A Code-less BIST Processor for Embedded Test and in-system configuration of Boards and Systems

CJ Clark, Intellitech Corporation, 70 Main Street, Durham, NH 03824 cjclark@intellitech.com
Mike Ricchetti, ATI Research, Inc., 62 Forest St. Marlborough, MA 01752 mike_ricchetti@ieee.org

Abstract

A code-less processor that enables designers to achieve optimal in-system FPGA configuration as well as embed built-in self-test capabilities into boards and systems is presented. This system BIST architecture enables designers to lower system costs and design effort while satisfying test and field engineering requirements for simplified product test.

Motivation

A number of products now utilize programmable logic devices, such as FPGAs, CPLDs, and programmable non-volatile memories, such as EEPROM, Serial EEPROM and FLASH. These devices support the adoption of programmable architectures, which enable system designers to achieve a quick time to market through field upgrade-able fixes and enhancements. In-the-field reconfiguration provides a compelling value proposition as the enhancements can extend the life of the product and also provide downstream revenue.

In order to remotely upgrade a system in the field, access to each volatile and non-volatile device over in-system mission-mode busses is increasingly becoming more difficult. Especially for boards with mezzanine cards or multi-board backplane based systems. For the system designer, these capabilities can add to the costs and design effort required to develop, and later to manufacture such configurable products. Often, the designers create their own ad-hoc methods, which are a costly and time-consuming and do not provide for a solution that is readily re-usable on future product designs.

FPGA based products as shown in Figure 1 are evolving and now routinely use high-speed memory interfaces, 4 or more different power domains, embedded FPGA based CPUs with FLASH memory, LVDS signal interfaces, and gigabit serial I/O. As FPGA I/O technology has changed, it poses increased problems for testing the product. Consider FPGA U1 in the figure, test engineers would like to use IEEE 1149.1 techniques to test the connections with U1 programmed, its I/O using LVDS, so opens testing to U2 can be performed. They would also like to test U1 un-programmed (so the termination resistor will look like a short) and fault isolation can be achieved for stuck at faults not possible with LVDS alone. Fault coverage on the PCB can be enhanced by downloading 'helper circuits' over IEEE 1149.1 into the FPGAs to facilitate tests to high-speed memories such as DDRRAM and FCRAM. Since GigaBit serial I/O typically does not have 'boundary-scan cells' on the I/O pins, the only method available to test them over 1149.1 is via downloadable BERT (Bit-Error Rate Tester) circuits. Furthermore, on-board FLASH and Serial EEPROM configuration speeds can be enhanced through patented "on-chip FLASH programmers" simply by downloading them to the FPGAs [6].

The authors therefore were motivated to provide an integrated approach to Built-In-Self-Test at the PCB and System level. The BIST Processor described here provides a structured approach to solving these problems, using infrastructure IP designed for the board and system levels.

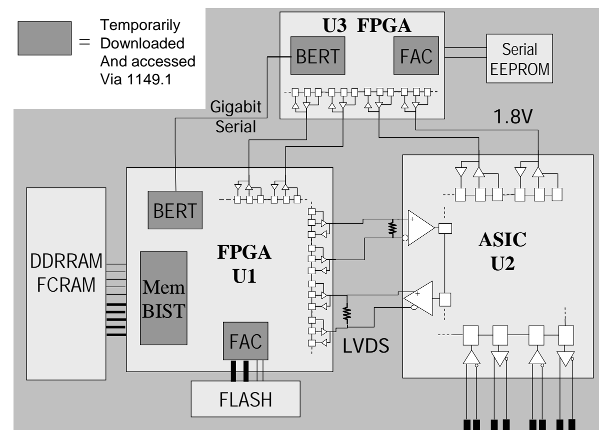

Figure 1. PCB requiring coordinated FPGA programming with 1149.1 based Test

Previous Approaches to Embedded Test and Configuration

Various methods have been used to implement embedded test and configuration at the board and system levels. Details of these approaches can be found in the preliminary description of the BIST Processor [1]. A summary of these approaches is provided in this section.

Software-Based Embedded Test

The typical approach used to embed test into PCBs and systems has been to utilize functional diagnostic code. This diagnostic code is developed by test engineers and systems designers and stored on-board the product in the

0-7803-8580-2/04 $20.00 Copyright 2004 IEEE

CPU's FLASH memory. These embedded tests are then used as a means to test the integrated systems, both in manufacturing and in the field. There are several disadvantages to this approach:

- Diagnostic code development is not automated.
- Engineers must be familiar with the functional designs, the effects of various faults and fault classes, in order just to begin to write software.
- Requires longer development times than boundary-scan based ATPG or BIST and hence higher development costs
- Requires further engineering resources to maintain software long after product ships.
- Test quality and fault coverage can not be easily or automatically measured as with boundary-scan ATPG.
- Software based embedded test requires a (mostly) working CPU to execute.
- Diagnostic isolation for board repair is often poor.

