

Solutions to Embedded System Design Challenges

Part I



Time-Saving Tips to Improve Productivity In Embedded System Design, Validation and Debug

Hello, my name is Mike Juliana. Today's presentation is the first of two sessions that will examine ways to improve productivity in embedded system validation and debug. This session, "Begin With the End in Mind", will look specifically at things that can be done early in the design cycle that can lead to improved productivity. Part II will then highlight specific tips that can be used in the lab to shave valuable time out of the validation cycle.

Embedded System Design Challenges

- ▶ Real-World Conditions of Time Constraints and Cost Limitations
 - Time to market
 - Compressed schedules
 - Additional features
 - Improved quality
 - Increased complexity
 - Cost reductions
- ▶ Solution to the Challenge
 - Right instruments to save time
 - Right approach to address problems



To stay competitive in today's market place, we are being asked to complete our design in less time with fewer resources. We are being asked to improve reliability even though the designs are increasing in complexity and we are having to integrate new technologies such as serial buses into our products. Added up, it is enough to create a stressful environment at times.

To meet these challenges, we have relied on advances in our design tools that allow us design more efficiently these complex designs. Designing million-gate FPGAs can be readily done with today's FPGA tools. PCB layout tools handle 20-layer boards with increasing ease. But once our designs get into the lab, we need to find these same time-saving improvements to match the increased complexity of our designs. We have to reduce the debug and validation time. Planning for this challenge and matching the right instrument to the task and using the right measurement approach is crucial in being able to achieve this.

Right Instrument Saves Time

Example: MyScope® Customizable Oscilloscopes

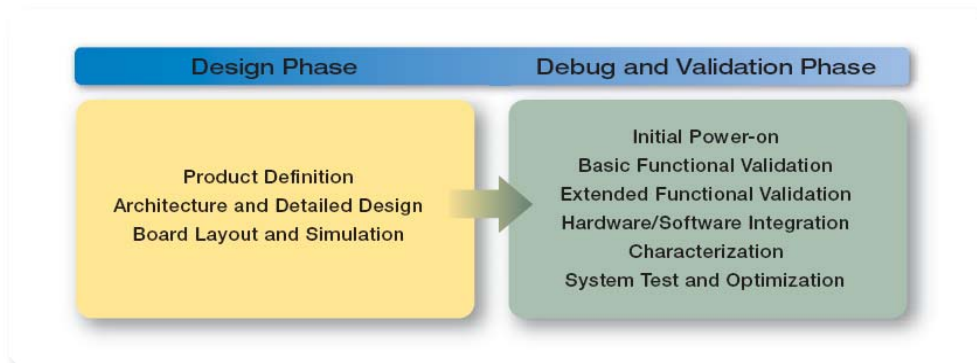
- ▶ Focus On Your Task, Not Your Instruments
 - Versatile tool that can be customized for many users
- ▶ Enables New Levels of Productivity and Efficiency
 - Allows you to pull frequently used features into one control window
- ▶ Simple to Learn, Without Programming
 - Simple drag and drop setup



Each of your test tools should allow you to focus on your debug task without being distracted by the instrument itself. For example, MyScope allows you to quickly and easily customize an oscilloscope to meet your unique measurement needs. With MyScope you can create an unlimited number of custom control windows using a simple drag and drop procedure. This could, among other things, enable multiple users of a shared oscilloscope to each have their own custom interface. It could also maximize efficiency in your day-to-day work by allowing you to focus your energy on the task at hand rather than navigating back and forth through multiple menus and submenus. This time-saving translates into increased productivity.

Again, you need to be able to configure and use each of your instruments to provide you the answers to your problems – not just the raw data.

Design Process Overview



► “Begin With the End in Mind”

- Debugging embedded systems starts with the design phase
 - Identify potential problem areas
 - Design for validation and debug
- Excitement can quickly turn to frustration during validation and debug
 - Rationalize and streamline your validation and debug process

Let’s step back a minute and look at the entire design process. There are really two major phases: these are the design phase and the debug and validation phase.

In the design phase, the marketing group defines the customer requirements. These requirements are then translated into a product by the design team. Architecture is done and hardware/software trade-offs are made. Finally, the detailed design is done.

Once the design is complete and a prototype is built, we start the next phase – the debug and validation phase. This phase validates the correctness of the design, corrects any problems and ensures the design can be produced reliably. It’s during this time that we spend a lot of time in the lab using an oscilloscope, a logic analyzer, or some other tools looking at our system; it is our job to find the problems and fix them.

