Testability Features of the Alpha 21364 Microprocessor

Scott Erlanger, Dilip K. Bhavsar*, Richard Davies***

*Intel
334 South Street
Shrewsbury, MA 01545

**Hewlett-Packard Company
334 South Street
Shrewsbury, MA 01545

Scott.Erlanger@intel.com Dilip.Bhavsar@intel.com RDavies@hp.com

Abstract

The custom testability strategy of the Alpha 21364, Hewlett-Packard's most recent Alpha microprocessor, builds upon its Alpha 21264 embedded core. Several additional DFT features integrate to meet the testing challenges of the new generation.

1. Introduction

The Alpha 21364 [1] is the fourth generation Alpha microprocessor. This 139 million transistor 1443-pin chip, originally a 0.18 μm bulk CMOS device, has been recently implemented using 0.13 μm SOI technology [2].

The chip design combines the previous generation Alpha microprocessor (Alpha 21264 [3]) as an embedded core with a 1.75 MB second-level cache, two high-performance memory controllers and an on-chip interprocessor router. Figure 1 shows the chip photomicrograph.

Like its predecessors [4], the Alpha 21364 microprocessor follows a custom testability strategy. That is, the total test solution consists of a blend of custom design for test (DFT) techniques, handcrafted test patterns, and test engineering tools and procedures. Each aspect of the test methodology specifically targets a unique testability goal.

The implementation of this strategy on the Alpha 21364 faced several interesting challenges. First, since the chip embeds the previous generation microprocessor as its core, new DFT solutions needed to seamlessly integrate with the inherited DFT and testability access infrastructure. Second, the chip's high performance interfaces and large number of pins required special attention to ensure that the chip is testable on affordable testers. Third, aggressive time-to-market goals necessitated the need for rapid prototype debug capabilities. Finally, unlike previous generations, the Alpha 21364 relies entirely on external foundries in its production. This created additional test requirements including the need for significantly enhanced debug and diagnosis, increased production test pattern suite stability, and chip testability on automated test equipment (ATE) available at the foundry.

This paper focuses on the testability features of the Alpha 21364 microprocessor. A companion paper [5] details the chip's debug features. The paper is organized as follows: Section 2 describes the testability features, Section 3 briefly describes the testability access architecture, Section 4 summarizes some key results, and Section 5 concludes the paper.

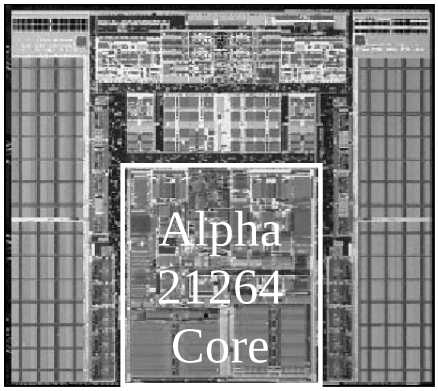


Figure 1: Alpha 21364 Microprocessor

*The work for this paper was performed while the authors were with Compaq Computer Corporation/Hewlett-Packard Company.

2. Testability Features

The Alpha 21364 employs a two-level hierarchical test organization. In this scheme, similar test targets collectively form a test target group. The test targets within a group employ similar, but dedicated testability resources. Members of a group are daisy-chained and transport data and results serially. A single central controller interfaces with the test features using a small number of wires. External access to the test features is provided via an IEEE 1149.1 test access port (TAP) [6] interface.

Figure 2 shows the top-level view of the Alpha 21364 testability feature organization. The features located in the Alpha 21264 core are shaded in black.

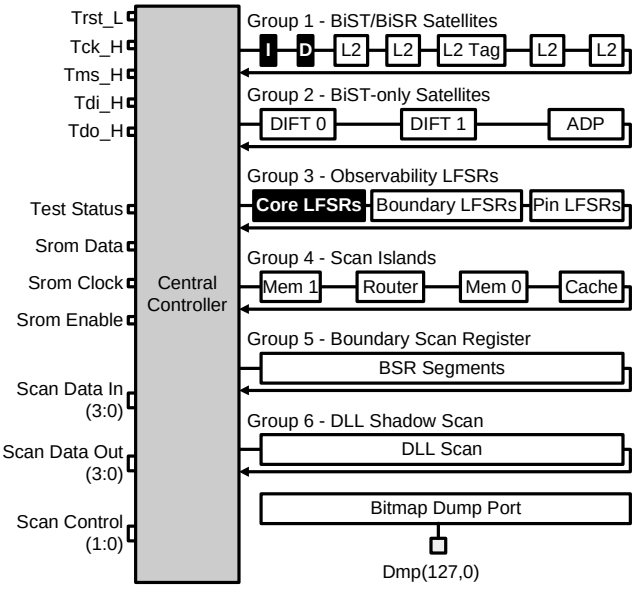


Figure 2: Alpha 21364 Testability Features

Group 1 - BiST/BiSR Satellites

This group includes seven test satellites that control the testability features of all major RAM arrays. These arrays include: the L2 cache and tags, L1 instruction cache and tags, and L1 data cache, tags and ECC. Due to its size and location on the die, the L2 cache contains four quadrants, each accessed by its own test satellite.

The Group 1 satellites support the following test operations:

- Automatic power-on self-test
- Quick-initialization upon special reset
- Freeze and dump for system debug on certain arrays
- Manufacturing self-test and self-repair
- Retention self-test
- Bitmap

Each of the Group 1 arrays performs self-repair using redundant state elements. To further improve yield, the larger arrays employ two-dimensional self-repair [7]. The self-repair allows a one-pass wafer probe, streamlining chip production. The Alpha 21364 supports permanent cache repair through the use of laser-blown fuses.

The Group 1 arrays allow the bitmap operation to be performed while performing BiST at full speed. Throughout the BiST operation, the bitmap data is serially transported to the pins and outputted via a 128-bit bitmap dump port. The bitmap dump port shares its pins with the chip's functional interface.

Group 2 - BiST-only Satellites

This group is comprised of three satellites targeting two small RAM arrays located in the memory controller unit and a RAM/CAM combined array in the L2 cache controller data path. These arrays are an order of magnitude smaller than the Group 1 arrays. Consequently, they include no repair overhead.

The Group 2 satellites support a wide assortment of features to achieve their test goals as well as aid in the debug process including:

- Automatic power-on self-test
- Freeze and dump for system debug

- Manufacturing initiated self-test, which allows viewing of pass/fail status of individual satellites
- Address map

Group 3 - Observability Registers

The Alpha 21364 includes observability registers [8] based on linear feedback shift registers (LFSRs). They serve three objectives: enhance the fault coverage of the hand-crafted functional tests, support reduced-pin count testing, and provide internal observability for chip and system debug. The chip contains observability register nodes at the boundary of and throughout the Alpha 21264 core as well as at the system bus interfaces. In total, the Alpha 21364 has 86 LFSR chains observing approximately 6000 chip nodes.

The Alpha 21364 partitions the observability registers into several independent LFSR segments, which are daisy-chained together for serial access. This organization provides several benefits. It confines and contains the impact of non-deterministic data resulting from the capture of uninitialized state. It increases failure isolation. Finally, it facilitates debug by allowing the bypass of all LFSR segments except those of interest.

The observability registers support two primary modes of operations single-cycle snapshot mode used for debug and multi-cycle signature mode used for production test. In both cases, the observability registers operate at full CPU speed, allowing data capture at any CPU cycle. The IEEE 1149.1 TAP and the LFSR control logic located inside the central controller control the oberservability register operation.

Group 4 - Scan Islands

The Alpha 21364 implements partial scan in the new logic outside of the Alpha 21264 core. While the primary goal was to facilitate rapid debug and diagnosis, the implementation also allows leveraging scan for a variety of production test purposes, including ATPG-based patterns.

The Alpha 21364 supports the following scan test modes:

- Traditional scan testing from tester
- At-speed manufacturing scan-BiST from on-chip random pattern source
- Manufacturing scan with patterns supplied from tester via the scan wheel [9]
- Quick-initialization of the scannable state upon powering up
- A dump of scannable state for system debug

To increase the effectiveness of ATPG and test operation, the partial scan implementation employs a partitioning scheme called scan islands [10]. This architecture allows ATPG to be performed on partitions independently. During production test, islands may be tested separately and simultaneously or chained together to form an archipelago, which permits the testing of interfaces between islands.

The IEEE 1149.1 TAP and the central scan controller located inside the central controller control the scan operations. A scan interface port called an isle scan port provides local control to each island and its scan chains.

The Alpha 21364 has four scan islands, each with four parallel scan chains. The penetration of scan in the islands is in the range of 25-40%. The total number of scan-flops is approximately 13,000.

Group 5 - Boundary Scan Register

The Alpha 21364 incorporates a boundary scan register (BSR) architecture to facilitate testing of its high performance interfaces. It supports three main objectives:

- Traditional board-level interconnect testing using IEEE 1149.1 EXTEST and SAMPLE instructions
- AC-parametric testing for characterization and production screen of high performance interface pins on a limited-capability tester
- Observability of unconnected output pins during functional testing in a reduced-pin tester environment

AC-parametric testing of high performance pin interfaces is a challenge that is new for this generation of Alpha. Three features play together to meet this challenge. First, all high performance interface pins incorporate on-chip loop-back capability [11]. To facilitate this, all pins are implemented as bidirectional. This required adding test receivers to output-only pins and test drivers to all input-only pins. Second, the pin interface and internal registers provide control of the skew of the driver and receiver clocking. This allows margining data loopback across a range of driver and receiver clock frequencies. Finally, the BSR structures allow an on-chip method for test stimuli generation and response evaluation.

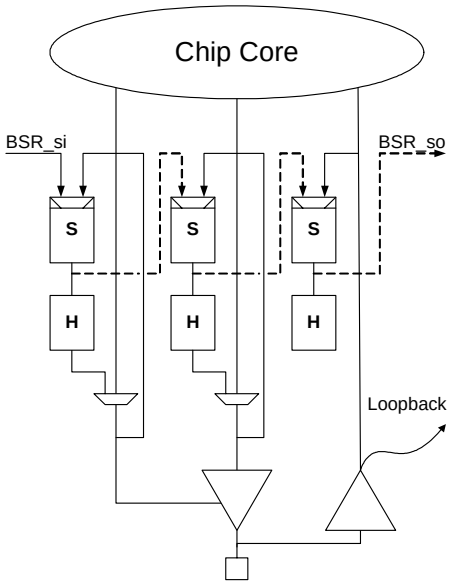


Figure 3: Alpha 21364 BSR Structure

Figure 3 shows the BSR structure at each pin. It consists of three BSR cells, one each for incoming data, output data and the pin direction control. Each BSR cell has a shift flip-flop and a hold flip-flop. The six flip-flops present in the BSR cell may be reconfigured to form a 6-bit LFSR.

This configuration is known as pin-LFSR mode. The LFSR of the pin under test generates the test stimulus data while the LFSR of an adjacent pin captures a signature using the loopback data. Switching the compressor and generator roles allows testing of the full interface in two passes. The generation and receiving of data is entirely contained on-chip and performed at the full speed of the pin interface. AC-parametric testing of the pins is thus achieved with reduced dependency upon the ATE timing accuracy.

Another mode of operation, called chip-LFSR mode, concatenates all pin-LFSRs of a high-speed interface to form a very wide LFSR. This mode captures and compresses the data driven to the output pins, allowing functional testing on a reduced-pin tester environment. The chip-LFSR mode comprises an integral part of the Group 3 observability LFSRs described earlier.

The per pin tristate control provides the ability to perform DC-parametric testing in a reduced-pin tester environment by ganging several unconnected pins together on a single tester channel. Each pin is enabled one at a time, while all of the other pins sharing the tester channel are tristated. This allows the serial DC-parametric testing of all outputs using a reduced number of tester channels.

Included within the boundary scan shift path are special control registers that configure the pins under test. This includes features such as driver strength selection, driver enabling, and pin termination control. Although this is a deviation from the IEEE 1149.1 standard, the boundary scan offers a convenient access mechanism to these controls and greatly improves the testability

of the pin interface both during chip manufacturing and board-level testing.

Group 6 - DLL Shadow Scan

The Alpha 21364 possesses several delay-locked loops (DLLs) in its high-speed pin interfaces and in the new logic outside of the 21264 core. The DLL shadow scan architecture allows access to all DLLs operating in different clock domains via a single scan-path. This technique provides a basic functionality to characterize and test the digital logic and the digital/analog (D/A) interfaces present in these DLLs. Specifically, DLL shadow scan facilitates three test objectives:

- Testing of the DLL digital logic via ATPG-generated test patterns
- Providing direct access to digital DLL control logic to facilitate testing of the analog logic
- Observability of DLL digital state during normal chip operation

The architecture implements special DLL shadow scan-flops at the D/A interfaces and throughout the digital logic of the DLL. Figure 4 shows the DLL shadow scan-flop. It consists of two flip-flops: a system flop from the normal DLL interface and a shadow flop that is part of the DLL scan path. The shadow flops provide the serial scan path to load and unload data to and from the DLL logic under test. The system flops receive test data from the scan path and also load the scan path with the circuit response.

