Tektronix
Enabling Innovation

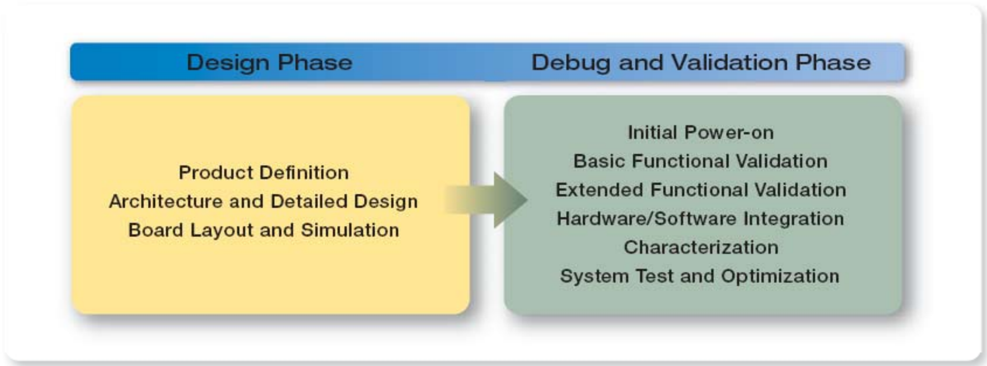
Solutions to Embedded System Design Challenges
Part II

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

Hi, my name is Mike Juliana. Welcome to today’s eLearning. This is Part II of a discussion on how to improve productivity in validating and debugging our embedded systems. This session will continue the discussion we started in Part I and will focus on the things that we can do in the lab to increase our productivity.

Tektronix
Enabling Innovation

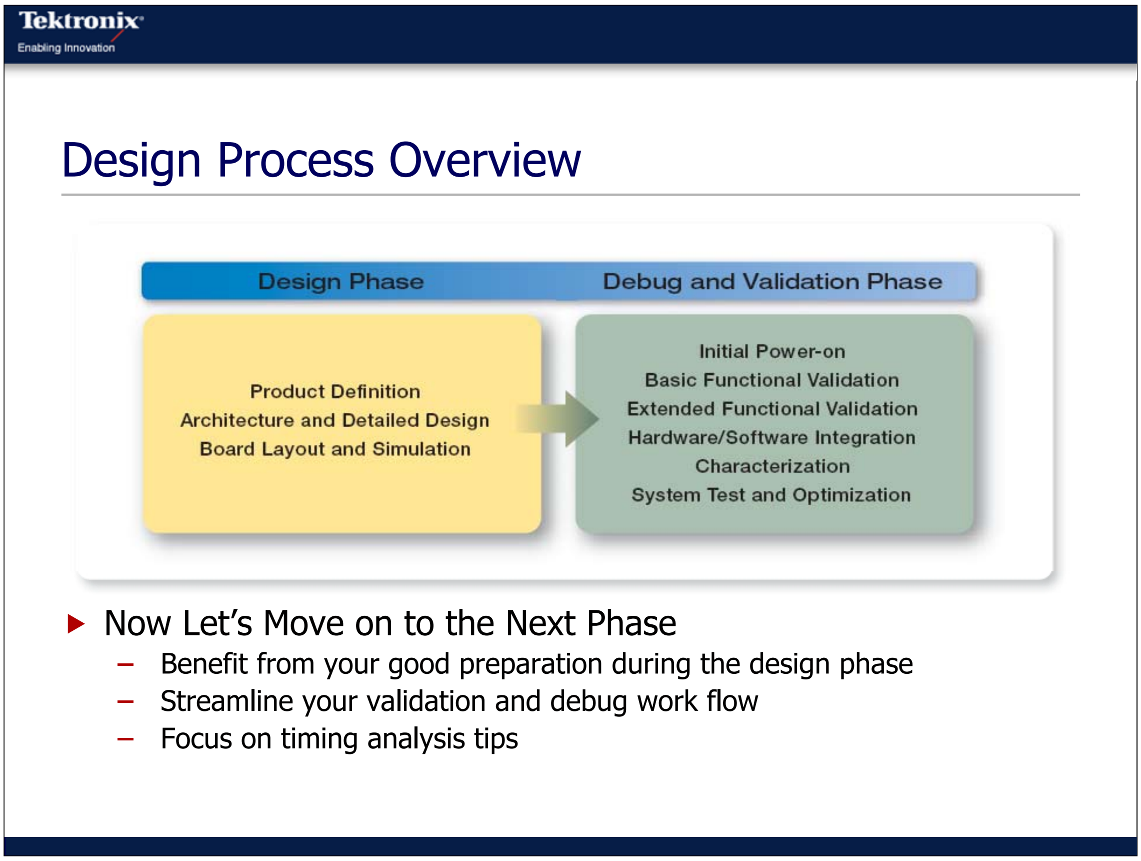
Review of Part I


Design Phase
Debug and Validation Phase
Product Definition
Architecture and Detailed Design
Board Layout and Simulation
Initial Power-on
Basic Functional Validation
Extended Functional Validation
Hardware/Software Integration
Characterization
System Test and Optimization

- "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

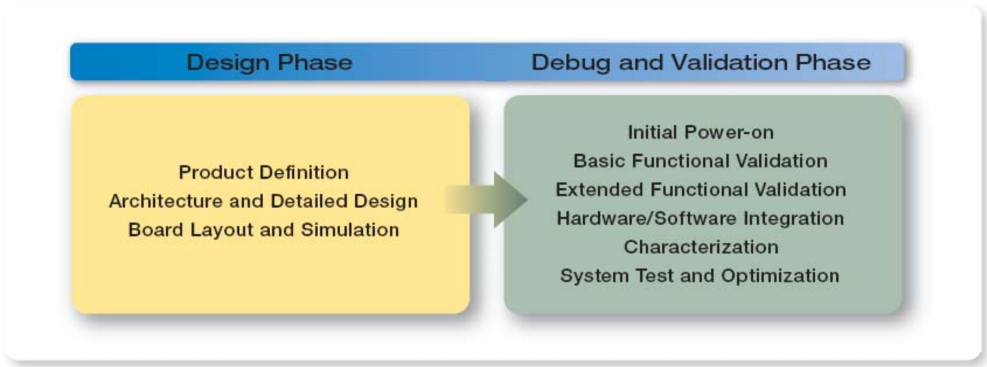
Before we get going on today's material, let's quickly review some of the key things we talked about in Part I.

First, we talked about this idea of "Beginning with the End in Mind". This is simply the concept that our debugging actually needs to start in the design phase In the design phase we need to identify the potential problem areas and document them and come up with a test and validation plan. We discussed that a key part of this plan is a good probing strategy. It's this probing strategy that identifies the signals that we need to have access to and how we will have access to those signals. This is probably the single-most important thing that we can do in the design phase to help ourselves out once we get into the lab. This will save us time, so this is a key step we need to go through.



Tektronix
Enabling Innovation

Design Process Overview


Design Phase
Debug and Validation Phase
Product Definition
Architecture and Detailed Design
Board Layout and Simulation
Initial Power-on
Basic Functional Validation
Extended Functional Validation
Hardware/Software Integration
Characterization
System Test and Optimization

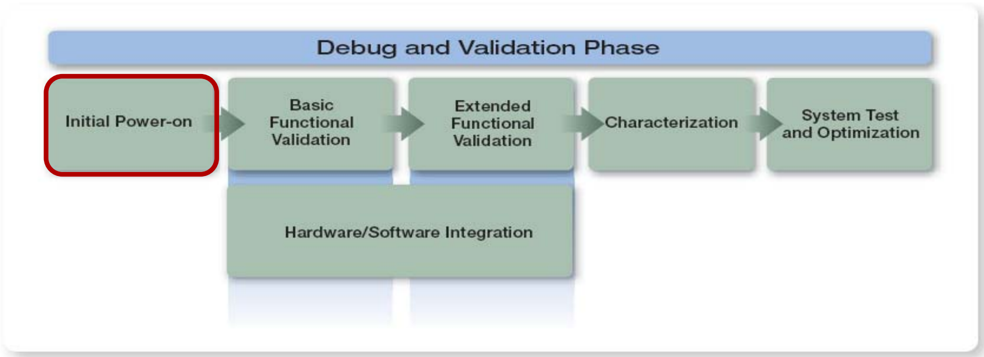
- Now Let’s Move on to the Next Phase
  - Benefit from your good preparation during the design phase
  - Streamline your validation and debug work flow
  - Focus on timing analysis tips

With that quick review, let’s move on to the second part of the design process – the debug and validation phase. This is where, hopefully the preparation that we have done in the design phase will pay off.

