

# How to Detect Potential Memory Problems Early in FPGA Designs

System compatibility testing for FPGA memory requires methods other than traditional signal integrity analysis.

by Larry French  
FAE Manager  
Micron Semiconductor Products, Inc.  
lfrench@micron.com

As a designer, you probably spend a significant amount of time simulating boards and building and testing prototypes. It is critical that the kinds of tests performed on these prototypes are effective in detecting problems that can occur in production or in the field.

DRAM or other memory combined in an FPGA system may require different test methodologies than an FPGA alone. Proper selection of memory design, test, and verification tools reduces engineering time and increases the probability of detecting potential problems. In this article, we'll discuss the best practices for thoroughly debugging a Xilinx® FPGA design that uses memory.

## Memory Design, Testing, and Verification Tools

You can use many tools to simulate or debug a design. Table 1 lists the five essential tools for memory design. Note that this is not a complete list as it does not include thermal simulation tools; instead, it focuses only on those tools that you can use to validate the functionality and robustness of a design. Table 2 shows when these tools can be used most effectively.

This article focuses on the five phases of product development, as shown in Table 2:

- **Phase 1** – Design (no hardware, only simulation)
- **Phase 2** – Alpha (or Early) Prototype (design and hardware changes likely to occur before production)
- **Phase 3** – Beta Prototype (nearly “production-ready” system)
- **Phase 4** – Production
- **Phase 5** – Post-Production (in the form of memory upgrades or field replacements)

## The Value of SI Testing

SI is not a panacea and should be used judiciously. SI should not be overused, although it frequently is. For very early or alpha prototypes, SI is a key tool for ensuring that your system is free of a number of memory problems, including:

- Ringing and overshoot/undershoot
- Timing violations, such as:
  - Setup and hold time
  - Slew rate (weakly driven or strongly driven signals)
  - Setup/hold time (data, clock, and controls)

Tool	Example
Electrical Simulations	SPICE or IBIS
Behavioral Simulations	Verilog or VHDL
Signal Integrity	Oscilloscope and probes; possibly mixed-mode to allow for more accurate signal capture
Margin Testing	Guardband testing and four-corner testing by variation of voltage and temperature
Compatibility Testing	Functional software testing or system reboot test

Table 1 – Memory design, test, and verification tools

- Clock duty cycle and differential clock crossing (CK/CK#)
- Bus contention

By contrast, SI is not useful in the beta prototype phase unless there are changes to the board signals. (After all, each signal net is validated in the alpha prototype.) However, if a signal does change, you can use SI to ensure that no SI problems exist with the changed net(s). Rarely – if ever – is there a need for SI testing in production.

SI is commonly overused for testing because electrical engineers are comfortable looking at an oscilloscope and using the captures or photographs as documentation to show that a system was tested (Figure 1). Yet extensive experience at Micron Technology shows that much more effective tools exist for catching failures. In fact, our experience shows that SI cannot detect all types of system failures.

### Limitations of SI Testing

SI testing has a number of fundamental limitations. First and foremost is the memory industry migration to fine-pitch ball-grid array (FBGA) packages. Without taking up valuable board real estate for probe pins, SI is difficult or impossible because there is no way to probe under the package.

Micron has taken several hundred

Tool	Design	Alpha Proto	Beta Proto	Production	Post-Prod
Simulation – Electrical	Essential	Very Valuable	Limited Value	Rarely Used	No Value
Simulation – Behavioral	Essential	Very Valuable	Limited Value	Rarely Used	No Value
Signal Integrity	Unavailable	Critical	Limited Value	Rarely Used	No Value
Margin Testing	Unavailable	Essential	Essential	Essential	Essential
Compatibility	Unavailable	Valuable	Essential	Essential	Essential

Table 2 – Tools for verifying memory functionality versus design phase

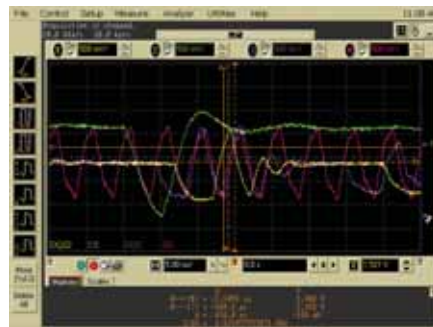


Figure 1 – Typical signal integrity shot from an oscilloscope

thousand scope shots in our SI lab during memory qualification testing. Based on this extensive data, we concluded that system problems are most easily found with margin and compatibility testing. Although SI is useful in the alpha prototype phase, it should be replaced by these other tests during beta prototype and production.

Here are some other results of our SI testing:

- SI did not find a single issue that was not identified by memory or system-level diagnostics. In other words, SI found the same failures as the other tests, thus duplicating the capabilities of margin testing and software testing.