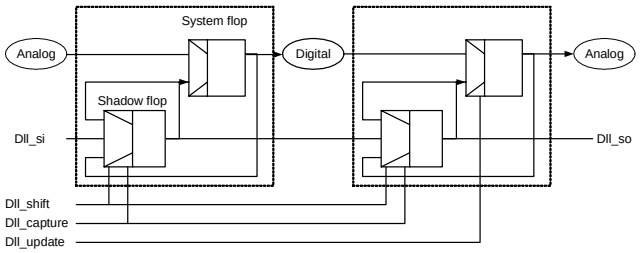


Figure 4: DLL Shadow Scan

The IEEE 1149.1 TAP controls the operation of the DLL shadow scan architecture. The TAP controller initiates all data transfers between the system and shadow flops. The system flops receive data from the shadow flops upon leaving the Update-DR state. The system flops transfer data back into the scan path upon entering the Capture-DR state.

The Alpha 21364 has ten high performance ports, each with multiple DLLs operating in their independent clock domains. This offered the unique challenge of stringing shadow scan-flops in different ports into a single chain.

The DLL shadow scan architecture mitigates this challenge by synchronizing the test control signals locally in each port with the clock domain of the DLL under test (DCLK). The test data crosses clock domains when loading from the shadow flops into the system flops. Likewise, the response data crosses back into the scan path clock domain (NCLK) when captured by the shadow flops. The entire shadow scan path shifts within the NCLK domain. Internally to the chip, the TCK pin is synchronized with NCLK. From the standpoint of the user, the DLL shadow scan appears to operate entirely using TCK even though it actually accesses several clock domains.

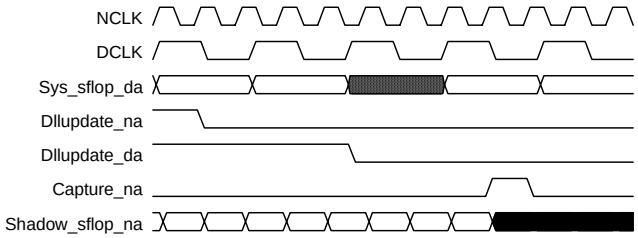


Figure 5: DLL Digital Test Timing Diagram

Figure 5 illustrates an example of the DLL digital logic test operation. The deassertion of the update signal from the central controller (Dllupdate_na) in the NCLK domain begins the operation. Synchronization logic crosses this signal into the DCLK domain (Dllupdate_da).

Shadow scan data is transferred to the system flip-flops upon the falling edge of this signal.

Once the digital logic under test reaches a stable state, the shadow scan may capture the response. The capture command from the central controller begins the response sample operation. The assertion of the Capture_na signal transfers the response data from the system flops into the shadow flops that will shift the data to the pins.

In addition to testing the digital logic, the DLL shadow scan also provides test access to the reference loop clock generators and the phase mixing logic. Access to these circuits provides the ability to manually force the generation of different clock waveforms and provides observability into the internal DLL state.

3. Testability Access Architecture

To address the challenge of accessing test features scattered widely across the chip using a minimum number of wiring channels, the Alpha 21364 employs a two-tier hierarchical access scheme.

At the top-level is the central controller, which broadcasts global control the test target groups. All targets within a group respond identically to global broadcasts. The global broadcasts transfer test data and results and initiates test actions.

At the bottom-level are the local controllers located at each test target. One or more test control registers in a local controller provide customized test target-specific control as well as any parameters for test operation.

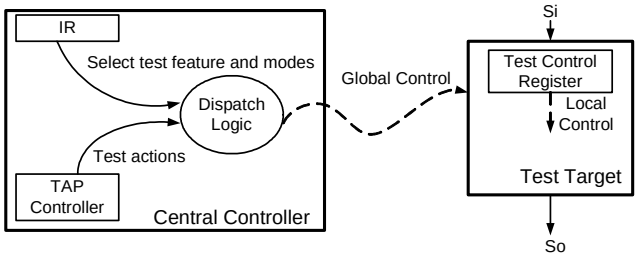


Figure 6: Testability Access Architecture

Figure 6 illustrates the control organization. A single IEEE 1149.1 TAP provides the interface to the outside world. The TAP instruction-register selects testability features and test modes. The TAP controller initiates all test operations as well as controls the shifting of data and results to and from the test satellites. The central controller broadcasts the instruction register and TAP controller state to the testability features using a small number of wires. As mentioned earlier, the testability features themselves are daisy-chained to transport test data and results via a serial shift register path.

The testability access architecture faced a unique challenge in interfacing the internal scan chains with external testers. All scan-flops on the Alpha 21364 use a data-mux implementation. Consequently, the scan chains shift at full speed with the normal CPU clock. Ordinarily, a tester cannot source or sink data such a high rate.

The Alpha 21364 solves this problem with the innovative scan wheel interface [9] that allows a slower speed ATE to load/unload scan chains operating at full CPU speed. Connecting the last scan-flop to the first scan-flop in a chain forms a scan wheel. This circulates the captured data to keep it alive in a closed loop. A sampling mechanism unloads and injects new data into the scan wheel periodically until all scan-flops have been read and written.

In addition to interfacing the scan chains with a slow speed tester, scan wheel provides a more general-use mode of viewing data generated at the full CPU speed using a slow ATE interface. The dumping of arrays and observability registers use the scan wheel interface.

4. Results

In all, the testability features of the Alpha 21364 use approximately 150K transistors (not including scan overhead) and occupy approximately 2% of the total die area. The testability access architecture uses eight

dedicated pins and shares the remainder with the normal functional pins. Synthesis and automatic place and route tools implemented the majority of the testability logic.

All of the testability features are operational and have been fulfilling the Alpha 21364 testing and debugging needs since first silicon.

The Group 1 and Group 2 satellites have been useful from both a test and debug standpoint. Together, they successfully self-test over 116 million of the 139 million devices on the chip. The self-repair has allowed a one-pass wafer probe, eliminating an extra testing step and reducing test costs.

The ability to freeze and dump the contents of the L2 cache and tag arrays and the Group 2 arrays has been extremely valuable in both chip and system debug. The Alpha 21364 also contains an on-chip logic analyzer that writes information related to program execution and memory accesses to the L2 cache. When used in combination with the array dump, this ability has provided essential information for the debug of the chip.

The array bitmap and address map have also been used for debug and yield engineering. In particular, the Group 2 address map helped in tracking down CAM-cell voltage sensitivity.

The observability registers at the high performance pin interfaces allow the 1443-pin Alpha 21364 to be thoroughly tested at wafer-probe using a 512-pin tester, thus weeding out the defective parts early and saving the cost of expensive packages. The fault coverage gains from the other internal observability registers is not explicitly measured, but based on previous experience, is speculated to be in the range of 1-3%.

The implementation of partial scan and the scan island architecture yielded mixed results. The scan island architecture itself has proven to be an extremely effective mechanism for the generation and delivery of ATPG patterns. Commercial ATPG tools generated the ATPG patterns using scan island-level wirelists. The ability to partition the chip into smaller sections greatly eased the burden on the ATPG tools and sped up the pattern generation process. Adding a simple pattern header easily converted the scan island ATPG patterns to chip-level patterns. All ATPG patterns generated for each scan island ran successfully at first silicon.

Despite the success of the scan architecture itself, the ATPG coverage is not at all impressive. The 15-40% penetration of scan in the islands proved to be too little to obtain any meaningful coverage. However, this has no consequence on the Alpha 21364 coverage levels as the overall test strategy relies on at-speed functional tests for production screens. The design and test teams learned several valuable lessons in implementing and using scan.

Similar to scan, the scan-BiST feature yielded mixed results. The grossly partial scan did not yield meaningful coverage. However, scan-BiST turned out to be extremely useful for burn-in testing. Its pseudo-random patterns generate toggle activity on 50-60% of the nodes throughout the scan islands.

The scan wheel mechanism paid off in two ways. First, it allowed ATPG patterns to be run using a slow ATE interface. Second, the ability to dump the scannable state to the pins was vital in tracking down several bugs.

The Alpha 21364 is the first Alpha microprocessor to extensively exploit many test features for the purpose of system debug. A simple notebook PC-based IEEE 1149.1 master controller provides access to the test features in system environment. This gives a direct method to validate and debug chip functionality without relying on system firmware. This access mechanism found and resolved many production-related problems early in the prototype debug process. This backdoor test access also provides a higher degree of precision and easier bug

diagnosis when compared to software-based approaches.

Both ATE and system-based testing utilize the boundary scan logic heavily for testing the chip interfaces. ATE and system board-level test use the BSR structures for their AC-parametric and DC-parametric test capabilities as well as characterizing signal integrity.

The DLL shadow scan has been used extensively for both circuit characterization and debug. The DLL state observability provided invaluable insight used to validate the DLL operation. The ability to drive the analog DLL circuitry with test data was also used for characterizing the frequency response of the chip DLLs. This insight into the actual operation of the DLLs allowed refinement of the circuit simulation models and helped to understand the impact of manufacturing process upon the DLL performance.

5. Concluding Remarks

The paper presented the testability strategy and the testability features of the Alpha 21364 microprocessor. This strategy builds upon the custom testability strategy of its embedded Alpha 21264 core and adds several new features to solve the testability problems of the new generation. The features of the two generations seamlessly integrate to present one unified testability solution for the entire chip. All features are operational in manufacturing and delivering their promised functions. Furthermore, the design and test teams learned several valuable lessons that they will carry forward to future generations of microprocessors.

The Alpha 21364 testability architecture boasts several innovations. The two-level hierarchical organization of daisy-chained test targets provides uniform testability access over a minimal number of global wires. The two-dimensional row-column self-repair, observability register architecture, scan island partitioning, scan wheel interface, and comprehensive BSR and DLL shadow scan architecture are all innovations that play together on the Alpha 21364 to solve a variety of test and debug challenges.

The Alpha 21364 testability features represent a stage in the continuous evolution of the Alpha microprocessor test strategy. Although this is the last Alpha microprocessor, we hope the techniques developed and lessons learned will find their way into future microprocessor designs.

Acknowledgements

We express our gratitude to W. Anderson, A. Angle, P. Bannon, V. Bettada, J. Gebhart, D. George, D. Gowda, B. Gran, S. Kabadi, P. Kovijanic, A. Kumar, T. Litt, R. Mueller, S. Nadig, B. Nguyen, S. Russell, R. Russo, M. Safer, M. Shen, M. Steinman, R. Tan, S. Tomljenovic, and the entire Alpha 21364 design team for their help in making this project successful.

References

[1] A. K. Jain, *et al.*, "A 1.2 GHz Alpha Microprocessor with 44.8 GB/sec of Chip Pin Bandwidth," *Int'l Solid State Circuits Conf.*, pp. 240-241, February 2001.

[2] J. A. Kowaleski Jr, *et al.*, "Implementation of an Alpha Microprocessor in SOI," *Int'l Solid State Circuits Conf.*, pp. 248-249, February 2003.

[3] B. Gieseke, *et al.*, "A 600 MHz Super-scalar RISC Microprocessor with Out-of-Order Execution," *Int'l Solid State Circuits Conf.*, pp. 176-177, February 1997.

[4] D. K. Bhavsar, *et al.*, "Testability Access of the High Speed Test Features in the Alpha 21264 Microprocessor," *Int'l Test Conf.*, pp. 487-495, October 1998.

[5] T. Litt, “Debug Support in the Alpha 21364 Microprocessor,” *Int’l Test Conf.*, pp. 584-589, October 2002.

[6] IEEE Std.1149.1 – 1993 “A Test Access Port and Boundary Scan Architecture”.

[7] D. K. Bhavsar, “An Algorithm for Row-Column Self-Repair of RAMs and its Implementation in the Alpha 21264,” *Int’l Test Conf.*, pp. 311-318, September 1999.

[8] D. K. Bhavsar and R. Tan, “Observability Register Architecture for Efficient Production Test and Debug of VLSI Circuits,” *IEEE/ACM 14th Int’l Conference on VLSI Design*, pp. 385-390, January 2001.

[9] D. K. Bhavsar, “Scan Wheel – A Technique for Interfacing a High Speed Scan-Path with a Slow Speed Tester,” *VLSI Test Symposium*, pp. 94-99, April 2001.

[10] D. K. Bhavsar and R. A. Davies, “Scan Islands – A Scan Partitioning Architecture and its Implementation the Alpha 21364 Processor,” *VLSI Test Symposium*, pp. 16-21, April 2002.

[11] M. P. Kusko, *et al.*, “Microprocessor Test and Test Tool Methodology for the 500 MHz IBM S/390 G5 Chip,” *Int’l Test Conf.*, pp. 717-726, October 1998.

THE TESTABILITY FEATURES OF THE ARM1026EJ MICROPROCESSOR CORE

Teresa L. McLaurin, Frank Frederick, Rich Slobodnik
ARM Inc.
1250 S. Capital of TX Hwy, Bldg 3, Ste 560
Austin, TX 78746

Abstract
The DFT and Test challenges faced, and the solutions applied, to the ARM1026EJ microprocessor core are described in this paper. New DFT techniques have been created to address the challenges of distributing a DFT core solution that will ultimately end up in many different environments. This core was instantiated into a test chip. The new DFT features were utilized successfully in the SoC.