To frame our discussion today, we are going to break the debug and validation phase into six distinct activities: initial power-on, basic functional validation, extended functional validation, hardware/software integration, characterization of our system, and system test and optimization. The reality is that often times we overlap these phases, but in general the activities that we do in the debug and validation phase can land in one of these buckets.

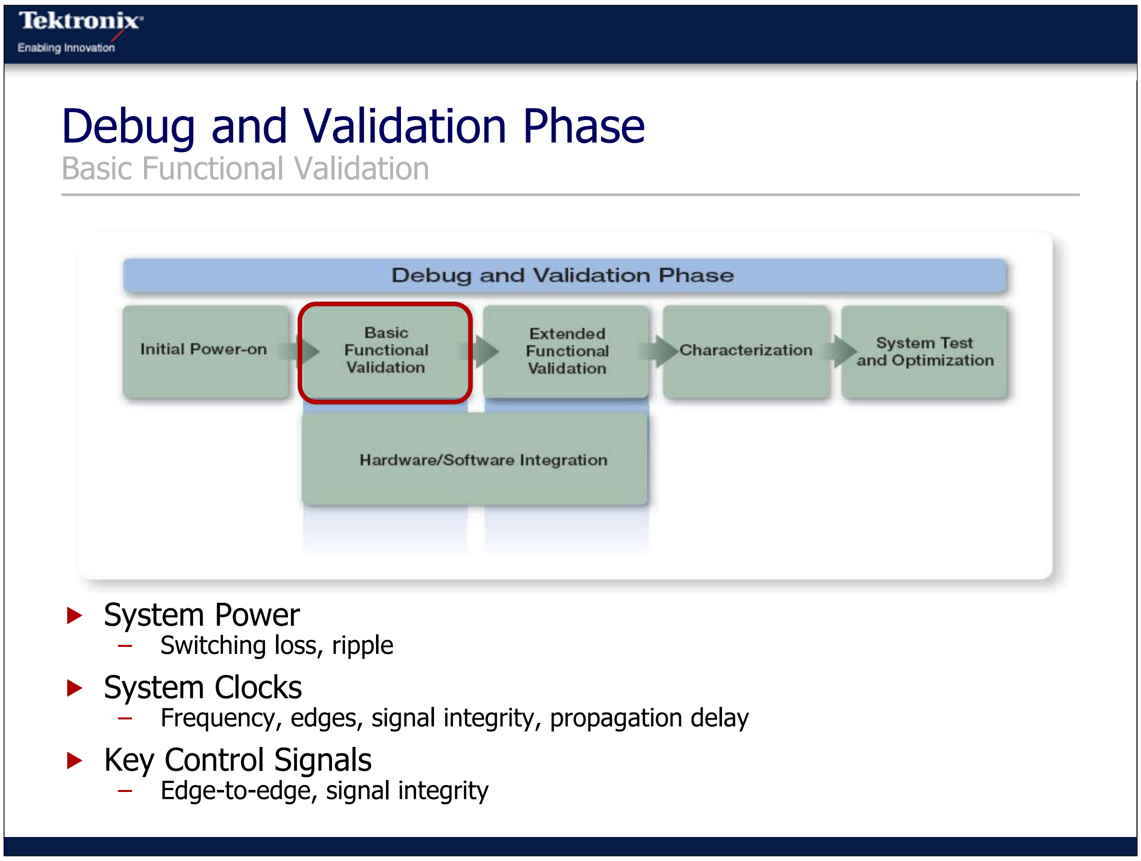
For today’s discussion, we are going to focus on functional validation, both basic and extended validation.

Debug and Validation Phase
Initial Power-On


Debug and Validation Phase
Initial Power-on
Basic Functional Validation
Extended Functional Validation
Characterization
System Test and Optimization
Hardware/Software Integration

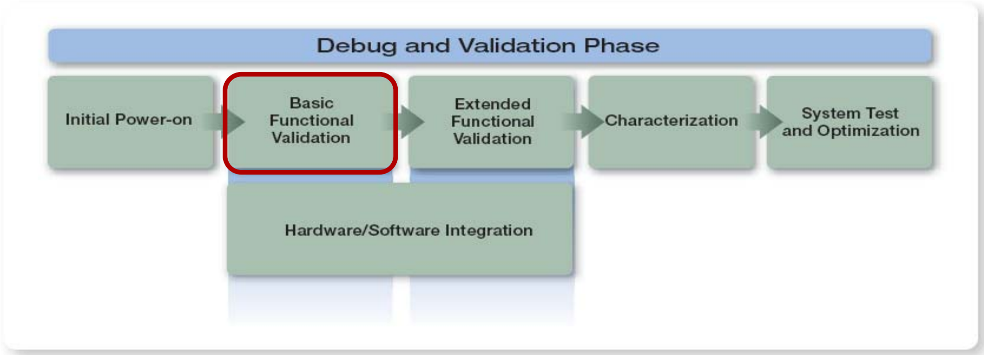
- Initial Power-On
  - Check for shorts
  - Apply power
  - Check for smoke

The first step is actually the initial power-on. This is the stage where the board come back from the manufacturer for the first time, we get it in the lab, we hook up the power supplies and with a little bit of fear, we apply the power to the circuit board for the first time. If all goes well, we don't see or smell any smoke. Usually this is the case, but occasionally smoke will happen and that's never a good thing. But most of the time, nothing happens here and we can move on to the next step.



Tektronix
Enabling Innovation

Debug and Validation Phase
Basic Functional Validation


Debug and Validation Phase
Initial Power-on
Basic Functional Validation
Extended Functional Validation
Characterization
System Test and Optimization
Hardware/Software Integration

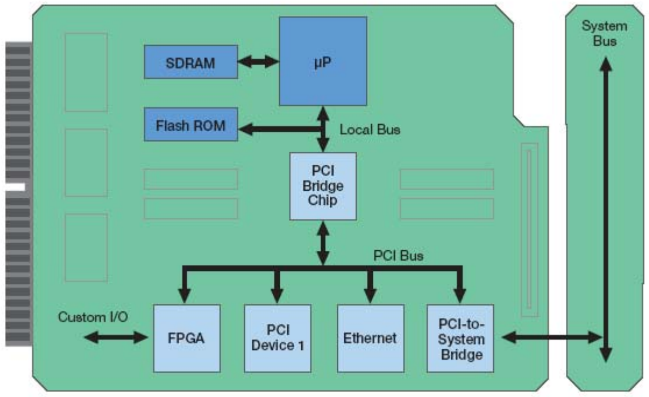
- System Power
  - Switching loss, ripple
- System Clocks
  - Frequency, edges, signal integrity, propagation delay
- Key Control Signals
  - Edge-to-edge, signal integrity

The next step is the basic functional validation step. This stage, we are simply trying to validate the very core of our design. I use the analog that we are just trying to get the heart and brains of our system working. We are not concerned about the legs, we are not concerned about the arms, we are just concerned about the key things that are required so we can get the arms and legs working later on. So we want to focus on the critical few things that are necessary at this point in time, not our entire system.

Typically at this point in time, we are focusing on a selected group of components and a selected group of key signals. For example, we'll want to look at the power supplies to make sure they are at the proper levels, they are sequencing in the proper sequence, maybe the ripple is within tolerance, those sorts of things. We'll also want to take a quick look at the clocks to make sure all the clocks are running at the right frequency and that they are all present in our system. Then from there we will move on to the key buses to look at the relationship with the key signals of these buses.

