An Embedded Test and Configuration Processor for Self-Testable and Field Re-Configurable Systems

CJ Clark and Mike Ricchetti
Intellitech Corporation, 70 Main Street, Durham, NH 03824
cjclark@intellitech.com and miker@intellitech.com

Abstract

This paper presents the architecture of a configuration and test processor that enables designers to embed test capabilities, and build products that are readily re-configurable. The architecture supports built-in testing and in-the-field re-configuration for PCBs and systems.

Introduction

An emerging issue for system designers is the desire for configurable products with a quick time to market. This has motivated designers to increase the adoption of programmable architectures. As such, a growing number of products now utilize programmable logic devices, such as FPGAs and CPLDs, and programmable non-volatile memories, such as EEPROM and FLASH. In order to support continuously changing industry standards, field upgrade-able fixes and enhancements are becoming a common product requirement. Designing systems with the ability to remotely upgrade the programmable logic and non-volatile memory of the system, while in the field, is a typical approach to addressing these requirements. However, obtaining access to all of the programmable devices is increasingly more difficult, especially for PCBs with mezzanine cards or multi-PCB 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 functional based methods for debug, configuration and test. This is a costly and time-consuming approach, and does not provide for a solution that is readily re-usable on future product designs.

Traditionally, the approach used to embed test into PCBs and systems has been to utilize functional diagnostic code, as developed by test engineers and systems designers, stored on-board the product. For example, stored in the 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. Such functional test programs are ad-hoc, custom, embedded software applications. They require specialized resources to develop, validate and maintain, which results in high costs due to long development times and related engineering resources. Furthermore, Quality Managers cannot easily measure the fault coverage of functional tests. That is, there is no way to mathematically calculate the percentage of pins that are being tested for stuck-at faults versus those not tested when functional software based test is used. The traditional method is to inject stuck-at faults through physical access, but this is time consuming and difficult to do on a modern PCB assembly with BGAs and mezzanine cards. Collecting data on field returns tells only what faults are being caught not what faults are missed, and the major disadvantage of this approach is that it is done after customers receive the product.

While significant resources can be invested, any software based functional tests that can't identify faults in the field or isolate faults enough to repair a failing system offer little value over running the system in mission mode and identifying that it doesn't function. Any functional test will also require a (mostly) working system in order to execute, and so they offer limited value in system-bring up and debug of prototypes. As systems grow more complex, it is becoming impractical to continue with this approach to embedded system test, just as it became impractical to continue testing ICs with functional tests.

Current Approaches to Embedded Test and Configuration

The following sections provide some background on methods that have traditionally been used for facilitating product re-configuration, and for embedded test of PCBs and systems.

Re-Configuration of Programmable Devices

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.

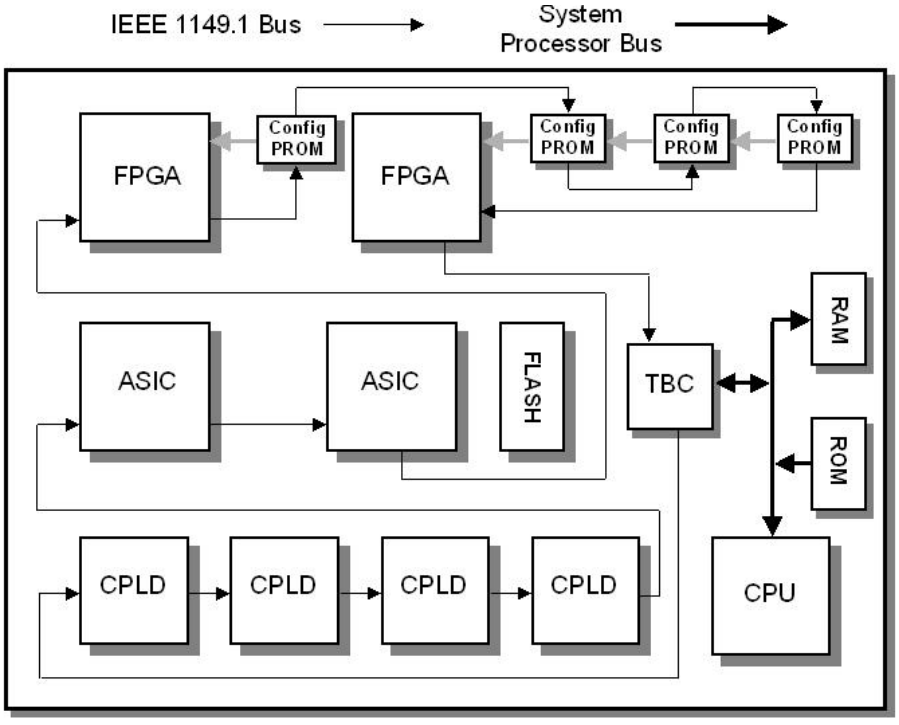
A large FPGA, on average, requires up to four configuration PROMs for a single design. In some extreme cases, with the largest FPGAs and designs, up to seven PROMs could be needed. The number of PROMs impacts PCB parts costs, board area and layout time. Moreover, configuration PROMs are limited in that they can only hold one FPGA design. This is an issue because in order to satisfy requirements for re-configurable products, it is often necessary to have the ability to load a product with different FPGA designs based on the target use of the product by the customer. For example, it is sometimes desirable to load different protocols or algorithms into an FPGA based on the target communication medium, or the geographical location where the product will be used. A further issue is that configuration PROMs employ a proprietary method for programming FPGAs, so they are not interchangeable between different programmable device vendors. Also, the PROMs will require an on-board mechanism that enables them to be re-configured in-system.

