Agilent Technologies

InfiniBand System Validation and Debug

June 6, 2001

presented by:

David Fincher

Agenda

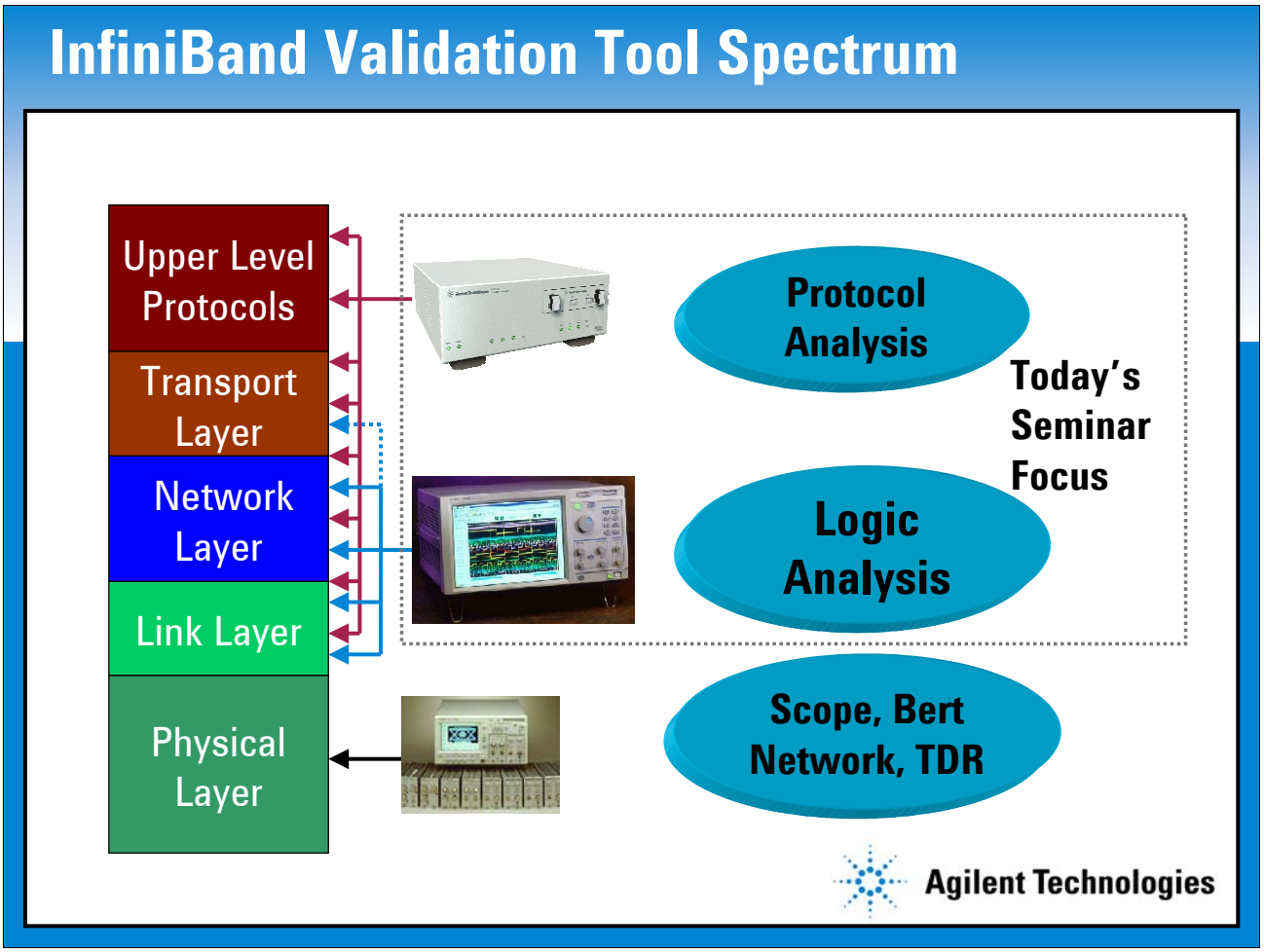
- Introduction
- System Level Debug Concepts
- Preparing for InfiniBand Validation
- InfiniBand Validation/Debug Examples
  - Link Powerup Negotiation
  - RDMA Transfer Failure
  - Switch Fabric Validation
  - Other Applications

Agilent Technologies

Agenda/Introductory Remarks:

This presentation covers key concepts underlying system level debug and how you prepare for validation of InfiniBand systems. Specific examples of typical debug sessions using logic analyzer and protocol analyzer tools to debug InfiniBand systems at the whole system level are covered.

