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

0-7803-8106-8/03 $17.00 Copyright 2003 IEEE

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.

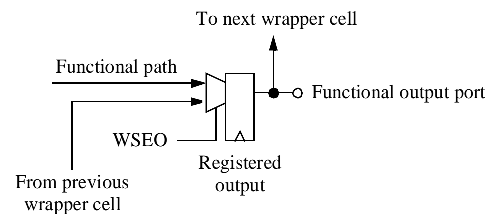
4.2 Strategy

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

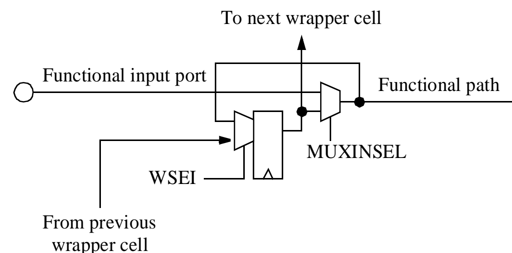
The test wrapper utilizes both dedicated and shared wrapper cells. A shared wrapper cell is one that serves a dual purpose of both functional mode and the control and observe function needed for the wrapper cell. A shared wrapper cell can only be used on a input or output that is registered. Figure 1: "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

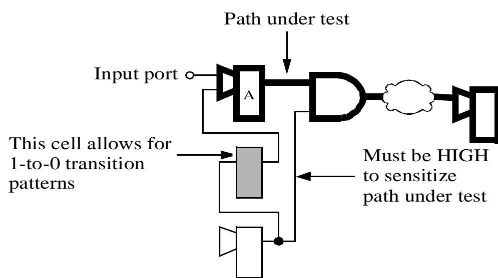
There are two wrapper scan enables. One wrapper scan enable, WSEI, connects to wrapper cells adjacent only to input ports. The other wrapper scan enable, WSEO, connects to wrapper cells adjacent only to output ports. The primary reason that the wrapper scan enables are separated is to allow the capability 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

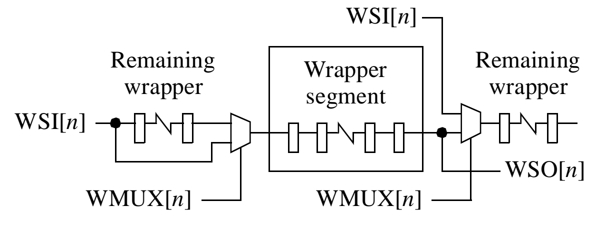
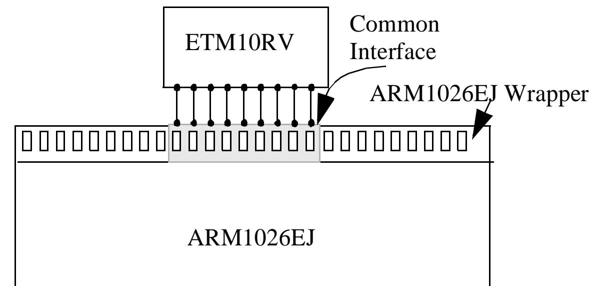


Figure 5: ETM10RV/ARM1026EJ Common Interface

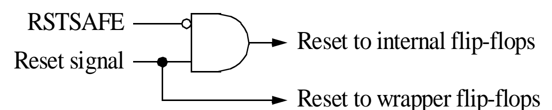


4.4 RSTSAFE

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.

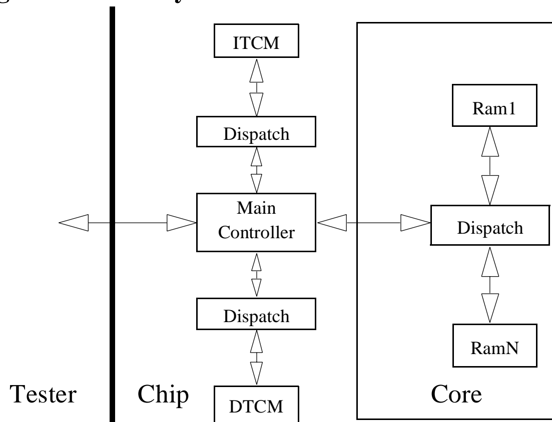
5.3 Implementation

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.