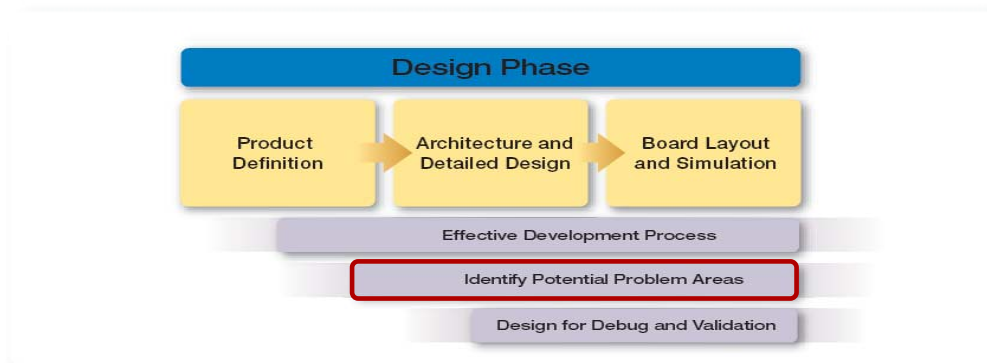
Beginning with the end in mind is a simple reminder that as we progress through the design phase, we make choices that have an impact on our ability to debug our system. Some of these choices are obvious. For example, do we provide an easy way of accessing the signals on our board? Other choices are not so obvious: our choice of technology, components, or the mechanical packaging can have a great impact on our ability to debug our system.

If only we could look into the future to see where the debug challenges were going to be. Unfortunately, the reality is that we rarely know exactly where problems will be but experience, that is learning from past mistakes, can help us. As we move from one design to another, we gain insight that can help us prepare for the next design.

Experience shows us that there are two types of problems that we will encounter: functional problems and signal-integrity related problems.

Design Phase

Identify Potential Problem Areas



► Functional Issues

- Test benches, stimulus models, test cases
- Simulation
 - Find and fix problems in the early stages
 - Limitation of simulations: problems related to synchronizing signals across clock boundaries, construction errors

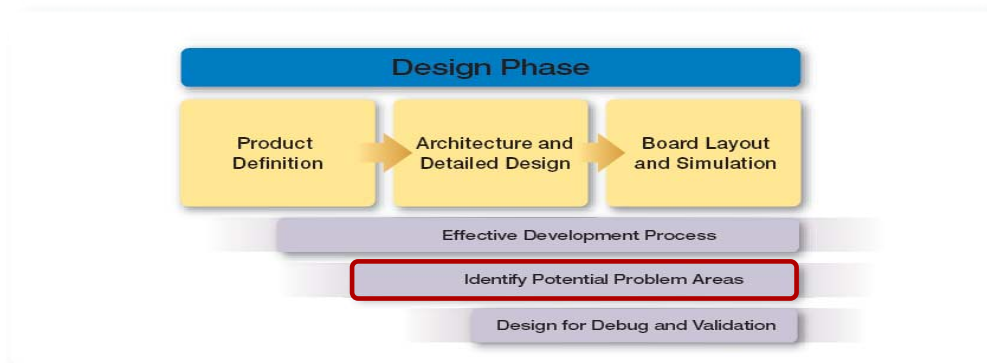
Functional errors occur for a variety of reasons. Maybe we have a misunderstanding of how a purchased component really operates. Maybe we have errors in the RTL-code of our FPGA or, worse, errors in a purchased IP block that is in our FPGA. We could also have problems with the software/hardware interface due to incomplete specifications or because of a misunderstanding between the hardware and software teams. The point is that there are many different reasons that functional errors might be present in our design. The trick is to find these problems as early as possible in the design process. Doing so will reduce the time required to fix them as well as reducing the cost of fixing them.

One of the best ways to catch functional problems early is to make sure you are using an effective development process. Teams with a defined process that incorporate peer reviews and gated milestones can catch many problems before the board is even made. Even if your company does not use formal reviews, you should develop your own checklist. Things like power distribution, power sequencing, clock distribution, and resets should be on this checklist and reviewed before a board is released to be built.

