

On the synergy of boundary scan and emulation board test: a case study

Jim Webster (BAE Systems), Billy Fenton (International Test Technologies), Dave Stringer (ASSET InterTech), Ben Bennetts (Bennetts Associates)

Abstract

Boundary Scan is a structural test method, whereas CPU emulation is a functional test method. Functional tests, although difficult to generate, will cover some (usually unknown) structural defects whereas structural tests are unlikely to cover any functional misbehaviours. Discussions between vendors of boundary-scan and emulation products and a system user showed a potential synergy between the boundary scan and CPU emulations approaches. That is, use boundary scan to structurally test a board, and then use CPU emulation to structurally test those parts not covered by boundary scan, and to implement a basic functional test of the board.

1. Introduction

In the past, boundary-scan products have sold primarily into the prototype board debug market whereas emulation products have sold emulation systems into field-service environments to assist in the debug and diagnosis of faulty microprocessor boards. Recently, some emulation users have begun to use emulation for the functional testing of loaded boards before they enter into customer service i.e. after production volume structural test, either as a free-standing board or as a part of a system (collection of boards in a rack). This innovative use of emulation moves it back into the production test environment and effectively backs up to where the boundary-scan usage stops.

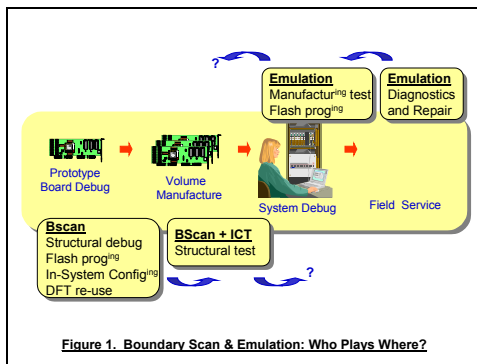


Figure 1. Boundary Scan & Emulation: Who Plays Where?

The paper describes the results of a collaboration between a military systems user, an emulation supplier and a boundary-scan tester supplier and presents a case study showing the synergy between boundary scan and emulation test.

This paper is organised as follows:

- Section 2 is a short technology backgrounder on the meaning and use of emulation techniques. The reader is assumed to be familiar with the IEEE 1149.1 Boundary Scan Standard [1]
- Section 3 introduces the case study problem
- Section 4 discusses the solution based on a combined boundary-scan and emulation test environment
- Section 5 concludes the paper and comments on future work.

2. Basics of Emulation Testing

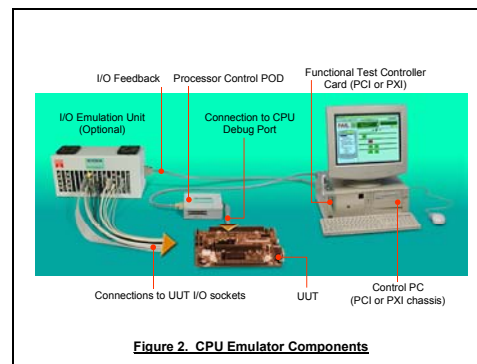


Figure 2. CPU Emulator Components

Emulation testing is used with bus-based boards. The emulator typically replaces some part of the on-board circuitry, and takes control of the board via one of its buses. Once in control, the emulator can load and run tests and diagnostics on the UUT. Types of emulation include: Processor, ROM and Bus

A processor emulator replaces the board's processor, and gives full control over all the processor buses. Once in control, full

read/write access to all parts of the UUT's memory and I/O is available via the emulator. It provides the ideal method of test access for processor boards. However, processor emulation is not viable on higher speed processors (>20-30MHz), and the real processor must be socketed, so that the emulator can easily replace it.

A ROM emulator replaces the boot ROM, and substitutes diagnostic code for the processors normal boot code. In addition, a ROM emulator has a bi-directional communications pathway, so that the UUT can communicate with the tester. Its primary shortcoming is its inability to diagnose the processor to boot ROM area. In addition, it can be difficult to use with soldered-in ROMs unless some circuit modifications are made at the product design stage.

A bus emulator connects to a bus slot or edge connect, and provides the ability to perform read/write bus cycles on the UUT, thus providing test access to the various UUT circuits and functions. It's a good solution for testing plug-in bus cards (e.g. VME, PCI, etc.).