Another issue with the configuration PROM method is integration of FPGA configuration with board and system test. For instance, many FPGAs now support various I/O logic families such as GTL, SSTL, HSTL, PECL and LVDS. To maximize fault coverage in a complex system, the FPGA based PCBs need to be tested in at least two scenarios, with the FPGA I/O configured and with the I/O unconfigured. It is also desirable to program the FPGAs during boundary-scan test with small 'test' helper circuits to maximize test coverage and add at-speed tests.

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. This is not cost effective due to the added design time and part cost, increased engineering verification and debug time, lack of support by commercial design and test tools, added software integration costs and lack of re-use in the next version of the product design. When using a proprietary method for FPGA configuration, the lack of coordination between configuration and test through IEEE Std. 1149.1 or 1532 [1], [2] based test development and validation tools, adds to engineering time and increases test cost and complexity. Especially the complexity of embedded self-test. In addition, these approaches also result in loss of fault coverage.

Another common approach used to configure programmable devices is to interface the system processor to an 1149.1 Test Bus Controller (TBC) IC [3], and then use this access mechanism to reconfigure the FPGA PROMs. This method is shown in Figure 1.

Figure 1. Traditional CPU and Test Bus Controller



This however has serious drawbacks in that it doesn't reduce configuration PROM costs or PCB area, since the PROMs are still needed for FPGA configuration. Another major drawback is that there is no interoperability between the PROM vendor's external programming tools and programming with the TBC method. Successful programming of the PROMS with the vendor's tools does not guarantee that the TBC and CPU firmware can program the PROMs. Consequently, additional validation and debug are needed. Configuration of FLASH based PROMS and FPGAs can be timing dependent, so it is difficult to predict the timing, throughput, and delivery of the configuration data with this approach. Interfacing to the CPU is not always straightforward, as it requires glue logic, for example using a CPLD, to provide peripheral addressing on the CPU address/data bus to the TBC. Finally, glue logic is also needed around the TBC to allow access by external PROM and FPGA programming tools during bring-up and validation

Designers may be tempted to use the mission mode processor to program FPGAs directly. However, this creates hidden engineering and product costs downstream. The slow configuration speed prohibits practical application of multi-design FPGA loading. For systems that use this approach, configuration times can exceed several minutes to configure all of the FPGAs in the system. For products that require "on-the-fly" re-configuration, this is prohibitive. The poor programming performance of this approach also affects throughput during manufacturing, adding to the cost of the product. A difficult problem to avoid with this approach is system wide resets that are controlled by programmable devices, such as CPLDs that also have IEEE 1149.1 test capability. This creates undue complexity since basic Boundary-Scan instructions can toggle system and FPGA reset pins in ways that the designer did not envision. This will affect the 1149.1 pin level diagnostics capability, as a simple stuck at fault in a critical interconnect between an FPGA and the CPU will negatively affect all of the serial data for the Boundary Scan test. Thus, it is best to avoid integrating mission mode logic with serial configuration and test mechanisms.

