EMBEDDED BOUNDARY-SCAN TESTING

Bradford G. Van Treuren, vantreuren@lucent.com
Jose M. Miranda, jmmiranda@lucent.com
Lucent Technologies, Inc.
Test Solutions Engineering Group
101 Crawfords Corner Road
Holmdel, New Jersey

Abstract

Boundary-Scan testing is used more and more to overcome many of the testability issues facing today's higher density designs. In the past, Boundary-Scan has been used successfully with ATE's and external PC based test systems. Since Boundary-Scan tests are structural in nature, they can be reused with often minor modifications in the embedded arena. Further, these same tests can be used in the H/W design lab, S/W development lab, EST chambers, functional test, factory system test, field test, and repair center test. We present cases, within Lucent Technologies, where this has been successfully achieved for many Wireline and Wireless product families. This paper also discusses the mechanisms used to achieve these successes.

1. Introduction

Board level Boundary-Scan testing has been used for a number of years. Typically this has been performed in the lab with a PC based tester and in manufacturing with ATE's. One of the primary advantages of Boundary-Scan tests is that they are structural in nature and have explicit coverage metrics. Also, most tests are generated automatically using structural information about the design (e.g., netlists, device pin attributes). All this leads to a higher quality assurance and reduced test generation time.

In the embedded environment, board level tests are typically functional tests aimed at some specific operation of a system. These tests cover one or more hardware modules within a board or system. The percentage of circuit coverage by these tests is often difficult to calculate. Further, these tests are typically hand written specialized cases designed to give a GO/NO-GO status of a functional block of the circuit.

Since Boundary-Scan tests are vector-based tests, the application of these tests is independent of the content of the vectors. This leads to a decoupling of the test data from the control software used to apply the data. Thus, only one software module needs to be written to apply any Boundary-Scan test. Using Boundary-Scan in the embedded environment reduces the software development effort required to test the Unit Under Test (UUT). Further, most tests can be reused from manufacturing with little to no modification. All this leads to a reduced testing cost for the embedded environment. Boundary-Scan will never eliminate the need for functional test cases, however, the test engineers writing functional tests can focus on more specific areas of the design that are not covered explicitly by the Boundary-Scan tests. These areas are clearly defined based on the coverage metrics of the Boundary-Scan tests provided by the test generation tools.

We will describe how embedded Boundary-Scan is used in two typical types of telecommunications systems produced by Lucent. However, these same techniques may be used in any type of system where Boundary-Scan tests may be applied.

2. System Test Architectures

Depending on whether you are performing simplex testing (self test) or duplex testing (mate or slave testing) there are different hardware architectures used to connect Boundary-Scan boards together. The first is the serialized chain where all Boundary-Scan chains in the system are connected as one single chain (Figure 1). The second and more flexible architecture is the multidrop configuration[1] where the Boundary-Scan chain is bussed to each board and special logic is used to connect a board to the test bus at appropriate times (Figure 2).

Within Lucent, we use an extended IEEE Std 1149.1 multidrop architecture, based on the Texas Instruments Addressable Scan Port (ASP)[2], for systems that contain significant complexity. We have found that the ASP provides us the ability to reuse tests for common circuit boards within a system easier than with other technology. This is because the selection protocol is separate from the test data. For simple

systems, architects choose whether they want to use multidrop or serial scan chains to test the system. With either architecture, a TAP controller must be resident within the system. In the two cases presented in this paper, the TAP controller is the Agere Systems Boundary-Scan Master 2 (BSM2)[3].

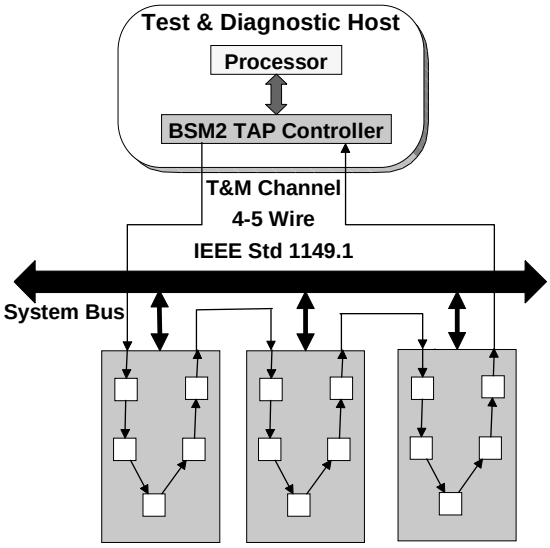

Figure 1 – Serially connected scan chain

To develop tests for serial scan chain architectures, Lucent uses a tool (developed by Bell Labs and OEM'ed commercially by ASSET InterTech, Inc. as SystemMerge™) which merges the netlists of the individual boards and backplane into a single netlist that allows the use board level test generation tools to test the individual boards and backplane interfaces in aggregate.

