily isolatable through the boundary-scan method included backplane faults, memory decode faults, internal bus faults, and scan-path faults. These types of faults are typically very difficult to isolate with traditional methods. The scan-path fault was isolated through the use of the preload feature built into scan instruction register command of the SCOPE octals, as defined by 1149.1.

- Design Partitioning and Test Access

  Experimentation using the fault insertion switches reinforced some important guidelines regarding boundary-scan placement and design partitioning. It is very important that boundary scan be implemented at functional partitions, especially the board-to-board interface, to allow functions to be exercised and isolated completely. Also, boundary scan access to key control signals, such as the microprocessor HOLD signal, is important to avoid bus contention when attempting to drive buses with boundary-scan registers.

  By providing controllability and observability of each board's backplane signals through boundary scannable devices, each board could be tested independently as it arrived from manufacturing. Emulation of board interfaces, whenever required, was controlled through the backplane boundary scan. Using this method, board-to-board dependencies were eliminated, and each could be verified standalone. This was very beneficial during the integration phase, since the system could be debugged without requiring all boards in place, thereby reducing the ambiguity size whenever a problem was detected. The MUM and GPL boards were verified completely using ASSET and the boundary-scan interface. Once two boards were debugged, a system backplane test then could be performed between boards using ASSET. This allowed the backplane interconnects between two boards to be verified using a pattern-specific test, as opposed to implicitly testing them through interaction between the MPM and either of these modules. In this way, faults caused by the backplane wiring or connector seating were detected and isolated easily.

- Ease of Use (ASSET)

  Using ASSET to debug/test each board proved to be less time-consuming than taking the traditional approach. For example, the designer of the GPL was able to write the ASSET debug routines and complete debug of his board in a few days with no experience in writing ASSET code or using the system. Once the GPL tests had been compiled into a production test for the board, the remaining GPL boards were verified quickly, without resorting to any of the traditional methods.

  Both test engineers and software engineers also were involved in the use of ASSET to debug the system and create production tests.

- Scan-Path Design

  While there are obvious benefits to using boundary scan within a design, there are also problems that can be created if it is implemented incorrectly. The instances in which this occurred during the design/development of the prototype system dealt with scan clock control and changing of scan-

path lengths. These problems can be eliminated through careful adherence to scan-path design rules.

Some BIST designs, such as the one implemented on the MUM, may require explicit clock control over independent scan paths and, therefore, mandate that scan clocks be gated. Careful design should be used when such gating is necessary. During debugging of the MUM, this gating caused several problems when attempting to use scan paths under control of the MUM BIST. When inactive, the BIST disabled certain scan clocks, and all scan data beyond this point were made inaccessible. The problem would have required manual probing to isolate had it not been for the 1149.1 instruction register scan preload of data. When commanded to do an instruction register scan, the SCOPE octals preload a binary 10 on the two least-significant bits of the scan data output.[7] Those octals' scan paths that are obstructed in some way, however, will output all zeros (0s) or all ones (1s), depending on the fault. By analyzing the information retrieved during a single instruction register scan, the point at which the scan data were corrupted can be determined.

While the 1149.1 specification allows scan clocks to be disabled, it does not encompass the system design considerations necessary to ensure the operation of a device's additional test capabilities, such as those available in the SCOPE octals. A system design problem of this type was discovered when developing backplane tests that use the SCOPE PRPG/PSA modes. The BPM backplane scan path is implemented as a separate scan ring from the other PWBs within the system (MPM, MUM, GPL). The two separate rings are connected in parallel with each interfacing to the host through the same scan-path signals. Because of this architecture, circuitry was added to ensure that only one of the rings could be active (scan clock enabled) at any given time. As a result, backplane PRPG/PSA tests between modules on separate scan rings could not be performed; for example, a test between the BPM and MPM could not be performed. This forced the development of two separate backplane tests: one using the BPM (discrete patterns) and the other using only PWBs on the secondary ring (free-running PRPG/PSA). To avoid this problem, the same free-running clock must be provided to each ring if backplane tests using SCOPE PRPG/PSA modes are to be performed between boards on different rings.

Another problem encountered concerns the collapsing and expanding of scan rings within a design. For example, while designing the MUM BIST, it was found that, while collapsing and expanding of scan rings is acceptable, it should not be performed arbitrarily with respect to the test bus controller. Changing the scan-path length without notifying the test bus controller will cause the controlling software to become confused and possibly not recover. All scan-path length changes should be done in conjunction with the test bus controller to be as graceful as possible.