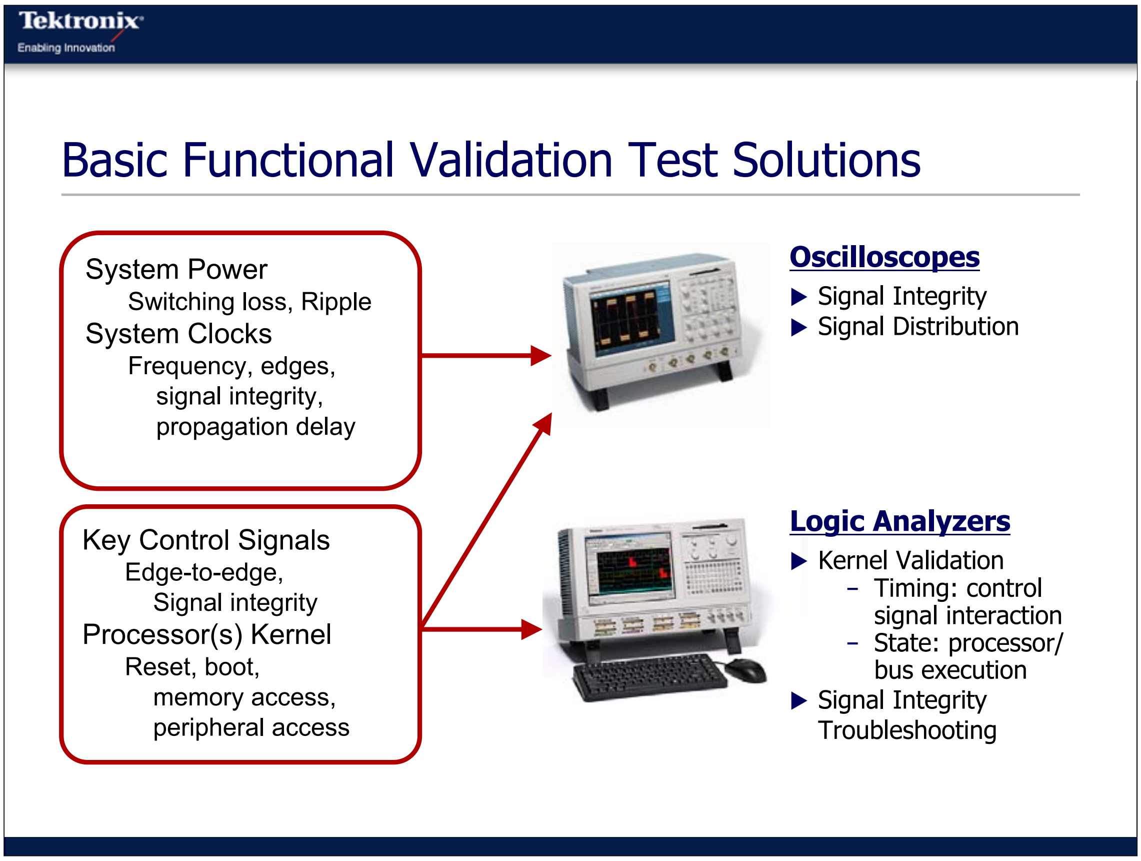
At this point, the primary goal is not to do an exhaustive analysis of all of the signals and the components but just to verify the relationship and that things look okay so we can move on to the next step. The goal here is to do this a quickly as possible.

Basic Functional Validation



So just quick review. Here is the system that we used in Part I to talk about probing strategies. It is a simple microprocessor-based system that has a local bus. It also has am SDRAM, it could be a DDR interface hooked to the microprocessor. Off the local bus we have the Flash ROM that contains our boot code, and off that local bus we also have a PCI Bridge Chip that bridges to a internal PCI bus. On the internal PCI bus we have several peripherals, an ethernet chip. Also we have an FPGA where we have done a lot of intellectual property work. This is the piece of the design that distinguishes our product from our competitors' products. And then this internal PCI bus is tied to our system bus through a PCI-to-System bridge.

During the part of the basic validation phase, we are mainly looking at the highlighted components (in blue), the microprocessor, the Flash ROM and the SDRAM and the buses that tie these things together. We are not concerned about the rest of the system.



Tektronix
Enabling Innovation

Basic Functional Validation Test Solutions

System Power
- Switching loss, Ripple

System Clocks
- Frequency, edges, signal integrity, propagation delay

Key Control Signals
- Edge-to-edge, Signal integrity

Processor(s) Kernel
- Reset, boot, memory access, peripheral access

Oscilloscopes
- Signal Integrity
- Signal Distribution

Logic Analyzers
- Kernel Validation
  - Timing: control signal interaction
  - State: processor/ bus execution
- Signal Integrity Troubleshooting

As I’ve mentioned, the goal at this stage is to take a quick look at the key signals and their relationships to each other. To do that, we can use one of two common tools – an oscilloscope and a logic analyzer.

The scope is obviously a great tool for looking at the analog things in our design, looking at the power supplies, maybe digging deeper to look at switching loss, ripple and things like that. It can also be used to look a timing relationships and clock distribution of a few signals.

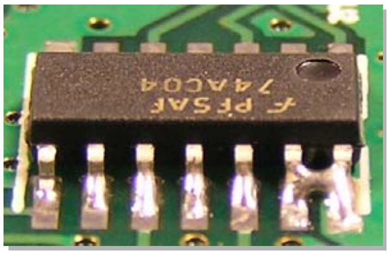
The logic analyzer is also good at looking at the timing relationship of signals and as you will see, also finding signal integrity problems. The big difference between a scope and a logic analyzer is the number of signals that can be observed. Because a logic analyzer can observe a large number of signals, it is ideal for looking at the large buses and looking at the boot process of our system.

For most engineers, the tool of choice is the scope. It is a tool we are very comfortable with, but sometimes we try to stretch its usage a little too far which can actually be counter-productive. The key here is to recognize the strength of each instrument and use each accordingly.

Tektronix
Enabling Innovation

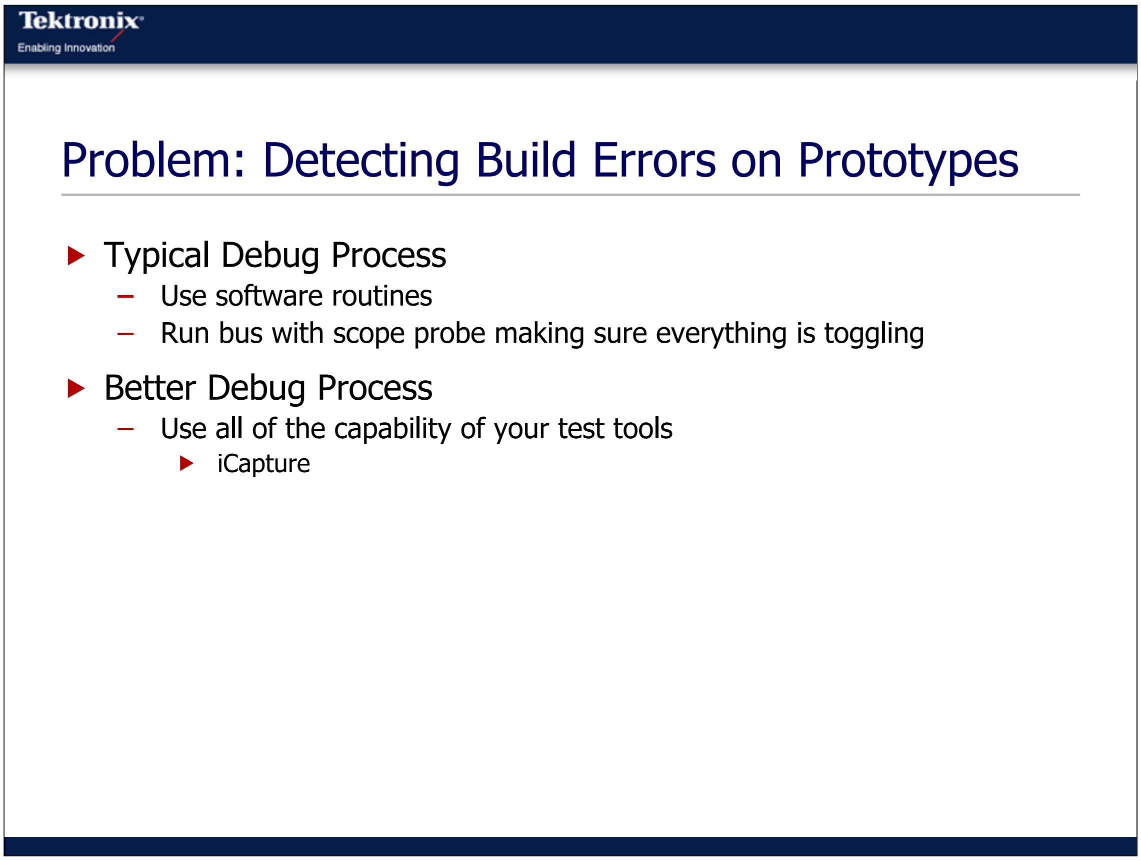
Key Challenges

- Design Errors
  - Logic errors, inverted logic, bus contention, addressing errors, misunderstanding of commercial parts/circuits
- Build Errors
  - Shorts, opens, wrong parts, etc.
- Accurate Signal Acquisition



Obviously during this step, we are looking for any design errors that would keep the heart and brain from functioning properly. During the design process we wanted to eliminate these through design reviews, and simulation and things link that. But sometimes, despite our best intentions, design errors do creep into our systems. It could be something we overlooked in the design, it could be a wrong assumption about how a commercial part behaves, maybe a misunderstanding between two members of the team. Whatever the reason is, it is our job to find the problem and get the heart and brain working as fast as possible.