1. Introduction

The ARM1026EJ is a member of the ARM10 family of cores and implements the ARMv5TEJ architecture. It is a high performance, low power, cached processor that provides full virtual memory capabilities. It is designed to run high-end embedded applications and sophisticated operating systems such as Linux, Microsoft, WindowsCE, NetBSD and EPOC-32 from Symbian. It supports the 32-bit ARM, 16-bit Thumb and 8-bit Jazelle instruction sets.

The test chip was processed in .13 micron technology. Memory for an external IP vendor was incorporated into the core and the test chip.

The ARM1026EJ is currently available as a soft core, but the DFT methodology put in place is designed for both a soft or hard core implementation. More information must be supplied in order for the soft core users to attain the same coverage as was achieved in-house, but the core user can choose the technology. A hard core requires DFT configurability to fit the integrator's test needs, while a soft core should allow enough flexibility for the integrator to choose their own DFT methodology. Soft cores and hard cores deliver different benefits and the SoC designer must decide which scenario is better for them.

This paper will mainly discuss the DFT methodology of the ARM1026EJ hard core. This paper will also briefly discuss the instantiation of this core into a test chip that includes the Embedded Trace Macrocell (ETM10RV) and the DFT methodology for the test chip.

2. Goals

The main DFT goals were high coverage of both logic and RAMs, configurability of some of the DFT, such as scan chains, and methodologies that allowed for easier connectivity checking after instantiation. It is always important to keep the costs low and the timing impact to the functional design to a minimum.

All patterns must be able to use a minimal pin set so that the core can be instantiated into many different package and tester environments. We also had to create patterns and methodology to address delay defects and speed sorting capabilities.

3. Strategies

3.1 ARM1026EJ Strategy

The DFT strategies used to meet the stated goals for the ARM1026EJ were addressed with scan and the ARM memory BIST. Only patterns for these two scenarios are delivered with the hard core. ATPG scan patterns are generated for this core that include stuck-at, transition and path delay. Wrappers are segmented functionally to address other issues that will be discussed later in the paper. There is also a scan chain configuration block included.

In addition to the test patterns, speed and power indicative functional source code is delivered. A core customer can choose to employ these if they feel it is necessary.

This paper will describe the DFT challenges and solutions related to the following areas:

- Wrapper functional segmentation.
- A wrapper with the capability to put at-speed multiple values into the input paths for delay testing.
- An ARM memory BIST controller designed and added for maximum coverage of the memories and

the ability to choose algorithms not run in our “go-nogo” test.
- A way to reduce power during external testing without having separate clock domains
- A memory BIST controller that allows for address scrambling to account for different physical mapping of different memories.
- Configurable scan chain capability.

3.2 ARM1026EJ Test Chip Strategy

The ARM1026EJ test chip strategy allows the test chip to stay on a lower cost tester and sets up a strategy for testing all of the pieces of the test chip.

This paper will describe the DFT challenges and solutions related to the following areas:

- Logic attaching to the PLL that allows for slow shift and at-speed launch-to-capture.
- Logic allowing for single scan chain throughout the test chip for IddQ or burnin.
- Test strategy for entire test chip.

4. ARM1026EJ Wrapper Methodology

The ARM1026EJ instantiates a test wrapper to isolate the core, much like the IEEE P1500 wrapper boundary register [1]. This allows control and observe to the core as well as to the logic adjacent and external from the core.

4.1 Background

The test wrapper incorporates some methodologies that have been seen in other cores. In addition, a couple of new capabilities have been added to the wrapper. The new wrapper methodologies allow segmentation of the wrapper chain by function and a way to reset the core without resetting any of the shared wrapper cells. This paper will briefly review old wrapper methodologies and discuss the reasons and structure of the new methodologies.

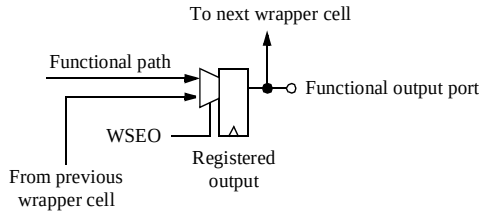
4.2 Strategy

The goal of the wrapper strategy was to implement an isolation ring around the core with minimal impact to area and timing. In addition, the capability for delay testing of the input paths was needed. Some new methodologies were derived to address power concerns and consideration of peripheral cores that could be attached to the core. The ports to the tightly couple memories are not wrapped. The methodology to test the logic attached to these ports will be discussed later in the paper.

4.3 Implementation

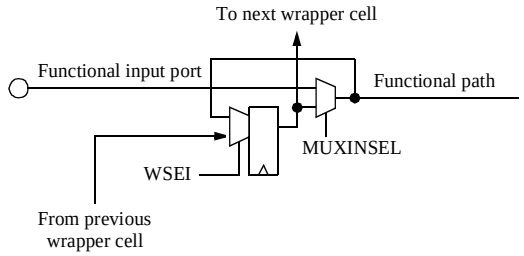
The test wrapper utilizes both dedicated and shared wrapper cells. A shared wrapper cell is one that serves a dual purpose of both functional mode and the control and observe function needed for the wrapper cell. A shared wrapper cell can only be used on a input or output that is registered. Figure 1: "Shared Output Wrapper Cell" shows an example of a shared wrapper cell. The advantage of shared wrapper cells is that less flip-flops must be added, there is no mux delay in the functional path and external test mode can test the timing of the actual path.

Figure 1: Shared Output Wrapper Cell



Inputs and outputs that are not registered use dedicated wrapper cells as shown in Figure 2: "Dedicated Input Wrapper Cell".

Figure 2: Dedicated Input Wrapper Cell



4.3.1 WSEI and WSEO

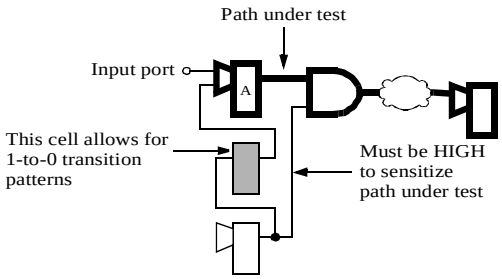
There are two wrapper scan enables. One wrapper scan enable, WSEI, connects to wrapper cells adjacent only to input ports. The other wrapper scan enable, WSEO, connects to wrapper cells adjacent only to output ports. The primary reason that the wrapper scan enables are separated is to allow the capability of delay testing on the input paths of the core. It also allows the same type testing to be done on paths external to the core and connected to the outputs of the core.

During internal test mode, the WSEI can be held in shift mode for all of the patterns. There is never a need to capture into the input wrapper cells during the internal test mode. However, in most cases there is no issue with allowing WSEI to toggle during internal test mode. The shift only capability give us the ability to input data pairs (1-to-0 or 0-to-1) without having multiple cells per input port.

Delay testing requires data pairs to be input during the

capture cycles. Constraining WSEI active during the capture cycle allows the tool to shift data the pair via the wrapper chain. Some cores put a dual flip-flop wrapper cell on the wrapper to allow for the data pairs to be input accurately [2]. If the scan enable on the ARM1026EJ input wrapper cells was to enable captures, then the second clock of the capture would hold the previous value in the wrapper cell. Note that the same input wrapper cell that delivers the second piece of data from the data pair may also affect sensitization of the path under test and prevent that path from being tested. In this case, the wrapper cells would be rearranged or a second wrapper cell could be added in to allow for the second piece of data to be delivered (see Figure 3: "Example Input Path with Extra Flop").

Figure 3: Example Input Path with Extra Flop



4.3.2 Wrapper Segmentation

The ARM1026EJ microprocessor core can be partnered with other peripheral cores such as a coprocessor core or the embedded trace macro module. The ARM1026EJ microprocessor wrapper can be configured such that it can be used for the peripheral cores and the peripheral cores will not need their own wrappers. This requires compliance across a microprocessor core family in order to be successful. The benefit of not having a wrapper on the peripheral core is faster interface timing on inputs and outputs that are not registered and smaller, simpler peripheral cores. The ARM1026EJ wrapper is functionally segmented into three separate wrapper chains to accommodate these requirements (see Figure 4: "Wrapper Segment Example").One segment interfaces with the ETM10 core, one segment with the coprocessor core logic and one segment interfaces with the user-defined logic (UDL). The WMUX signals are used to manipulate the wrapper chain.

Figure 5: "ETM10RV/ARM1026EJ Common Interface" shows an example of how the coprocessor core would attach to any ARM10 processor core. All ports of the coprocessor core attach to the microprocessor, so the coprocessor does not require a wrapper at all. The patterns that are created for a hardened coprocessor core utilize one functional portion of the ARM1026EJ wrapper. All subsequent ARM10 cores will have the exact same wrapper segment so that the coprocessor vector set will work with any ARM10 core attached.

Figure 4: Wrapper Segment Example

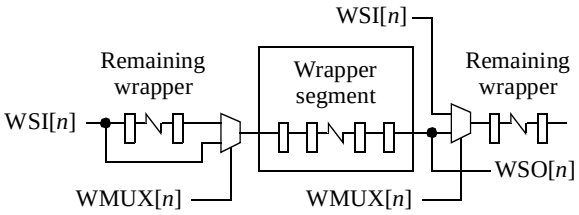
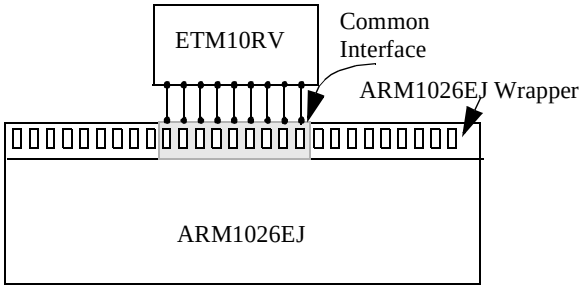


Figure 5: ETM10RV/ARM1026EJ Common Interface

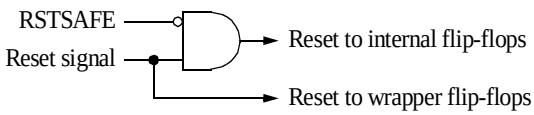


4.4 RSTSAFE

The ARM1026EJ contains a single clock domain. The ARM1026EJ wrapper includes shared wrapper cells and is part of this single clock domain. It would be ideal to have the wrapper on a separate wrapper clock domain, so that during external testing mode the bulk of the core could be made quiet (no clock). This would reduce the power during test. Since this was not feasible, a methodology was developed to help save power during test.

One way to help reduce power during external test mode is to put the core in reset. However, many of the shared wrapper cells have reset ports. These wrapper cells cannot be in reset during external test mode. So, a signal (RSTSAFE) was created to reset only the resettable core flip-flops and not the wrapper flip-flops. Figure 6: "RSTSAFE Example" shows how the active low reset is handled on the ARM1026EJ.

Figure 6: RSTSAFE Example



5. ARM Memory BIST

5.1 Background

As a silicon IP provider, ARM has an assortment of customers all with different requirements for memory test. The challenge was to provide a solution that would offer both a high quality test and considerable flexibility in order to satisfy the requirements of as many customers as possible. The decision was made to provide a custom ARM solution rather than using third party IP from an EDA supplier for two main reasons: No currently available solution offered the flexibility and breadth of test choices desired, and the potential legal and financial complexity of delivering non-ARM IP to customers was unattractive.

5.2 Strategy

The ARM memory BIST solution is separated into a main controller and one or more dispatch units. The function of the controller is to provide an interface to the tester and issue instructions to the dispatch units. The dispatch units perform the RAM accesses, compare the read data, and store the results for analysis. This architecture allows timing critical logic associated with the RAMs to be physically located near the RAMs without duplicating the entire controller multiple times. It also facilitates IP reuse since only the dispatch units are design specific.

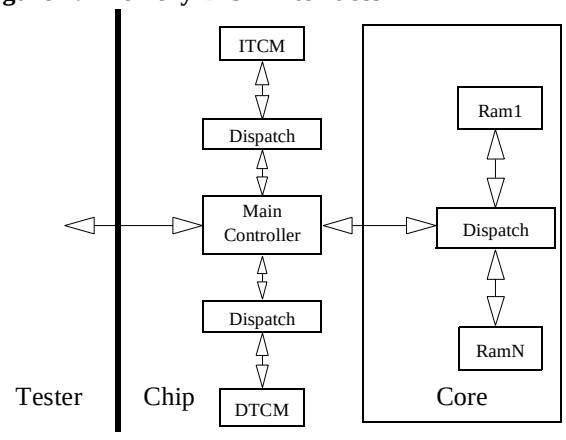
5.3 Implementation

This sections details the implementation of the main features of the ARM memory BIST solution.

5.3.1 Interfaces

The communication interfaces for the memory BIST are shown in Figure 7: "Memory BIST Interfaces".

Figure 7: Memory BIST Interfaces



The main controller has one port per dispatch unit (three for the ARM1026EJ test chip). The relatively small number of signals in these two interfaces limits the routing requirement for memory BIST and the tester channels required to control it. The interfaces between the dispatch units and the RAMs are design specific, but generally consists of address, write data, read data, write enable, and chip selects.