Another 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 for the entire life of the product. Software such as "JAM/STAPL" for CPLD/PROM configuration has been offered by FPGA vendors [4]. However, much consideration must be made in order for it to be used in a complex multi-board system. For instance, when this software is executed by the mission mode CPU and FLASH, additional care must be taken into account so when new FPGA designs are distributed in the field and programmed in the FLASH, the FLASH has no possibility of becoming corrupt and hanging the mission mode CPU. Finally, it should be noted that with the adoption of IEEE Std. 1532, the JAM/STAPL approach has become obsolete.

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 teams change must be accounted for. These factors can not be overlooked.

C++/STAPL Code-Based Embedded Test

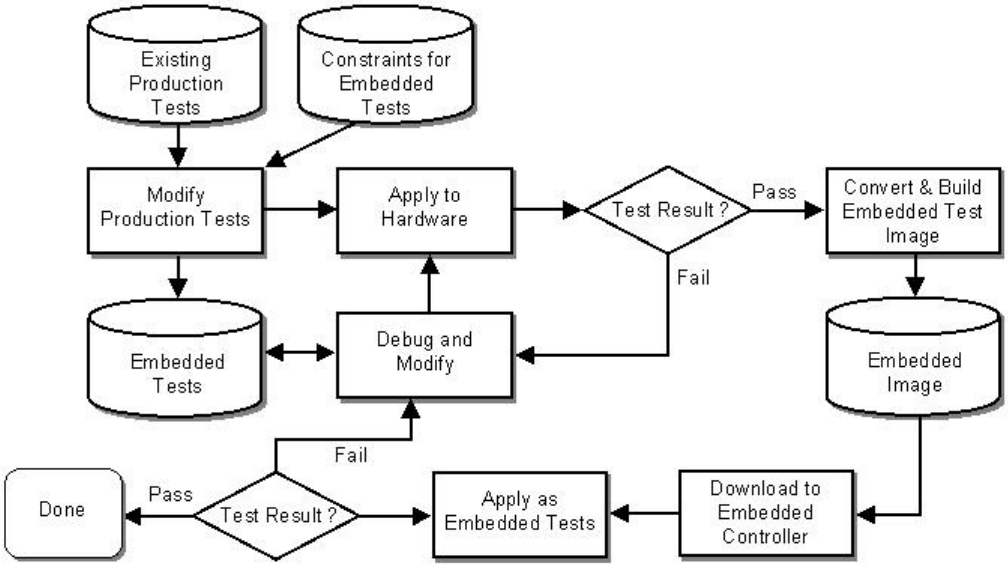
Besides functional based testing, as was discussed in the introduction, using the system CPU and an 1149.1 TBC is another approach that is often used to embed test [5]. The approach enables a foundation for both embedded structural test and functional tests, but has its own shortcomings. Figure 2 shows a typical flow used in this type of approach. Notice that it is necessary to modify existing tests and re-validate them after each modification.

The modifications of the production tests are needed since the CPU, FLASH, and connected devices cannot be tested while the CPU is applying the tests. Once they are modified, they must be converted for the embedded environment and validated a second time as embedded tests. The engineering resource cost of this approach, when used for every test and every FPGA design, on every system configuration, should not be overlooked.

A Novel Structured Approach to Embedded Test and Configuration

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

Figure 2. Development and validation flow for traditional CPU and TBC



testing would typically be run using ICT. However, if we embed a dedicated configuration and test 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 aboard spends sitting on higher cost capital equipment. This approach to embedded test then allows the same set of high quality tests to be used in many different environments and through 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.