Emulation was a popular approach in the 80's and early 90's; its decline was driven by the absence of socketed processors and ROMs in new designs, and by increasing processor speeds, which made emulators difficult or impossible to design. However, with the advent of processor debug interfaces, there is now a renaissance of this powerful approach to functional test and diagnosis.

2.1 What is a Microprocessor/DSP Debug Interface?

Designers require test tools, such as processor emulators and logic analyzers, for debugging both hardware and software. However, it is difficult to design emulators for high speed processors (>20MHz), and most modern processors operate at frequencies far in excess of this. Processor vendors realized this and solved the problem by incorporating the required debug functions on the processor die. In essence, a 'processor emulator' is now inside the processor!

Typically, the debug functions incorporated include the following low-level functions:

- Stop the Processor
- I/O Read/Write
- Memory Read/Write
- Breakpoints
- Single Step Code
- Code Trace

Access to the debug functions is typically via the processors 1149.1 Test Access Port (TAP), sometimes known as an extended-JTAG (eJTAG) port, plus 2-3 special purpose signals. Processor designers simply extend the 1149.1 instruction set to include vendor-specific instructions for controlling the processor core.

How can the debug interface be used for manufacturing test and diagnosis? The debug interface allows the tester to take control of the processor, and then the processor is simply used as a 'vehicle' for testing the rest of the UUT. For example, the debug interfaces read/write functions can be used to perform memory testing.

2.2 Some Debug Interface Examples

Many modern microprocessors and DSPs now include a debug interface. Some examples are:

- Intel's Pentium Family
- Motorola's and IBM's PowerPC Family
- ARM [2]
- MIPs
- SPARC
- Motorola DSP
- AMD Elan [3]

Additionally, the IEEE recently published a standard (IEEE-ISTO 5001) for debug interfaces [4].

2.3 Using a Debug Interface for Functional Test

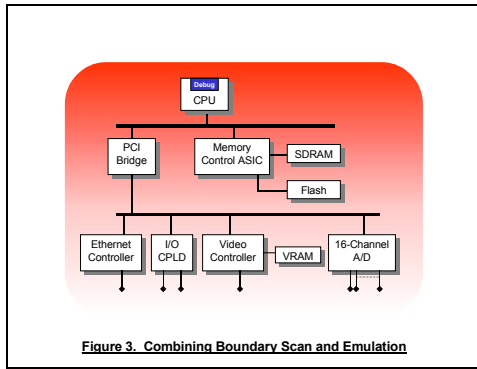


Fig. 3 shows an example processor board. The 'Debug Interface' Tester connects to the processors debug port. This connection requires about 5-8 test points on the UUT. The tester takes control of the processor before it runs any code, using the debug interface. Once in control of the processor, the tester simply uses the processor as a 'vehicle' for testing the rest of the board. Typically, the tester can perform the following low-level functions via the debug interface:

- Start/stop the processor
- Read/Write Memory
- Read/Write I/O
- Breakpoints
- Download test code to the UUT memory

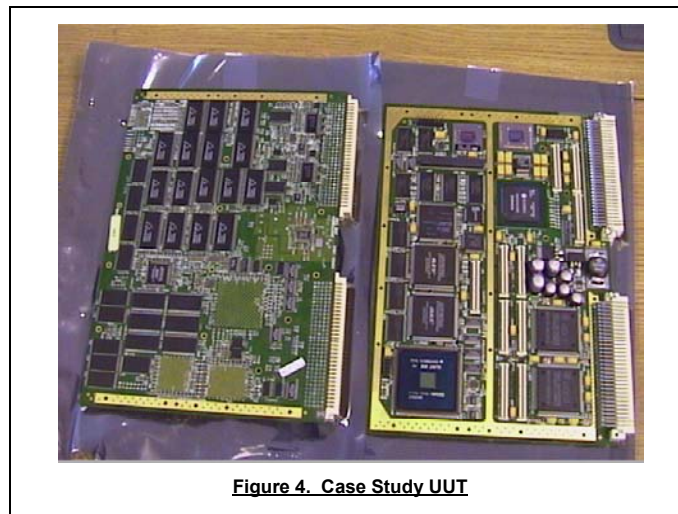
- Control execution of test code in the UUT memory
- Collect test results from UUT memory

A UUT can be divided into functional blocks. For example, the UUT in Fig. 3 contains the following functional blocks: processor, PCI bridge, SDRAM, video controller, etc. Each functional block contains arrays of memory or I/O registers. These registers control the operation of the functional blocks, and are accessed using the debug interface low-level functions. Therefore the debug interface low-level functions are used to build up a test program for each functional block, and these test programs can then be sequenced to create a complete UUT test.