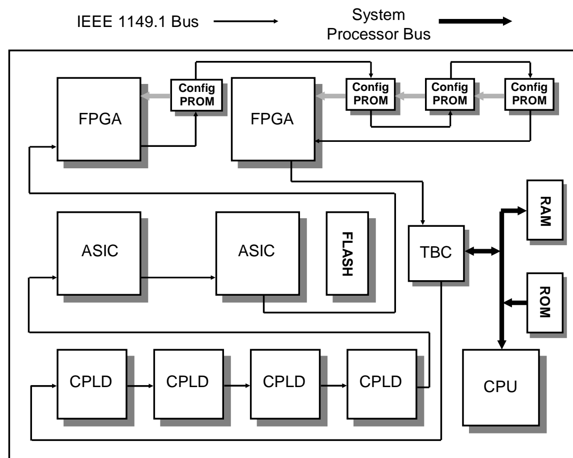

Figure 2. CPU and Test Bus Controller

In addition to functional based testing, using the system CPU in combination with an 1149.1 Test Bus Controller (TBC) is another approach that is often used to embed test, see Figure 2. The approach enables a foundation for both embedded structural test and functional tests, but has its own shortcomings in that it is necessary to modify existing tests, convert them, and re-validate them after each modification [2]. This is necessary since the CPU, FLASH, and connected devices cannot be tested while the CPU is applying the tests. The engineering resource cost of this effort should not be overlooked when compared to other methods.

Configuration PROMS

The method that is predominately used to configure FPGA logic employs application specific configuration PROMs. These PROMs are programmed with the configuration data for the design, which is then loaded into the FPGA's configuration memory at power-up.
However, there are a number of issues with regards to this method:

- Large FPGA designs will require multiple configuration PROMs, impacting board area, layout, and parts cost.
- Configuration PROMs can hold only one design, making it difficult to design an in-the-field re-configurable product.
- Configuration PROMs employ proprietary programming methods, so they are not interchangeable between different device vendors.
- PROMs only program FPGAs, they are not able to update CPLDs, or other on-board non-volatiles like FLASH or Serial EEPROM. Additional circuitry is needed for in-the-field updates of these devices adding to complexity and cost.

Issues surrounding PROM based configuration such as these have driven design engineers to explore new methods for in-system configuration. Unfortunately, this has meant that many design teams have had to design and develop their own custom configuration capabilities. A custom configuration method often ends up being a 'one-off' solution and so it is not cost effective. When using a proprietary method for FPGA configuration, the lack of coordination between configuration and test through IEEE Std. 1149.1 or 1532 [3], [4] based test development and validation tools increases test cost and complexity. Embedded Self-Test through 1149.1 requires this coordination (as discussed in the motivation section) and this can be difficult to achieve when the self-test circuitry cannot control or access the FPGA programming circuitry. The result will be an undesirable loss of fault coverage.

Another common approach used to configure programmable devices is to interface the system processor to an 1149.1 TBC [5], and then use this access mechanism to reconfigure the FPGA PROMs. This method is shown in Figure 2. The major drawback with this approach is that custom firmware must be developed for the target PCB and each individual FPGA configuration must be validated and debugged. Another drawback is PCB area and parts cost since both the PROMs and now another device the TBC and support circuitry need to be added when ideally having one IC that could do both jobs would be more cost and area efficient. During prototype development FPGA engineers will be forced to have some reliance on the software engineers to rebuild software images to incorporate changes. With complex multi-board systems additional care must be taken into account so when new FPGA designs are distributed in the field and the CPU's PROM or FLASH is re-programmed, there is no possibility of corrupting the PROM and hanging the mission mode CPU. These types of risks should be

assessed before in-house development is considered. Also the on-going software development and software maintenance costs when target CPUs change, architectures change, or software development personnel change must be accounted for. These factors can not be overlooked when assessing the true cost of implementing an in-house solution.

CPLD-FLASH based FPGA configuration methods

Another popular method to configure FPGAs in-system is by using a NOR based FLASH. Major FPGA vendors provide downloadable designs to facilitate FPGA programming from a FLASH on their web-sites. This design is then programmed into a CPLD. The CPLD design is used to sequentially access the FLASH memory contents and provide 8 bit wide FPGA programming or the vendors' proprietary serial based programming data streams. The approach has several drawbacks that could be solved with a more unified approach to in-system configuration. Some limitations to consider are:

- There is no data compression. There is a 1 to 1 correspondence between the FPGA configuration bits needed and the size of the FLASH memory needed.
- There is no bit stream re-use. If "Design A" requires 8 Megabits, 4 FPGAs on the PCB require 32 Megabits of FLASH memory.
- The CPLD is a simple blind sequencer, it does not have flexibility for determining the presence or the size of FPGAs and loading the appropriate design based on that information.
- Customization and circuitry is needed to support loading different designs into the FPGAs in the field.
- Updates in the field require erasing and reprogramming the entire FLASH device. Anecdotal evidence suggests more than one vendor has created systems that fail the update process and cannot restart since the CPU required a programmed FPGA to access the FLASH functionally.
- There is no structured 'rollback', a method to return the system to a previous version when new versions updated in the field are problematic.
- It is a custom design, difficult to debug when FPGA does not program correctly and there is little/no direct FPGA vendor tool support. There are no diagnostics.
- On-board programming of FLASH using 1149.1 and EXTEST mode of CPLD is too slow for production lines. A CPLD with 300 bits in its boundary register would take 2.5 minutes to program an externally connected 64 Megabit FLASH. Manufacturing line cycle times are less than one minute. FLASH would need to be programmed off-board first increasing product costs over other methods that exist. Reviewers and readers please see reference [6][18] to understand more about how FLASH is programmed with boundary-scan.
- The CPLD-FLASH method requires more test engineering resources to develop boundary-scan tests for the PCB. Difficult to integrate FPGA programming with a boundary-scan based test flow - especially since CPLD needs to be in EXTEST for PCB level interconnect tests.
- This method is not suitable to re-configure other devices in-system such as CPLDs, I2C/SPI EEProm etc. A separate method must be designed and developed to support configuration of these devices in the field.
- Only NOR based FLASH is supported, lower priced, higher density NAND FLASH would require a NAND interface which would not fit in a CPLD
- The FLASH device typically cannot be shared with the CPU software code, requiring more PCB area for an extra FLASH.
- The CPLD-FLASH method is limited to programming four FPGAs using the vendors slave serial interface. This means that multiple CPLD-FLASH designs are needed for larger systems
- The CPLD-FLASH programming method is not suitable for multi-board systems without engineering circuitry to deliver the programming data throughout the multi-PCB system