5.3.2 Memory BIST Instruction Register

The memory BIST instruction register is shown in Table 1.

Table 1: Memory BIST Instruction Register

Pattern		Control		MaxX		MaxY		Dataword		ArrayEn	
37	33	32	28	27	24	23	20	19	16	15	0

The pattern bits encode the algorithm selection. The encodings are shown in Table 2.

Table 2: BIST Algorithm Selections

Size	Description
1N	Write dataword to all entries
1N	Read dataword from all entries
1N	Write alternating data & $\overline{\text{data}}$ to all entries
1N	Read alternating data & $\overline{\text{data}}$ from all entries
14N	March C+: Incr/decr X fast RWR march
14N	March C+: Incr/decr Y fast RWR march
6N	Test BIST failure detection
6N	Incr/decr wordline fast RW march
6N	Incr/decr bitline fast RW march
8N	Incr/decr wordline fast RWR march
8N	Incr/decr bitline fast RWR march
18N	Incr/decr wordline fast bitline stress

The control bits provide for the configuration of the memory BIST engine behaviors such as real time or sticky fail flag and debug mode. The rest of the bits in the instruction register are duplicated in each dispatch unit to save on routing. The MaxX bits store the maximum row address required by the arrays under test and the MaxY bits store the maximum column address required. The row and column address space required by an array depend on the logical to physical address mapping performed on that array (see Section 5.3.3 "Address Scrambling"). The dataword bits store the background data used by the selected algorithms. Finally, the ArrayEn bits control which arrays will be enabled for testing. Any number of arrays can be enabled in parallel during production testing (within the limits of the power rails). During debug mode, only one array can be enabled at a time.

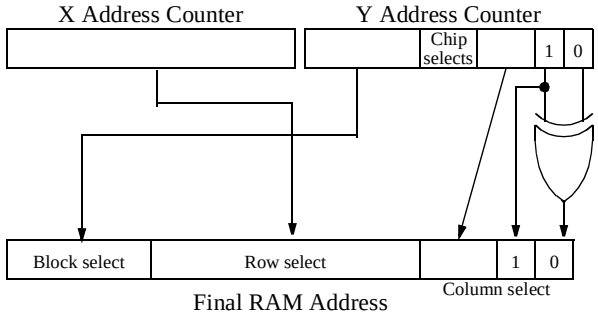
5.3.3 Address Scrambling

For the memory BIST controller to properly perform

bitline stress testing and true physical checkerboard testing, logical to physical address mapping is required, which means the memory BIST must be capable of addressing the physical rows and columns of each array separately. To accomplish this, a separate row (X address) and column (Y address) counter is maintained. One will only change when the other has expired, depending on whether the algorithm selected calls for row fast or column fast operation. These two address counter values must be merged, or “scrambled”, to create the physical RAM address. This requires knowledge of the column width (number of bits the RAM uses for the column select), which can be different for each RAM type. The column widths for the ARM1026EJ RAMs are tied off internally.

A common configuration for memories is to use the LSBs of the RAM address as the column selection. The LSBs of the Y address counter are used for this, with bit zero defined as the XOR of counter bits one and zero (this is to prevent both LSBs of the column address from changing at the same time, which is how columns are physically accessed for timing reasons). The next group of bits are used for the row selection, which comes from the X address counter. A maximum of 256 rows are supported per column. If the array consists of multiple physical RAMs to be tested serially, then the chip selects for those RAMs are assigned to the next bits of the Y address counter after the column selection bits. Finally, any unassigned RAM address space comes from the rest of the bits of the Y address counter. This is also known as the RAM block selection. This is shown graphically in Figure 8: "Address Scrambling".

Figure 8: Address Scrambling



5.3.4 Memory BIST Debug Mode

If failures are detected during normal testing of the arrays, the memory BIST controller can be placed in debug mode for further analysis. In this mode, only a single array can be tested at a time. The memory BIST controller will automatically stop on each failure and wait for the tester to serially shift out the data in the datalog register, shown in Figure 9: "Memory BIST Datalog Register".

Figure 9: Memory BIST Datalog Register

Chip Selects		Write Data		Address		Failing Data Bits	
92	89	88	85	84	64	63	0

The ARM memory BIST supports pipelining of the RAM inputs and outputs if required for functional timing purposes. This means that there may be functional state elements between the memory BIST dispatch unit and the RAM. During debug mode, the memory BIST controller stalls after issuing a read access to the RAM and waits until the read data has been compared before releasing the next RAM access. This is to prevent missing back to back failures that are in the pipeline when a failure is detected.

6. Scan Chain Configurability

6.1 Background

Customers of a hardened ARM core may have different requirements for the number of scan chains. For example, one customer may have the capability of testing a minimal pin set with lots of memory, while another customer of the same core may want many more chains to save tester memory. The environment determines the needs of the core user. The core provider has no knowledge of that environment.

6.2 Strategy

To solve this problem, the ARM1026EJ hard core provides special scan chain muxing logic that allows the number of scan chains to be configured by the tester. The internal scan chains and the wrapper scan chains can be configured independently. The muxing configurations supported for internal chains on the ARM1026EJ are 7, 14, 28, and 56 chains, while those for the wrapper chains are 1, 2, 3, and 4 chains. This provides for a minimum of 8 and a maximum of 60 total chains.

6.3 Implementation

A series of muxes are placed on the scan chain inputs that select between the top level scan input pin for that chain and the scan chain output of the chain that will be concatenated with it for a lower chain count configuration. The top level scan output pins are always connected to the same scan chain outputs, but may not be used for lower chain count configurations. The scan patterns are always created using the maximum chain count configuration, but are converted to other configurations without re-running ATPG. Currently, this is done using a

custom pattern conversion tool written and maintained by ARM that manipulates the WGL formatted vector data.

7. CheckTest

When a hard core is instantiated into an SoC there must be some way to check the connections between the SoC pin and the port of the core, without having knowledge of the core itself. Functional connections are checked with an encrypted, RTL-based core called a Design Simulation Module (DSM). Scan is not introduced into the core until synthesis, so the RTL-based DSM has no scan. The scan patterns cannot be run on this model. CheckTest provides one solution to this dilemma.

7.1 Background

The scan patterns are all verified against the hard core by delivery time. Once a core user instantiates the ARM1026EJ hard core into an SoC, there must be some way of verifying that the test connections out to the SoC pins will not cause the core ATPG patterns to fail (e.g. a flip-flop in the path of a test signal). If the core was delivered as a black-boxed, hard core, the core user cannot verify that the test connections to the core are correct. A way had to be devised to address this issue.

7.2 Strategy

The basic idea was to add control and observe cells on the dedicated test pins, somewhat like wrapper cells. The CheckTest cells, as they are called, on the test inputs are used to both observe the test inputs and control the test outputs.

7.3 Implementation

There are a couple of ways that the CheckTest cells can be incorporated. Dedicated wrapper cells can be reused or new cells can be added to the dedicated test ports. The second option was chosen for the ARM1026EJ.

Figure 10: CheckTest Example

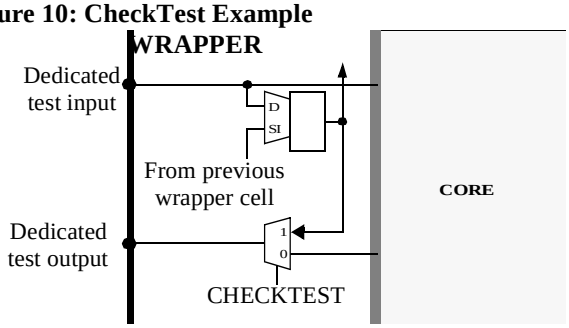


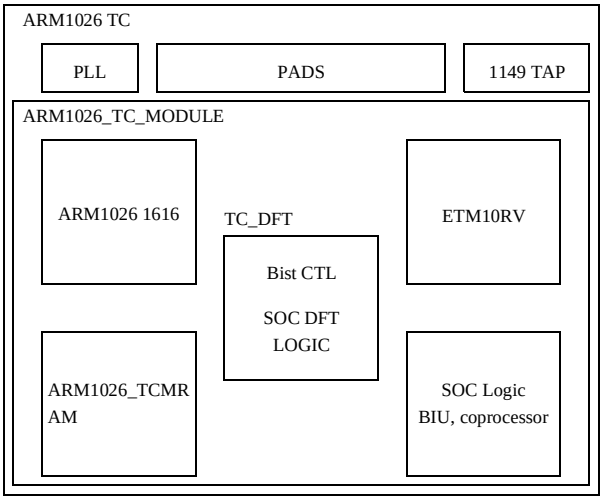
Figure 10: "CheckTest Example" shows a dedicated test input attached directly to the core, but also attached to the D input of a MuxD-flip-flop. A pattern is created that exercises the dedicated test input as it would be exercised during an intest mode. Consider an example of a test signal that is static, such as a SCANMODE signal. The SCANMODE signal would be pin constrained high during the creation of an intest pattern. If this signal is connected improperly in the SoC (driven to the wrong state or no direct connectivity) the pattern delivered would discover that misconnection.

The CheckTest scan chain is included with the gate level model of the wrapper, so the core can remain black-boxed. The CheckTest scan chain is not part of the wrapper chain during intest or extest mode as it will make the wrapper chain longer than necessary.

8. ARM1026EJ Test Chip Implementation

The ARM1026EJ test chip comprises the ARM1026EJ microprocessor core, the ETM10RV core and SoC logic which includes tightly coupled memories for the ARM1026EJ, coprocessor validation logic for validation of the coprocessor interface in the ARM1026EJ core and a PLL macro with dft clock control logic (see Figure 11: "ARM1026EJ Test Chip"). The ARM1026EJ and ETM10RV are being hardened for the first time in this test chip.

Figure 11: ARM1026EJ Test Chip



8.1 Digital PLL logic

8.1.1 Background

ARM macrocell IP is delivered without a PLL as it is usually a single component of a larger SoC. This establishes the need for a standard ARM PLL macro which can be configurable for multiple cores and requires the development of a generic DFT flow for at-speed test of ARM cores. This external PLL is

utilized for at-speed ATPG and memory BIST testing. The ARM1026EJ test chip is the first ARM test chip to follow this new PLL flow.

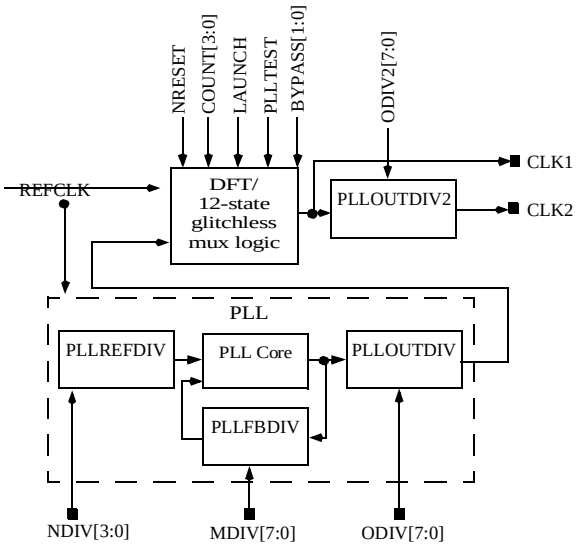
8.1.2 Strategy

The functional requirements of the PLL include the ability to switch between reference clock domains and fast (VCO) clock domains on the fly and without glitches in the output clock. DFT requirements are similar but with an added need to deliver precise counts of fast clock bursts with deterministic cycle-to-cycle behavior windows.

8.1.3 Implementation

A PLL output control block was created with muxes selecting between REFCLK and VCO-based clocks. This mux is controlled by an asynchronous state machine which has primary control signals and clocks as inputs. Synchronous logic was not used as it is subject to lockup or unsafe switching since the clock used to control it may be slower OR faster than the clock that is being switched. Synchronizer logic prevents external controls from influencing the state machine until previous control changes have reached safe states. Figure 12: "PLL Conceptual Diagram" shows the digital PLL logic.

Figure 12: PLL Conceptual Diagram



8.1.3.1 PLL DFT Control

The PLL is controlled during test by two separate mechanisms. First, primary input control allows control of all multiply/divide ratios as well as DFT test modes. This mechanism is used in the ARM1026EJ test chip as pad I/O are already allocated for PLL control. A second mechanism allows for serial shift in of the control signals. This shift-in will set the PLL into test mode and use the internal scan register for configuration of the PLL (See Figure 13: "PLL Control Chain Override"). This second mechanism is provided as SoC integrators may not wish to allocate test chip pins for this purpose. The following signals are significant to PLL testing:

PLLTEST - Enables DFT logic and signals.

PLLSE - The scan enable to enable shift of the control signal scan chain

PLLSI - The scan input where data is loaded into the control or observe scan chains.

PLLSO - The scan output for the control or observe scan chains.