If the design incorporates FPGAs, you should do simulation to verify as much of the functionality as possible before the boards are built. This will find many of the problems that could be more difficult to find later on. Doing simulation should not be thought of as a cure-all however; simulation has limitations that need to be recognized. While simulation is great for finding functional issues, more complex interactions such as synchronizing signals across clock domains can often be difficult to thoroughly test. Simulation should not be your only way of verifying functionality. You should plan on having problems that need to be debugged once you get into the lab.

Design Phase

Identify Potential Problem Areas



► Signal Integrity Issues

- Power distribution
 - Power plane, by-pass capacitance on IC pins, clean low-noise voltage regulation, low impedance distribution, dynamic current changes
- Clock distribution
 - Phase-Locked Loop (PLL), margin
- Others concerns
 - Electromagnetic Interference (EMI), component tolerance

The other potential problem area can be broadly classified as signal integrity problems. In the ideal world, all of our signals would be complete and unimpaired. That is, a digital signal would have clean, fast transitions; stable, valid logic levels; accurate placement in time; and would be free of transients. Getting close to this ideal is getting harder and harder to do. Recognizing this and planning on having to find some these problems can eliminate many stressful hours in the lab.

At fast clock frequencies and fast edge rates, every design's detail becomes important. A host of variables can affect signal integrity: signal path design, the stack-up of our circuit board, transmission line effects and power distribution.

When problems do come up, we are likely to encounter them as problems in the digital form. That is, the binary signals on a bus or a device output will have incorrect values. The errors may appear in the waveform or timing view on a logic analyzer or they may show up at the state or even protocol level.

Problems in the digital domain are usually timing related. Bus contentions, setup and hold violations, metastability, and race conditions are examples of problems in the digital domain. Each of these can cause erratic signal behavior on a bus or a device output.

Problems in the analog domain such as low-amplitude signals, slow or fast transition times, glitches, crosstalk, and noise usually have they origins in the design of the circuit board or signal termination.

Not surprisingly, there is a high degree of interaction and interdependence between digital and analog signal integrity issues. For example, a slow rise time on an input can cause the output pulse to be delayed which could in turn cause bus contention in the digital environment.

Signal Integrity (SI) Challenges

► Sources of SI Problems Include

- Low speed logic families with fast edge rates can have SI problems
 - Faster edges turn lumped circuit trace loads into transmission lines
- Faster edges represent larger transient currents
 - Increased crosstalk, ground bounce; especially on wide buses



► SI Problems Result from Complex Interactions

- Output drivers
- Signal path layout
- Signal path loads
- Signal path termination
- Ground and power distribution

Any discussion of signal integrity should focus on the signal transitions in our system. This is where the action really is. It is the transitions that cause signal integrity problems. It is important to note that many digital systems have slower clock rates but may have very fast edges. The advances in semiconductor technology has made signal integrity a concern for designers using virtually any logic family. These fast edges carry high-frequency components regardless of clock rate. Fast edges bring lots of benefits but can also add many complications to our job.

These fast transition times require us to take extra care in making component, termination and layout choices. Remember – a circuit board trace of just 6 inches acts like a transmission line when driven with edges rates faster than 2 nanoseconds. When edges transition at high speeds, new signal paths are created. These paths don't show up in our schematics – they are parasitic. Nevertheless, they provide a means for signals to influence one another in unpredictable ways. For example, ground and power planes become part of the transmission line system that are formed by the signal runs, and in turn interact with each other. These interactions are known as crosstalk and ground bounce.

These are classic analog effects that lie at the heart of many of the failures in our systems. It is our responsibility to minimize these problems during the design phase and then to correct any issues that come up in the debug phase.

Design Phase

Identify Potential Problem Areas

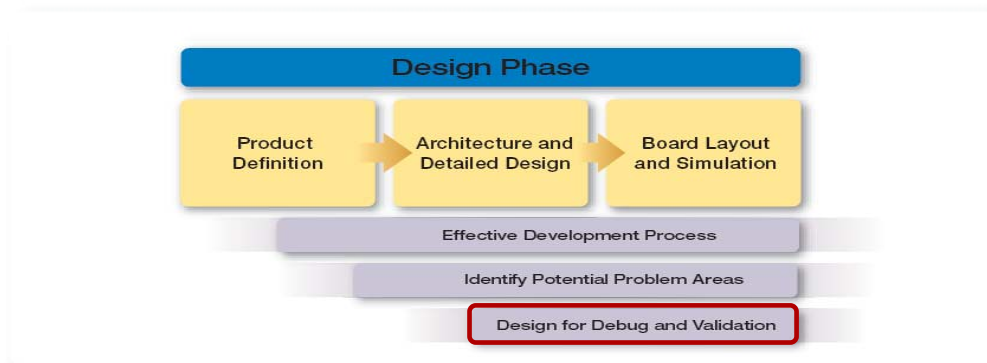
► Summary

- Think through your system
- Document potential problem issues
- Try to minimize them during design phase
- Think about how to address them if happens