There are three examples at different levels of abstraction. Starting with the most detailed showing link power-up negotiation and then moving up the protocol stack to RDMA operations.

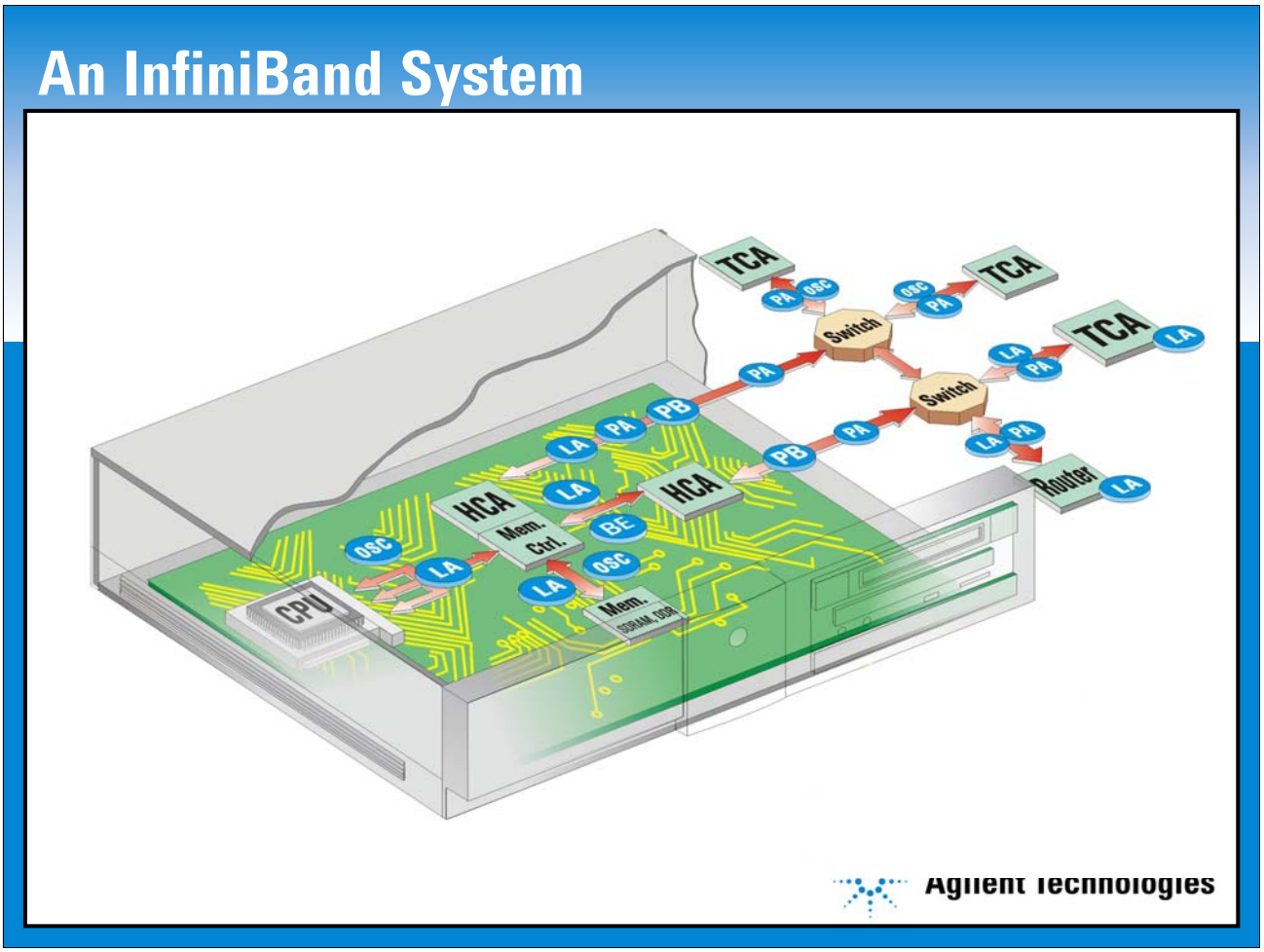


Validation Tool Context

When you look at an InfiniBand system, there's a huge dynamic range of technologies required to support InfiniBand. From microwave down to the lowest physical layers, all the way up through many layers of the protocol stack.