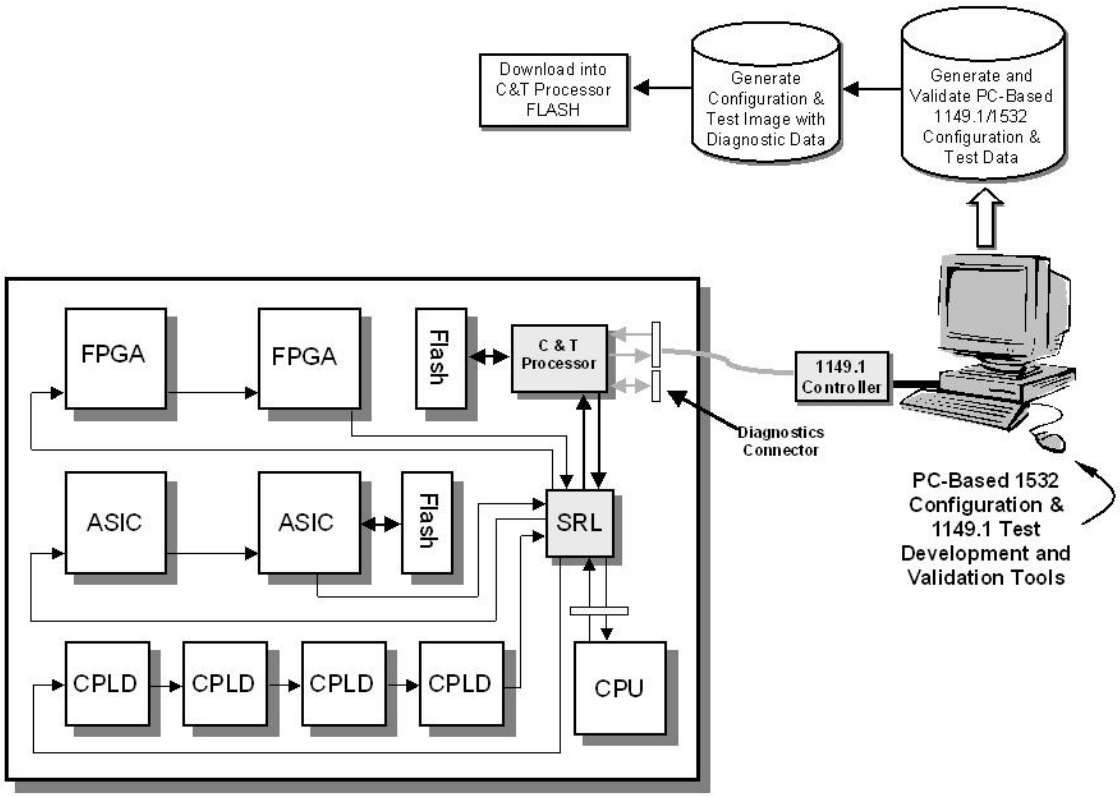
A dedicated configuration and test processor is able to manage multiple system configurations, so system re-configuration can take place anywhere and engineering changes can be easily made at any time during a product's life cycle. There are many examples of dedicated, special purpose, processors in systems today. For example, network processors, audio processors, digital signal processors and video processors. For such tasks, dedicated processors offer many advantages over using the mission mode general purpose CPU. In this case, a dedicated architecture for embedded test enables testing of the general purpose CPU and its' support logic, and logging of all failures without the need for the system to function.

For these reasons, we developed a dedicated embedded configuration and test processor (C&T processor) that will function as a centralized manager for configuring and testing PCBs and systems [8]. This 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. First, by including a dedicated embedded configuration and test processor in products, board and system designers can simplify in-system device configuration while enabling comprehensive structural test throughout the system, including the mission mode CPU. The dedicated processor can be provided as an IC or as infrastructure IP, which reduces the design time for engineers. Consequently, the embedded test and configuration processor enables a structured methodology for designing field-configurable and self-testable systems. This reduces test-engineering development effort and reduces test execution time. 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.

Centralized Management for Embedded Configuration and Test

The configuration and test 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

Figure 3. Embedded Configuration and Test (C&T) Processor for a single PCB system



subsequently automatically embedded into the system. The processor is a vendor independent solution, eliminating the need for proprietary PROM or FLASH based solutions. As infrastructure IP for configuration and test of boards and systems, designers no longer need to develop customized methods and designs for embedded in-system solutions. A single configuration and test processor at power-up, or under CPU control, can automatically run the entire manufacturing test stream, including scan 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 configuration and test processor can be used at the board level. This illustrates how the configuration and test processor connects externally to automated development tools that are used for developing and validating configuration data and test programs. The flow for development and validation with this architecture is also show in Figure 3, which is much simpler than the embedded test flow required in Figure 2.