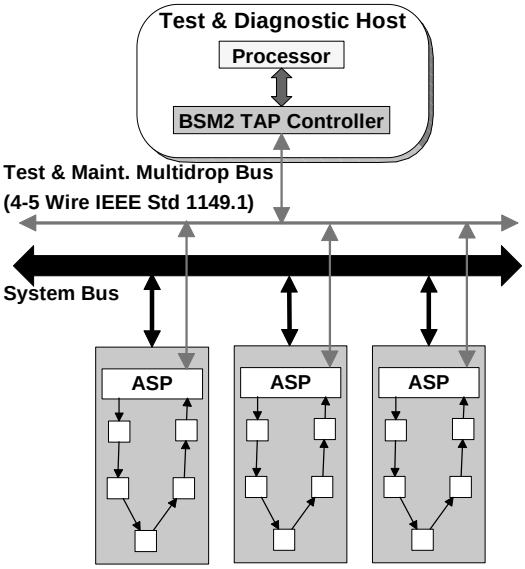

Figure 2 – Multidrop scan channel

For multidrop systems, board tests are developed independently for each board, using commercial board test generation tools, and applied to the board when the ASP is selected in the system. During board test development, external tests are applied to the board with the ASP bypassed (the BYPASS pin of the ASP is wired to the board edge connector and controlled by the test fixture).

3. Embedded Test Software

The software used to apply the tests in the embedded environment is written in C++ and embodies the Test Flow Control Language™ (TFCL), which is a development of Lucent Technologies. With TFCL, a test engineer is able to apply various test steps by name in a specified order. TFCL provides for conditional branching as well as various forms of looping. Listing 1 shows a simple example of a TFCL program to select an ASP address for a board, apply a scan path integrity test, and apply an interconnect test to the UUT. The types of test steps supported by TFCL in the embedded

```
ENTITY sys
FLOW sys IS
APPLY ASP FROM 0x10 TO 0x10 DIAGNOSE;
IF SYSTEMERROR THEN
   PRINT("System Error!");
   STOP;
END IF;
IF NOT FAIL THEN
   APPLY integ1 DIAGNOSE VECTORS;
   IF NOT FAIL THEN
      APPLY inter1 DIAGNOSE LINE PIN NET;
      IF FAIL THEN
         PRINT("Interconnect test failed!");
      ELSE
         PRINT("All tests pass!");
      END IF;
   ELSE
      PRINT("Integrity test failed!");
   END IF;
ELSE
   PRINT("Unable to connect to board.");
END IF;
END FLOW;
END ENTITY;
```

Listing 1 - Example TFCL Program

environment are: SVF tests, ASP selection requests, BSM assembler programs, BSM2 assembler programs, and STAPL programs. The use of TFCL provides a powerful and flexible means for separating the

embedded program from the tests to be applied to the UUT. To apply any test is as simple as passing the TFCL program and supporting test data to the TFCL interpreter. Thus, this architecture is able to support not only resident embedded tests, but also remotely downloaded tests. The size of the embedded program for a standalone application running on a 68060 is 92Kbytes for the text and data sections of the standalone application code in FLASH memory.