By beginning with the end in mind and identifying potential problems early in the design cycle, we can avoid many painful hours in the lab. Think through the system. Where are problems likely to occur? Is this the team's first design with clock rates of 200 MHz? First time using a new serial bus technology? How would you address problems in these areas? The answer to these questions are the start of the validation and test plan that we need to develop during the design phase.

Design Phase

Design for Validation and Debug



► Issues to Address

- Which instruments to use
 - Signal source, oscilloscope, logic analyzer
- Which factors to include in the decision criteria
 - Banner specs, memory depth, modularity, analysis features
- How to address problem areas
 - Test points, probing

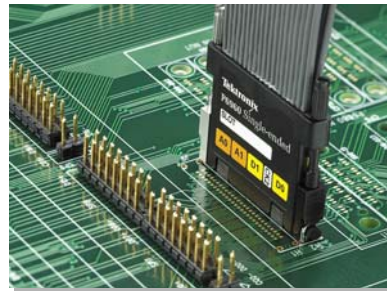
After identifying the potential problem areas, the next step is to make sure we design for debug and validation. The best way to do this is to develop a test plan. This plan needs to be developed early in the design phase because it will most likely impact the design of our board in some way. Any potential problem areas previously identified should be noted as risks and ways of reducing that risk should be in the test plan. The goal is to remove as many surprises as possible.

The plan should identify the functionality to be tested and how it will be done. It should also identify the interfaces and signals that need to be validated. Finally, it should detail the types of measurements that will need to be made and how they will be made.

By doing this in the design cycle, we can avoid the worst possible scenario – that is having no debug strategy when a problem does occur. No engineer is good enough to eliminate the need for troubleshooting altogether. The best we can do is plan for debug early.

Develop a Probing Strategy

- ▶ Probing Strategy Needs to be Developed at the Onset of the Design
 - Validation board/breakout board
 - Microprocessor/bus visibility
 - Key signal accessibility
 - Probe loading simulation



A key aspect of the test plan is a good probing strategy. Direct signal observations and measurements are the only way to discover many functionality problems and the causes of signal-integrity problems. This part of the test plan needs to consider many factors: where, what, and how to probe. For what instruments do we provide access? Logic analyzers? Oscilloscopes? Or both? Obviously the number of signals to be probed will factor into the decision. If we need to look at a 32-bit data bus, it could be hard to do that with an oscilloscope but if we only need to look at three critical signals, an oscilloscope is probably sufficient. For most embedded designs, it is best to plan for both logic analyzer and oscilloscope access. These are the two tools that we will use heavily in the validation and debug phase.

The goal, is obviously, to provide as much visibility into our system as possible. The more the better because as engineers we are great at solving problems once we can see them. But saying more is better is very simplistic. Where do we really need to provide access points? It would be nice to place them everywhere but in most cases this is just not practical. We are caught between two competing requirements. On one hand you have the design for debug requirement competing with the limited real estate on your board. Sometimes just getting the required functionality on a board is a problem – let alone adding test connectors. The question then becomes – how do we resolve this conflict?

Plan Ahead for Total Real-Time System Visibility

► Accessibility to the Signals

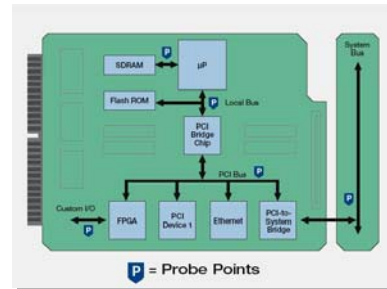
- Possible probing points for your logic analyzer

► Plan to Validate Your System as a Whole

- Total System Visibility

► Prioritize Your Needs

- Do I care about microprocessor/bus visibility?
- What happens if I can't see these signals and will it impact my schedule?
- What is the impact of the probing on my design?
- How do I minimize design risks?
- If I cannot access that signal, do I have any other alternative way?



The key again, is to begin with the end in mind and ask where the problems are likely to be. Let's look at the example here to walk through the process of identifying suitable logic analyzer probe points.