3. Case study: statement of the problem

The case study centres on the difficulties associated with the test of the Motorola PowerPC circuit board where the need to test beyond the boundary scan interconnections becomes a real requirement.

The following picture shows both sides of the evaluation board. There is a MPC603e PowerPC, with an MPC106 PCI bridge and a CA91C142 VME bridge device as well. The only other scanable ICs are 3 Altera programmable logic devices



Currently the only test methods in place are Boundary Scan and functional in-system test. The latter has many

drawbacks not least of which is that the PCB is required to be treated with a conformal coating and this makes test

probing very difficult since this coating contaminates any test probes. The other option that was available to us was to generate a loadable set of test routines into Flash and then either remove them or make them only available under specific test conditions, that could not be entered into during normal operation. Additionally testing under these conditions has some major drawbacks:

- Time consuming
- High degree of skill for the debug engineer
- Needs a system around the PCB – an expensive test harness!
- Knowledge tends to disappear in the long product life cycle, making field

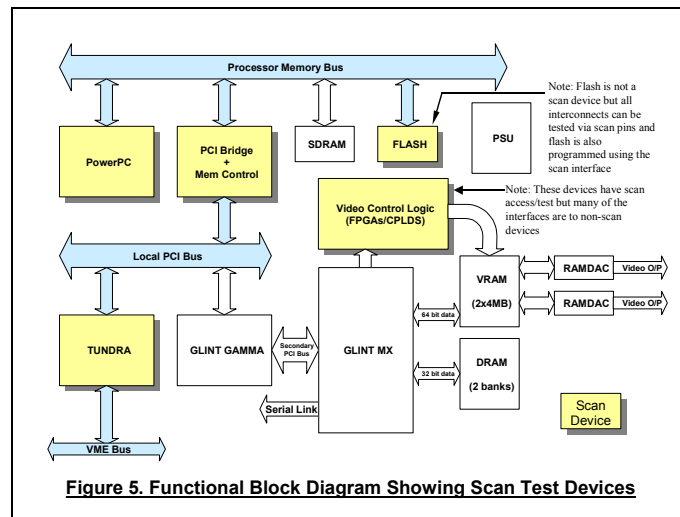
returns expensive to diagnose and repair

- Lack of physical access makes first pass fault isolation to single device level difficult

3.1 Use of Boundary Scan

As stated earlier, Boundary Scan is currently used as the test method for this unit, but there are many areas of the circuit that boundary scan does not reach

- digital domain (those where no scan devices available)
- digital-to-analogue and analogue-to-digital domains
- analogue domain



From this it is clearly shown that there is a limit to the test coverage offered by Boundary Scan and this can be further highlighted by showing actual devices in the scan path. I have included flash as part of the scan test since there are now exhaustive scan test capabilities for flash devices.

We have also used the Boundary Scan to test the VME Buffer Interface, which although has no scan devices, can by

simple “wrapback” connections plugged into the main edge connectors, allow a scan controlled test to drive out through the bi-directional buffers to be read back into a scan device. This is done for both directions and covers the 32 bit data and address busses of the VME Interface.

Boundary Scan is also used to program the 3 programmable logic devices, test Flash memory and program the processor bootROM.

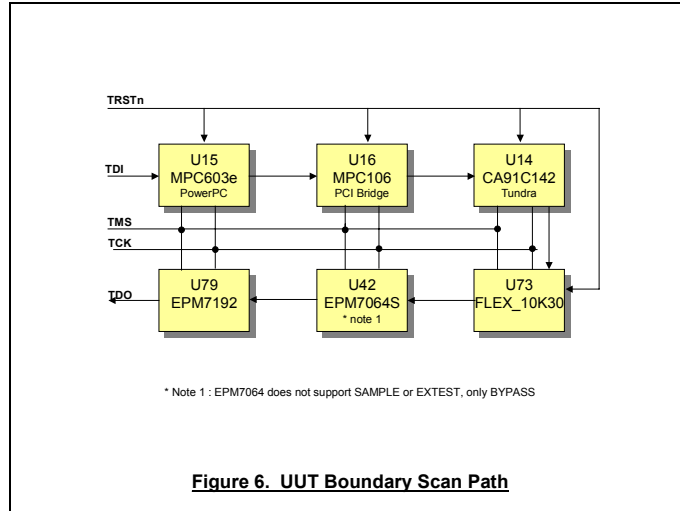


Figure 6. UUT Boundary Scan Path

3.2 Use of Emulation

The prime considerations for investigating and adding emulation testing were to extend the testing into those areas of the circuitry that crossed the border from pure digital to digital/analogue and pure analogue. In addition emulation testing potentially offers to use the on-board processor and test

