A Comprehensive Memory Validation Strategy Using Advanced Processor Features

Introduction

Experience has shown that stable memory is crucial for reliable use of modern electronic devices. Electronic devices use entire memory subsystems for storing applications, running programs, and storing data. Faulty memory can prevent a device from running, cause it to crash unexpectedly, or behave unreliably. Understanding test strategies for various memory technologies is of the utmost importance.

What's Inside

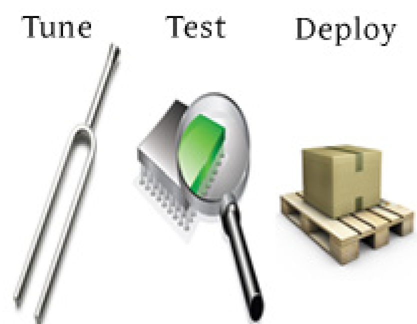
INTRODUCTION
TUNE, TEST AND DEPLOY
COMPLEX MEMORY SUBSYSTEMS
MEMORY TEST SETUP
DDR MEMORY TUNING
INSIDER TIP #1: USE SIMULATED SPD DATA
MEMORY VALIDATION STAGES
INSIDER TIP #2: USE DMA CHAINING
DEPLOYING THE TEST SOLUTION
INSIDER TIP #3: FOCUS ON ASSEMBLY ERRORS
A CUSTOMER STORY
SUMMARY
COMPANY INFORMATION

Memory sizes, data rates, and frequencies continue to increase and modern testing has not kept up. Using electrical, emulated, or user applications to test memory subsystems have proven unable to detect and isolate some memory issues. Certain techniques, such as using scan chains to test memory, cannot reach the memory when it is contained within a complex system-on-a-chip (SoC). Additionally, the layout and assembly of a memory device may be without issue, however, the configuration results in memory that mostly works, but hangs or crashes under various environmental conditions. Finally, many memory devices, such as Flash, have a limited life span requiring a reliable test solution to ship with the final device.

Presented is a comprehensive memory validation strategy: for automated tuning of memory, accurately determining the reliability of entire memory subsystems, pinpointing faults when repair is required, and deploying the same core solution used in design and manufacturing with the final product.

Tune, Test and Deploy

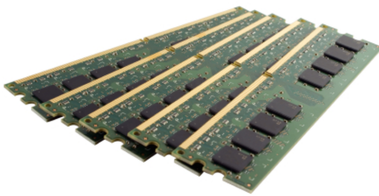
This strategy uses functional software running at full speed on the device to tune and test entire memory subsystems. Then deploy that software, in a format most suitable for field use, with the final application for self-testing the entire memory system.



Described is a test mechanism that functionally validates the memory, testing the memory at the fastest speeds possible, and stressing the memory using advanced processor features.

Complex Memory Subsystems

By memory subsystem, we are referring to all types of memory technologies, on-chip, on-board, on-bus, embedded in other ICs, across interconnects, and from board to board. This includes cache memory (data, instruction, level one, and level two), random access memory (SRAM, CRAM, MRAM, and PRAM), SDRAM (using DDR1, DDR2, DDR3, or future data rates), Flash (NAND and NOR), and memories contained within other integrated circuits.



The need for new test techniques is required with the release of new memory technologies such as those embedded inside a System-on-a-Chip (SoC). In addition, faster memories such as double-data-rate (DDR) memory require additional configuration and tuning steps that pertain directly to the circuit board layout.

Many new system designs using DDR memory require a manual process of configuring memory, tuning it for performance, and testing the memory at those settings. Environmental testing, testing at a variety of temperatures, is also essential for determining the stability of a memory system. Our experience has found that DDR memory may work mostly at a given configuration, but after long durations the operating system and application crashes with minimal information for fault isolation. We have seen engineers spend months attempting to isolate and correct memory faults. Our strategy has discovered issues in an hour or two, with fault isolation and correction completed in a day or two.

Memory Test Setup

This paper describes test techniques using software running on the target device. The goal is to use software written in a high-level and portable language, such as the "C" programming language, which requires memory for code execution. A stable test setup separates the memory under test and memory used by software. For example, when testing SDRAM, the software uses only SRAM, or cache memory. This configuration allows the test software to alter some DDR configuration settings on the fly.