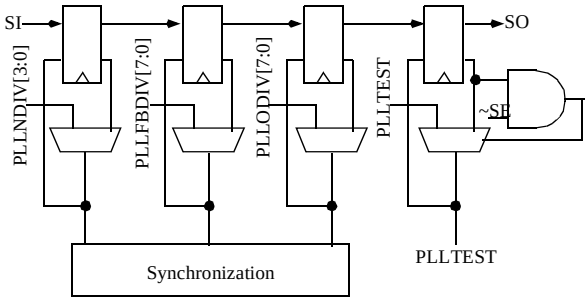
SCANSEL - Selects between control and observe chain. Chains must be separate to insure pll lock is not lost.

BYPASS: Selects REFCLK as the primary clock output to the core.

LAUNCH: When asserted, VCO (fast) clocks will be delivered after the next rising edge of clock.

COUNT[3:0]: Set prior to launch, this signal dictates the number of fast clocks to be delivered during the LAUNCH cycle, allowing for pattern sets to contain from 0 to 14 clocks during capture.

Figure 13: PLL Control Chain Override



The ARM1026EJ memory BIST also allows for at-speed testing of the memories. Shifting data out of the memory may require a slower clock speed due to package or tester constraints. PLL control for this capability is accomplished by setting the COUNT[3:0] to 0xf, which results in continuous fast clocking. The fast clocking is started and terminated by the LAUNCH control pin setting. BYPASS is used to start and terminate delivery of the slower REFCLK clock.

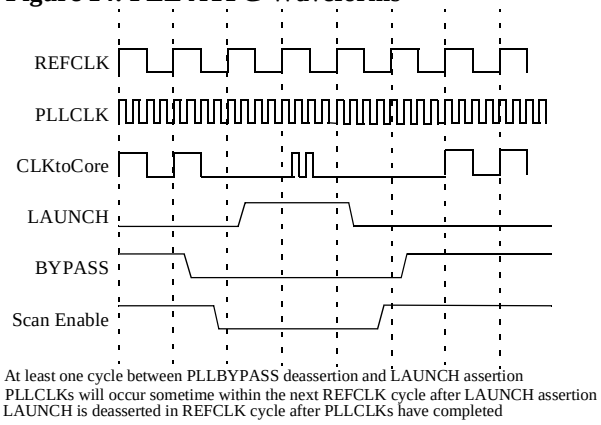
BYPASS and LAUNCH are never toggled in the same REFCLK cycle. This is a necessity for ATPG input setups/controls and also allows for simpler

PLL design.

A second scan chain is provided in the PLL for observation of control signals provided by SoC logic (for use during SoC scan testing) and also provides a counter chain which counts the number of fast clocks delivered during the ATPG capture cycle (an engineering PLL debug feature).

Figure 14 “PLL ATPG Waveforms” shows how the core clock is output during ATPG patterns that utilize the fast launch-to-capture sequence using the PLL.

Figure 14: PLL ATPG Waveforms



8.2 ATPG Patterns

The core ATPG patterns must be post-processed for the other scan chain configurations and to add, delete or change any of the signals while taking the core pattern to the test chip level.

A dummy “dft_bus” is added to the netlist to assist with the pattern manipulation. The dft_bus is controlled in the ATPG scripts and provides key information for our post processing software. The dft_bus signals can be stripped from the test chip pattern delivery, but are left in the core pattern deliverable for the core user to utilize if needed. Signals in the dft_bus allow us to add the PLL test signal activity and to manipulate the patterns for the different scan chain configurations.

The ARM1026EJ test chip utilizes the 28 internal chain scan width mode for testing. These vectors are converted to the test chip level by use of internal ARM software. Features within the conversion tool include

- Pin addition/deletion
- Setting pin constraint values (force 0/1 on an Input)
- Swap pin names
- Pin equations (value set by RPN formula of other signals)
- PLL COUNT bus addition
- PLL test setup stretching to achieve lock.

During the conversion process, the PLL COUNT value is determined by counting the number of capture clocks that occur and is inserted for all vectors during that capture sequence.

Two sets of patterns are automatically created for the test chip use; those utilizing PLL BYPASS and those utilizing PLL fast clocks. All patterns are created in PLL BYPASS mode, and those tagged as pll compatible will also be created in PLL fast clock mode.

The ARM1026EJ pattern sets include stuck-at, transition delay, and path delay patterns. Memory models that were provided from the synthesized ram vendors were re-written to reflect true silicon behavior since provided models are pessimistic and result in longer ATPG run times.

Two ATPG vector solutions are offered to customers to allow them to best match their test needs. In the first solution, a full stuck-at pattern set is offered. In the second solution, an efficient transition pattern set is created and fault graded against the stuck-at model. A ‘topoff” stuck-at pattern set is then generated to capture the remaining stuck-at faults.

Transition patterns are created in two passes. The first pass creates patterns with a minimum detect/pattern requirement (higher pattern efficiency) and a second pass is executed without restriction (transition topoff). Stuck-at fault grading and topoff patterns are generated against the high efficiency transition vector set only. The customer has further choice by trading off vector counts of the two transition sets against transition coverage. This tradeoff is significant when transition patterns are generated with memories. All transition patterns delivered in the first release had black boxed memories due to unresolved tool bugs.

8.3 Memory BIST Patterns

The ARM1026EJ soft core comes with a memory BIST testbench for use in creating validation runs. This testbench was modified to instantiate the ARM1026EJ test chip. Successful execution relies on a properly configured test environment as set by the test chip’s DFT logic.

Patterns were created for both the BYPASS mode (no fast clocks) for debug and the PLL clocking mode.

8.4 Test Chip control of ARM1026EJ

The ARM1026EJ wrapper was utilized for creation of ATPG vectors for the ETM10 core and for SoC logic (validation coprocessor and user-defined logic). The ETM10 ATPG generation utilizes the ETM segment of the wrapper chain while the SoC logic utilized both the UDL and coprocessor segments of the wrapper

chain.

A test chip level block of DFT logic was implemented to configure the cores and PLL for ATPG testing. There is a test mode configuration for each of the cores, as well as the logic external to the cores. These test modes configure the wrappers correctly, control all static signals properly and give pin access to an dynamic test signals for each test.

8.5 ARM1026EJ TCM Shadow Logic Test

Tightly coupled ram interfaces were not wrapped due to their critical timing paths. Since ARM delivers a hard core and the TCM configuration and memory type is unknown, ATPG vectors must work with all TCM configurations and memory types. Test coverage of the TCM interface and their cones of logic were obtained by performing separate ATPG runs for each possible TCM memory size. All pattern sets are delivered with the core and the integrator chooses the ATPG pattern set that correlates to their implemented TCM size. To insure safe ATPG testing with different RAM vendors, TCM ATPG memory models have been created with additional pessimism.

8.6 ARM1026EJ Test Pattern Validation

All core ATPG WGL patterns are tested against gate level models, with timing, to insure quality deliverables.

This process is repeated on test chip deliverables for both slow clocking and PLL fast clocking configurations.

Validation of memory BIST patterns are also performed in a similar fashion.

9. Results

As of the paper deadline, silicon was pending. Silicon results are expected in time for the presentation.

- *Successfully implemented an at-speed architecture utilizing an external PLL to provide the test clocks.*
- *Functionally segmented wrapper successfully created and successfully used it to create the ETM10RV internal test patterns and the user-defined logic patterns.*
- *Two full pattern sets created - one utilizing bypass clocking and one utilizing pll clocking and successfully validated against back annotated models.*
- *Successfully delivered IddQ, stuck-at, transition, and path delay vectors; simulated with tester timing in time for first silicon.*
- *Tool developed to convert patterns for other chain configurations.*
- *Tool developed to convert core patterns into testchip patterns.*
- *ATPG scripts created to insure successful conversion into other confiigurations/hierarchy.*
- C*umulative stuck-at coverage for the ARM1026EJ_88 is 99.4%.*
- *87% transition delay coverage with black boxed memories.*
- *Path delay patterns generated for 4000 top paths including memory paths.*

Two ATPG vector flows delivered:

- *Generic Stuck-at flow.*
- *Transition and stuck-at flow.*

Table 3: ARM1026EJ_88 Stuck-at Test Coverage

Type	Pattern Count	Vector Count	% Test Coverage
Stuck-At	2,381	913,361	99.43%
TOTALS		**913,361**	**99.43%**

Table 4: ARM1026EJ_88 Transition Test Coverage

Type	Pattern Count	Vector Count	Transition Coverage	SA Test Coverage
Transition Delay	2,307	672,322	84.9%	94.7%
Stuck-at topoff	1,648	722,050	na	99.4%
Transition topoff	1,219	355,547	87.4%	na
TOTALS		**1,749,919**	**87.4%**	**99.4%**

Path delay pattern files are segmented and in slack order. A portion of the pattern set can be used if all of the patterns do not fit in the memory.

Table 5: ARM1026EJ_88 Path Delay Coverage

Type	Pattern Count	Vector Count	#Faults/ Paths
Path Delay Slack < 1ns	2857	827100 56 chain	3543/3171

10. Lessons Learned

At the time of this paper silicon was due out imminently. Results of the silicon should be in the presentation

Memory Bist

- The memory BIST worked as expected. It passed simulations at-speed with back annotated timing.
- The memory BIST had high overhead. Smaller cache implementation made some arrays' overhead larger than desired due to full bitmapping features.
- The original design point had a separate master BIST controller to be utilized for all memories on the SoC. This resulted in an external controller with respect to the delivered core and further overhead for implementation teams to support. The external at-speed memory BIST controller is also more difficult to manage for clock tree management as maximum frequency increases.

ARM1026EJ_88 Wrapper

- The functionally segmented wrapper had different chain polarities per synthesis run, resulting in ARM1026EJ core specific pattern sets for ETM10RV test vector sets.
- The functionally segmented wrapper did work as expected. In the case of the user-defined logic, it worked very well, minimizing the UDL wrapper chain. The ETM10RV segmentation was more complex as some of the ports were connected to the ARM1026EJ and some were connected elsewhere (needing another wrapper chain). This resulted in four wrapper scan enables that had to be properly managed.

ATPG timing hazard

ARM1026EJ memories (including TCMs) have an inverted clock relative to the core clock. Standard pattern sets have the falling edge of clock near the end of the scan clock cycle. Scan enable is deasserted early in the cycle after shift completion. This results in potential timing hazards as the falling edge of the core clock (rising edge of the memory clock) and the scan enable deactivation are too close together. This resulted in memory corruption from the previous capture cycles. An extra post-shift cycle was added to avoid the hazard.

PLL Timing

- The PLL digital macro logic worked very well in simulation with back annotated timing, with ATPG patterns.
- A DFT speed path was identified just beyond the maximum target frequency. A separate timing accurate validation environment for this block was required.

CheckTest

- This feature was provided as a mechanism to check for correct DFT hookup of IP in the customer's design and worked as expected. The WSEI pin must always be pin accessible for intest mode in order for CheckTest to work. However during actual intest testing, WSEI is held to a static high. This was the only variant to our intest DFT methodology.

RSTSAFE

- RSTSAFE functioned properly, but no measurements on power reduction have been taken.

Scan Chain Configurability

- Patterns only had to be generated in one scan chain configuration. Scripts were written to convert the initial set of patterns to all other configurations. This worked very well.

WSEI/WSEO

- WSEI being enabled during path delay did allow some input paths to be caught.

11. Future Directions

- Future cores will have an internal memory BIST controller per delivered core.
- Future segmented wrappers will enforce true polarity on all elements.
- The wrapper segmentation will be available on the wrappers, but will only be used for user-defined logic, cores with critical paths at the boundary of the core (cannot tolerate the mux delay of a dedicated wrapper cell) or area critical cores (if dedicated wrapper cells needed).
- Future designs will not have CheckTest as gate level model (scan included) encryption techniques have been created.
- Develop a better methodology for testing the shadow logic of external memories.
- Ensure that clock timing delay to the core is reasonable.
- Not all input paths have been tested as tight slack paths were generated first. A separate input/output path pattern should be generated.

12. References

[1] E.J. Marinissen, T. McLaurin, Rohit Kapur, Maurice Lousberg, Mike Ricchetti, Yervant Zorian, "On IEEE P1500's Standard for Embedded Core Test", Journal of Electronic Testing: Theory and Application 18, 2002, Kluwer Academic Pulishers, pp. 365-383

[2] D.Amason, et al., "A Case Study of the Test Development for the 2nd Generation ColdFire Microprocessors", International Test Conference, 1997, pp. 424-432

An Optimized DFT and Test Pattern Generation Strategy for an Intel High Performance Microprocessor

David M. Wu, Mike Lin, Madhukar Reddy, Talal Jaber, Anil Sabbavarapu, Larry Thatcher

Intel Corporation

Contact Author: david.m.wu@intel.com

Abstract

This paper describes an optimized DFT architecture and its implementation strategy for an Intel high performance (>3 GHz) microprocessor. Major DFT features and ATPG techniques implemented are described and key results are presented to show the return-on-investments (ROI) in the high volume manufacturing (HVM) test environments.