- fast test time, 1-2 minutes (yes - this is fast for military applications)
- accurate diagnostics
- at speed performance testing for
- SDRAM
- VRAM

The MPC603e also utilises the 1149.1 interface as part of a Common On-chip Processor (COP) interface that allows processor and emulator testing.

- Graphics Processors
- RAMDACs
- analogue video outputs
- other functional control blocks such as timers, memory mapping

However there are 2 Graphics control devices which have no 1149.1 support but have considerable digital interfaces as shown in the following block diagram.

Emulation should offer

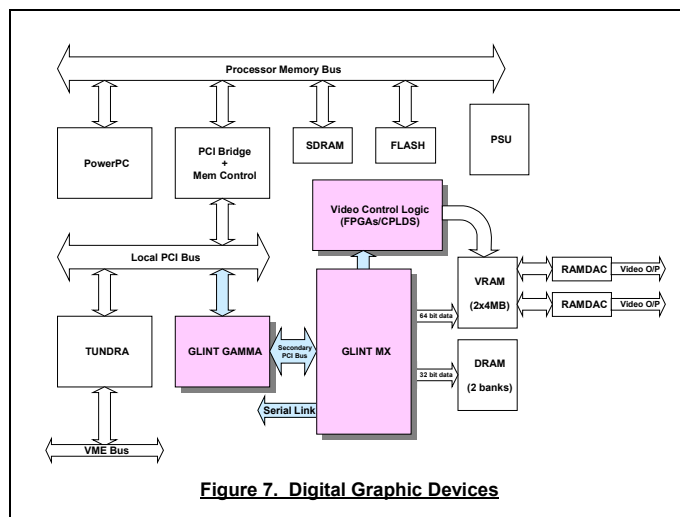
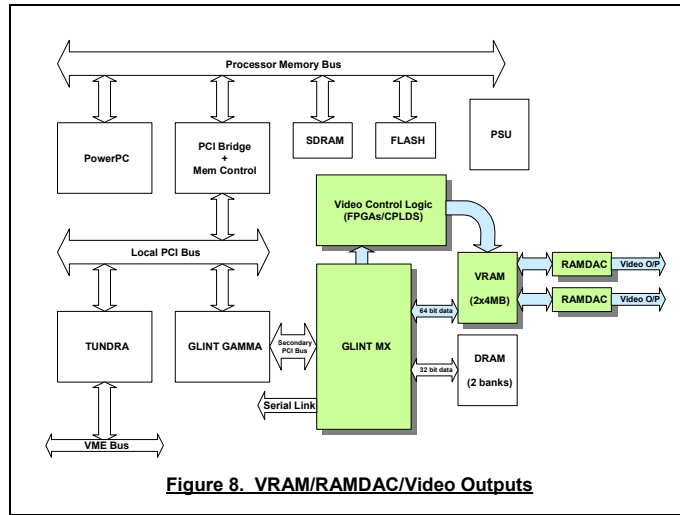


Figure 7. Digital Graphic Devices

Beyond these graphics devices there is yet more digital circuitry, as well as a

RAMDAC converter taking us into the

analogue domain where video signals are conditioned and output.