Physical layer debug and validation is with tools such as scopes, Bert Network, and TDR (Time Domain Reflectometry).

This paper focuses on functional validation of InfiniBand systems and that's where logic analyzers and protocol analyzers really add value.



An InfiniBand System

This is a block diagram of a typical InfiniBand system. The lid of the box is cut away, and what this shows here is the CPU, chip set, and InfiniBand HCA (host channel adapter). Outside the box is the InfiniBand traffic on the fabric.

In red are all the different busses or channels that connect up these components, and the blue dots represent the different kinds of test equipment that you might need to connect to those channels or busses, in order to do debug of a complete system.

OSC = Oscilloscope

PA = Protocol Analyzer

LA = Logic Analyzer

BE = Bit Error Rate Tester

System Level Debug: Key Concepts

- **Broad Visibility**
  - If you knew where the problem was, you wouldn't need to test!
  - Must see everything from pS to packets
- **Cross Correlation**
  - Cause and effect can span entire system
  - Between busses AND over time
- **Stimulus Diversity**
  - Mix of "real world" and regression test

Agilent Technologies

When debugging a system, there are really three main concepts that need to be kept in mind to do effective system-level validation, broad visibility, cross-correlation, and stimulus diversity.

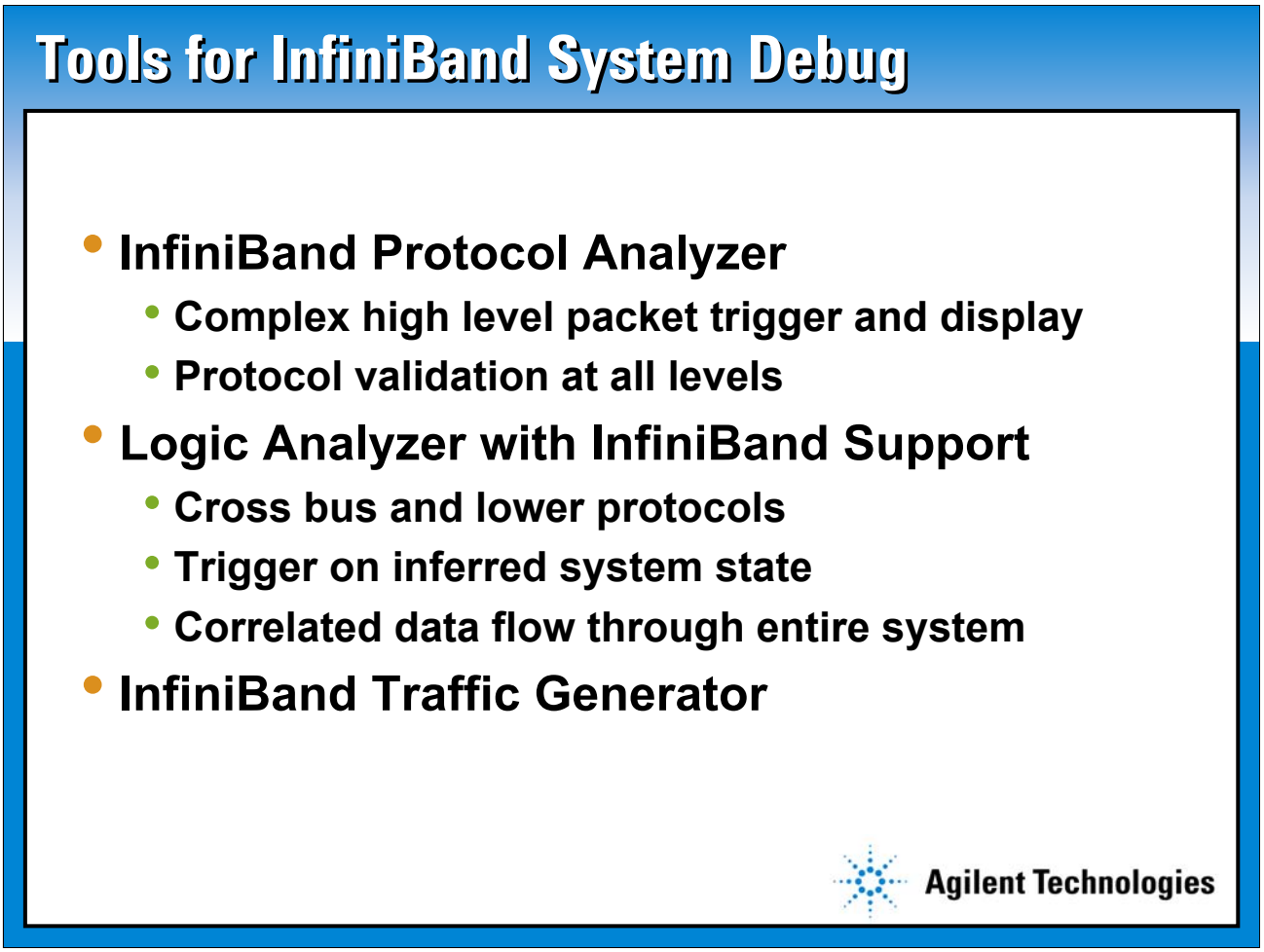
You need to have broad visibility into the system's behavior. If you know exactly where the problem is so that you only have one bus to look at, then from a system validation standpoint, you probably don't need to be testing the system; you are really just testing one component, or one bus, or one channel. When you are trying to track down system-level problems, you generally need to have broad visibility across a range of busess and links within the system. You also need to see behavior in the system at the picosecond domain, which is especially true for InfiniBand all the way up to the packet level.