While this list appears to be lengthy, CPLD-FLASH based FPGA configuration is one of the predominate methods used. This is probably due to the fact that most of the costs associated with this method are in other segments of the product development that the designer does not get measured on. The lack of flexibility and limitations are also overcome with more engineering resources and design time that most companies don't have good cost models on or are not easily measured in some cases. Many of the issues only surface later during prototype bring-up, in-the-field updates or test engineering. At that point the design is done and it is too late to take another approach. The next design is done the same way, of course, as management is convinced the problems encountered and FPGA configuration issues on the previous generation product are solved for the next generation product.

Embedded Test and Configuration Processor

It has been estimated that a PCB will be tested up to seven times during its product life [7]. This coupled with the desire to perform test and configuration in geographically disperse areas, provides a compelling reason to embed test and configuration into the PCB or system itself. For example, consider that in production manufacturing, board level device configuration and testing would typically be run using ICT. However, if we embed a dedicated BIST Processor, we can eliminate the need to run boundary-

scan based digital tests on ICT equipment. This lowers manufacturing costs by greatly reducing the time a board spends sitting on higher cost capital equipment. In addition, it allows the same set of high quality tests to be used in many different environments and throughout all phases of the product's life cycle. This includes lab prototyping, volume PCB manufacturing, system integration, vibration test, HALT/HASS test, power-up self-test, field service and depot repair.

A dedicated BIST Processor is able to manage multiple system configurations, so system re-configuration can take place anywhere and engineering changes can be easily made at any time during a product's life cycle. A dedicated architecture for embedded test also enables testing of the general purpose CPU and its' support logic, and logging of all failures without the need for the system to function.

To address this application, a dedicated BIST Processor was developed, [8], [9] [10]. It functions as an embedded centralized manager for configuring and testing PCBs and Systems. The architecture was specifically designed to addresses the problems associated with the previously discussed methods, and it has many advantages and benefits when compared to these other approaches. By including the BIST Processor in products, board and system designers can simplify in-system device configuration while enabling comprehensive structural test throughout the system, including the system's CPU. The BIST Processor can be provided as an IC, download-able IP binary or as infrastructure IP that can be embedded in an ASIC. The processor can be used at the board and system levels and allows designers to take advantage of cost efficiencies over the entire product life cycle. It also provides for a scalable and reusable methodology, which augments existing test and configuration standards. Finally, the architecture was designed to off-load ICT equipment, such that structural digital test and device configuration can be done in-system, while expensive ICT equipment can be better leveraged for analog testing.

<u>Centralized Management for Embedded Configuration and Test</u>

The BIST Processor is designed around a novel architecture enabling manufacturing tests and device configuration suites to be developed, and validated, with automated PC-based tools, and subsequently automatically embedded into the system. The processor is a FPGA vendor independent solution, eliminating the need for proprietary configuration PROMs or ad-hoc FLASH based solutions. As a result, designers no longer need to develop customized methods and one-off designs for embedded in-system solutions. At power-up, or under CPU start, a single BIST Processor can automatically run the entire manufacturing test stream, including ASIC ATPG tests, logic BIST, memory BIST, and board/system interconnect tests – as well as configure all the programmable logic devices in the system.

Figure 3 shows an example of how the BIST Processor can be used at the board level. This also illustrates how external automated development tools that are used for developing and validating configuration data and test programs connect to the BIST Processor. The flow for development and validation with this architecture is also shown in the figure. This three step flow is much simpler than the embedded test flow using a Test Bus Controller as described by Van Treuren, et. al. [2]. The TBC method requires a second debug and validation step since the TBC interprets results, TCK frequencies, 'wait-times' and some scan operations differently than the external tools used.

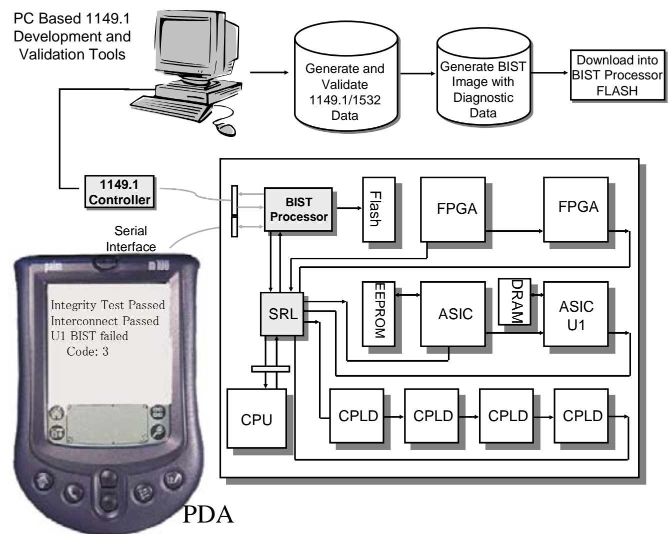

Figure 3. BIST Processor on single board system

The BIST Processor uses PC-based IEEE 1149.1 software tools for ATPG and debug. This development environment then interfaces with an IEEE 1149.1 controller, which connects to the processor on the PCB. The embedded test and configuration processor then interfaces with an optional Scan Ring Linker (SRL) [10] that partitions the scan paths at the board level.