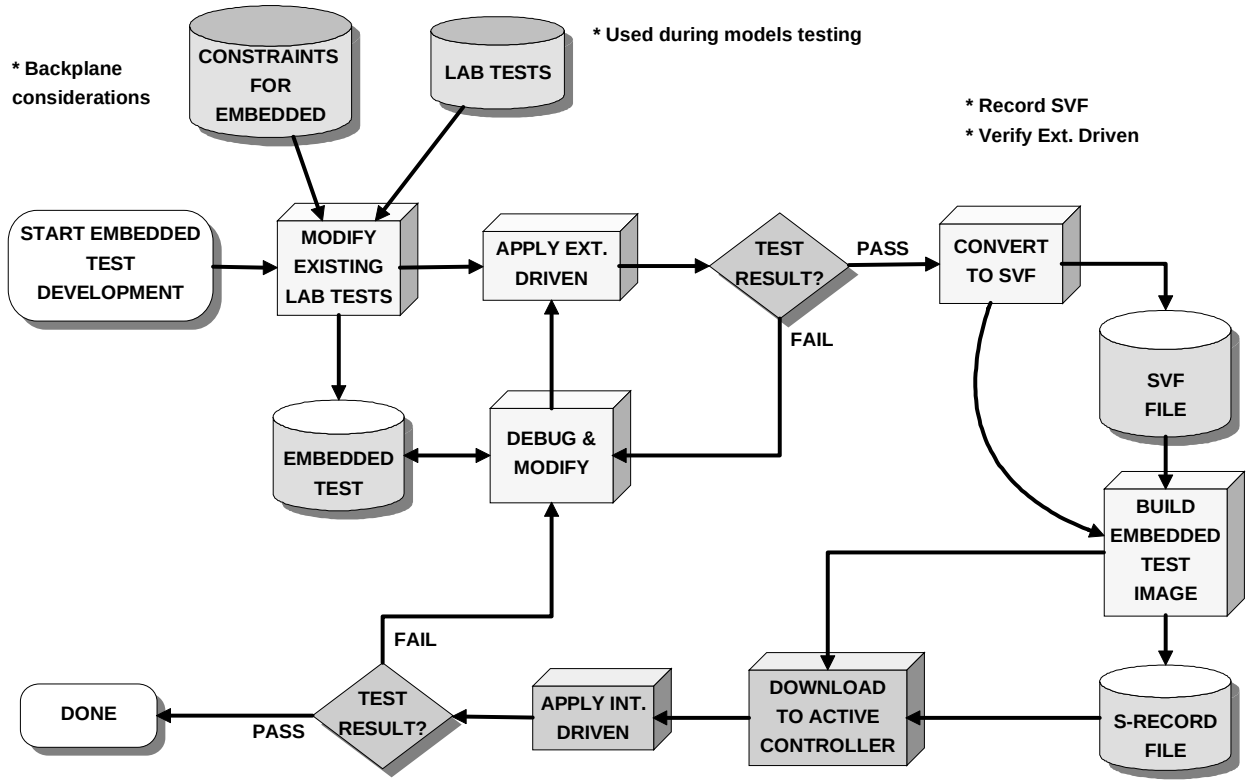

Figure 3- Embedded Test Process Flow

4. Embedded Test Generation Process

Figure 3 describes the embedded test generation process flow. The embedded Boundary-Scan test generation process may be described in three phases: 1) embedded vector generation; 2) vector porting to target application environment; and 3) embedded test application.

Embedded Vector Generation: The Boundary-Scan board tests are generated off-line using a PC based test tool – in our case, the ASSET system. Typically, this would involve selecting a suitable subset of existing tests, such as those used for circuit board models testing, then refining and tuning them to constrain signals that may affect system operation during the test (e.g., backplane signals). The modified tests are applied to the UUT using an external tester. Once they have been verified on the external tester, they are ready to be ported to the embedded environment.

Embedded Vector Porting: The embedded tests are first converted to Serial Vector Format (SVF)[4] files. SVF was chosen as the intermediate Boundary-Scan vector representation format due to its popularity and status as a defacto standard. SVF also supports the ability to easily add diagnostic support due to its static model structure. Other languages, such as the Standard Test and Programming Language (STAPL)[5], are used for programming devices, however, they do not lend themselves to provide diagnostics easily in the embedded environment due to their dynamic model support.

Before an SVF file can be used in the embedded environment, the data has to be compiled and compressed. A suite of tools, developed by Bell Labs, process the SVF file(s) to create a binary or an S-record file suitable for downloading to the target circuit board. The S-record image is loaded into the

target's DRAM for immediate application or as an interim store for eventual loading into a non-volatile memory resource (e.g., flash memory). The binary image may be copied into the system via a high speed network or stored as a resource file accessible by the system. Vectors used for In-System Configuration may be created as SVF or STAPL files. If STAPL is used, the file is compiled into Jam™ STAPL Byte Codes[6] for use in the embedded environment.

Embedded Test Application: The S-record image includes embedded Boundary-Scan control software that is used to apply the tests in a stand-alone environment. Some diagnostic tools are available to debug tests that fail in the embedded environment. Test engineers may choose any of the following diagnostics for their embedded tests: GO/NO-GO, Failing vector data, SVF line number of failing vector, the device pin of the observed failure, and the net containing the observed failure. Once the tests have been validated in this environment, they may be incorporated into the production firmware load. This allows the tests to be proven in and used in parallel to the development of the production system software.