- SI is time-consuming. Probing 64-bit or 72-bit data buses and taking scope shots requires a great deal of time.
- SI uses costly equipment. To gather accurate scope shots, you need high-cost oscilloscopes and probes.
- SI takes up valuable engineering resources. High-level engineering analysis is required to evaluate scope shots.
- SI does not find all errors. Margin and compatibility testing find errors that are not detectable by SI.

The best tests for finding FPGA/memory issues are margin and compatibility testing.

### Margin Testing

Margin testing is used to evaluate how systems work under extreme temperatures and voltages. Many system parameters change with temperature/voltage, including slew rate, drive strength, and access time. Validation of a system at room temperature is not enough. Micron found that another benefit of margin testing is that it detects system problems that SI will not.

Four-corner testing is a best industry practice for margin testing. If a failure is

### How Does the Logic Analyzer (or Mixed-Mode Analysis) Fit In?

You may have noticed that Table 1 does not include logic analyzers. Although it is rare to find a debug lab that does not include this tool as an integral part of its design and debug process, we will not discuss logic analyzers in this article. Because of the cost and time involved, they are rarely the first tool used to detect a failure or problem in a system. Logic analyzers are, however, invaluable in linking a problem, after it has been identified, to its root cause. Like signal integrity (SI), logic analyzers should be used after a problem has been detected.

## ...margin and compatibility testing will identify more marginalities or problems within a system than traditional methods such as SI.

going to occur during margin testing, it will likely occur at one of these points:

- Corner #1: high voltage, high temperature
- Corner #2: high voltage, low temperature
- Corner #3: low voltage, high temperature
- Corner #4: low voltage, low temperature

There is one caveat to this rule. During the alpha prototype, margin testing may not be of value because the design is still changing and the margin will be improved in the beta prototype. Once the system is nearly production-ready, you should perform extensive margin testing.

### Compatibility Testing

Compatibility testing refers simply to the software tests that are run on a system. These can include BIOS, system operating software, end-user software, embedded software, and test programs. PCs are extremely programmable; therefore, you should run many different types of software tests.

In embedded systems where the FPGA acts like a processor, compatibility testing can also comprise a large number of tests. In other embedded applications where the DRAM has a dedicated purpose such as a FIFO or buffer, software testing by definition is limited to the final application. Thorough compatibility testing (along with margin testing) is one of the best ways to detect system-level issues or failures in all of these types of systems.

Given the programmable nature of Xilinx FPGAs, you might even consider a special FPGA memory test program. This program would only be used to run numerous test vectors (checkerboard, inversions) to and from the memory to validate the DRAM interface. It could eas-

ily be written to identify a bit error, address, or row – in contrast to the standard embedded program that might not identify any memory failures. This program could be run during margin testing. It would be especially interesting for embedded applications where the memory interface runs a very limited set of operations. Likely, this type of test would have more value than extensive SI testing of the final product.

### Tests Not To Ignore

The following tests, if ignored, can lead to production and field problems that are subtle, hard to detect, and intermittent.

#### Power-Up Cycling

A good memory test plan should include several tests that are sometimes skipped and can lead to production or field problems. The first of these is power-up cycling. During power-up, a number of unique events occur, including the ramp-up of voltages and the JEDEC-standard DRAM initialization sequence. Best industry practices for testing PCs include power-up cycling tests to ensure that you catch intermittent power-up issues.

Two types of power-up cycling exist: cold- and warm-boot cycling. A cold boot occurs when a system has not been running and is at room temperature. A warm boot occurs after a system has been running for awhile and the internal temperature is stabilized. You should consider both tests to identify temperature-dependent problems.

#### Self-Refresh Testing

DRAM cells leak charge and must be refreshed often to ensure proper operation. Self-refresh is a key way to save system power when the memory is not used for long periods of time. It is critical that the memory controller provide the prop-

er in-spec commands when entering and exiting self-refresh; otherwise, you could lose data.

Like power-up cycling, self-refresh cycling is a useful compatibility test. If an intermittent self-refresh enter or exit problem is present, repeated cycling can help detect it. Applications that do not use self-refresh should completely skip this test.

### Sustaining Qualifications

One last area to consider is the test methodology for sustaining qualifications. That is, what tests should you perform to qualify a memory device once a system is in production? This type of testing is frequently performed to ensure that an adequate supply of components will be available for uninterrupted production.

During production a system is stable and unchanging. Our experience has shown that margin and compatibility testing are the key tests for sustaining qualifications. Because a system is stable, SI has little or no value.

### Conclusion

In this article, our intent has been to encourage designers to rethink the way they test and validate FPGA and memory interfaces. Using smart test practices can result in an immediate reduction in engineering hours during memory qualifications. In addition, proper use of margin and compatibility testing will identify more marginalities or problems within a system than traditional methods such as SI. No “one-size-fits-all” test methodology exists, so you should identify the test methodology that is most effective for your designs.

For more detailed information on testing memory, see Micron’s latest DesignLine article, “Understanding the Value of Signal Integrity,” on our website, [www.micron.com](http://www.micron.com). 