1. Introduction

This paper describes the design for testability and debug, and ATPG strategies for a high performance (>3 GHz), high density microprocessor (>150M Transistors) with super rich system features for Intel high end platforms. Our DFX team had committed to develop and implement the most advanced and cost effective DFX strategy that would meet the quality goal predicted by an Intel Internal prediction tool developed by the Intel CQN (Corporate Quality Network) group. To meet this challenge, a highly skilled team was formed with key DFX experts in the various DFX areas. In the technology readiness phase, a DFX portfolio was developed, feasibility analyzed, gaps, mitigation plan and backup plans were identified. At the end of the technology readiness phase, we decided to optimize industrial style scan and DFT methodologies [1-5] with some innovative DFT features to reduce overall test cost. In order to handle test data volume limitation and the potential Vcc droop problems during the HVM test process, we developed a partition ATPG strategy [6-9] and implemented required scan DFT controls to isolate the targeted logical units or clusters. In addition, some innovative methodologies for other DFT/DFD (Design for Debug) areas such as array DFT, I/O DFT, Design for Burn-In and Design for Debug were developed.

Section 2 describes a Partition ATPG (PATPG) architecture and the required scan DFT features to overcome the limitation of the commercial ATPG capacities and to reduce the potential di/dt impact during manufacture tests. To meet the overhead constraint and performance goal, we developed a 'Skip-Scan' technique and established a set of skip scan design rules and a very strict scan waiver process to ensure meeting the test coverage requirements even if we minimize the silicon area overheads.

Section 3 depicts the array DFT design and validation strategies. Since the allowed DPM budget for entire arrays on the chip is extremely low, it is necessary to have a very comprehensive array DFT test strategy. We use Programmable Array BIST (PBIST) [10] to test the largest arrays and an optimized array BIST technique to test smaller and medium arrays. Direct Access Testing (DAT) [11] for array access and diagnosis and Programmable Weak Write Test Mode (PWWTM) [13] for memory cell stability test to reduce the test time.

Section 4 describes an enhanced TAP controller called 'Integrated Test Controller (ITC)' that has special hooks to control the additional DFT and DFD features we have implemented in the silicon. The key DFD and IO DFT features inherited design from the previous Intel microprocessor [14] are addressed in this section.

Section 5 describes burn-in DFT techniques. A built-in self test feature is used to generate burn-in toggling test patterns for the logic circuits. Two BIST schemes were used to provide high toggle coverage for burn-in of the arrays.

Section 6 highlights the key difference of design-for debug features mostly described by the previous paper on the Intel microprocessors [15]. A scanout system is developed and validated. A system clock freeze feature during scan test is implemented to help at-speed debug.

Key learnings from the DFX implementation processes and results of ATPG data for key logical partition units and clusters are depicted in the last section.

2. Logic DFT and ATPG

Generating scan ATPG vectors for the targeted microprocessor 'PX' poses many formidable challenges due to the sheer complexity of the tasks. One of the key challenges is imposed by the extra large number of logic

gates and faults presented in the full chip netlist that created problems for ATPG tools capabilities to effectively generate test patterns. In this section, we describe a Partition ATPG (PATPG) methodology adopted by us to break down the problem into a set of smaller and manageable sub-problems. We partition the design into smaller blocks and generate ATPG vectors at full-chip level for these blocks so that complex transformation of ATPG vectors is not needed to apply them at full chip level. The required Scan DFT and PATPG methodologies are described in this section.

The PX microprocessor design is highly complicated with multiple clock domains, multi-cycle paths, domino circuit including OTB (Over-Time Borrowing) and static circuits, and extraordinary high number of transistor circuits. Analysis shows that a cluster size of logic can be handled by the commercial ATPG tools used by PX. In addition, to overcome the potential di/dt problem, a lower level of partition becomes necessary. To accommodate these requirements, PX scan DFT adopted a Hierarchical Scan Architecture (HSA). The HSA divided the full chip into a group of "Clusters". A cluster is usually a top level functional logical entity within the microprocessor such as the floating-point execution cluster. A scan control logic block was designed to enable the testing of a single cluster or a combination of multiple clusters. Test patterns can be generated accordingly for either a single cluster or for a combination of multiple clusters.

Similarly, a cluster can be divided into a group of 'Units'. A sub-level test controller was designed to enable the testing of either a single unit or a combination of multiple units. In the ATPG process, we expect that there should be no issue for the commercial ATPG tool to handle size and complexity of a cluster. However, we have prepared the 'unit' level ATPG test generation capability in the case of intolerable di/dt problem during the HVM testing process.

There are 36 scan chains distributed throughout the full chip hierarchically. Each "Cluster Partition" is bounded by scannable scan chains. Most "Unit Partitions" are also bounded by scan chains although there are exceptions. There could combinational logic at the boundaries of partitions. Our overall methodology benefits from a scan architecture with partitioning as one of the primary goals and has the following related features:

- Partitions targeted for test are flexible and configurable. For example, partition under test (PUT) can be a cluster, a unit, or combinations of units/clusters. This enables us to scale the size of PUT according to the complexity presented to the ATPG tool or desired test application, and also target ATPG for faults in the "exposed" logic between partitions.
- Scan chains in partition not under test (PNUT) can be bypassed and one can configure scan chains for any PUT selected to connect to the chip pins at full-chip level. This optimizes test time as only scan chains in PUT are configured between chip pins
- Our methodology requires neither separate scan chains nor separate functional clock control for scan instances at the boundaries of ATPG partitions. This reduces scan design complexity and overhead significantly and leads to "design friendly" scan design.

Note that partitioning netlists is an old topic in EDA literature. There are several published papers on strategies for partitioning netlists, faults and patterns for parallel and/or distributed ATPG/fault simulation [6-9]. The objective of this paper is not to propose another such algorithm. The main contribution of this paper is to describe a practical method illustrating the implementation challenges for such a "divide and conquer" strategy in complex and high performance processors and using a commercial ATPG and simulation validation tools.

2.1 SCAN DFT ARCHITECTURE TO SUPPORT PATPG

PX has seven logical clusters that make up the full chip. Each one of those clusters has one cluster test controller (CTC) module and at least one unit test controller (UTC) module. These CTCs and UTCs are used to configure the scan chains and control the clocking of PUTs and PNUTs in ATPG mode. Every CTC module buffers 36 scan chains. It also contains control and staging logic for scan control signals. The outputs of the CTC drive the UTC modules. Each UTC contains a control register for PATPG control. These register bits are connected into a signals and clock control special scan chain that connects all UTC control registers into one global chain. This register consists of control bits for scan shift clock enable, controlling various functional clocks, and the "functional clock gating test override" signal. An example of scan chain routing and partitioning for ATPG is shown in Figure1.

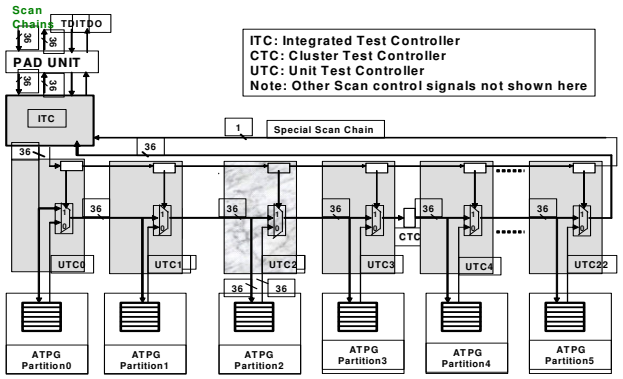


Figure 1: Scan Chain Routing & Partitioning for ATPG

The scan shift clock enable controls scan chain selection and scan shifting. This bit controls a bypass mux in the UTC for the 36 scan chains. If the scan shift clock enable is active then the 36 scan chains that are connected to the UTC are part of the active chains for ATPG, or if it is not active the scan chains that are connected to the UTC are bypassed with a mux like structure.

2.2 Skip Scan Methodologies

The ATPG process for PX has been a challenging task. Not only we have to deal with highly demanded test coverage numbers, we also have to deal with the limitation of budgets in terms of the silicon area, leakage power, and scan performance impact. To meet our challenge, a set of well documented design-for-test rules are communicated to all PX design teams and a well developed scan waiver process was 'enforced' throughout the development phase of the project. As a result, a very cost effective scan implementation was very successfully implemented in PX. One of the very valuable techniques is the 'Skip Scan' Technique. It is also called Datapath Interleaved Scan (DI-Scan) Technique since it is the most effective to be applied in the datapath logic following some Skip Scan design rules. The Skip Scan technique helped us significantly in meeting an aggressive scan area budget and still maintain performance target and test coverage goal. Figure 2 shows simple examples of DI-scan datapath pipelines in which scan are skipped in one sequential stage between two scanned sequential stages. Data flow direction is from left to right as indicated. The top portion in this figure shows a flop based datapath design and the bottom portion shows a latch based design. The scan type employed LSSD-Like Design with edge triggered flip-flop. In Figure 2, CLK and CLK# are ph1 and ph2 clocks respectively. The skipped functional logic parts are shown in shaded boxes, the scan logic parts are shown in white boxes and the "clouds" represent combinational logic. Latches driven by phi2 clock, CLK#, are transparent when clock is inactive and hence are not scanned even in full-scan design. In a full-scan design, the non-scan stages would also have been converted to scan stages. Note that, both the datapath pipelines start with scan and end with scan. Skipping scan in at most one sequential stage between two scanned sequential stages is the predominant DI-scan technique employed in datapath pipelines. This minimized the risk of lowered test coverage from scan based ATPG tools. However, for some data paths, we relaxed this restriction and allowed for skipping scan in two consecutive sequential stages for improved area and timing after completing careful analysis and based on ATPG test coverage results. In a pipelined microprocessor designs, often the micro-architecture calls for delaying certain signals by several clocks in order to synchronize them with the rest of the logic. This is accomplished by using several back-to-back register stages. In other cases, often FIFO structures are used to capture data that is transmitted at a high burst rate for consumption later. These FIFOs also have back-to-back register stages with no logic in between. In this paper, we call such stages as "staging pipelines". We further extended the skip-scan technique more aggressively into staging pipelines by taking advantage of absence of combinational logic (except for buffers and inverters) in staging pipelines and skipping scan in more sequential stages. From ATPG studies we have done, we concluded that we can skip scan in up to a maximum of three consecutive stages and with out significant degradation in ATPG test coverage and with some increased ATPG execution times.

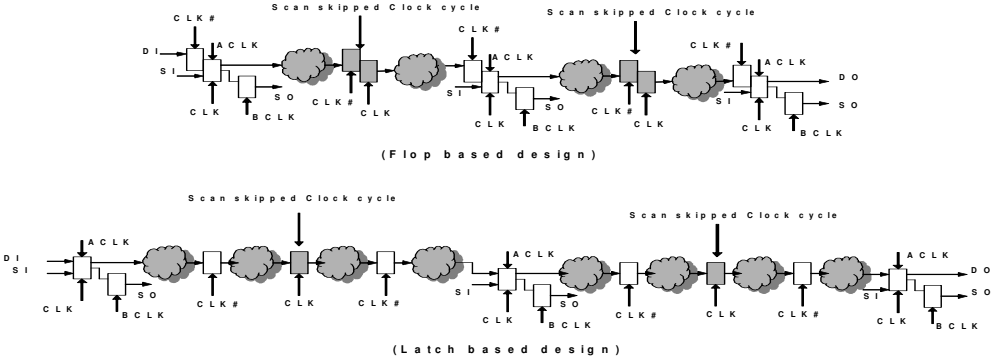


FIGURE 2: DATAPATH INTERLEAVED SCAN IN FLOP BASED DESIGNS AND LATCH BASED DESIGNS

In PX we architected scan with partitioning for ATPG as one of the primary goals. The DI-scan technique is employed in PX in such a way that we can generate ATPG tests for DI-scan pipelines in an integrated manner with rest of the logic in Partition Under Test. Since our goal is to generate ATPG patterns using the scan based ATPG tools, we have restricted the number of sequential stages where we are skipping scan and additionally defined required attributes of the data paths where DI-scan could be employed. In section 2.3, we first define DI-scan rules for ideal datapath pipelines from ATPG point of view and illustrate how a scan ATPG tool with limited sequential capabilities would be able to generate tests for such pipelines. As some of these rules are violated in some real datapath pipelines, we will explain what complexity such violations add to ATPG and how they may impact the test coverage.

2.3 Skip Scan Design Rules in the Datapath: DI-Scan Rules

To make the DI-Scan work, it is important to establish some DFT rules. Key DFT rules for DI-Scan rules including the following:

1. DI-scan is used only in datapath pipelines.

2. Control logic must be full-scan.

3. DI-scan pipelines are "uniform" and the dataflow direction is only forward. That is, no feed forward or feed backward through sequential stages. This prevents logic of uneven sequential depth to converge and enables known states from scan chains to propagate forward through datapath pipeline in a ”wave” like manner when functional clocks are exercised. In case of pipelines with latch based design, both P-latches and N-latches must be considered as sequential stages. N-latches are at clock cycle boundaries. P-latches are at half-cycle boundaries and are transparent when clocks are inactive. P-latches are not scanned. (Note: No feedback loops in combinational logic that create sequential state are allowed. No feedback loops made of a latch and combinational logic are allowed. These rules are part of basic scan DFT rules).