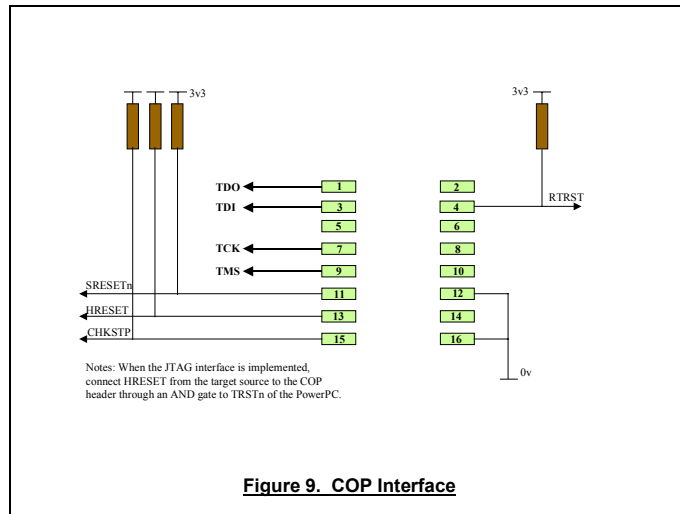
3.3 COP Interface [5]

The COP function of the PowerPC processors allows a remote computer system, typically a PC, to access and control the internal operations of the processor. The COP interface connects primarily through the JTAG port of the processor with some additional status monitoring signals. In order to fully control the processor, the COP port requires the

ability to independently assert HRESETn or TRSTn.

A COP header has many benefits – breakpoints, watchpoints, register and memory examination/modification, and other standard debugger features are possible through this interface.

The arrangement of the COP interface, as recommended by Motorola, is shown in the following figure.

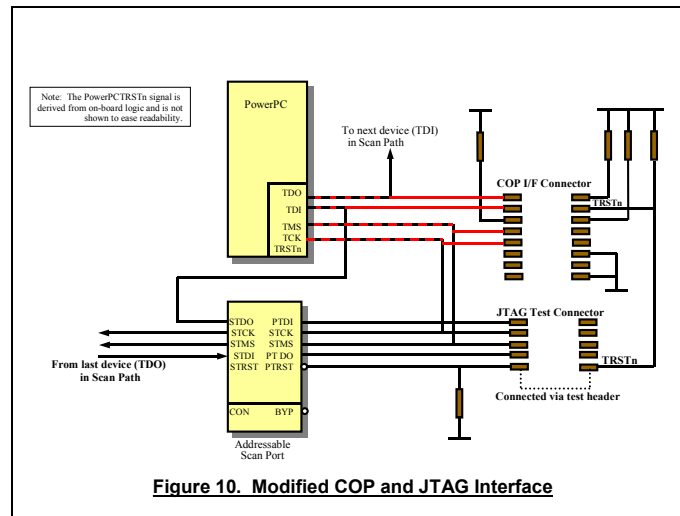


Notes: When the JTAG interface is implemented, connect HRESET from the target source to the COP header through an AND gate to TRSTn of the PowerPC.

This header can also be used as a JTAG test interface for the PowerPC, although in today's complex boards there is a need to include the PowerPC into the standard

boundary scan chain, thus permitting interconnection test to be implemented at manufacturing without the need for separating/dividing scan chains.

Figure 10 shows how we have combined the COP and JTAG interfaces.



The COP header allows full access, by the emulator, direct to the PowerPC and all recommended connections are catered for.

The JTAG Interface, now using an Addressable Scan Port (ASP) brings the PowerPC into the scan path for interconnect testing. It is important to note that TMS and TCK are routed to the PowerPC from the input of the ASP and that TRSTn is connected via a link in the JTAG header. TMS/TCK being provided from the input of the ASP ensures that other scan devices are not under emulator controlled signals. The TRSTn link is necessary to allow Boundary Scan to control TRSTn when in scan testing but allow TRSTn to be included in the reset circuitry of the PowerPC for emulation testing.

3.4 Additional Emulation Requirements

In addition to the functional requirements of the emulation there are other constraints that require to be applied

- Must be a stand-alone solution. These PCB's will have a long product life cycle and the maintenance and repair of these are a major consideration. This would also make field repairs a much simpler undertaking.
- Minimum external hardware interfaces. This must be the case in order to reduce the reliance of the system level diagnostics which can be expensive to maintain, if testing

requires other system PCBs, motherboard and its Power supply.

- No Probing. These PCBs have a conformal coating and it is impractical to remove this for a test & repair operation – since the conformal coating would then need to be reapplied on completion of testing
- Re-use wherever possible. In an ideal world it would be nice to re-use as much of the emulation tests as possible as the product design moves forward.