You need visibility that is very broad and you need to be able to get insight into the behavior of each link in the system at the picosecond level all the way up to the highest level of packets, such as matching up request/response pairs in a protocol.

Once you've got measurements from all the different parts of the system, you need to be able to correlate the activity that you're seeing in the different parts of the system. Because cause and effect for a particular system level problem is not usually located in just one part of the system; often it can span the entire system. Something can happen in one area where you see the symptom, but the root cause of it may actually be somewhere else. So you need to be able to correlate activity in one part of the system with what's going on in the other part of the system.

That correlation takes two forms. One can be spatial correlation; that is, what's happening at this point in the system at the same time that something's happening at this point in the system. So, you're correlating across two different points in the system so that you can see behavior on both of those pieces at the same time.

(Continued -- next slide)



Tool Selection for InfiniBand System Debug

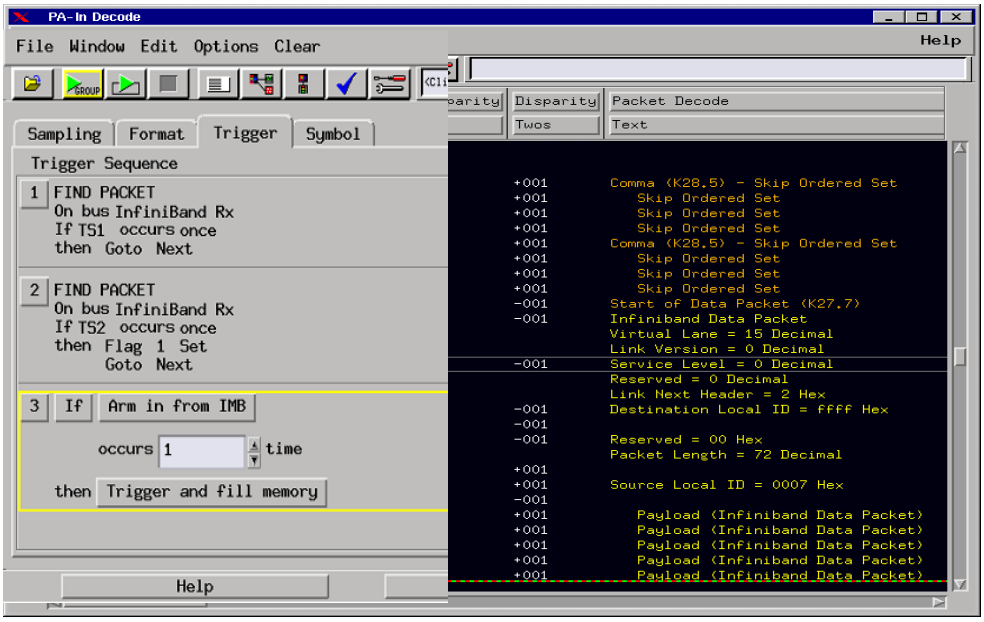
As you prepare to do system-level validation, you have to select the tools that you're going to use and set up your test bench for the system validation. We're really only talking about three basic types of tools in this paper and you need to understand what the tools are good at and how they complement other tools that you might choose to use for each job.