As can be seen by comparing Figure 3 with Figure 2, the embedded BIST Processor replaces the configuration PROMS and interfaces to a FLASH memory device. The FLASH stores the test and configuration suites of the processor and the processor drives the 1149.1 scan chains on the board in this example via the SRL. The processor also interfaces with an external IEEE 1149.1 connector, which allows communication to and from the PC-based tools. This interface is used to develop and validate configuration and test vectors using the external PC-based tools. This is a major advantage in that the tools can

communicate through the BIST processor directly to the devices in the scan-chain during initial bring-up. This guarantees the equivalent drive and signal integrity for the on-board embedded configuration and test mechanism, as was achieved with the external PC-based tools. The external controller and the embedded processor essentially contain the same 'engine' for interpreting the scan test data and the FPGA configuration data, so their behavior is exactly the same, including critical timing elements needed for CPLD programming, DDRRAM tests and advanced interconnect tests such as those performed through IEEE 1149.6. The result is that only one configuration and test validation step is needed, eliminating the need to re-validate the vectors and scripts in the embedded environment. After the test and configuration suites are finalized through the external connector and tools, they are downloaded into the FLASH for embedded execution. Now the external IEEE 1149.1 equipment can be disconnected from the board, and the BIST Processor will assume control of running the embedded test suites and programming the FPGAs.

Test and Configuration Suites

The boundary-scan development tools enable the engineers to create test and configuration 'suites' that hold an unlimited number of test vectors and test scripts, flow control scripts, FPGA configurations, diagnostic codes and text messages. The current implementation of the BIST Processor can accommodate up to 16 suites. The processor can apply one of these 16 test and configuration 'suites' at power-up based on the binary value on the Test_Select bus. When all Test_Select pins are '0', the test selection corresponds to Suite '0', reserved as the 'reset' suite. This reset suite is a set of user defined procedures which are executed automatically whenever a failure occurs. The 'reset' suite can be as simple as causing an 1149.1 Test-Logic-Reset, or more complex, such as addressing PCBs in a system and performing an orderly shutdown. An example test suite is shown in Figure 4. In the figure, the suite includes running standard tests, like interconnect tests, and more advanced tests such as an ASIC internal self-test (not to be confused with the PCB level BIST which is achieved through the BIST processor. The BIST Processor also has the capability to make complex run-time decisions, for example identifying hardware configurations and acting to test and configure the boards and system appropriately. The BIST Processor can also control important hardware functions needed during configuration and test. For instance, the TCK frequency can be lowered to 1MHz on-the-fly, in order to program a slower device, such as a CPLD

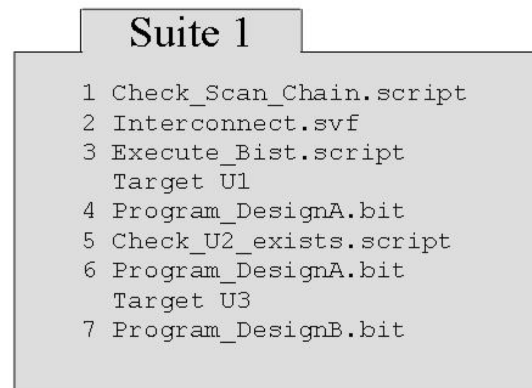

Figure 4. Example Test Suite

Also shown in Figure 4 are the diagnostic codes, labeled 1 through 7. As test and configuration data are added in the PC based development software, a new diagnostic code is assigned. The BIST Processor uses these codes to indicate status of the suite as it is executed. If a test fails on power-up, its' diagnostic code is displayed. The test suites allow one set of configuration and test suites to be applied at power-up and another set of suite to be applied based on 'select' pins on the processor. Alternate test suites can be executed at a time other than power-up, such as during a maintenance mode or update of the system. For instance, CPLDs do not need to be programmed at power-up, however, by including a CPLD reconfiguration capability on suite 2, all (or just specific ones) of the CPLDs in a system could be updated in the field. New or updated suites generated with the off-line PC-based tools can be distributed and uploaded to the FLASH of the BIST Processor enabling the processor to manage all non-volatile updates from a centralized source.

Efficient BIST Data Storage

The BIST Processor development tools analyze duplicate data found not only in a single suite, but also across all 16 suites. The novel way the storage image is created enables scalability and pattern re-use, since in multi-board systems much of the tests and configuration is replicated. For example, many large telecom PCBs have multiple data channels, and so typically 4, 8 or 16 FPGAs have duplicate design data in them. The storage technique used by the BIST Processor reduces the size of the data image beyond the built-in data compression, and hence the size of the FLASH needed for storage is smaller than needed by commercial configuration device offerings or the ad-hoc CPLD-FLASH method.

For example, consider a PCB where three 16-Megabit FPGAs out of six have the same FPGA design, as

illustrated in Figure 5. Each FPGA requires 8Megabits of programming data. Using a CPLD and a FLASH to program the FPGAs requires 48 (6 x 8Mbit) Megabits of FLASH. Since a CPLD and FLASH can only program 4 FPGAs in slave mode, two CPLDs and two FLASH devices would be required. With the BIST Processor, compression of the FPGA data results in an approximately 18 Megabit image. With just a few bytes of data overhead, the same image can be used for all three FPGAs, providing an effective savings of 30 Megabits of FLASH memory. Consequently, the BIST Processor's FLASH needs are considerably reduced compared to using PROMs, commercial configurators from the FPGA vendors, or an in-house designed sequencer using a CPLD coupled to a FLASH. The reduced storage needs allow designers more flexibility in making in-the-field re-configurable systems, since one FPGA change doesn't require an entire duplicate set of configuration data for the PCB. For instance, consider this same design in Figure 5, where FPGA 1, 2 and 4 perform a DSP function. At power-up rather than loading Design A, an alternate design, Design D is to be loaded based on the target use or as an enhancement to the original Design A. If both images were to be used, the CPLD-FLASH method would require 96 Megabits of FLASH memory. However the BIST processor approach would only require an incremental impact on storage area of approximately 6 Megabits.