- First, how important is microprocessor visibility? Am I concerned just about visibility for hardware debug or will visibility for software debug be important as well?
- Second, which internal buses do I want to look at? Can this be done with just oscilloscope test points or do I need logic analyzer test points as well?
- Next, where do I think design margin will be an issue. How will this be validated? Will I need to do this across temperature and other environmental factors?
- What happens if I can't see my signals? How would this impact my schedule? Would it require a spin of my board?
- And finally, how will the test points and probes interact with my signals? Will they cause any problems because of excessive loading?

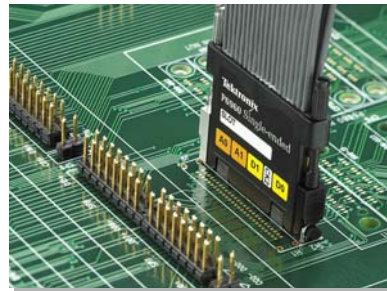
The design shown here is a typical embedded system with a microprocessor with flash and SDRAM, a local PCI bus, several peripherals, and a bridge to the system PCI bus. As a starting point, let's take the position that we want to provide access to all of the points shown with a P in the block diagram. Why is this?

First, access to the local bus would allow us to monitor and debug boot issues. There is where unknown hardware and software come together for the first time – often a high risk area. The SDRAM interface access would allow us to see data structures and system code stored in SDRAM. It would also allow us to do a real-time trace of the software execution. This will allow software performance analysis as well as debug of any hardware/software interaction problems. This could eliminate finger-pointing between the hardware and software teams.

One can make an argument for each of the other points in the system. Of course, this is the ideal list and what we should strive for but the reality, as we have mentioned, is that in most cases, we just don't have the board space to do this. We have to cut these probe points down. How do we cut down the ideal until it is something realistic? The trick is to minimize the impact. Ask the questions that we have stated about and in this case, at a minimum, we would want to keep visibility of the FPGAs since this is where much of the functionality or secret sauce of our system may be implemented. And once we have our logic analyzer probe points identified, we should identify any required oscilloscope probe points that are not part of logic analyzer probe points. These may be power supplies, clocks, and resets. And finally, remember to place an adequate number of easy to access ground points around the board to may probing with an oscilloscope easier.

Probe Selection

- ▶ **Oscilloscope Probes**
 - Impedance loading, bandwidth (>3x DUT frequency), rise time, connectivity
- ▶ **Logic Analyzer Probes**
 - Impedance loading, density, size, connectivity, propagation delay, crosstalk
- ▶ **Reduce Loading**
 - Design-in connectors/footprints, eliminate double probing



Once we have decided on the test points, we still need to answer the how – that is, which probes that we will use. The cardinal rule is that any probe we use needs to accurately transmit signals from the board to the acquisition system input.

For oscilloscope probes, there are a variety of things to consider. Two of the most important factors are the bandwidth and step response of the complete probe/acquisition system. These characteristics ensure adequate capture of the signal aberrations that define signal integrity. Ease of use and how the probe is connected to the system should also be considered.

A logic analyzer's probing scheme plays a crucial role in helping find many of the problems in our systems.

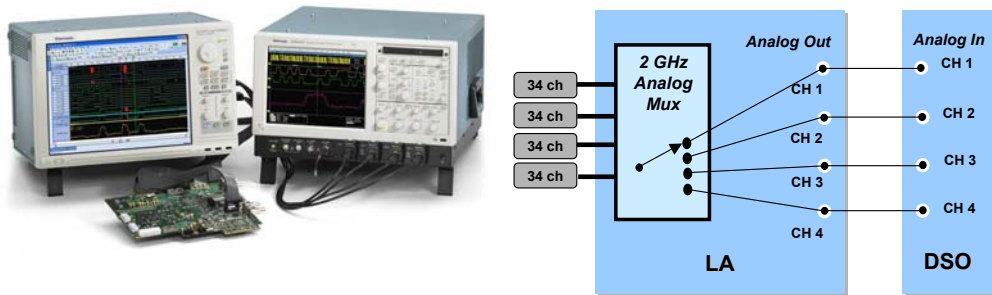
In most embedded systems, a dedicated test point is usually the most practical way to measure signals with a logic analyzer – especially if we are trying to capture a large number of signals such as a 32-bit address/data bus.