The first is a protocol analyzer for InfiniBand; the second is a Logic Analyzer that has InfiniBand support, and the third is a Traffic Generator to generate control in InfiniBand traffic for system validation. These each have different strengths. The protocol analyzer is really focused at a comprehensive view of what is going on on the InfiniBand channel it provides a very robust view of InfiniBand packets, robust triggering capabilities, and validation protocols at all layers of the protocol.

The Logic Analyzer (LA) is focused on looking at cross bus or multibus correlation. Looking at various points in the system at the same time the logic analyzer is more focused or optimized for looking at the lower levels of the protocol. It doesn't necessarily go all the way up to the application layer. But the major strength that the Logic Analyzer has is you can make measurements essentially on things that you can't actually see directly. You can set up the analyzer to infer internal state of your system from events that are observable across the entire system.

Some of the examples will show how that can be done. The logic analyzer is a very powerful tool for being able to reach inside ASICS that you can't probe directly. Then the LA will correlate information, the measurements that are made across the whole system and allow you to correlate measurements taken in one part of the system with measurements taken in another part of the system so you can see how, in time, events in one part affect events in another part.

InfiniBand Protocol Logic Analysis



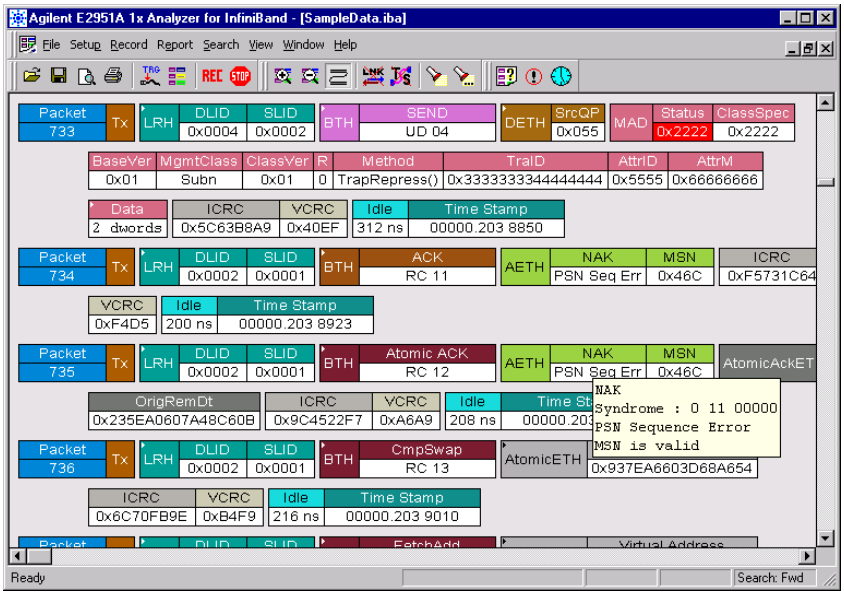
Agilent Technologies

Logic Analyzer Protocol Triggering and Decode

This is an example of the Logic Analyzer and how you might set up an InfiniBand trigger on a Logic Analyzer, along with an example of what the display looks like. The thing I would point out here is that this way of setting up triggers and looking at results is very similar to the way that it's traditionally been done with microprocessor busses and other kinds of busses in the system. The Logic Analyzer is optimized for cross bus measurements and so it tries to provide a consistent way of looking at data no matter what bus that you're on.

The Protocol Analyzer is optimized for protocol measurement; a lot of protocols are hierarchical and so the protocol analyzer will provide a high hierarchical view of the protocol. You can click on specific packets and pieces of packets and expand them to get more detail, so it provides a much richer, more robust view of the InfiniBand protocol.