A third goal is flexibility, accomplished by providing various images that can execute directly from bootable memory, such as Flash, or execute directly from RAM. This option allows the user to use memory that appears stable for testing alternate memories.

Finally, a desirable set up loads the test code into memory using a JTAG interface. Once the test application is running on the target, it uses a serial or Ethernet interface to transfer new images from a host to the target. This configuration provides very fast image transfers and the fastest device programming speeds.

DDR Memory Tuning

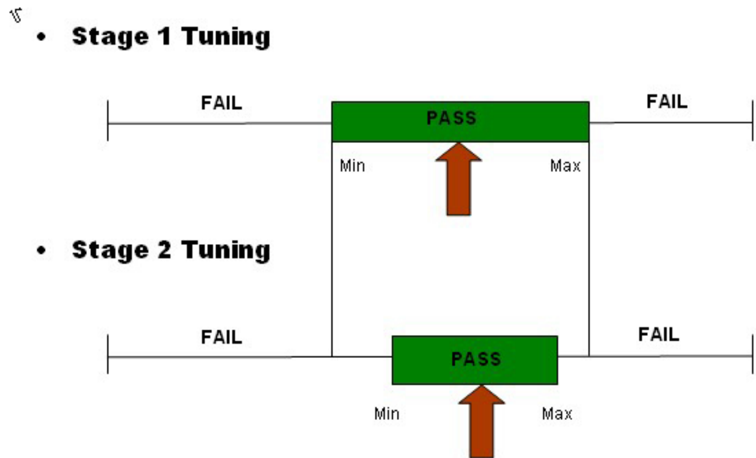
Our process begins with memory tuning. We use two stages for tuning. The first stage is an automatic single-stage tuning process followed by an interactive two-stage tuning process. During all tuning steps, a comprehensive suite of memory tests validate that the device continues to function properly with the new configuration settings.

The first step in tuning DDR memory is to configure static settings based on the memory's technical datasheet. These settings typically vary from one board design to another. We have found it is best to

store these values in a single text file, that the user can easy update and use to regenerate the test application specific to the target circuit board. This step may take a try or two to get stable enough settings for minimal code execution. Once you are able to run software on the target successfully, and have it execute test commands, the next step is to run the suite of memory validation tests. Once all tests pass, you can move on to automatic tuning.

Stage One Tuning:

1. Adjust either the read or write delay clock timing value, and hold the other constant
2. Increment through all possible timing values and run a memory test on SDRAM
3. Determine a range of timing values for the field for which memory passes
4. Selects a value in the middle and continue

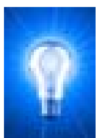


When manual interaction is required, we found it best to provide direct access to processor registers. This allows you to perform reads and writes with a single command avoiding the use of low-level assembly language commands. You may also probe the SDRAM interface with a logic analyzer, perform reads and writes, and compare the timing waveforms with the memory datasheet

Stage Two Tuning:

1. For better memory performance and stability, we suggest adding the capability of interactive tuning and optimization.
2. Execute an auto-tuning scan. As part of the tuning algorithm, try all possible values for the read-delay-timing value. This adjusts the timing of reads to SDRAM. Pick the optimal value. (This step assumes that writes to SDRAM are working. There is not a good way for software to adjust both the read timings and the write timings simultaneously, but you can hold read timings constant and tune the write timing value instead.)
3. Execute SDRAM test suite. Exception handling is critical because exceptions will occur as we adjust DDR settings. Handle the exception and dump pertinent information.

Insider Tip #1: Use Simulated SPD Data



Use simulated SPD data as a central mechanism for configuring SDRAM. Serial presence detect (SPD) is information stored in an electrically erasable programmable read-only memory (EEPROM) chip on a synchronous dynamic random access memory (SDRAM) module that contains the module's size, data width, speed, and voltage.

- Serial presence detect (SPD) is information stored in DIMM modules.
- SPD provides module's size, data width, speed, and voltage.
- Software reads and uses SPD data either form the actual DIMM or from an initialized data structure to automatically configure memory settings, enabling stable memory use.

Memory Validation Stages

Our memory validation strategy consists for four stages, with each succeeding stage adding a higher level of memory validation.