4. DI-scan pipeline begins with scan at first clock cycle and ends with scan at last clock cycle in the pipeline. Scan is skipped only at sequential stages at alternate clock cycles when traced forward from the first clock cycle. It is OK if scan is not skipped at some clock cycles. In case of pipelines with latch based design, skip-scan pipelines should look like this: Scanned N-latch -> P-latch -> unscanned N-latch (scan skipped clock cycle) -> P-latch -> Scanned N-latch -> P-latch and so on. Note that only a section of pipeline is described here.

5. Non-scan state elements in skip-scan must hold states during scan shifts (excludes P-latches which are transparent). This is trivially satisfied in PX scan architecture as functional clocks are inactive during scan shifting.

6. It is imperative that all functional clock gating be implemented within the Local Clock Enable (LCE) block. This ensures DFT overrides for functional clock gating are implemented. No further clock gating is used after LCE.

7. It must be possible to control the functional clock gating logic combinationally from scan flops. This ensures patterns can be generated to test clock gating logic with DFT overrides set to inactive state.

8. For flops with enables on data input, it must be possible to control the enable logic combinationally from scan flops. This includes self-loop situations for flops.

9. In DI-scan datapath pipelines, selects for multiple tri-state driven nodes are (needed for contention-free ATPG patterns): Fully decoded within combinational logic, or Fully decoded within a DI-scan stage (i.e., within the two clock cycles logic of a DI-scan stage), or If coming from control logic, fully decoded within combinational logic, or If coming from control logic and not fully-decoded within combinational logic Hold Scan is used.

2.4 ATPG Results for Key Units and Representative Clusters

During the process of establishing the DFT strategy, we faced some challenges in making decision on the ROI (Return-ON-Investment) tradeoff, with no negotiation of performance and quality goals. For clusters or units that do not have any performance sensitivities, the full scan/skip scan methodologies were very successfully implemented and the DFT rules were strongly ‘enforced’. However, for some clusters or units that have little room for area overhead that could cause performance issues, we exercised very extensive test coverage analysis to isolate certain logic blocks to be ‘non-scannabel’ blocks (NSBs). These NSBs are distributes in relatively small amounts of areas across the whole chips and the design engineers were responsible to generate Functional Test Patterns either using structural based functional test (SBFT) or traditional functional test methods to coverage the gaps. The following ATPG data are test coverages results (before the functional tests) that reflect the test coverage gaps due to NSB blocks.

In Table 1, the ATPG data of 5 Units that has implemented very aggressive Skip Scan methods are depicted. The number of logic gates in these units ranged from ~64K to ~460K. The percentage of scan sequentials of these Units ranged form ~31% to ~ 76%. Even though

the actual implementation is 'Skip Scan', we also simulated the 'full scan' (excluding the NSBs) implementation to compare the test coverage results. In Unit 1, the Skip Scan case has a lower percentage (~73%) of scannable sequential but it obtained a better test coverage (~93%) comparing to the case of 'full scan' (~85% scan percentage, exclude NSBs, and ~79% test coverage). One of the key reasons of this result is due to the decoder logic spreading across multiple stages of sequential elements that actually makes the ATPG harder for full-scan circuit. The Unit 2,3,4,5 demonstrated some impressive results of Skip Scan techniques. With a relatively low scan percentage (from ~31% to ~ 77%); the Skip Scan test coverage loss for Unit 5 (~50% scan) is nearly 0%. For Units 2, 3, 4, the Skip Scan coverage gaps comparing to 'full scan' is ranged from ~1% to 6%. As previously stated, additional functional test patterns will be applied to coverage these gaps in order to ensure the targeted test coverages both in stuck-at faults and delay faults are satisfied.

Table 1: Unit level ATPG results, Skip Scan vs. Full Scan

Unit	# Gates	S@ Faults (Uncollapsed)	Total # of flops	UNIT with DI-Scan Datapath		UNIT with Full-Scan Datapath		Comments (ATPG untargeted logic not completely nofaulted. Other T-scan exceptions present. PIs controlled, Pos observed)
				% scan	Test Covg.	% scan	Test Covg.	
Unit 1	140382	294226	4263	73.73%	93.32%	85.78%	79.4%	4345 buses. Sequential decoding. Contention prevention. Untestable faults in select logic.
Unit 2	85747	153896	3626	31.05%	91.24%	98.79%	98.71%	1576 buses. Sequential decoding. Contention prevention. Untestable faults in select logic.
Unit 3	125252	234598	5571	76.84%	97.94%	92.44%	98.19%	2652 buses. Sequential decoding. Contention prevention. Untestable faults in select logic. 2 stage deep non-scan EBB embedded.
Unit 4	464093	361538	17157	40.06%	85.60%	45.38%	86.95%	5930 buses. Sequential decoding. Contention prevention. Untestable faults in select logic. 4 stage deep non-scan EBB embedded. Abort limit of 500 not sufficient
Unit 5	64105	125806	4845	50.28%	99.97%	N.A.	N.A.	EBB level run. 128 pass-gate MUXes, combinationally fully decoded. Unit level coverage for this EBB is 99.13%

Two representative Clusters ATPG results are depicted in Table 2. Cluster 1 has ~900K stuck-at faults with an approximately ~14K sequential elements. It is mostly synthesizable random logic and the test coverage reached 98.2% without taking credits for the ignorable faults and possibly detected faults. Cluster 2 has an approximately 3.45 M faults with 146K sequential elements. This Cluster is a very performance critical and area budget limited cluster, we have implemented many aggressive scan waiver techniques in addition to the Skip Scan methodology and the NSBs isolation to optimize the cost factor but still preserve the defined coverage target. We used only ~48% of scan sequential in this cluster and obtained ~91% test coverage for the entire cluster.

Table 2: Cluster ATPG results

Name	# Faults	Skip Scan	Total # of seqs	% scan	RLS/CBD/EBB	Test Cov	Comments
Cluster 1	**900K**	No	**13.9K**	**99.56%**	RLS	**98.2%**	Nearly Full Scan
Cluster 2	**3.45M**	Yes	**146K**	**48.65%**	CBD/EBB	**91.5%**	Aggressive Scan

3. Array DFT

Our Array DFT test strategy is to use PBIST (Programmable Built-In Self Test) [10] to test the second level cache and use DAT [11] to test the remaining arrays. In addition, we use PWWTM (Programmable Weak Write Test Mode) [13] to test cell-stability defects of 7 large small-signal memories. The following are the detail for each of the 3 modes.

3.1 Programmable BIST (PBIST):

We uses PBIST to test 4 Level 1 (UL1) arrays (Data, Tag, LRU and State arrays). PBIST is a widely used technique within Intel. It not only supports application of all kitchen sink patterns (> 50 Algorithms). It also supports testing of 7 different configurations of UL1 sizes. Figure 3 depicts the diagram of PBIST structure. PBIST can be thought of as a micro controller connected to fairly sophisticated address generation logic and distributed data generation/comparison logic. Access to all portions of the PBIST is available through the JTAG TAP controller. PBIST is used for all at-speed production testing of the UL1. Because the testing is done at operational speed, it can detect many delay related subtle defects. It can also raster memory data at the operational speed for memory repair, fault diagnosis, and yield improvement. PBIST can also be used during POST (Power-On Self-Test) to check the health of UL1. POST runs as part of the chip microcode reset sequence and has the responsibility of reporting to microcode whether the UL1 cache is operable. The maximum size of PX UL1 is 2 M bytes. However, there are 7 possible configurations with 3 possible sizes: 2MB, 1MB, and 0.5MB. Since a PX chip may be sold at any one of the 7 configuration when either some part of UL1 is defective or market demands for smaller cache size version, PBIST is enhanced to test all 7 configurations. Note that such testing is done on top of the possible reconfigurations to substitute the defective column with the spare column.

3.2 Direct Access Testing (DAT): The DAT (Direct Access Test) mode in PX as shown in Figure 4 is very similar to that presented in [11]. The DAT mode allows direct, parallel access from FSB pins to the I/O of the target array through CRB (Control Register Bus) and the local DAT circuit adjacent to the array. This parallel connection provides a much higher bandwidth to transfer array test information back and forth between the tester and the targeted array allowing 100 times faster production test than accessing though the TAP's serial control Register Bus. PX DAT mode covers 99.9% of all arrays including UL1 arrays. In addition, it supports back-to-back test of all arrays except UL1 arrays, which can be tested back-to-back by PBIST (covered above). The key reason to support simple read-write to UL1 arrays is to enable SBFT (Scan Based Functional Test) and FRIT (Functional Random Instruction Test) [12]. In addition, it can be used for dumping 4 UL1 arrays for debug. While PX has more arrays (>110) and more multi-ports arrays with more ports in such arrays in any earlier Pentium 4 CPUs, the percentage of arrays and the percentage of ports in multi-port arrays covered by DAT mode is higher in PX than any previous CPU. As such, the array test coverage (note) of PX is 99.3% - the highest in Pentium 4 family.

3.3 Programmable Weak-Write Test Mode (PWWTM):

In [13] WWTM (Weak-Write Test Mode) was used to detect many stability types of defects prevalent in memory cells. In this mode, a value is opposite to the cell value is "weakly" written into a memory cells such that if in the presence of defect, the value of the memory cell will be flipped. On the other hand, if the cell is stable – free of major defects, the cell value will remain unchanged. Production data, however, show that it is very hard to choose the optimal design point of the weak right circuit such that the stress caused by the weak-write is just strong enough to push the unstable cell to flip the stable without cause significant over-kills or under-kills. To overcome the issue, the WWTM was enhanced to make design point of the weak-write circuit become programmable. This way design point can be programmed to the optimal condition after the enough data is collected from silicon. This technical has been shown to drastically reduce both over-

kills and under-kills in detecting stability defects of small signal memories.

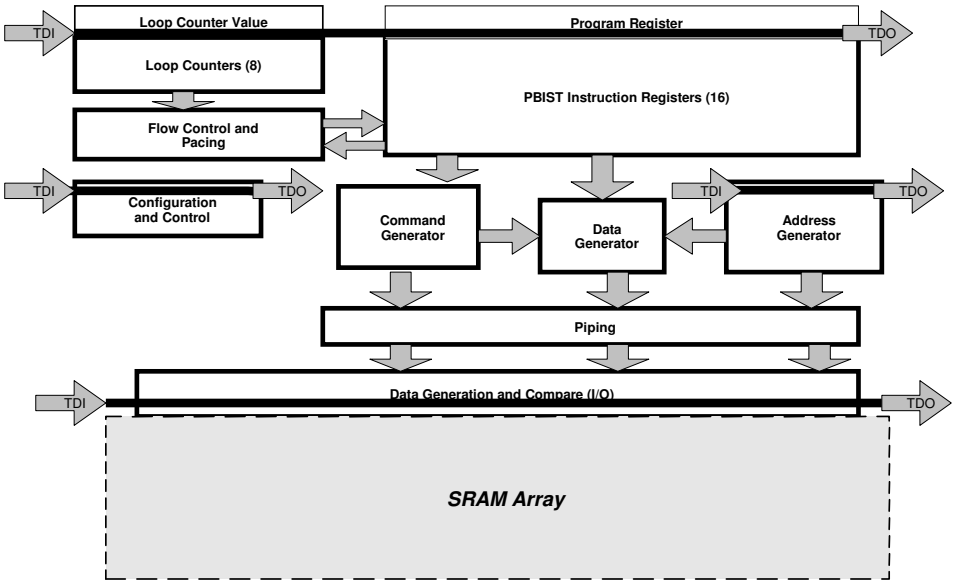


Figure 3: Programmable Built-in Self Test (PBIST) Architecture

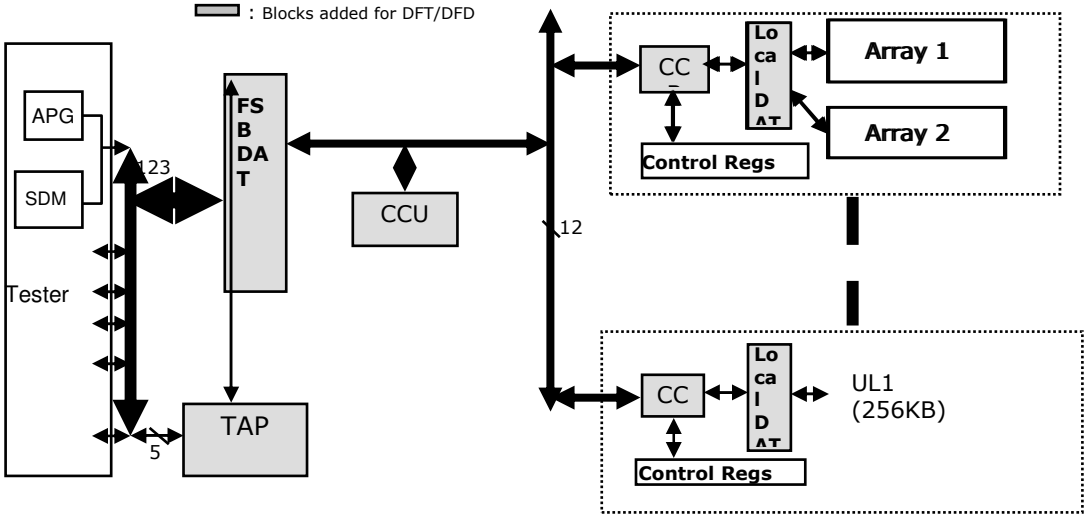


Figure 4: Direct Access Testing (DAT) architecture

4. Integrated Test Controller and IO DFT

The Integrated Test Controller (ITC) includes the Test Access Port (TAP) logic and all the controller logic that controls various debug and test features. The TAP complies with the IEEE 1149.1 ("JTAG") test architecture standard, and additionally provides access to most of the testability and debug features including Micro-breakpoints, Control register bus access, LBIST, Scan, Scanout, Signature mode, thermal sensor control, I/O self-testing, fuse programming and DAT mode . The TAP supports an instruction set of many functions. It provides not only a rich set customer-visible features, but also crucial proprietary functionalities for silicon debug, system validation and production testing .