One option is to use square pins. This does provide a way of connecting the logic analyzer probe to our system there are issues. Most importantly performance will be lower than other options because of the added loading of the square pins. This loading will always be present and have an effect on the board's performance even when the logic analyzer is not connected.

Another choice is to mount a mictor connector on the board. These compact, high-density connectors provide a convenient connection point to the logic analyzer. Again, they provide fast, positive connections for large groups of signals but still will have an impact high-speed signal operation.

Even higher-density options such as the D-Max probing technology shown here have emerged to provide an alternative to conventional mictor probes. These probes require no connectors on the board. Instead, they mate directly to land pads on the board and are held in place by small retention posts. These probes address lead inductance and reduce capacitive loading to about 0.5 picofarads. They come in single-ended and differential versions. These types of probe offer the highest performance as well as the highest-density.

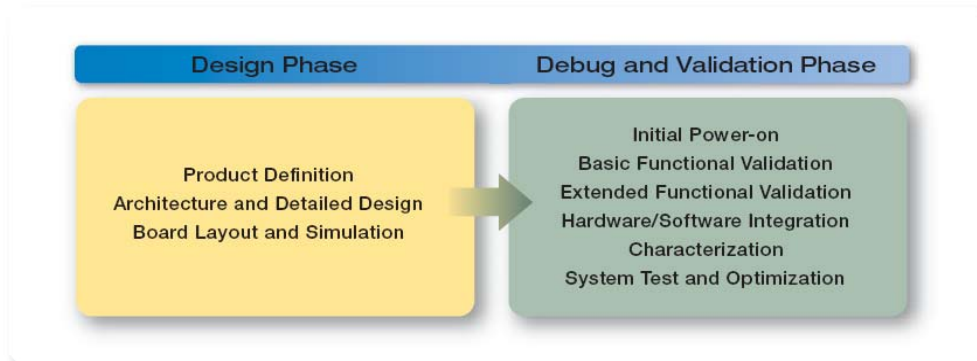
Eliminate Double Probing



- ▶ iCapture™ Multiplexing: 2-in-1 Single Probe Solution
 - Reduce probe loading
 - ▶ Single-point digital and analog probing
 - ▶ 2 GHz analog bandwidth on all channels
 - Save time in probing
 - ▶ Analog probe outputs are always 'live'
 - ▶ Any 4 of the 136 LA input channels are multiplexed to 4 analog output BNCs

One of the more recent advances in probing technology is a probe that can carry both digital information to the logic analyzer, and also deliver these same signals to an oscilloscope as analog signals. One logic analyzer probe can then handle most aspects of our design. Any pin of the probe can be used for both digital acquisition and analog acquisition. An analog mux in the logic analyzer is used to route the signal to an external oscilloscope as shown in the block diagram. In this way, any four of the signals being probed by the logic analyzer can be viewed on an oscilloscope. By this means, it is possible to determine almost instantly whether a digital error is associated with an analog fault.

Review



- ▶ “Begin With the End in Mind”
 - Identify potential problem areas
- ▶ Develop a test plan that includes a probing strategy
- ▶ Minimize functional and signal integrity problems with good design practices

In Part II, we are going to move on to the next phase, “The Debug and Validation Phase”. This is where the preparation done during the design phase will pay off with a streamlined validation and debug flow. We will also examine in more detail how a single probe that can be used for both digital and analog acquisitions can simplify many common troubleshooting tasks.

So let’s review the key things we have discussed. First, we need to begin with the end in mind. This simply means to start our debug process in the design phase. We do this by identifying potential problem areas. Once identified and noted as risks, we can develop a test and validation plan – a key part of which is a good probing strategy. This section needs to document where and how we are going to probe the system.

We also examined the types of problems that experience shows that we will encounter. First are functional problems. Good design practices and simulation can help reduce the number of problems that we see in the lab. The second type of problems are signal integrity related. These type of problems can occur in any system – even those with slower clock rates. It all depends on the edge rate of the signals. Signal integrity problems are caused by a wide-range of things and the health of signals is influenced by many different factors – all of which deserve attention during the physical design of our system.

With a well-thought through test plan, we will have minimized the risk and put ourselves in the best position to reduce our total debug and validation time.

Thank You!

Additional information on troubleshooting tips
is available at:

www.tektronix.com

So, thanks for listening today and I invite you to listen to Part II to learn more about saving time when in the lab. For more troubleshooting tips, please visit our website at www.tektronix.com.