The architecture 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) [6] that partitions the scan paths at the board level.

As can be seen by comparing Figure 3 with Figure 1, the embedded configuration and test 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 an SRL. The process 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 development and validation tools. This is a major advantage in that the external Boundary Scan tools can communicate through the processor directly. Therefore, 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 tools 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. The result is that only one configuration and test validation step is needed, eliminating the need to re-validate the vectors 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. Then the external IEEE 1149.1 equipment is disconnected from the PCB and the processor assumes control of running the embedded test suites and programming the FPGAs.

Efficient Storage Architecture

The PC based software tools enable the developer to create test and configuration 'suites' that hold an unlimited number of test vectors, test scripts, flow scripts, and FPGA configuration data files. The PC based software and C&T processor can hold up to 16 suites at one time. The first suite, suite '0' is reserved as the 'reset' suite, enabling a set of procedures to be executed automatically when a failure occurs. The 'reset' suite can be as simple as causing a TEST-LOGIC-RESET or more complex, 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 and more advanced tests such as BIST. Note that complex decision-making is a capability, allowing the C&T processor to identify PCB and System configurations and acting appropriately. Important controls used during PC based 1149.1 test are also executed by the C&T processor, for instance, the TCK frequency can be changed to 1MHZ on-the-fly to program slower devices such as a CPLD. The C&T processor can apply one of these 15 sets of test and configuration 'suites' at power-up.

Figure 4. Example Test Suite

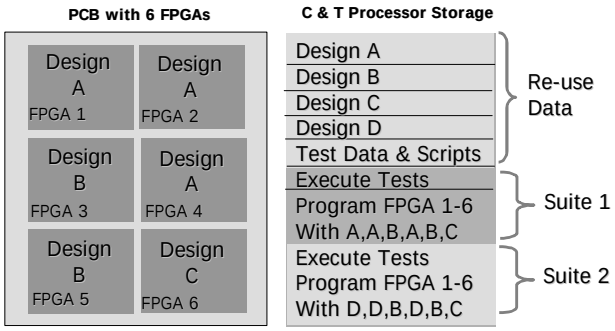
Suite 1

```
1 Check_Scan_Chain.script
2 Interconnect.svf
3 Execute_Bist.script
  Target U1
4 Program_DesignA.bit
5 Check_U2_exists.script
6 Program_DesignA.bit
  Target U3
7 Program_DesignB.bit
```

Also shown is the embedded diagnostic code. As tests and configuration data are added to the suites, a diagnostic code is assigned that will be used by the C&T processor to indicate which test failed during power-up. The test suites and corresponding test select pins of the processor allow the engineer to create one set of configuration and test to be applied at power-up and another test and configuration suite to be applied at a different time, such as during maintenance and updates of the system. A single incremental image generated by the PC based tools could be distributed physically or electronically to customers and uploaded to the storage area of the C & T processor. 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, without writing general purpose CPU code to accomplish the task.

The PC based software 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, as in multi-board systems many of the tests and configuration data are the same. This capability reduces the size of the image beyond the built-in data compression and hence the size of the FLASH needed for storage. For example, consider a PCB where three 16 Megabit FPGAs out of six have the same FPGA design, as illustrated in Figure 5. Using a CPLD and a FLASH to program the FPGAs requires 48Megabits of FLASH. With the C&T processor, compression of the FPGA data results in an approximately 10 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 38 Megabits of FLASH memory. Since many large telecom PCBs have multiple data channels, typically 4, 8 or 16 FPGAs have duplicate design data in them. The C&T 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. An FPGA that performs a DSP function can be loaded with different DSP algorithms based on the target use of the product, with only an incremental impact on storage area.

Figure 5. Processor Storage Example



Using the C&T Processor at the System Level

The configuration and test processor can also be used at the system level, this is shown in the example multi-board system of Figure 6. This system uses a multi-drop 1149.1 bus called the Parallel Test Bus (PTB), along with an addressable Parallel Test Bus Controller (PTBC) on each board. These are described in the Parallel Test Architecture (PTA) references [6], [10], [11]. In this configuration, only the Master/Slave PCB (Type A) has an embedded configuration and test processor. This provides a single, centralized, processor that is dedicated for managing all system configuration and test. Engineers can use the external IEEE 1149.1 controller and embedded processor for validation, before embedding the configuration and test data. As can be seen, 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 (3 Type B), and system level interconnect test.