This is often compounded by the fact that, not only is this the first time we have seen the design, but it is also the first time that manufacturing has built this design. So what appears to be a design problem could actually be a build problem. For example, The manufacturer could have placed wrong components on our board, maybe there is a wrong resistor value or capacitor, maybe even there's a resistor where a capacitor should have gone. Sometimes there's a solder bridge or unsoldered pins. So finding and detecting these problems can sometimes keep us from finding the real design problems in our board. So we need to be able to quickly get through and find any build problems in our board.



Problem: Detecting Build Errors on Prototypes

- Typical Debug Process
  - Use software routines
  - Run bus with scope probe making sure everything is toggling
- Better Debug Process
  - Use all of the capability of your test tools
    - iCapture

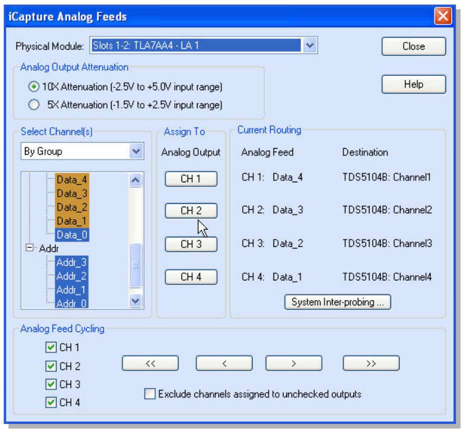
Since this is such a common problem – having to find design errors on our prototypes – let's take a look at a typical example here, the process of finding either an open pin or a solder bridge on a bus. Typically the way that we would find a problem like this is to use an oscilloscope and a software routine that will loop on a particular address or address range and toggling each bit on the data bus or the address bus. We would then use a scope to run the bus, probing each pin to make sure that it is toggling. But when you think about this activity, it is actually very time-consuming. To do the first bit, we would need to find Bit-0 of the data bus on the schematics, then we would locate that U number on our circuit board, say Bit-0 was on U29 pin 14. Then we would need to find it on the circuit board, count to pin 14 and then probe pin 14. And then we would have to repeat that another 31 times to completely verify a 32-bit bus. This is not only time-consuming but it can be frustrating. Luckily there is a way that can streamline the process if we use the capability in all of our test equipment.

Tektronix
Enabling Innovation

Cycling Channels

- Use iCapture™ Multiplexing to Browse Your Bus Channels

Logic Analyzer



iCapture Analog Feeds

Physical Module: Slots 1-2: TLA7AA4 - LA 1

Close

Help

Analog Output Attenuation

[✓ 10X Attenuation (-2.5V to +5.0V input range)]
5X Attenuation (-1.5V to +2.5V input range)

Select Channel(s): By Group

Data_4
Data_3
Data_2
Data_1
Data_0
Addr
Addr_3
Addr_2
Addr_1
Addr_0

Assign To: Analog Output

CH 1
CH 2
CH 3
CH 4

Current Routing

Analog Feed	Destination
CH 1: Data_4	TDS5104B: Channel1
CH 2: Data_3	TDS5104B: Channel2
CH 3: Data_2	TDS5104B: Channel3
CH 4: Data_1	TDS5104B: Channel4

System Inter-probing ...

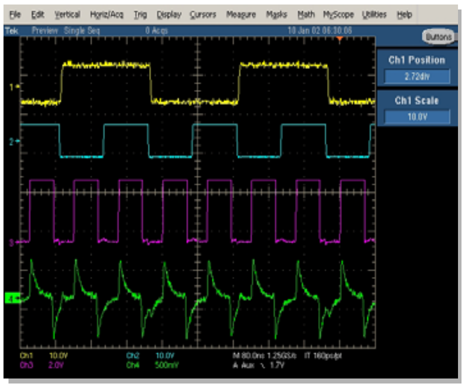
Analog Feed Cycling

[✓ CH 1]
[✓ CH 2]
[✓ CH 3]
[✓ CH 4]

<< < > >>

Exclude channels assigned to unchecked outputs

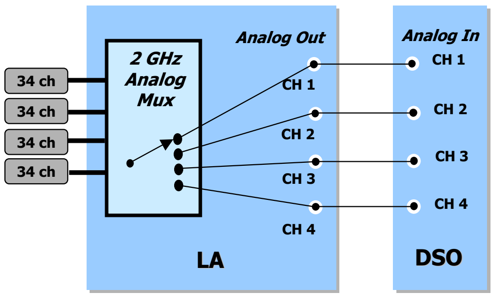
Digital Storage Oscilloscope


File Edit Vertical Horiz/Acq Trig Display Cursors Measure Masks Math MyScope Utilities Help
Tek Preview Single Seq 0 Acqs 10 Jan 02 06:30:06
Buttons
Ch1 Position
2.72div
Ch1 Scale
10.0V
Ch1 10.0V Ch2 10.0V
Ch3 2.0V Ch4 500mV
M 80.0ns 1.25GS/s IT 160ps/pt
A Aux 1.7V

In this case, we want to take advantage of the probing strategy that we defined in the design phase. By doing this, by hooking up a logic analyzer, we can take advantage of a feature that is available on some logic analyzers, a feature called iCapture™. With this feature, through a single probe we view both analog and digital data. This makes it real convenient to do certain things. So because we have added these convenient logic analyzer test points to our systems, all we need to do is hook up our logic analyzer probe to our system under test and then, not only can be see the digital data on the logic analyzer like we traditionally do in the form of 1s and 0s, but we can also see selected analog signals on an oscilloscope as well, all through a single logic analyzer probe. We'll see how this works on the next slide.

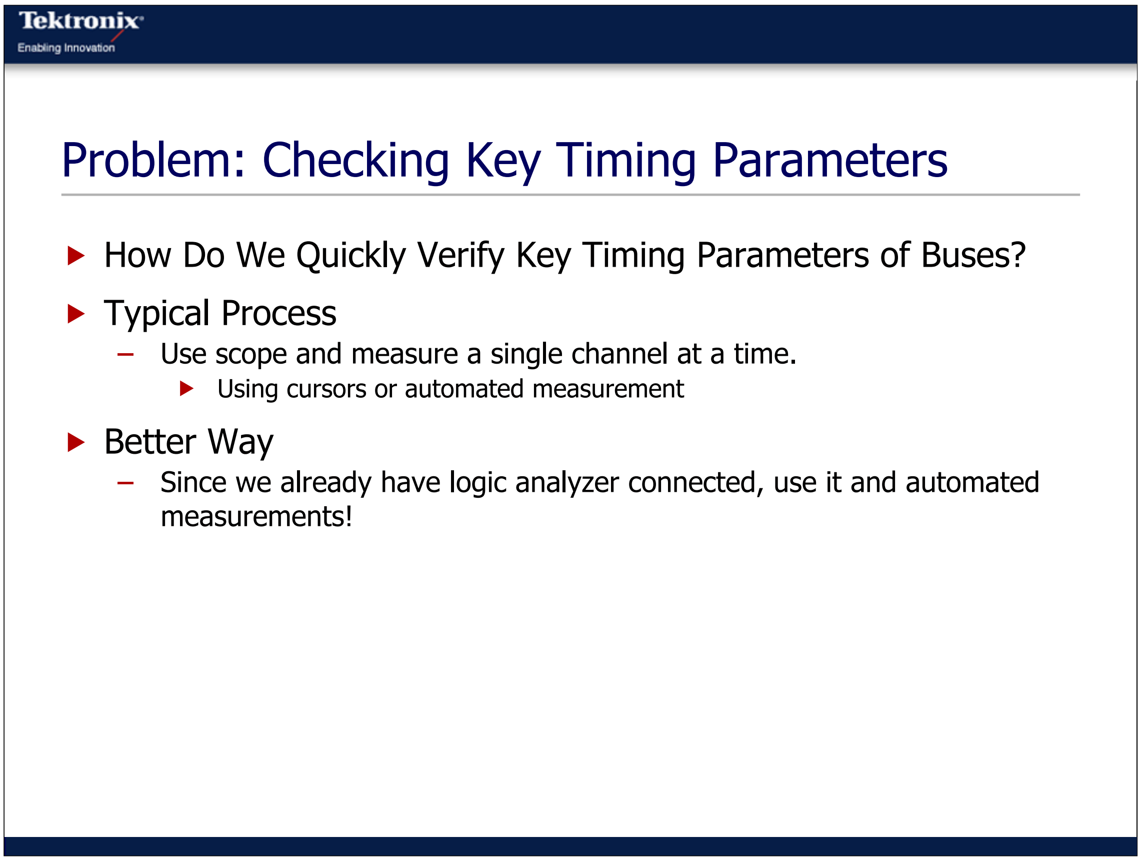
For now though the desire is to quickly scan 32 bits of the data bus to make sure that each bit is toggling. With iCapture we can do this without having to move the single probe. We simply connect the logic analyzer probe to the system that we are testing and then we use the logic analyzer user interface to assign the data bus to one of the analog feeds on the logic analyzer. Again, we will see what this looks like on the next slide. Changing which bit of the data bus we are looking at on the scope is a simple matter of clicking an arrow on the logic analyzer dialog box. Each time we click the arrow, we route a different data bus bit to the scope where was can verify that it is toggling. Now instead of having to probe 32 different signals manually and move the probe in between each one, that could take a half an hour, we can do this in a minute by simply clicking a mouse button 31 or 32 times.