5. Duplex Test Case Study

For an early transmission product application, the controller boards implement embedded Boundary-Scan testing – hosting the JTAG interface device. These controllers are installed in matched pairs with one processor functioning as the active controller and the mate acting as the standby processor. The active processor performs the Boundary-Scan testing of the mate standby processor within the system.

The following embedded tests were performed for the subrack controller with the coverage listed:

- 1200 nets (51.3%) covered by ATPG tools
  - Scan path integrity test (201 nets covered, 8.6%)
  - Interconnect test (999 nets covered, 42.7%)
    - 456 fully covered nets (19.5%)
    - 234 shorts and some opens covered nets (10.0%)
    - 309 shorts only covered nets (13.2%)
- Power/Latch test (hand generated)
- IC level testing
  - 497AA BSM BIST
  - ASIC BIST (2 ASICs)
  - CPLD INTEST (4 devices)

The test vectors for each board configuration are stored locally in a segment of local FLASH memory on the UUT. The active mate is able to access the standby board's FLASH memory and retrieve the vector data to local DRAM at the time of testing. Thus, configuration management is not an issue because each board contains its own test data locally.

Also located in FLASH is a version of the control software that can execute as a standalone program from the DEbug MONitor (DEMON). A second version of the control software is integrated with the Factory System Test Software (FSTS) operating from FLASH disk on the CHORUS operating system (OS). Both software systems use the local test vectors from the UUT's FLASH memory.

The standalone FLASH-based version of the software has proven useful in the design lab, software lab, EST chamber testing, and manufacturing functional test areas of the product life cycle. This is because it embodies the concept of *Built-in Test* or *Test Anywhere* philosophy. The FSTS version is useful in the Factory System Test operations in manufacturing and functional test since it contains the functional test software as well as the Boundary-Scan software.

Our experience has been that once the test feature was available, more people identified other uses for it than previously targeted. For example, using embedded Boundary-Scan in EST applications.

6. Simplex Case Study

In this case the system incorporates a single controller module as the master test unit. This module is able to test sections of itself using Boundary-Scan. The master also tests the feature boards using the multidrop test architecture.

The vectors associated with testing each module are not stored locally with each board because the controller module does not have direct access to the FLASH on each slave board. Instead, the vectors are downloaded into DRAM when the tests are to be performed or versions of the tests are stored in available FLASH on the controller module. The vectors for testing the controller itself are stored locally in FLASH. The vectors for testing the other boards in the system are downloaded via a network interface if needed.

There is a version of the control software that is called during the Level 2 boot operation that tests the controller, which does not depend on the OS to be operational. A second version is available at run-time executing under the VxWorks OS based diagnostic

software for testing the rest of the system. A third version, based on the same code used for the second, is planned to provide the capability for remotely downloading tests and applying them to the system. This last version is thought to be good for performing updates to the FLASH and PLDs in the system as these devices may be programmed using the IEEE 1149.1 Boundary-Scan protocol.

7. Benefits

The factory engineers for the first case have applied embedded Boundary-Scan tests as part of their overall functional test suite for all controller circuit boards. They have been able to use the embedded testing in-line as there is no special setup required to apply the tests. This capability is providing enhanced coverage at functional test, and is serving as a diagnostic tool for boards failing at system test or for boards returned from the field.

Once the embedded tests have been certified in a factory setting, they may be deployed as part of the generic diagnostics for periodic execution in the field.

The fault diagnostic resolution is very different across a product's life cycle. For lab and manufacturing test, the diagnosis down to the failing device pin or net is preferred. Fortunately, external testers may be used here to isolate the fault to that detail once a board has been indicted by the embedded tests. However, failure diagnostics to the pin and net level are available in the embedded testing case. In the field, diagnosis down to the smallest field replaceable unit is required. Using additional data compression techniques, such as vector signatures, allow more tests to be stored in FLASH. This allows the quality of the test coverage to be improved, but the diagnostic resolution is sacrificed.

The infrastructure used to acquire the test data from the UUT, in case one, was also able to detect a failing circuit board when the data was inaccessible due to faults. Indirectly, the embedded test system described here was able to identify failing circuit boards prior to applying any Boundary-Scan tests.

8. Traps and Pitfalls

When a field-deployed system is online, one embedded Boundary-Scan issue that must be considered is fault recovery. While an embedded Boundary-Scan test is in progress, a mechanism must be provided to enable the embedded test master to respond in a timely way to alarms or other failures detected in the system.