The embedded configuration and test architecture provides for various mechanisms to report status and output diagnostics information. This includes go/no-go LEDs, 7-segment LED displays for diagnostic failure codes, and a serial port for text-based diagnostics. The diagnostics information, which can be used for field replaceable unit (FRU) identification and PCB-level repair, is embedded in the storage image generated by the PC based software.

A major advantage of this approach is that it uses a "code-less" architecture, which greatly reduces engineering time and enables the same test and configuration vectors developed for prototype bring-up and production to be reused by embedding them in the FLASH memory for use by the dedicated configuration and test processor. Using this approach embedded C++, Java or STAPL software development time is eliminated. Embedded software diagnostic engineers can build software-based test and diagnostics on a solid base of deterministic structural tests and diagnostics provided by the configuration and test processor.

Figure 7 shows a top-level block diagram of the embedded test and configuration processor. This 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.

Figure 6. Top-Level Block Diagram of the C&T Processor

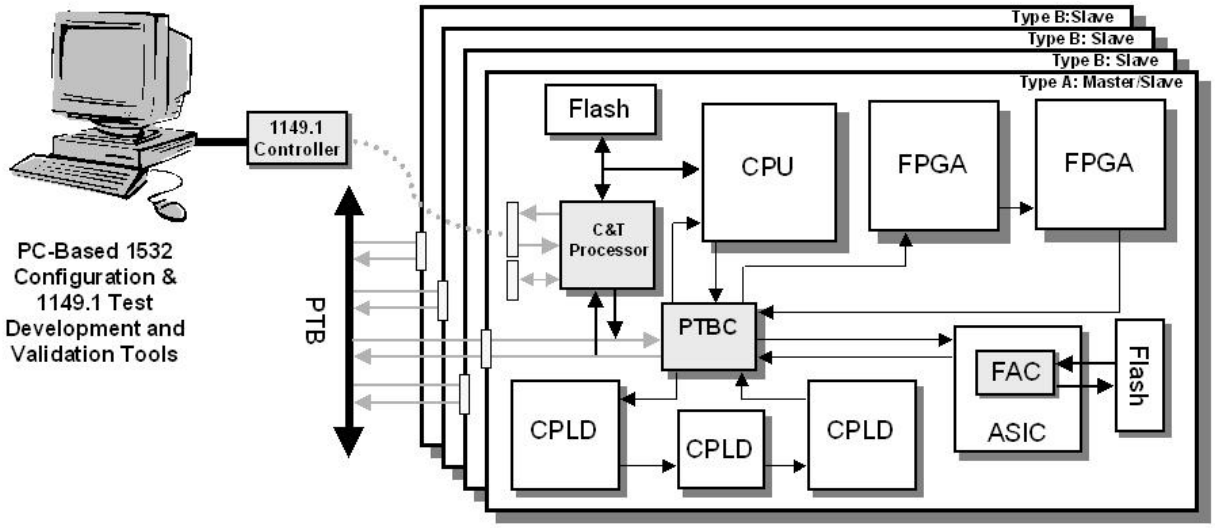
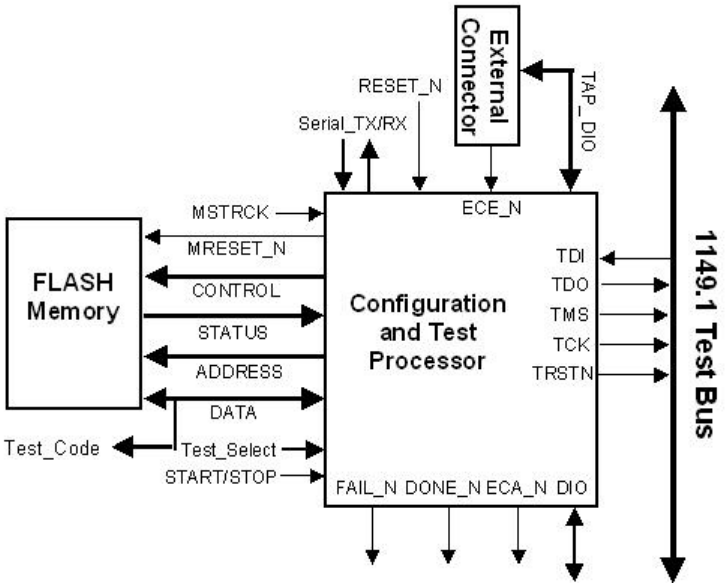