iCapture™



- 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

Here we see a simple block diagram of the iCapture system. This is a single probe that allows us to look at analog and digital data. We use this single probe to connect up to our system under test, and as you can see, one of the things that happens when the signal comes into the logic analyzer, it goes through an analog mux; this is a 2 GHz analog mux. So I can take any of my logic analyzer signals that I am probing and route them out one of four BNCs on the logic analyzer module. I then can route these signals from the BNC on the logic analyzer module over to my oscilloscope using a simple BNC cable. So with one probe, I can see both the digital data represented as 1s and 0s, and I can also see the real analog data behind that signal. This is key in helping to increase our productivity as we want to look at different views of our data.



Tektronix
Enabling Innovation

Problem: Checking Key Timing Parameters

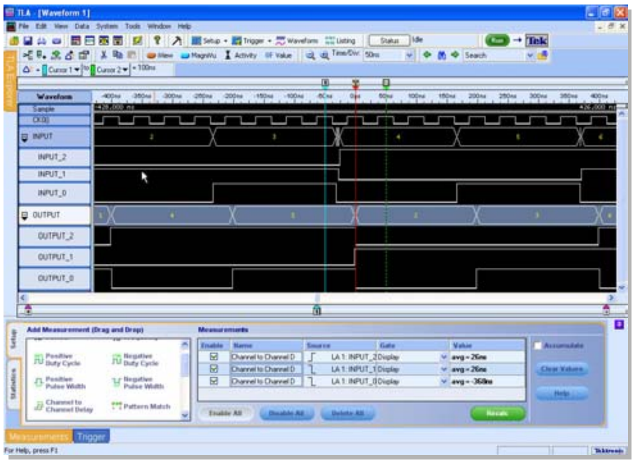
- How Do We Quickly Verify Key Timing Parameters of Buses?
- Typical Process
  - Use scope and measure a single channel at a time.
    - Using cursors or automated measurement
- Better Way
  - Since we already have logic analyzer connected, use it and automated measurements!

Another common problem in this stage of the basic functional validation stage is, how do we quickly verify some of the key timing parameters of our system. Typically, again we would use our scope and we would measure one signal at a time using either cursors or automated measurements. But again, since we have developed a probing strategy that allows us to conveniently hook up our logic analyzer to these buses, we can use it instead and use the automated measurement capability of the logic analyzer.

Tektronix
Enabling Innovation

Verifying Key Timing Parameters

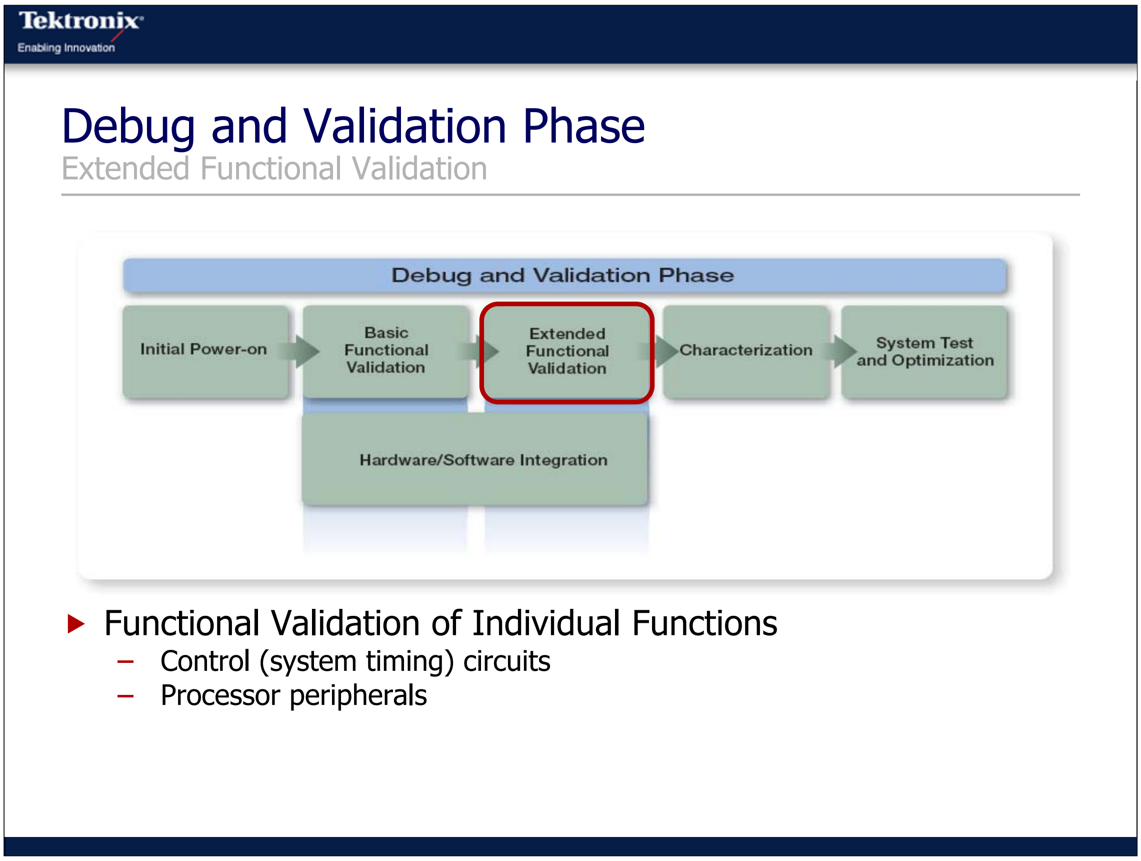
- Clock Check-Up
  - Clock frequency
  - Duty cycle
  - Pulse width
- Propagation Delay
  - Relationship of key signals



Measurements

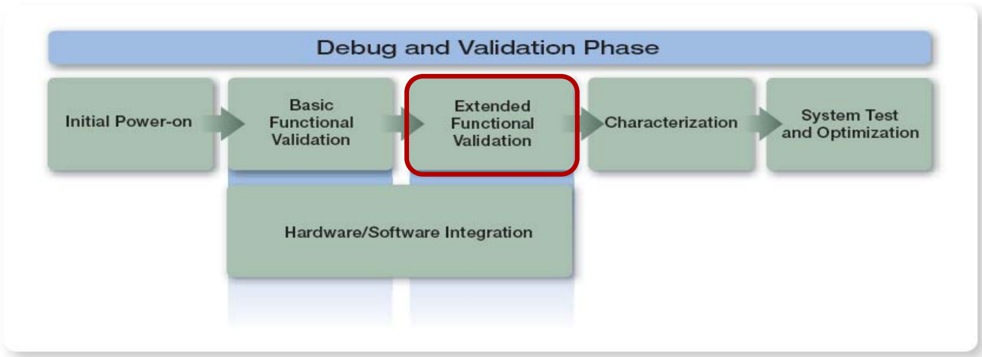
Enable	Name	Source	Gate	Value
☑	Channel to Channel D	LA 1: INPUT_2	Display	avg = 26ns
☑	Channel to Channel D	LA 1: INPUT_1	Display	avg = 26ns
☑	Channel to Channel D	LA 1: INPUT_0	Display	avg = -368ns

So for instance, if we were probing a key, critical control bus on our system, we could acquire the data using the logic analyzer and then using drag-and-drop measurements within the logic analyzer, quickly verify our clock signals. We could do measurements for clock frequency, the duty cycle, pulse width and a variety of measurements. We could also use these drag-and-drop measurements to quickly define and see what the relationship the propagation delay between some key signals are. So again, by using a single probe, the ability to probe many signals at once with our logic analyzer, not only can we see the analog as well, but we can quickly make some key measurements using automated and drag-and-drop measurements to help us understand the relationship of these signals.