Stage	Description
Focus	Validate a single memory device
Flow	Validate data flow from one memory device to another
Stress	Burst data through all permutations of memory-to-memory devices
Performance	Measure read and write performance across all transfer widths

Focus Testing

Provide validation tests focused on validating individual IP blocks by exercising as many IP block internal functions and configuration options as possible. These tests exercise other parts of the chip indirectly, but focus on testing one IP block at a time. Provide numerous tests and an order that will be most effective in isolating errors.

The key of this stage of testing is to examine the various options associated with a given memory and use those advanced processor features to test that memory. The following is an example of a suite of tests that take advantage of advanced PowerPC processor features to test data cache.

Component	Test Case	Short Description
Data Cache	Parity Error Detection	Verifies that the data cache can detect and report parity errors
	Multi-hit Detection	Verifies that the data cache can detect when two cache lines match the same address
	Write Back Operation	Verifies write-back operation of the data cache
	Cache Line Replacement	Verifies the round-robin replacement algorithm of the data cache
	Cache Block Invalidate	Verifies that a single cache block can be marked invalid
	Cache Congruence Class Invalidate	Verifies that the entire data cache can be marked invalid
	Cache Line Locking	Verifies that data can be locked into the cache
	Cache Memory Tests	Verifies reads/writes to a locked portion of the data cache.

The following sequence provides a logical order of testing for SDRAM: check data bus lines, check address lines, stress the data bus, verify the SDRAM device, verify data caching and bursting. You can reconfigure the SDRAM tests in many ways to troubleshoot problems according to your needs. To reconfigure the SDRAM tests, you modify the appropriate environmental variables and run the test again. For example, set the bus width, set the address range, and configure the incrementing pattern test, configuring the data-bus noise test, or improvising your own test combinations. Key features are the ability to display all memory settings, read and write a single address, display memory regions, fill memory, and verify memory. Here is a synopsis of useful SDRAM test cases.

Component	Test Case	Short Description
SDRAM	Data Bus Walking 0, 1	Verify each data bit can be set to 0, 1
	Address Bus	Verify each address bit can be set to 1
	Simultaneous Switching Output	Stresses the data bus over a range of memory
	Data Bus Noise	Stresses the data bus to a single location
	Byte Access	Verifies byte access with incrementing pattern
	Word Access	Verifies word access with incrementing pattern
	March	Verifies that all bits can be programmed to a 0 and 1 over a memory range
	Data Bus Burst Noise	Tests the data bus for noise issues using burst transfers
	Burst Access	Verifies burst access with incrementing pattern

Flow Testing

This stage validates data flow from one memory device to another. These tests provide data path testing and concurrency testing of multiple IP blocks simultaneously. They take advantage of single channel DMA operations to execute bulk data transfers.

- Transfer from one on-chip memory source to another.
- Transfer from one on-board memory source to another.
- Transfer from one board to another.

Used are DMA operations to perform bulk data transfers with minimal CPU involvement. The CPU programs the DMA controller with the information necessary to perform the transfer. The controller then manages the transfer, and interrupts the CPU when the transfer completes or an error occurs. The DMA controller typically uses burst transfer modes to optimize performance.

Stress Testing

This stage uses advanced techniques to stress the memory device along with internal busses connecting the devices. One technique is to execute burst data transfers to stress test memory devices and memory buses. For example, with data cache enabled flush the cache to trigger a burst.

Next, use multiple DMA channels and chained operations to combine multiple bulk data transfers with other concurrent tests. The test software chains several DMA transfers together in a single DMA operation. Then the software performs multiple DMA operations concurrently on separate DMA channels. Each DMA operation starts and completes independently of other operations happening concurrently on other channels.

By transferring between different memories on separate channels, for example from Flash to SDRAM on channel 1, from SDRAM to PCI memory on channel 2, numerous internal and external busses of the system are utilized and tested concurrently.

Performance Testing

For performance testing, provided is a wide assortment of tests that you use to measure the read and write performance of your memory devices under various conditions. Provided is the ability to run the tests individually to measure the throughput and average access times to memory. You can also run the tests as part of a preprogrammed test suite.

Each performance category contains several variants. The tests command the processor to use 8, 16, 32, or 64-bit accesses, with and without caching. The CPU incurs a small amount of overhead as it measures access times, and this overhead is included in the performance measurements. Therefore, the performance statistics may be slightly less than the maximum possible from the device.