In line with the overall test strategy of “fixtureless” testing and the long life cycle of our product range these constraints are necessary. It has to be considered that these PCBs have conformal coating

4 Case study: solution

4.1 Introduction

Boundary scan provides a comprehensive, but incomplete, structural test of the case study board. A COP-based emulator can share the same test port as the boundary scan tool, and extends the boundary scan test in a number of significant ways, namely:

- Structurally test non-boundary scan parts, which are CPU accessible.
- Provide a functional test of both boundary and non-boundary scan parts
- Provide an at-speed test.

The COP port essentially allows an external instrument or emulator to control

the operation of a PowerPC microprocessor. At reset the processor is stopped by the emulator and CPU instructions, such as memory read/write operations, can be issued from the emulator. Essentially, the emulator provides the processor boot code. Using these instructions, a test program can be constructed for all parts of the board that are CPU accessible. Combined together, a full sequence to both structurally and functionally test the board is made available.

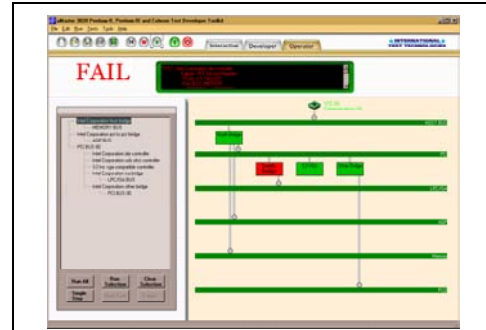


Figure 11. CPU Emulator GUI with Easy-To-Use Block Diagram Interface

4.2 Case Study Test Sequence

The sequence shown in Table 1 illustrates a COP-based test sequence for the case study board. All parts are both structurally and functionally tested.

This test sequence primarily consists of a series of CPU read and write operations to the boards various control ICs. The writes configure each component for normal operation and stimulate them into their different modes of operation, and the reads check for the appropriate responses. Alternatively, responses can come from an operator (e.g. video screen), or via another instruments such as a DMM attached to an I/O or bus connector.

The test program is embedded in the emulator not the board, and is typically written using standard programming languages such as C. The GUI can either be user written, or a standard test executives can be easily integrated. This means that changes can easily be made during the life-cycle of the product, which is often not the case for Built-In Test (BIT), which may be dependent on the designers for changes. Additionally, test executives are designed for technician or operator usage, so the training requirement is greatly reduced, as shown in Figure 11.

Board Part	Test Sequence	Comments
PowerPC 603R	Take control of CPU	Use COP interface to take control of CPU
MPC106 Bridge	Memory Controller Register Test	Use CPU read/writes to verify buses and communications from CPU to MPC106
	Configure Memory Controller for Normal Operation	Use CPU writes to configure MPC106 registers
	PCI Bridge Register Test	Use CPU read/writes to verify buses and communications from CPU to MPC106
	Configure PCI Bridge for Normal Operation	Use CPU writes to configure MPC106 registers
	Interrupt Controller Register Test	Verify CPU access to Interrupt Controller using CPU read/write instructions
	System Timer Test	Use CPU read/write instructions to start and verify timer operation
Flash Memory Test (x1) 8bit – Boot Flash	Bus Test using ROMBus Test algorithm	Use statistical algorithm to read ROM contents and verify and diagnose main buses.
	ID Test	Read back flash ID using CPU read/write instructions.
	ROM Checksum	Use CPU read instructions to calculate flash checksum and verify contents

Flash Memory Test (x8) 64bit	Bus Test using ROMBus Test algorithm	Use statistical algorithm to read ROM contents and verify and diagnose main buses.
	ID Test	Read back flash ID using CPU read/write instructions.
	ROM Checksum	Use CPU read instructions to calculate flash checksum and verify contents
SDRAM Memory Test	Verify all buses to SDRAM using appropriate memory test algorithm.	An initial memory test can consist of CPU read/write instructions run via the COP. This can be used to verify a small section of memory and all buses. Memory test programs can then be downloaded to this area, and run under the control of the COP port.
PMC Bus Test	Access test to plug-in PMC card.	Uses known good plug-in PMC card. Use COP CPU read/write instructions to construct a bus test.
Glint Gamma 3D Geometry and Lighting Processor	PCI Bridge Register Test	Use CPU read/writes to verify buses and communications from CPU to Glint GAMMA
	Configure for Normal Operation	Use CPU writes to configure Glint GAMMA registers
Glint MX 3D Video Processor	PCI Bridge Register Test	Use CPU read/writes to verify buses and communications from CPU to Glint GAMMA to Glint MX
	Configure Memory Controller for Normal Operation	Use CPU writes to configure Glint MX registers
	VRAM Memory Test, Local DRAM Test	Run memory test algorithms under COP control on VRAM and local DRAM
Video Controller	Configure for Normal Operation	Use COP CPU write instruction to set-up video controller for suitable graphics mode