Tektronix
Enabling Innovation

Debug and Validation Phase

Extended Functional Validation

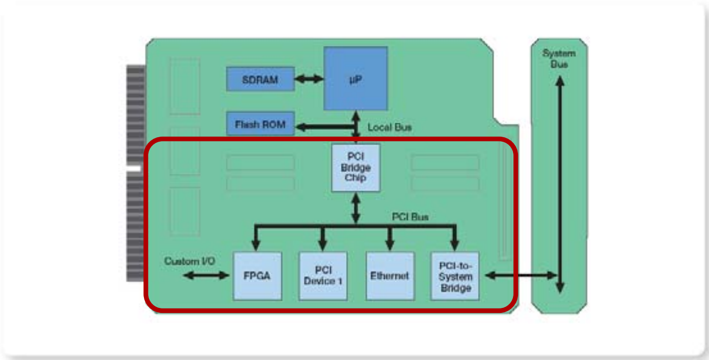

Debug and Validation Phase
Initial Power-on
Basic Functional Validation
Extended Functional Validation
Characterization
System Test and Optimization
Hardware/Software Integration

- Functional Validation of Individual Functions
  - Control (system timing) circuits
  - Processor peripherals

Once the heart and brain, the core of our design, is working then it’s time to move on to the other parts of our design. At this point in time we want to focus on one functional block at a time. So, for instance, we would move to one component on that internal PCI bus, maybe the ethernet chip, and we would focus getting that working properly. Once that was working properly, we would turn our attention to our FPGA and make sure that that was working properly. So we are going to move from the core of our design, the brain, and now we are going to focus on the extended pieces of it, but we are going to do it one piece at a time.

Tektronix
Enabling Innovation

Key Challenges In Bringing Up The Sub-Systems



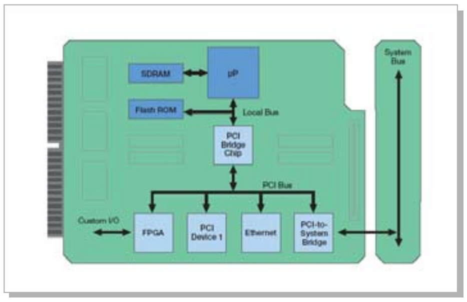
- Isolating Circuits and Sub-Systems
  - Must be planned in the design
- Quickly Validating Large Number of Signals
- Signal Integrity Problems

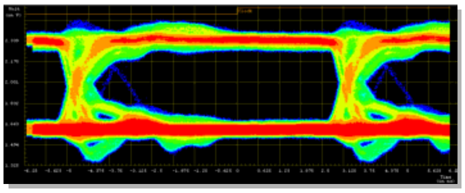
So, to be able to do it one piece at a time means that we need to isolate these circuits and subsystems as much as possible. Obviously that's not always possible, but it is something that we need to plan for as much as possible in the design part of this process. For instance, all the components on the local PCI bus will be impacted by errors on the PCI bus, so if there is a short or something like that PCI bus, it would impact all the peripherals on the PCI bus. But the trick and the goal is to isolate these components as much as possible so they are not interacting. Once we can do that then the next problem is we need to quickly validate a large number of signals. So we have gone past a small part of our design, and the number of signals quickly mushrooms and we need to be able to validate the relationship of those signals in the signal integrity aspects of all those signals. We need to be able to find ways of doing that quickly.

Tektronix
Enabling Innovation

Problem: How to Quickly Verify Signal Integrity and S&H Time on buses?

- Typical Process
  - Use scope and measure a single channel at a time
- Better Way
  - Use iCapture in combination with the eye-diagram capability of scopes – iVerify



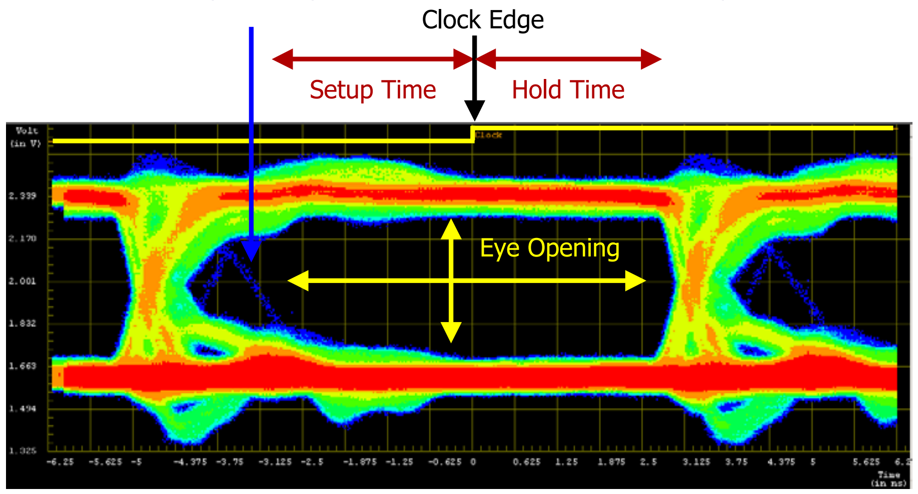


So let’s focus on that problem of how we can quickly verify signal integrity and setup/hold time on some of our buses. Say for instance we have a 32-bit data bus, we want to take a quick look at the signal integrity and how the setup/hold time looks on all 32 bits of that bus and we want to do that rather quickly. Again the typical process would be take our oscilloscope, take one probe and hook it to a clock signal or data strobe, something that we want to measure setup/hold time relative to, and then we take another probe and we would probe one data bit at a time, and we would build up a representation maybe using persistence mode to see what the setup/hold time relative to that clock signal or data strobe is. But there is a better way and a faster way of doing this. Again, by taking advantage of all the capability in our test tools, in this case we are going to take advantage of the iCapture™ capability in the logic analyzer that we talked about a few minutes ago, and we are going to combine that with the eye diagram capability of scopes, and we are going to end up with a feature set that we call iVerify™ and this will allow us to verify the signal integrity and setup/hold time on large buses rather quickly.