- Structured Process

One of the most important discoveries during development of the prototype system was the fact that the use of ASSET and boundary scan imposed a structured test and debug process for the engineer. As mentioned earlier, a great portion of the MPM was debugged using lengthy traditional methods. The process did not benefit the debug/test of the second and third MPMs when they arrived. Since the design engineer chose not to write ASSET debug routines, even after ASSET had become stable, only low-level ASSET library functions were available to manipulate the board's boundary-scan devices. Because the majority of the time was spent using traditional debug methods, no ASSET debug routines or MPM production test could be developed within the given timeframe.

In contrast, the engineers who used ASSET for design verification/debug/test of the MUM and GPL followed a structured procedure that was repeatable on subsequent boards. This procedure uses a building-block approach whereby individual pieces of the system are verified and then may be used to verify the remaining pieces. The procedure is shown in Figure 4 and can be divided into the following four phases:

- Hardware design
- Debug/test design
- Debug/test development and execution
- Test compilation

The hardware design phase consists of implementing boundary-scan devices into the design to partition it into easily testable functions.

During the debug/test design phase, a debug strategy for each board (and subfunctions) is defined and the low-level ASSET functions required to implement the debug routines are identified.

After the debug routines are defined, ASSET software may be written and executed on the designs. The debug routines may be combined to form test routines that, in turn, can be combined to form the board-level production test.

Once a production test exists for each board, they may be combined with a system backplane test to form a production test for the system.

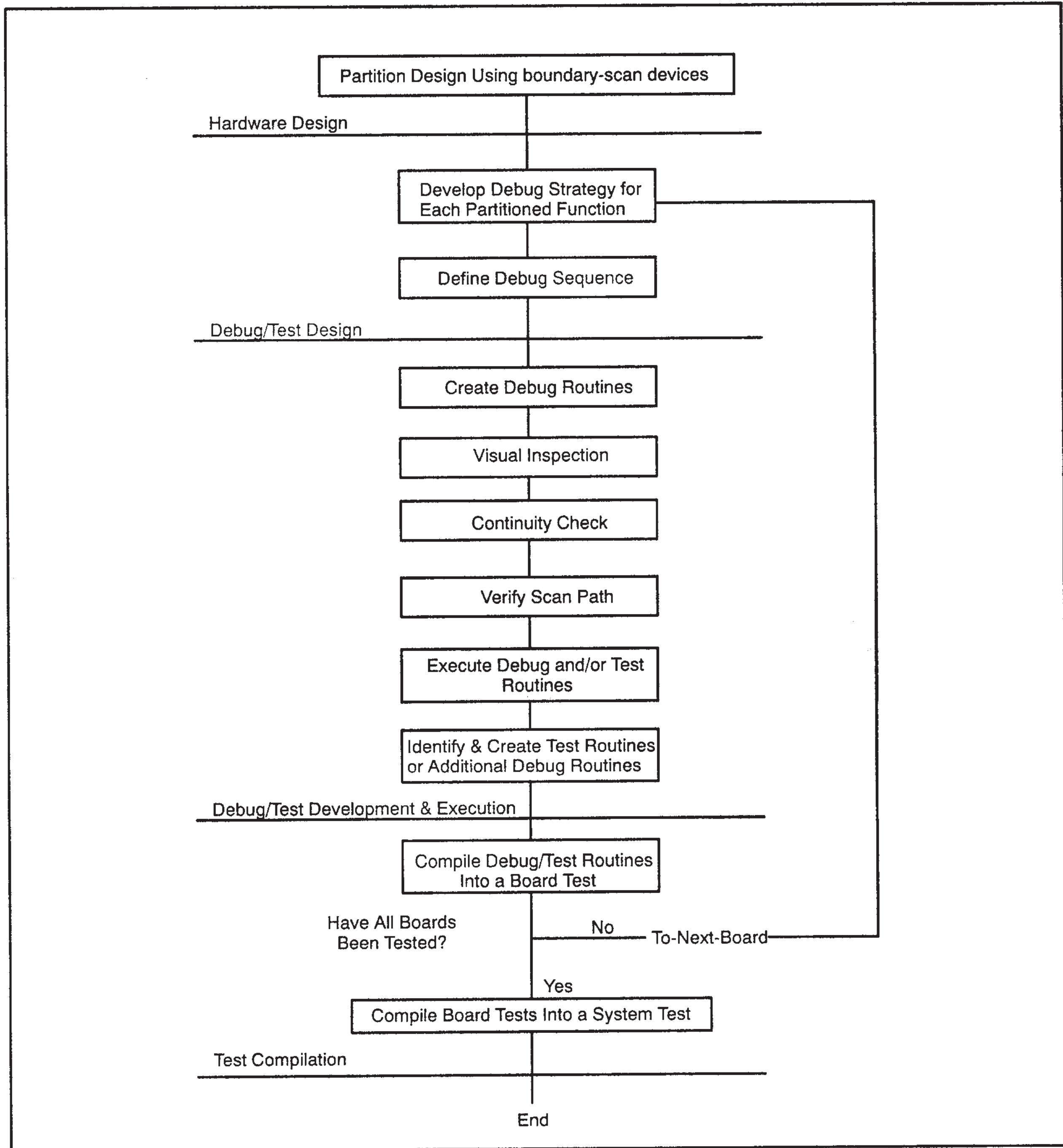

Figure 4. Structured Debug/Test Procedure

Benefits of Boundary-Scan Methods Versus Traditional Methods

The key benefits of boundary-scan methods versus traditional methods for design verification/debug/test are shown in Table 4 and discussed in the following paragraphs.