	RAMDAC_A, RAMDAC_B – Verify test pattern output.	Use COP CPU write instruction to set-up RAMDACs. Output test pattern to VRAM and get operator to verify on attached screen.
Tundra VME Test	Tundra VME Bridge Register Test	Use CPU read/writes to verify buses and communications from CPU to Tundra
	Configure Tundra for normal operation	Use CPU writes to configure Tundra registers
	VME Transfer	Establish communications from Tundra to adjacent card in the VME backplane under COP control.
Optionally run UUT as final verification		Using the COP reset and run the board under its own boot code.

Table 1: Case Study Emulator Test Sequence

4.3 Extended Test Coverage

In the case study, boundary scan has been used to structurally test the following parts (see figure 6):

- PowerPC Microprocessor
- MPC106 Memory and PCI Controller
- Tundra VME Bridge
- Various 'glue' FPGAs
- In addition, a basic memory interconnect test is provided to SDRAM and Flash.

Test coverage is extended using CPU emulation. A structural test of the following parts is added:

- Glint GAMMA
- Glint MX
- VRAM
- Glint DRAM
- RAMDACs

And a functional test of all board parts is added.

The added functionality now permitted a validation facility for both the bootROM and the main application program, by performing a checksum test on the flash devices we were able to determine,

particularly on repairs, that the latest versions of software were in fact installed.

Once the functional tests were complete it was now possible to reset the processor and allow it to bootstrap. When the UART was connected, via HyperTerminal, it was possible to monitor the normal powerup and run on-board test Built-In-Test (BIT) sequences. We also connected the video output to a RGB monitor to view an internally generated test pattern.

The test time to run the emulation test was approximately 64 seconds and was acceptable against the targeted requirements.

The diagnostics were accurate, however the messages are not aimed at a production technician.

Conclusions

Boundary scan provides good development times and excellent diagnostics for structurally testing boundary scan parts. Emulation takes over where boundary scan leaves off, testing the non-boundary scan parts and providing a functional test. The combined strategy provides maximum coverage, shortest development times, and optimum diagnostics.

Emulation test provides an ideal follow-on test from boundary scan, and when the design of the interface is suitable, it is possible to combine scan test and emulation test into a single interface. A minimal fixture may be required if there is a need to examine specific outputs from the board under test.

Fault coverage improvement is difficult to evaluate at this time. The fault catalogue for scan and emulation tests are different, but the greatest improvement is that the faulty board is being run in an automated fashion and that the tests are structured in a logical test sequence with the possibility to loop any individual test as an aid to diagnostics.

Further Work

Work is now on-going with our boundary scan and emulation providers to allow the execution of these emulation tests from

the same boundary scan interface and to bring the diagnostic messages to a standard format that is easily understood by a production technician.

The emulation tests need to be able to reformat the fault diagnostics to a single device level and, since the scan tools use netlists it is possible to actually generate net level diagnostics from the emulation test outputs.

The real test will be when we combine both scan and emulation interfaces into one and using discrete I/O signals switch seamlessly between scan and emulation in an automated test sequence.

Work is ongoing in developing a single on-board (JUT) design that allows switching between the COP and scan paths. This is necessary since the COP needs the powerPC device to be in a single device chain, but when implementing scan tests we wish to bring the powerPC into the full interconnect test scan chain.

6. References

- [1] IEEE 1149.1-2001 "Test Access Port and Boundary-Scan Architecture" Standard, <http://standards.ieee.org/catalogs/olis/index.html>
- [2] Application Note 28: The ARMTDMI Debug Architecture, ARM Ltd., 1995.
- [3] Daniel Mann, AMDDebug offers on-chip Debug Support, Embedded Systems Engineering, Dec/Jan 2000
- [4] IEEE-ISTO 5001-99 Global Embedded Processor Debug Interface Standard, www.nexus-standard.org
- [5] RISC Microprocessor Hardware Specifications – Motorola Common On-chip Processor (COP)