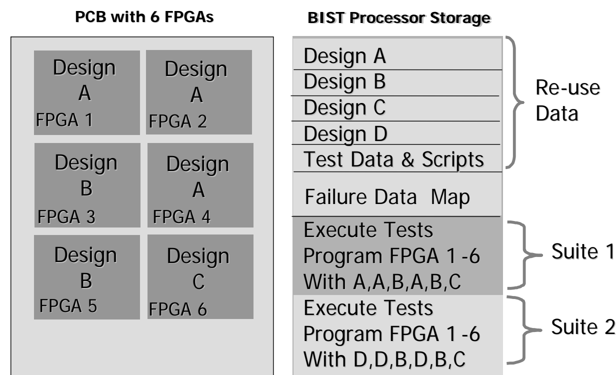

Figure 5. BIST FLASH Memory MAP

Multi-PCB Configuration and Self-Test

The BIST Processor can also be used with multi-board systems as shown in Figure 6 with the use of related infrastructure IP. This system uses a multi-drop 1149.1 bus called the Parallel Test Bus (PTB) and "PJTAG", along with an addressable Parallel Test and Configuration (PTC) IC on each board. These are described in the Parallel Test Architecture (PTA) references [10], [11], [12]. The PTC IC and "PJTAG" Bus extends the 1149.1 bus to allow simultaneous configuration of the FPGAs as well as on-board tests for all of the similar PCBs in the system. The details of this patent-pending technique will be published in another paper. With the technique simultaneous test is possible, but individual access to each PCB is preserved by the unique address of each PTC device in the system. The references and patent applications also describe how the PTC eases PCB-to-PCB interconnect test development when compared to traditional multi-drop architectures.

In the example of Figure 6, only the Master/Slave PCB (Type A) has an embedded BIST Processor. This provides a single, centralized, processor that is dedicated for managing all system level and on-board configuration and test. The architecture enables embedding test and configuration at the system level, including: PCB self test of the Master/Slave board (Type A in the figure), parallel configuration and test of the Slave boards (6 Type B PCBs), execution of ASIC self-test, execution of high-speed memory tests and system level interconnect test.

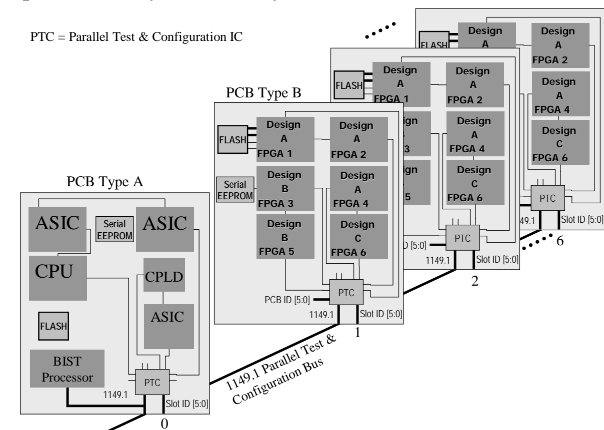

Figure 6. BIST Processor in a multi-PCB System

The reader will recall the calculations for FPGA configuration of the 6 FPGA PCB in Figure 5. When this PCB is replicated in a system six times, the FLASH storage needed by the BIST Processor on the master will only increase by a few bytes. The FLASH size needed for programming the FPGAs is still approximately 24 Megabits (holding two designs for FPGA 1, 2 and 4) versus the CPLD-FLASH method which requires 12 separate FLASH devices with a total of 576 Megabits and two CPLD sequencers per PCB. The BIST Processor approach provides more flexibility, embedded test with less PCB area and parts cost.

The reader should also note the other non-volatile devices on the "Type B" PCB. The on-board programming method with the FAC previously described would enable the BIST Processor to update the Serial EEPROM and FLASH associated with FPGA 1 over the parallel test bus as fast as off-board techniques. When the BIST Processor is combined with a structured scan architecture, such as the Parallel Test Bus access to the system non-volatiles for in-the-field updates becomes simplified.

BIST Processor Architecture

Figure 7 shows a top-level block diagram of the embedded test and configuration processor. It shows the interface of the processor to the FLASH memory, which is used to store test and configuration suites, the local IEEE 1149.1 bus, and the external connector interface, which is used develop and validate configuration and test vectors with the PC based tools. A major advantage of this approach is that it uses a "code-less" architecture, which greatly reduces engineering time. It enables the same test and configuration vectors developed for prototype bring-up and production to be reused by downloading them (via 1149.1) in the FLASH memory for use by the dedicated BIST Processor. Using this approach embedded C++, Java or STAPL [13] software development time, debug and re-validation of scan based tests is eliminated.