The TAP logic consists of a finite state machine controller, a serially-accessible instruction register, instruction decode logic and several data registers. The set of data registers includes those described in the 1149.1 standard (the bypass register, device ID register, BIST result register and boundary scan register), as well as several product sepecific registers. These pre-defined registers, together with control logic implemented in the TAP, provide access to all of the testability features.

The TAP logic and all test data registers are accessed serially through 5 dedicated pins on the chip package:

TCK (and thus the TAP itself) is designed to operate at any frequency between 0 Hz and a maximum frequency that will match the bus clock frequency (currently 200 Mhz). But remember that the tap clock frequency has no relation to any other clock on chip (i.e., there aren't any ratios or phase relationships). The tap clock is completely asynchronous to bus clock and the core clocks (although it is possible to purposely run the tap clock at exactly the same frequency as the bus clock, for testing reasons). The 0 Hz frequency means it is possible to stop the tap clock in the middle of a test, and then later on start tap clock back up and continue the test (although care must be taken with certain features to ensure they stay in sync).

TMS, **TDI** and **TDO** operate synchronously with **TCK** . **TRST#** is an asynchronous input signal. This 5-pin interface operates as defined in the 1149.1 specification.

4.1 The TAP Feature List

The testability features accessed through the TAP are summarized in the table . The second column lists, for each feature, whether it is defined as part of the 1149.1 standard, or product specific. The third column lists the TAP instructions which have been implemented to access each feature. The following table shows the testability features accessed through the TAP

Testability Feature	Feature. defined by	Supported TAP Instructions
Boundary Scan	1149.1	EXTEST, SAMPLE/PRELOAD
Bypass Register	1149.1	BYPASS, HIGHZ, CLAMP
Device Identification Register	1149.1	IDCODE
BIST	1149.1	RUNBIST
Serial Control Register bus access	Product Spec	CRBUSGO, CRBUSNOGO, CRPRELOAD, CRBUSPOLL, CRCANCEL
Array Freeze	Product Spec	ARRAYFRZ
Stalls (Allocator and Front End)	Product Spec	STALLREQ
Thermal sensor control	Product Spec	TSENCAT, TSENTHROT
DAT mode (Parallel Control Register bus access)	Product Spec	DATSERIAL, TESTMODE, ISCANDATMODE
StopClk	Product Spec	STOPCLK, STARTCLK, ALLCLKEN, ALLCLKDIS
Scanout chain, snapshot mode	Product Spec	SCANOUTLOAD, SCANOUTSHIFT
Scanout chain, signature mode	Product Spec	TESTMODE
Tscan	Product Spec	ISCANDATMODE, TSCANTIMER, TSCAN, TSCANSETUP, TSCANCREG, TSCANFNCNTR,
I/O Self-Test	Product Spec	IOTESTLOAD,IOTESTMODE, BSCANREAD
Micro-breakpoint mechanism	Product Spec	BRKPTCTL[A,B]
Probe Mode	Product Spec	PMENTER, PMEXIT, PMNOW, WRSUBPIR, READPDR0, READPDR1,PMSETTHID, PMCLRTHID
Fuse Programming	Product Spec	FUSECTL, FUSESHIFT, FUSECSR, FUSESSR,
Clock Comp Programming	Product Spec	PHASEDSHIFT
Secure/Unsecure modes	Product Spec	LOCK, UNLOCK
Status Reporting	Product Spec	TAPSTATUS, TAPSTATUSPRVT
ViewPLL	Product Spec	VIEWPLL

4.2 IO DFT:

This microprocessor inherits its bus protocol (including IO frequency characteristics - maximum of 800 MTS for Data IOs) from the previous microprocessor design. Key features including AC I/O Loopback DFT that is essential for screening out both functional and timing defects at the pads in order to reduce DPM. The detail design description and specifications were described in [14].

As we have described in the last year's ITC paper, a hybrid DFT architecture including test compression features has been implemented in this microprocessor. The detail description was presented in [17].

5. Burn-in (BI) DFT and methodology

PX implements a distinct feature WRPBIST-BI mode to increase toggle coverage of random logic, which is simple to program and easy to achieve target toggle coverage as shown from simulation data.

Pentuim4 CPUs use traditional ucode-based BIST patterns to toggle CPU circuit including both array and random logic. While BIST can achieve decent array toggle coverage, it doesn't achieve enough toggle coverages on random logic. Thus, SBFT and FRITS patterns were later added to achieve reasonable toggle coverage for random logic. For a subsequent processor, even all of the above methods were not enough to achieve the desirable logic toggle coverage. Therefore, scan patterns were used to achieve the required coverage. In addition, to avoid the significant efforts of generating ATPG patterns for every step of silicon, random patterns, rather than ATPG patterns, were applied to random logic. This was done after thorough analysis and some experiments to demonstrate that the contention during BI is not detrimental for circuit following certain design-for-BI rules.

In contrast, PX uses an on-die WRPBIST logic to generate pseudo-random pattern and shift the patterns thru scan chains to achieve very high toggle coverage. Throughout WRPBIST-BI mode (Figure 5), no functional clock of targeted logic is triggered, thus, the potentially detrimental contention is totally avoided, which eliminate the circuit design rules to tolerate contention and avoid any lingering doubt on whether contention can actually impact long term reliability of circuit. It is also easy to generate WRPBIST BI patterns because the patterns only need to set up the essential control and are independent of logic content of the block to be toggled. SBFT and FRITs are reserved as the backup methods and they will be used only if WRPBIST-BI has unexpected issue or even higher toggle coverage for random logic is desirable.

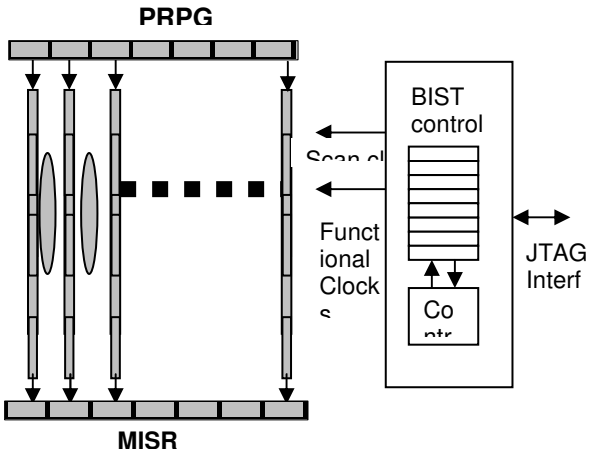


Figure 5 WRPBIST –BI Architecture

6. Design –for –Debug (DFD)

In the technology readiness phase, we had very active debate about either to treat Debug Scanout chains as part of the scan system. After many analyses, we decided to adopt the traditional design approach that completely separate scanout system from the ATPG scan chains. We allocated 10% of the full chip FFs as the signature cells and 1% scanout snapshot with an ability of performing system clock freeze and scan dump. The primary reason of this approach is to prevent the generation of excessive power due to scan at the system speed. The other key reason is to keep ATPG chains unpolluted by the scanout system. The detail of description of Intel micro-processor DFD features were described in [15].

7. Conclusion

The key contributions of the PX Microprocessor DFX team can be summarized as follows:

1. The PX DFX team has developed a highly competitive DFT and test strategy that enabled the first Intel full chip ATPG methodology with a very low scan overhead. By applying Skip Scan techniques and some innovative scan reduction methods, we were able to use ~52% of the full chip sequential and still remain nearly full chip ATPG (except NSBs) without scarifying the targeted performance, power and silicon budget. The total overhead estimated for scan DFT overhead is <3% of the chip area.

2. To overcome the issues of HVM power or Vcc Droop problems and the potential test data volume explosion, we developed a full chip partition ATPG structure and

methodology with a flexible scan DFT structure to enable the required clocking control logic. As a result, we have overcome the limitation of the commercial tools and manufacturing di/dt issues.

3. Novel Array DFT schemes were implemented to reduce approximately 50% of the total HVM test time for all arrays compared to earlier DAT scheme described in [11].

4. This is also the first Intel microprocessor that has a on-chip weighted random patterns BIST structure with a Hierarchical Structure including a test compression structure such s Illinois scan and X-Compact [17, 18, 19] The logic built-in self test structure also enabled the burn-in that saved significant efforts/resource in pattern generation and transformation.

5. In the DFD front, we addressed the HVM power issues by separating the scanout system and the traditional scan system. Scan for Debug techniques were explored by using system clock freeze features to take advantage of the large amount of scan sequential on chip.

In summary, the PX DFX team has developed a highly competitive DFT and test strategy that will serve as a BKM (best known method) for future Intel microprocessors to establish better microprocessor test methods that will significantly reduce product test cost and still achieving the highest product quality.

8. Acknowledgement

The authors would like to recognize the significant contributions provided by the PX microprocessor DFX team and the PX architecture and design teams throughout the entire project.

References

[1] Nandu Tendolkar, et. al. , "Novel Technique for Achieving High At-Speed Transition Fault Test Coverage for Motrola's Microprocessors Based on PowerPC Instruction Set Architecture" Pro of the 20th VTS'02

[2] C. Pyron et. al., "DFT Advances in the Motorola's MPC7400, a PowerPC Microprocessor",Proc IEEE ITC, 1999, pp 137-146

[3] R. Scott Fetherston et' al. "Testability Features of AMD-K6 Microprocessor", Proc. ITC 1997, pp 424-432

[4] M. Kruko et. al., "Microprocessor Test and Test Tool Methodology for the 500 MHz IBM S/390 G5 Chip", Proc. ITC 1998, pp717-726

[5] T. J. Wood , "The Test and Debug Features of the AMD-K7 Microprocessor", Proc. ITC 1999, pp130-136.

[6] L. Lai et. al. "Logic BIST Using Constrained Scan Cells", Proc. VTS 2004..

[7] Klenke, R.H.; Williams, R.D.; Aylor, J.H.; Parallelization Methods for Circuit Partitioning Based Parallel ATPG; Proc. of IEEE VLSI Test Symposium, 1993.

[8] Aguado, M.J.; Miranda, M.A.; de la Torre, E.; Lopez-Barrio, C.; A Dynamic Communication Strategy for the Distributed ATPG System DPLATON; Proc. of Design Automation Conference, 1993.

[9] Wolf, J.M.; Kaufman, L.M.; Klenke, R.H.; Aylor, J.H.; Waxman, R.; An Analysis of Fault Partitioned Parallel Test Generation; IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 15, Issue 5, May 1996.

[10] K. Zarrineh, et. al, "Self-Test Architecture for Testing Complex Memory Structures," Pro. Int. Test Conf. pp. 547-556, Oct, 2000.

[11] A. Sivaram, et. al. "Efficient Embedded Memory Testing with APG," Pro. Int. Test Conf. p 47-54, Oct.,2002.

[12] P. Parvathala, K. Maneparambil and W. Lindsay, "FRITS – a microprocessor Functional BIST Method," Pro. Int'l. Test Conf., Pp. 590~598, Oct., 2002.

[13] A. Meixner and J. Banik "Weak Write Test Mode: an SRAM Cell Stability Design for Test Technique," Pro. Int'l. Test Conf. pp. 1043~1052, Nov. 1997.

[14] M. Tripp, T.M. Mak, and A. Meixner "Elimination of traditional Functional Testing of interface Timings at Intel," Pro. Int'l. Test Conf., Sept, 2003.

[15] A. Carbine and D. Feltham, "Pentium Pro Processor Design for Test and Debug," pp. 294~303, Int' Test Conf. Nov. 1997.

[16] J. Waicukaiski at. Al. "Transition Fault Simulation by Parallel Pattern Single Fault Propagation," Proc. IEEE ITC, 1986, pp542-549

[17] D. Wu; M. Lin, et. al. "H-DFT: A Hybrid DFT Architecture for Low-Cost High Quality Structural Testing," Pro. Int'l Test Conf. pp. 1229~1238, Sept. 2003.

[18] Mitra, S., and K.S. Kim, "X-Compact: An Efficient Response Compaction Technique for Test Cost Reduction," Proc. Intl. Test Conf., pp. 311-320, 2002

[19] Hamzaoglu, I., and J.H. Patel, "Reducing Test Application Time for Full Scan Embedded Cores," Proc. Intl. Symp. Fault-Tolerant Computing, pp. 260-267, 1999.