In a live system, it is also necessary to have the board-under-test recover to a sane state after a Boundary-Scan test has been run. A Boundary-Scan interconnect test for example, will put the target board in a very unnatural state. There must be a means to reliably reset the target board so that it will be available to the system as soon as the test is complete.

The hardware architecture of the board is critical for Boundary-Scan to be successful in the embedded environment. The control processor and its associated resources needed for applying the embedded tests (e.g., DRAM, FLASH, BSM, Console I/O) need to be isolated from the UUT being tested. With the duplex example, the isolation came by the active processor testing the standby processor. These were two independent circuit boards on the backplane.

The controller board for the duplex case was not designed for maximum Boundary-Scan coverage in a simplex mode. Performing a simplex test on this controller results in significantly reduced test coverage (~51% down to ~25%) due to the overlap of nets required for controlling the test resources and available nets for Boundary-Scan testing. This was better than expected, however, because there was a clean separation between the processor core asset and the application hardware.

For the simplex case, the isolation was designed into the controller to allow the feature half of the board to be totally independent for test. This was a needed DFT practice for ensuring good test coverage in the embedded environment.

Whenever the design of a board changes in the system, new vectors need to be generated for that configuration/design. This can be a configuration nightmare if not managed well. Embedding the test vectors on each board reduces the configuration problem in the field. The only drawback is that the FLASH data on the slave needs to be accessible by the test master. We have found that some slave board designs providing access to the FLASH incorporated much of the board's logic that additional Boundary-Scan testing was not adding much value if the local test data was accessible.

9. Conclusion

Embedded Boundary-Scan testing is a natural extension to the board level external testing that already is being performed. There is much benefit of reuse/salvage of existing tests to obtain deterministic test coverage metrics ensuring a quantitative level of quality assurance. Further, use of Boundary-Scan

testing allows functional test engineers to focus on specific test areas (e.g., mixed signal testing, dynamic testing) without worrying about the covered digital interconnections on the board.

Boundary-Scan testing lends itself to decoupling the test data from the control software so one piece of software needs to be written for a system to apply Boundary-Scan tests and the test data may be updated independently of the software. Further decoupling of the control software was achieved through the use of the Test Flow Control Language. The functional testing of the BSM2 was decoupled through the use of a simple scripting language providing assembly language level support to the BSM2 registers.

For embedded Boundary-Scan to be effective in the embedded environment there needs to be Boundary-Scan friendly hardware architecture to support it. The processor core involved with applying the test needs to be partitioned separately from the remainder of the UUT being tested. Without this separation a significant loss of test coverage will result due to the overlapping of nets used in both partitions.

Lastly, the mating of test vectors with the UUT proved to be quite effective for the controller boards. This capability simplified the configuration management requirements in supporting Boundary-Scan testing of the product. Each board was responsible for its own test data version. One must be careful to design the board so only a small portion of the logic is used to access the UUT test data. Otherwise, there is no need to perform a Boundary-Scan test on the board if the hardware is exercised while obtaining the test data.

10. References

1. Texas Instruments Document SSYA002C, 1997, *IEEE Std 1149.1 (JTAG) Testability Pimer*. Retrieved September 4, 2002 from http://www-s.ti.com/sc/psheets/ssya002c/ssya002c.pdf.
2. Texas Instruments Data Sheet SCBS686A, April 1997, Revised December 1999, *SN54LVT8996, SN74LVT8996 3.3-V 10-BIT ADDRESSABLE SCAN PORTS MULTIDROP-ADDRESSABLE IEEE STD 1149.1 (JTAG) TAP TRANSCEIVERS*. Retrieved June 12, 2002 from http://www-s.ti.com/sc/ds/sn74lvt8996.pdf.
3. User Manual, April 2001, *497AE and 1215E Boundary-Scan Master 2 Advanced Operational Mode*. Retrieved June 12, 2002 from http://www.agere.com/bsm/docs/MN01037.pdf.
4. ASSET InterTech Specification ASSET-SVF-DOC, March 8, 1999, *Serial Vector Format Specification – Revision E*. Retrieved June 12, 2002 from http://www.asset-intertech.com/support/svf.pdf.
5. JEDEC JESD71 Specification, August 1999, *Standard Test and Programming Language (STAPL)*. Retrieved September 12, 2002 from http://www.jedec.org/download/search/jesd71.pdf.
6. Altera Application Note AN122, March 2000, ver. 1.0, *Using Jam STAPL for ISP & ICR via an Embedded Processor*. Retrieved June 12, 2002 from http://www.altera.com/literature/an/an122.pdf