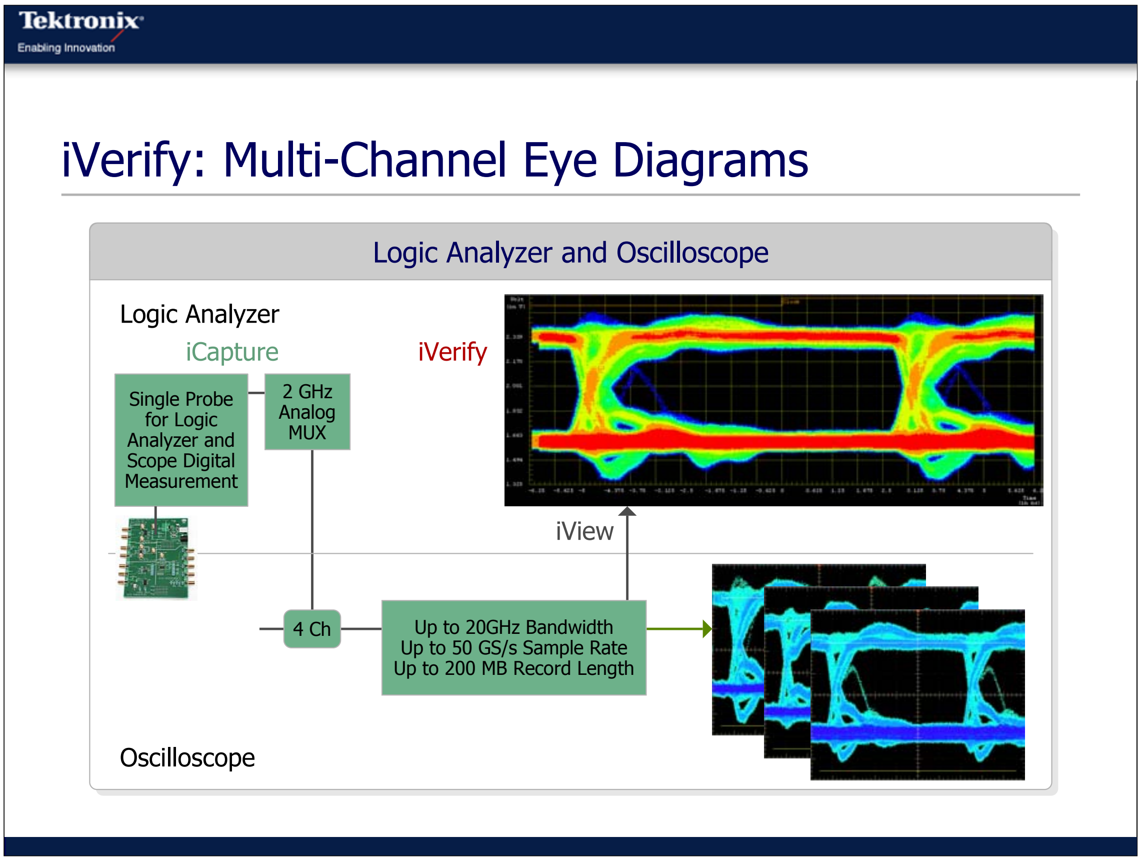
Tektronix
Enabling Innovation

Eye Diagrams Measure Data Valid Window

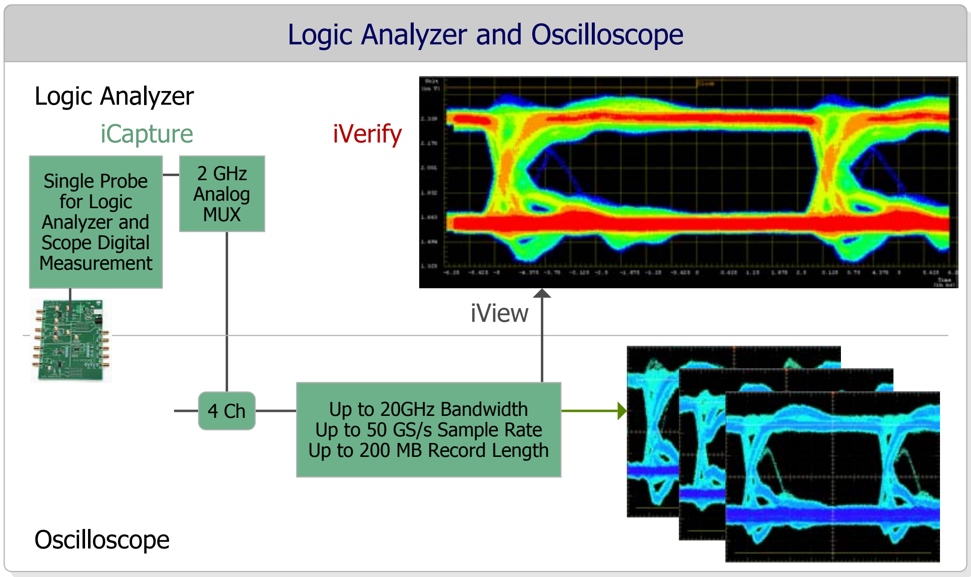
Faster Clocks = Smaller Eye Opening
Signal Integrity Problems = Smaller Eye Opening



So in this case, again what we are trying to do is we are trying to validate setup/hold time and the signal integrity of our buses around the points in time where it really matters. In a synchronous system, for instance, like a PCI bus, the only time the data really matters is on the rising edge of the clock. So we want to validate the control bus with the address bus relative to that clock edge. Again, one way of doing that is to use the eye diagram capability of the oscilloscope and use channel 4, in most cases, as the clock input and then take the acquisitions and build up an eye diagram of the signals around that clock edge. By doing that we can get a quick view of the health of the bus at that important point in time on the rising edge of the clock. We can build up this eye diagram representation that we see here and this will show us, relative to the clock edge, the setup time, the hold time, and any potential signal integrity problems that we have that are closing that eye or getting close to validating the required setup/hold time.


Tektronix
Enabling Innovation

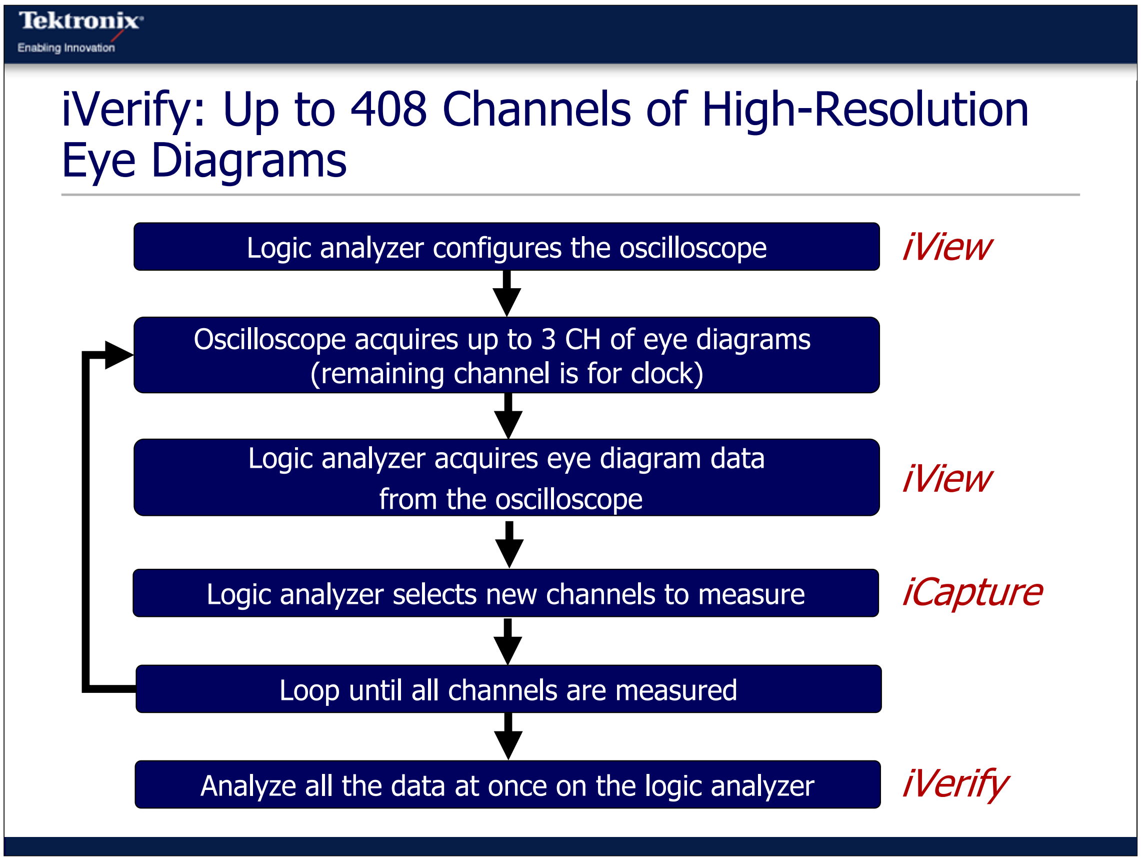
iVerify: Multi-Channel Eye Diagrams


Logic Analyzer and Oscilloscope
Logic Analyzer
iCapture
iVerify
Single Probe for Logic Analyzer and Scope Digital Measurement
2 GHz Analog MUX
iView
4 Ch
Up to 20GHz Bandwidth
Up to 50 GS/s Sample Rate
Up to 200 MB Record Length
Oscilloscope

Obviously, with an oscilloscope that has four channels, I can build up a three channel eye diagram rather quickly. The problem then, is how do I verify the other 29 bits of my data bus. I would have to move those three channels of my scope probes to a different three data bits and then build up the eye diagram again and I would lose all data from the previous three bits. So again, we are going to combine this capability of the oscilloscope to build eye diagrams and the iCapture capability of the logic analyzer and we are going to build something called iVerify which is multi-channel eye diagrams.