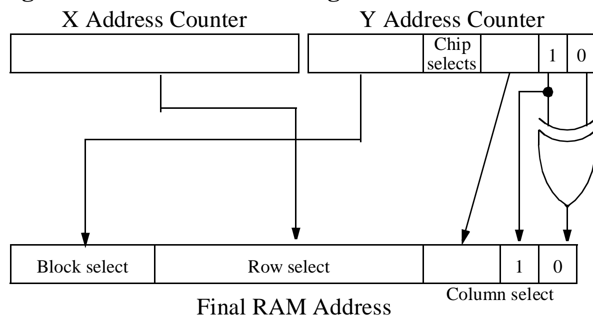
5.3.3 Address Scrambling

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

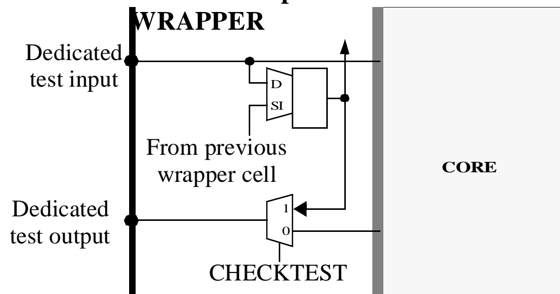


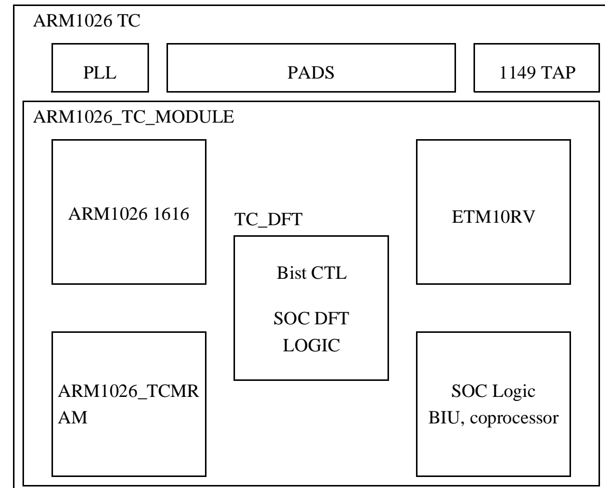
Figure 10: "CheckTest Example" shows a dedicated test input attached directly to the core, but also attached to the D input of a MuxD-flip-flop. A pattern is created that exercises the dedicated test input as it would be exercised during an intest mode. Consider an example of a test signal that is static, 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.

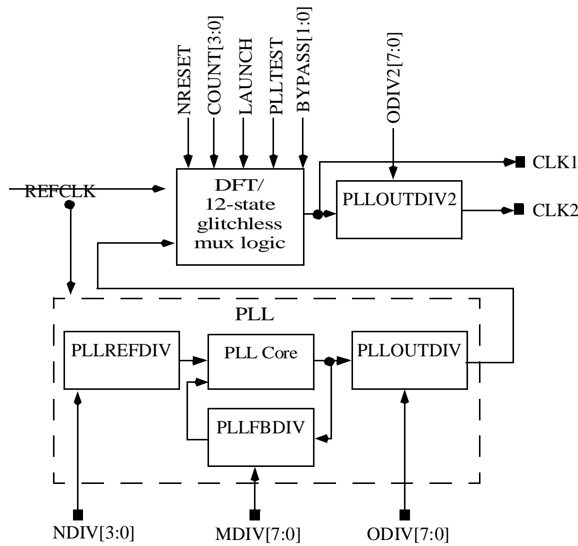
8.1.2 Strategy

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.

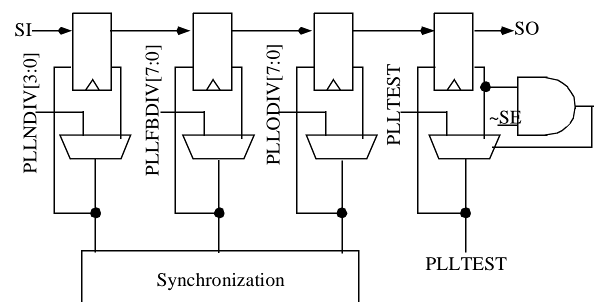
SCANSEL - Selects between control and observe chain. Chains must be separate to insure pll lock is not lost.

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.

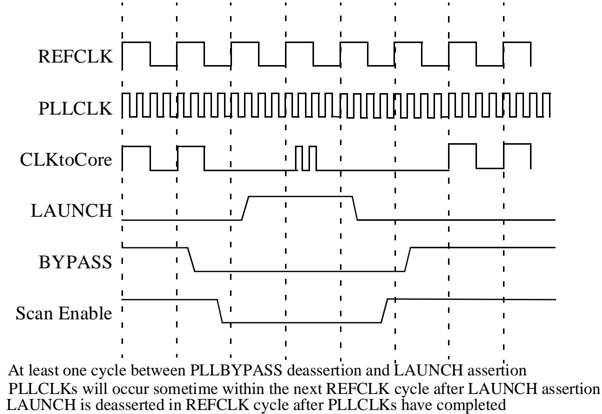
BYPASS and LAUNCH are never toggled in the same REFCLK cycle. This is a necessity for ATPG input setups/controls and also allows for simpler

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.*
- *Cumulative 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

Table 5: ARM1026EJ_88 Path Delay Coverage

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

order. A portion of the pattern set can be used if all of the patterns do not fit in the memory.

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

ARM and all other trademarks indicated as such herein are of ARM, Ltd. All other tradenames, trademarks and registered trademarks are the property of their respective owners.