Figure 7. Top-Level Block Diagram of the C&T Processor



In Figure 7**,** the RESET_N signal is used to reset the processor circuitry. When RESET_N is asserted low, the registers and state machines of the processor are reset to appropriate initial states such that it is ready to start applying vectors from the FLASH memory. The MSTRCK input to the configuration and test processor is a master clock that is used to run the processor. MSTRCK 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 Enable (ECE_N) input. When the ECE_N signal is asserted low, the circuitry is reset and the 1149.1 Test Bus and DIO will be controlled from the TAP_DIO signals on 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 configuration and test 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 for this suite 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 sequence, as determined by the 'reset' suite.

The memory interface in Figure 7 consists of circuitry and signals for controlling the FLASH device. The processor can interface to various other types of non-volatile storage, and any number of devices. In the example of Figure 7, a word-based FLASH device is used. Here, the signals function as follows. The MRESET_N output, when active low, resets the FLASH device. The CONTROL signals 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 STATUS inputs are for monitoring the Ready/Busy status of the FLASH. 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.

The results interface of the configuration and test processor contains circuitry that reports the outcome of a test and provides failure and diagnostic information. This interface id 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 it is no longer busy.

- Error_Code: This can output a pass/fail code to the memory interface circuitry. This code will then be driven out on the DATA bus when there is a failure and is used for diagnostics purposes.

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

A feature of the configuration and test processor is that it provides for a user definable error code to be associated with each scan vector suite as was shown in Figure 4. By providing the Error_Code to the DATA bus, the code may be displayed to an LED display, or read by a general purpose CPU connected to the bus. Text messages as developed in the PC based software along with the appropriate Error_Code are provided automatically via the UART port, on the Serial_TX/RX input and output of the processor. The text messages and error codes enable the customer or field service person to diagnose the problem down to the FRU (Field Replaceable Unit) with a low cost PDA. Tests can be made granular enough to allow identification of failing PCBs, plug-in daughter cards, and socketed components. A set of predefined error codes are used for failures of the configuration and test processor itself, such as FLASH is erased, can't read from FLASH, corrupt image and others.

Figure 8. PDA for displaying diagnostics

Failure code logging and failing boundary-scan bits can optionally be written to the FLASH at time of failure. This is particularly useful for systems and PCBs where displaying of the failure data is not possible in the field. When the failing PCB is returned to the factory, software can retrieve the failing bits and display detailed 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.

Conclusions

As Systems and PCBs become more complex, with less physical access, the test model will need to look more like that of the IC. IC designers and IC test engineers solved the lack of visibility into the IC by using structured scan, structured ATPG techniques and BIST. While various approaches have been written in the past, the IC test community has standardized on implementing scan architectures with very little dependency on the functional logic of the IC.

Likewise, we see this as the only direction for PCBs and Systems. Embedded structural test through 1149.1 will be the only way to maintain fault coverage and quality as the complexity of the PCBs and Systems increase and internal visibility decreases. Key to easy and robust test development and configuration is the separation of the infrastructure needed from the functional system logic. The embedded test and configuration processor described here meets that goal. For PCBs that have FPGAs, the test capability can be added with little to no overhead since the processor will replace the parts normally used for programming the FPGAs.

We have shown an embedded test and configuration architecture that can be used by board and systems designers to enable novel new approaches to configuration and test of systems. The problems associated with combining mission mode CPUs and mission mode logic for embedded configuration and test was described. Merging FPGA configuration and embedded test functions onto a dedicated high throughput configuration and test processor offers the best long-term strategy for building re-configurable and self-testable systems.

References

[1] 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.

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

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

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

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

[6] 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.

[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] 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. http://www.uspto.gov

[11] 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. http://ep.espacenet.com