Provided is a separate test capability to test I/O channel functionality and performance. This test engine works with any block-based device (SAS, SATA, IDE, Fibre) and measures I/O performance using custom batches of concurrent I/O operations. This test engine includes memory-resident pseudo-targets, when actual devices are not available, for live or simulated measurements. With this test capability, storage product suppliers can check raw hardware speed of the I/O channels out to block-based devices before any other software is running on the board.

Insider Tip #2: Use DMA Chaining

Use DMA chaining concurrently with interrupts to maximum stress testing. This method has been used repeatedly to successfully and quickly isolate memory contention issues.

- Set up multiple DMA operations to test from one memory source to another, across a bus, or even from board to board.
- Concurrently use outside stimulus to initiate interrupt-driven test routines.
- Use outside stimulus and interrupt driven routines to add additional stress on memory buses and devices.

Deploying the Test Solution

The final product lifecycle addressed is the ability to ship a memory test solution with the electronic device. This allows a field application engineer, or a returns engineers, the ability to run quickly memory diagnostics on a unit. In addition, shipping self-tests with the device allow it to be tested at your desired frequency, such as during every power on cycle.

The goal of the final step is to be able to reuse any or all tests developed in prior stages. Tests developed, tested, and used for design validation may prove very useful for manufacturing test, whether that is for internal use or for your contract manufacturer.

Insider Tip #3: Focus on Assembly Errors

Focus self-test on potential assembly errors, not design related issues.

- Wring out design issues during the earlier test phases.
- Compare and contrast coverage versus self-test execution time.
- Do you need to test all memory regions? Shorten test regions to speed up test times.

A Customer Story

A company in the United Kingdom spent several months attempting to debug a DDR memory failure using hardware tools and running the final application on the new circuit board. The application would run fine, but after many hours, or days, the system would crash unexpectedly. This customer found Kozio's web site and enquired whether we could help. We said yes and we did help.

They shipped a circuit board to us in Colorado and we had our software and memory tests running the same day the board arrived. The default memory test suite and simple DMA test suite resulted in successful executions, but a more stressful DMA loop test consistently produced memory failures in a matter of a few minutes.

Based on the customer's test description of the failure, heavy I/O to disk drives, Kozio was able to produce a test sequence that detected memory sequence failures on the customer's circuit board. The test used several concurrent DMA chains, with each DMA chain transferring 6 MB of data. With four chains per channel, four channels operating concurrently, and two 6 MB buffers for each chain the test utilized 192 MB of the DDR2 SDRAM. Also used were an Ethernet controller and external network traffic to generate additional memory contention.

The same test sequence was executed on an evaluation board 9600 times with no failures detected. Here is a sample of the information displayed after a test failure on the customer's board.

Memory Address	Expected Data	Actual Data	Byte Lanes with Error
0x40AA9AC0	0x11223344	0x11**DD**33**BB**	Bytes 1 and 3
0x40AA9ACC	0x11223344	0x1122**CC**44	Byte 2
0x40AA9AD0	0x11223344	0x11**DD**33**BB**	Bytes 1 and 3

Kozio provided a detailed write up and analysis of the test failures. The stress testing results clearly indicated that DMA memory writes were failing in single byte lanes. Using this information, the customer made power changes to the circuit board design and solved their memory failures.

Summary

Tune, test and deploy complex memory subsystems using advanced memory diagnostics that take advantage of advanced processor features.

- Stable memory is crucial.
- Stable memory results from extensive tuning and reliable and deterministic tests that pinpoint faults.
- Best solution – embedded at-speed memory diagnostics for tuning, testing, and deploying. One that uses advanced memory device features, advanced processor features such as DMA operations, and multiple testing techniques for focus, flow, stress, and performance testing.

Company Information

Kozio provides advanced memory diagnostics in an at-speed embedded test application along with fully integrated test management applications under the banner of kDiagnostics®. For more information please call or email us.

Trusted Hardware Validation

Kozio, Inc.
2400 Trade Center Ave
Longmont, CO 80503
www.kozio.com
info@kozio.com

© 2008 Kozio, Inc. All rights reserved. Kozio, kDiagnostics, SequenceRunner, SequenceReporter, and ValidationAssistant are trademarks or registered trademarks of Kozio, Inc.