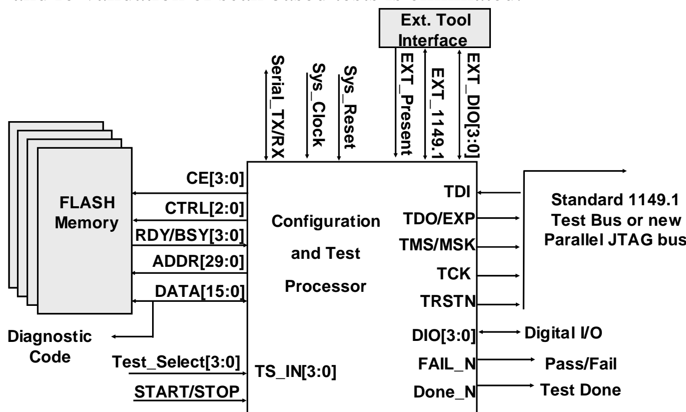

Figure 7. BIST IC Block Diagram

In Figure 7**,** the Sys_Reset signal is used to reset the processor circuitry. When Sys_Reset is asserted low, the registers and state machines of the processor are reset to their initial states, and the BIST Processor is ready to start applying tests or configuration data from the FLASH memory. The Sys_Clock input to the BIST Processor is a master clock that is used to run the processor. Sys_Clock is generally provided by a clock source external to the processor and can be used to derive the TCK frequency of the 1149.1 bus.

Circuitry in the processor enables selection of either an external test tool, or the processor, to be connected to and operate the 1149.1 Test Bus and Digital I/O (DIO). The selection is made with the External Controller Present (EXT_Present) input. When the EXT_Present signal is asserted low, the circuitry is reset and the 1149.1 Test Bus and DIO will be controlled from the external connector. This enables the external 1149.1 controller and allows the PC-based tools to be used for development and validation, and for the FLASH memory to be programmed.

The START/STOP input is used to cause a START or STOP sequence to occur in the BIST Processor. A start condition is issued with a rising edge on the START/STOP input. When a START occurs, the values on the Test_Select inputs determine what test suite the processor runs, and the processor starts accessing the configuration and test data in the FLASH. At this point, the processor begins applying scan vectors. While the processor is busy applying vectors, a falling edge on the START/STOP input will cause it to halt and begin a user defined clean up (reset) sequence.

The memory interface in Figure 7 consists of circuitry and signals for controlling up to 4 non-volatile FLASH ICs. The memory interface is coupled directly to the BIST Processor's internal Fast Access Controller for optimized on-board FLASH programming. The memory interface signals function as follows. The CTRL bus are used in controlling the FLASH's erase, program, and read operations. These include Chip Enable (CE), Output Enable (OE) and Write Enable (WE) signals. The ADDRESS outputs provide the address of the FLASH memory location to be read or programmed and the DATA signals provide data to be read from or programmed to the FLASH memory. A total of 4 Gigabytes of memory is available. During programming from the external tools, the RDY/BSY signals from up to 4 FLASH devices are used to optimize programming times by the internal FAC programming mechanism [14] [15].

The BIST Processor also has a results interface that reports the outcome of a test, and provides failure and diagnostic information. This interface is comprised of the following input and output signals:

FAIL_N: This output is asserted low to indicate that the processor detected a test failure.

DONE_N: After the processor is finished running a set of scan vectors, this output signal is asserted low to indicate that the processor is no longer busy.

DATA Bus: The DATA bus is re-used to output diagnostic codes of failing tests.

Serial_TX/RX: These signals are the transmit data (TX) and receive data (RX) for the Universal Asynchronous Receiver/Transmitter (UART) port of the BIST Processor.

The BIST Processor's architecture provides for various mechanisms to report status and output diagnostics information. One such feature is a user definable test code, which is associated with each test within a suite (see Figure 4), and is stored along with the scan data in the BIST Processor's FLASH. For instance, if the ASIC internal test of Figure 4 failed, the diagnostic code on the data bus at the end of the test execution would be "3"

indicating that the on-chip BIST for U1 failed. By providing the diagnostic code to the DATA bus, the code may be displayed to an LED display, or read by a general purpose CPU connected to the data bus.

Text messages can also be provided with each diagnostic code. These messages are assigned using the test and configuration development tools, and are stored in the flash with the scan vector suites. They are then transmitted via the UART port, on the Serial_TX/RX input and output of the BIST Processor. These messages can be displayed to any serial terminal, such a laptop or a low cost PDA or can be linked to the serial interface of the on-board CPU. The text messages and test codes enable a customer or field service engineer to diagnose a test failure down to the Field Replaceable Unit (FRU). Tests can be made granular enough to allow identification of failing PCBs, plug-in daughter cards, and socketed components.

Test code logging and failing boundary-scan bits can optionally be written to the FLASH at the time of failure. This is particularly useful for systems and PCBs where displaying of the failure data is not possible or useful in the field. When the failing PCB is returned to the factory, software can retrieve the failing bits and display detailed pin level diagnostics. By logging the failures automatically for the embedded structural tests, it avoids the common industry problem of NFF (No Fault Found) on returned PCBs. The failure that was in the field can always be identified, even if it can't be repeated again in the factory.

Logging the data failures is also instrumental for PCB and System test during burn-in. Since the burn-in process is long; continuous or on-demand structural tests can be executed by the BIST Processor simultaneously in each product going through burn-in. By embedding the tests, manufacturers can save on capital equipment costs and fixturing normally required to run boundary-scan tests during burn-in.