Incorporating boundary scan into prototype designs will give the engineers a structured approach to design verification, debugging and testing of prototype systems, which typically does not exist using traditional methods.

As the complexity and density of designs increase, the need for more equipment will be required when using traditional methods. The need for different types of equipment was shown to decrease when boundary scan was implemented into the design. The equipment required in debug and test of the prototype system described above included:

1. SCOPE octals incorporated into the designs
2. An IBM PC-AT compatible machine with a scan controller module and ASSET software
3. An ohmmeter.

In traditional methods, the software generated by engineers during design verification and debug usually is discarded after verification of the system. If the software is not discarded, it rarely can be used for board and system testing. In contrast, the hierarchical test method used on the prototype system allowed the software generated for board-level debug to be reused to generate board-level and system-level production tests.

The controllability and observability (C/O) in traditional debug and test methods usually requires the attachment of emulators, logic analyzers, and/or oscilloscopes to the UUT. In the boundary-scan method used on the prototype system, the C/O was achieved from the SCOPE octals. With this increased internal visibility, fault isolation and system software debug are accomplished more easily.

The use of the SCOPE octals, in the boundary-scan approach, required the use of four I/O pins and a slightly larger IC footprint on the boards. In the traditional approach, the amount of space used is dependent on the ad-hoc testability added to the designs. Space also needs to be allocated such that the boards can be probed.

Table 4. Differences in Traditional and Boundary-Scan Methods

Tradeoffs	Traditional	Boundary Scan
Approaches	Usually not structured	Structured approach
	Usually starts after design has gone to manufacturing	Starts during design
Test Equipment	Complexity/density of designs require more test equipment to test and debug systems	Limited equipment needed to test and debug systems. No need for expensive special test equipment during production test, depot test, etc...
	All "Hands On"	"Hands On" limited
	Hot mockup, logic analyzer, oscilloscope, ...	PC-AT software, ohmmeter test octals
Software	Tests generally not reusable	Reusable test software go from debug—board—production ...
Internal Visibility	Fault isolation depends on ad-hoc testability	Fault isolation increased
	Manual probing required	No manual probing required
	Required external hardware to look at internal nodes	Have internal node visibility
Real Estate	Additional real estate required to support ad-hoc testability and space to allow for probing	Additional real estate, four I/O pins, and a larger footprint IC

Conclusion

The increasing complexity of designs, the use of double-sided, surface-mounted boards, the spacing of components decreasing, etc., will warrant a new design verification and test approach. Devices incorporating boundary scan impose a minimal real estate overhead and change the process of design verification and test, which make it beneficial to both the design engineer and test engineer. The use of devices incorporating boundary scan will reduce the cost of test.

By using the devices that support the 1149.1 architecture in our prototype system, some of the problems and questions associated with the verification and testing of prototype systems (or even production systems) were solved. In addition to solving the problems, the verification and test process was simplified.

References

[1] *Proposed IEEE 1149.1, Standard Test Access Port and Boundary-Scan Architecture*, Draft D3, January 18, 1989.

[2] Pete Fleming. *An Advanced Test Architecture—What's in it for You?,* Texas Instruments Technical Journal, July-August 1988, pp. 17–27.

[3] Lee Whetsel. *A Standard Test Bus and boundary-scan architecture,"* Texas Instruments Technical Journal, July-August 1988, pp. 48–59.

[4] Lee Whetsel. *Standard Test Port and Cells Provide An ASIC Testability Toolkit,* Texas Instruments Technical Journal, July-August 1988, pp. 35–43.

[5] Lee Whetsel. *A Proposed Standard Test Bus and boundary-scan architecture*, Proceeding of the International Conference on Computer Design, October 3–5, 1989, pp. 330–333.

[6] Don McClean. *Making the Overhead of Scan Transparent (Almost!)*, Texas Instruments Technical Journal, July-August 1988, pp. 145–151.

[7] Don McClean and Javier Romeu. *Design for Testability With JTAG Test Methods*, Electronic Design, June 8, 1989.

[8] *Joint Test Action Group Specification*, Version 2.0.