In this case, we have the single probe going to our embedded system, and again through the single probe we can get both digital and analog data. So the signal comes into the logic analyzer, it goes through that 2 GHz analog mux, and we route out four signals over to the oscilloscope; one is the clock that we are interested in, it could be a rising edge of a PCI bus, it could be a data strobe, any point in time that is important to your system. The other three channels are three of the data bits. That then goes to the oscilloscope and the oscilloscope will build an eye diagram quickly, and that data will be transferred back to the logic analyzer using iView. So this is a connection between the logic analyzer and the oscilloscope by which this analog data can be transferred back to the logic analyzer. You can see here, what happens then is the oscilloscope will build a series of eye diagrams which get transferred back to the logic analyzer which the combines them into one eye diagram so you can quickly look at the 32-bit data bus.

The nice thing here is the logic analyzer will quickly switch, through the analog mux, the signals you want to look at, so you don’t have to move the scope probe. You simply hook up the logic analyzer to the 32 bits of interest here and the logic analyzer does all the work for you and the oscilloscope does all the work for you to build up this multi-channel eye diagram.



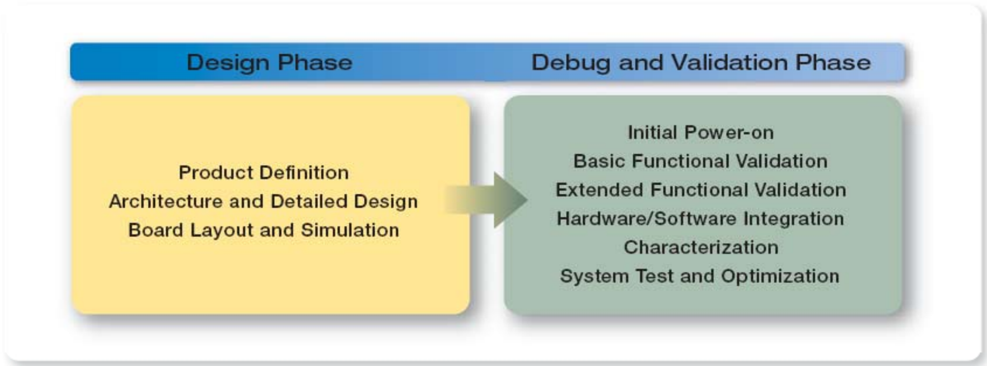
iVerify: Up to 408 Channels of High-Resolution Eye Diagrams

Logic analyzer configures the oscilloscope
iView
Oscilloscope acquires up to 3 CH of eye diagrams (remaining channel is for clock)
Logic analyzer acquires eye diagram data from the oscilloscope
iView
Logic analyzer selects new channels to measure
iCapture
Loop until all channels are measured
Analyze all the data at once on the logic analyzer
iVerify

Here is a high-level flowchart of how iVerify works with both the logic analyzer and the oscilloscope. Using iView, the thing the logic analyzer does is it sets up the oscilloscope to take the eye diagram. Into the logic analyzer you put a few parameters, what is your clock signal, the voltage range of the signals you are probing and the time range you want to look at. The logic analyzer will then set up the analog mux, so one signal is the clock you specified, that gets routed over to the oscilloscope and then it routes three other signals over to the oscilloscope that you want to put into this multi-channel diagram. The oscilloscope then will take an eye diagram, then it will capture that eye diagram back to the logic analyzer using iView, then the logic analyzer will switch three of the data bits to three new data bits you want to analyze leaving the clock signal alone and then it will loop on this until all the signals of interest have been analyzed and finally it is all combined into one eye diagram. So using this, you can build up to 408 channels of high resolution eye diagrams.

Tektronix
Enabling Innovation

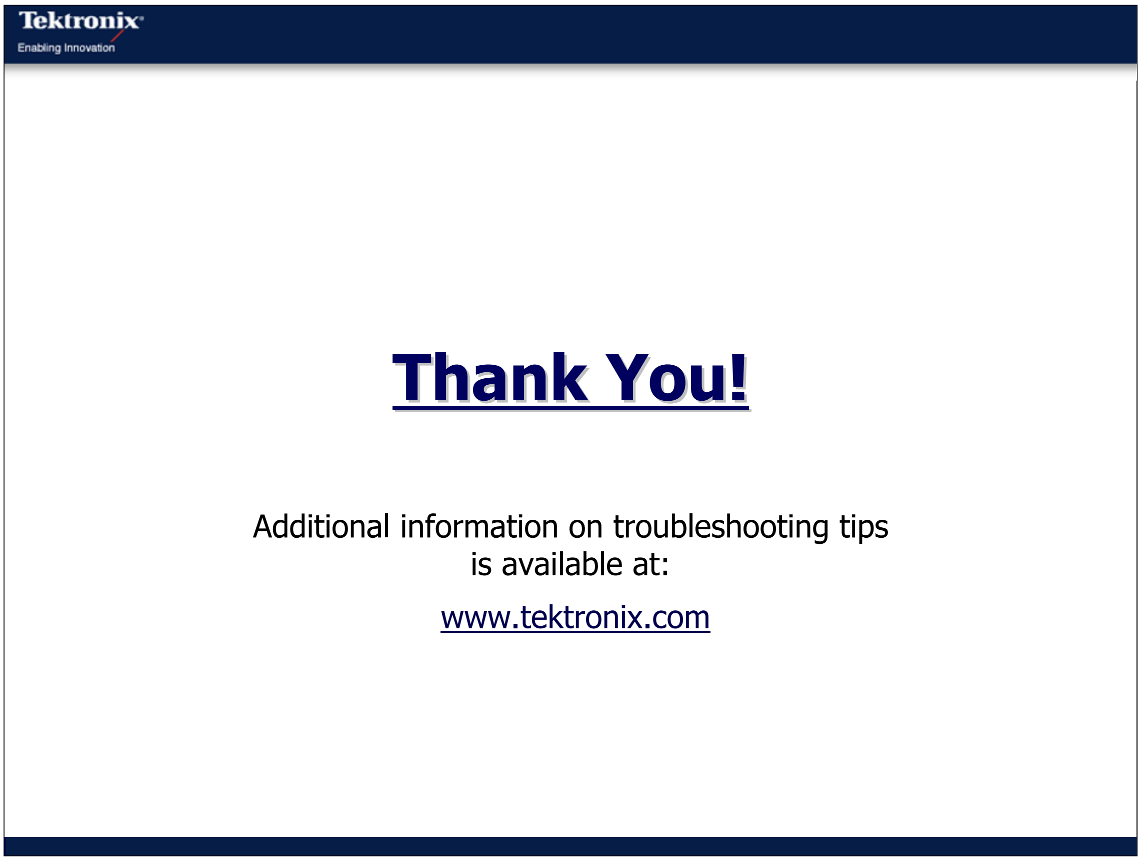
Review


Design Phase
Debug and Validation Phase
Product Definition
Architecture and Detailed Design
Board Layout and Simulation
Initial Power-on
Basic Functional Validation
Extended Functional Validation
Hardware/Software Integration
Characterization
System Test and Optimization

- Use the Right Tool in the Right Situation
- Understand the Capability of the Newest Tools
  - iCapture, automated LA measurements
- Take Advantage of the Added Power When Tools Are Combined
  - iView, iVerify

Let's review some of the key things we have talked about today. We have talked about a number of tips that will help streamline our debug and validation process. The first key thing is to use the right tool in the right situation. Sometimes we try to stretch the tool that we are using into situations that aren't appropriate; it would be better to switch to a different tool. The key here is to understand the capability of our tools and of the newest tools so we can take advantage of them; things like iCapture and automated logic analyzer measurements. Sometimes we can get some real power and added benefit if combine these tools into a single debug aid, something like iView or iVerify will combine the power of the logic analyzer and the oscilloscope.

So again, some of the key tips for streamlining our debug process is to begin in the design phase where we are doing our product definition and our architectural work, and the key thing here, the single most important thing to do during this design phase, is to develop a probing strategy. Engineers are great at solving problems once we can observe them. So the key is observability and the easiest way to do that is plan for it in advance. The worst debug strategy or probing strategy that we can have is one that we didn't think about. So again, the single most important thing to help us streamline and increase our productivity is to come up with that probing strategy because with that appropriate strategy we can simplify the tasks once we get in the lab and we can use all the power of our instruments.



Tektronix
Enabling Innovation

Thank You!

Additional information on troubleshooting tips
is available at:

www.tektronix.com

Thanks for listening today to the discussion on how to improve productivity in validating and debugging our embedded systems. For more troubleshooting tips, please visit our website at www.tektronix.com.