BIST Sequencer Comparison

When this paper was reviewed, one reviewer felt this papers concept had already been discussed in reference [16]. After the invention of the BIST Processor in 2000 by the authors, a paper at ITC 2001 was presented that described a 'BIST Sequencer' as part of a chip set for hierarchical boundary-scan. The paper devoted a few paragraphs to the "BIST sequencer" as a slave to a traditional multi-drop 1149.1 architecture (ie no parallel test or PJTAG). The BIST Sequencer was used to initiate (page 485) ASIC BIST tests on board, no discussion of the 'sequencer' covered FPGA configuration or 1149.1 interconnect tests at a PCB or system level. The solution appears to be a point solution designed for a certain PCB card used in a Motorola Satellite; it does not appear to be a generic solution that would work for other PCBs very well. The primary reason for this and the primary limitation with the approach by Harrison, et. al (and the commercial ICs that have sprung up as a result) is that LFSRs (Linear Feedback Shift Register) are used for collecting a signature for each test and comparison with the 'known good' LFSR signature stored in FLASH. While LFSRs are used in IC designs, they do not work well on PCBs. ICs that use MISRs and LFSRs have basic DFT rules that prevent what is called 'X' states in the scan data. Scan flip-flops in an IC that will have unknown behavior and cannot be predicted by ATPG are not allowed. However, PCB boundary-scan chains and tests have many 'X' states in them. "X" are unknown values in the "TDO shift data. For instance in SVF, expected and mask data is shown as:

```
SDR 25 TDI (0FF5FDF) TDO (1FFFFFD) MASK
(0000003);
```

Here all but two bits of the scan data is 'masked' (the MASK is an 'AND' function with the TDO data) that is because the other data is not 'deterministic'.

Typical PCB interconnect tests have hundreds if not thousands of bits that are 'masked' at various times during the tests, a cursory look at an "SVF" file for an interconnect test generated by commercial PCB ATPG tools will show the MASK data is variable and important in obtaining repeatable results. The first boundary-register scan during a typical 'PRELOAD' instruction causes the functional logic values captured in the boundary to be shifted out on TDO. This is problematic for LFSR approaches. Consider also how an 1149.1 device input pin connected to a free running clock would provide a 'toggling' input bit and variable TDO data. The same is also true for 'floating' input pins, FPGA configuration results, CPLD configuration or boundary-scan register bits marked 'internal'. These bits all vary the returned TDO data even if the boundary-scan cells are not part of the PCB test! Even simple 1149.1 instruction register scans will prevent an LFSR from collecting repeatable data. The 1149.1 standard allows, and many designers implement, special 'capture' bits in the upper bits of the IR. What's interesting is that while Harrison is using the approach for executing ASIC BIST tests, many commercial ASIC BIST solutions have non-predictable bits in their results registers (ie scan operations have both expected and mask data) precluding using an LFSR even for some ASIC BIST.

If a PCB level BIST is to be implemented, the EXPECTED and MASK data for every returned TDO bit, must be known by the BIST mechanism as with the code-less BIST Processor described in this paper and patent references. The Harrison approach would require separate tests from manufacturing, created specifically for LFSRs to avoid non-repeatable TDO bits - if at all possible. Validation of the LFSRs repeatability could only be done

after a large sample of PCBs were validated. A quick, but not comprehensive, comparison with the Harrison approach is given below:

- It appears to be a single board solution, with no reference to how multiple PCBs are handled.
- There is no support for FPGA configuration; presumably boards with FPGAs would need traditional FPGA configuration methods. This adds to the PCB cost and would also make certain PCB tests difficult to create and coordinate without the FPGA configuration integrated.
- There is no decision making or branching as described in this paper and the patent reference. Branching based on pop/de-pop conditions on a PCB or external interfaces are important for generic solutions.
- There is no test data compression or test data re-use. There is a 1 to 1 correspondence between the bits and the size of the FLASH memory needed by the Harrison approach. Typically a system, especially one with similar PCBs in it, has many tests that are exactly the same in terms of scan operations, without data re-use then the amount of memory required would be much larger. This also would preclude using the Harrison approach for FPGA configuration, since many PCBs have 4, 8 and even 16 FPGAs that have the same design loaded. A sequencer does not have data re-use requiring more memory needed than the BIST Processor.
- LFSRs prevent the granularity to diagnose a failing test beyond Go/No-go. This makes PCB manufacturing process improvements difficult. If the exact cause can't be found, then it will be difficult to improve the process. This is especially needed during ESS (Environmental Stress Screening) or in-the-field tests since action would be required to prevent the defect from surfacing again. Bit-by-Bit comparison as described in this paper and the patent references are necessary to identify the problem area to the pin.
- The Harrison approach requires each test to run to completion since they are LFSR based (the signature is completed on the last bit added to the LFSR). Certain applications need a controlled 'stop' and need to perform it the instant a fault is found. Consider if one used an LFSR approach to PCB BIST in a missile that was fired. It would be highly desirable to stop applying tests the instant one bit 'miscompares' and disarm the warhead. It's not clear if any action by the BIST sequencer is done on a failure since a general purpose processor and test bus controller is part of the Harrison approach described. The BIST Processor described in this paper has a flexible 'clean-up' or reset sequence when a failure occurs. Many times at the system level, a simple 1149.1 reset is not enough to bring the system to a safe state. The BIST Processor doesn't require an external CPU for access to results (which may not be functioning).
- The Harrison approach has similar problems as the CPU and Test Bus Controller approach in that there is little correlation between the sequencer and the external tools used. Data from the external tools is translated into a format compatible with the sequencer, however there is no guarantee that working external tests translate to working embedded tests. Anecdotal evidence shows that the way various 1149.1 controllers and ATE interpret data and apply it is different. Since the sequencer is LFSR based, if the test doesn't pass, it is difficult to know what caused the LFSR to miscompare.

Conclusions

The use of structured techniques for test, such as 1149.1 and BIST, will increase as boards and systems become more complex, with higher IC-to-IC speeds and with less physical access. Embedded structural test will replace software-based functional testing at all but the highest-levels of abstraction. . The limitations of CPU/OS based functional test development have also been cited by other authors [17]. At-speed 1149.1 based tests are becoming more common place, at-speed 1149.1 based memory tests is more the rule than the exception today.

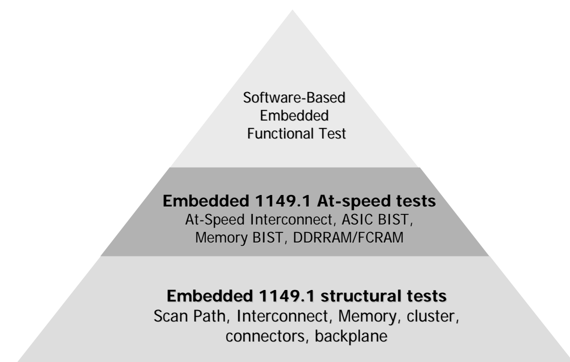

Figure 8. Embedded Test Pyramid

The code-less BIST processor provides an effective way to achieve embedded test with the added benefit of flexible FPGA configuration and simplified in-the-field re-configuration of non-volatile ICs in the field.

We have shown an embedded test and configuration architecture that can be used by board and system designers to enable novel new approaches to in-system configuration and self test of systems. A key to easy and robust test development and configuration with the BIST Processor is the separation of the infrastructure needed for test and configuration from the functional system logic. For PCBs that have FPGAs, test capability is added with little to no overhead, since the BIST Processor will

replace the ICs normally used for programming the FPGAs.
When this paper was reviewed, one reviewer wanted more cost comparisons. An ITC paper is probably not the right place to list cost comparisons, there are other places such as the internet to find these cost comparisons. In general, the BIST Processor and 64 Megabit FLASH is half the cost of a 16Megabit configuration only device and similar in cost to a single CPLD-FLASH FPGA configuration method. Better cost advantages are observed as more duplicate FPGA configuration data is needed since the BIST Processor requires less storage.

Merging FPGA configuration and embedded test functions onto a dedicated high throughput BIST Processor offers the best long-term strategy for building re-configurable and self-testable systems.

References

[1] Clark, CJ, Ricchetti, M., "An Embedded Test and Configuration Processor for Self-Testable and Field Re-Configurable Systems", IEEE 2nd International Board Test Workshop (BTW03), Charlotte, NC, October 2003.

[2] Van Treuren, Bradford G., Miranda, Jose M., "Embedded Boundary Scan Testing", Digest, Board Test Workshop (BTW02), Baltimore, MD, October 2002.

[3] IEEE Std 1149.1-2001, "IEEE Standard Test Access Port and Boundary-Scan Architecture", Institute of Electrical and Electronic Engineers, Inc., New York, NY, USA.

[4] IEEE Std 1532-2002, "IEEE Standard for In-System Configuration of Programmable Devices", Institute of Electrical and Electronic Engineers, Inc., New York, NY, USA

[5] Forstner, P., "Test-Bus Controller SN74ACT8990", Texas Instruments Application Report, SCAA044, August 2000.

[6] Clark, CJ, Ricchetti, M., " A Fast Access Controller for In-System Programming of FLASH Memory Devices", IEEE 2nd International Board Test Workshop (BTW03), Charlotte, NC, October 2003.

[7] Parker, Kenneth, et al.," Boundary Scan Signals Future Age of Test", EP&P, 7/1/2002.

[8] Ricchetti, M. Clark, CJ, "Method and Apparatus for Embedded Built-In Self-Test (BIST) of Electronic Circuits and Systems", US Patent Application 10/142,556, US Patent and Trademark Office, Washington, D.C., December 4, 2001.

[9] Ricchetti, M. Clark, CJ, "Method and Apparatus for Embedded Built-In Self-Test (BIST) of Electronic Circuits and Systems", US Patent Application 20030106004, US Patent and Trademark Office, Washington, D.C., May 10, 2002.

[10] C.J. Clark, Mike Ricchetti, "Infrastructure IP for Configuration and Test of Boards and Systems", IEEE Design & Test of Computers, vol. 20, no. 3, May-June 2003, pp. 78-87.

[11] Ricchetti, M. Clark, CJ, "Method and Apparatus for Optimized Parallel Testing and Access of Electronic Circuits", US Patent Application 2003009715, US Patent and Trademark Office, Washington, D.C., July 5, 2001.

[12] Clark, CJ, Ricchetti, M., "Method and Apparatus for Optimized Parallel Testing and Access of Electronic Circuits", PCT Patent Application WO03005050, World Intellectual Property Organization, Geneva, Switzerland, July 5, 2001.

[13] Anon., "Using JAM STAPL for ISP & ICR via an Embedded Processor", *Altera Corp.*, Application Note 122, March 2000.

[14] Ricchetti, M. Clark, CJ, Dervisoglu, B., "Method and Apparatus for Providing Optimized Access to Circuits for Debug, Programming, and Test", US Patent 6,594, 802, March 23, 2003.
http://www.uspto.gov

[15] Ricchetti, M. Clark, CJ, Dervisoglu, B., "Method and Apparatus for Providing Optimized Access to Circuits for Debug, Programming, and Test", PCT Patent Application WO0171876, World Intellectual Property Organization, Geneva, Switzerland, March 23, 2000.
http://ep.espacenet.com/

[16] Harrison, Stephen, et. al. "Hierarchical Boundary-Scan – A chip-set solution", International Test Conference 2001, Baltimore, Maryland.

[17] Nejedlo, Jay, "TRIBuTE™ Board and Platform Test Methodology", International Test Conference 2003, Charlotte, NC

[18] Clark, CJ, Ricchetti, Mike, "Parallel IEEE 1149.1 technique optimizes PCB test and configuration", *Test and Measurement World*", June 1, 2004.