InfiniBand Protocol Analysis

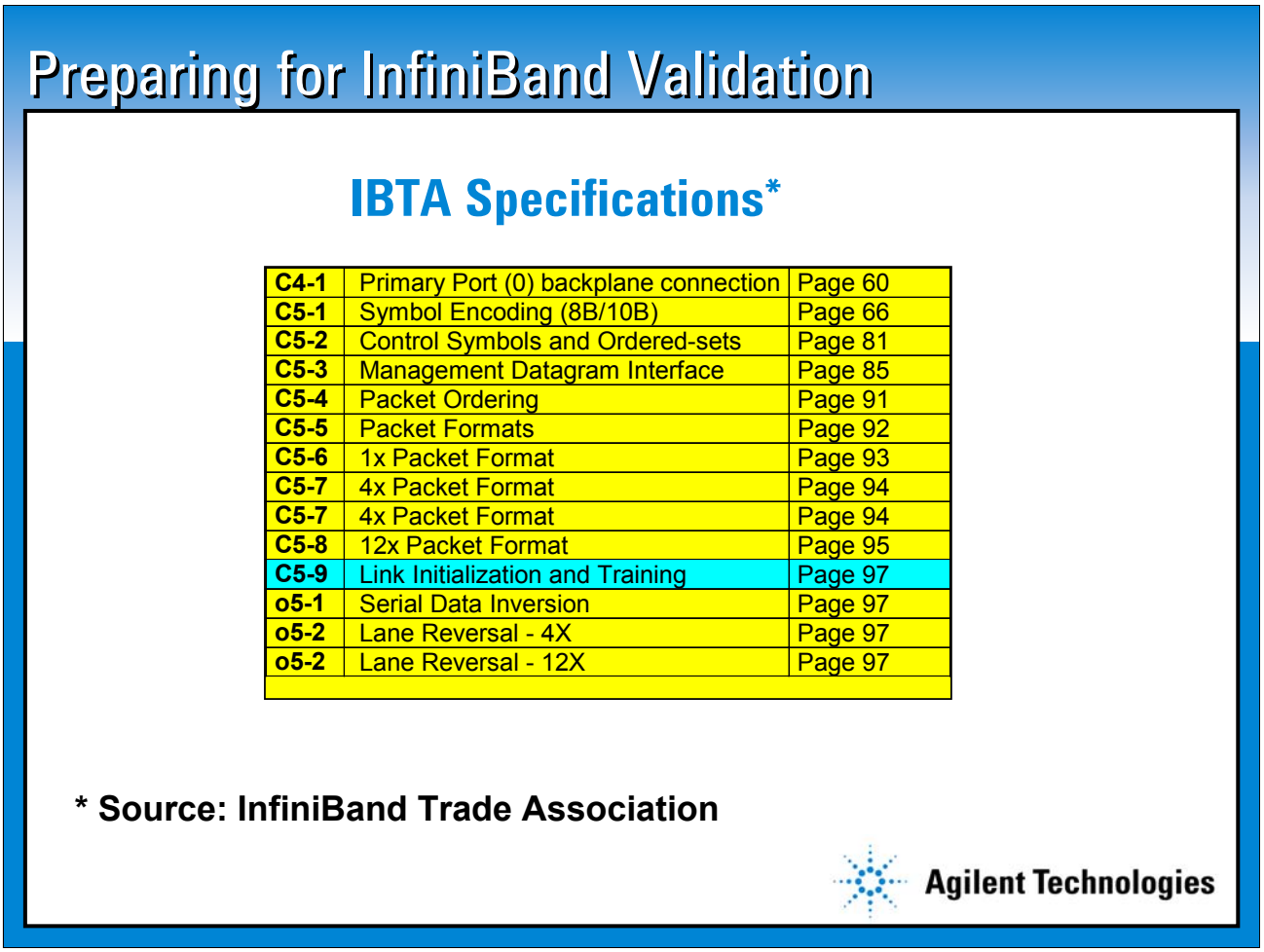


Agilent Technologies

Logic Analyzer Protocol Triggering and Decode

This is an example of the Logic Analyzer and how you might set up an InfiniBand trigger on a Logic Analyzer, along with an example of what the display looks like. The thing I would point out here is that this way of setting up triggers and looking at results is very similar to the way that it's traditionally been done with microprocessor busses and other kinds of busses in the system. The Logic Analyzer is optimized for cross bus measurements and so it tries to provide a consistent way of looking at data no matter what bus that you're on.

The Protocol Analyzer is optimized for protocol measurement; a lot of protocols are hierarchical and so the protocol analyzer will provide a high hierarchical view of the protocol. You can click on specific packets and pieces of packets and expand them to get more detail, so it provides a much richer, more robust view of the InfiniBand protocol.



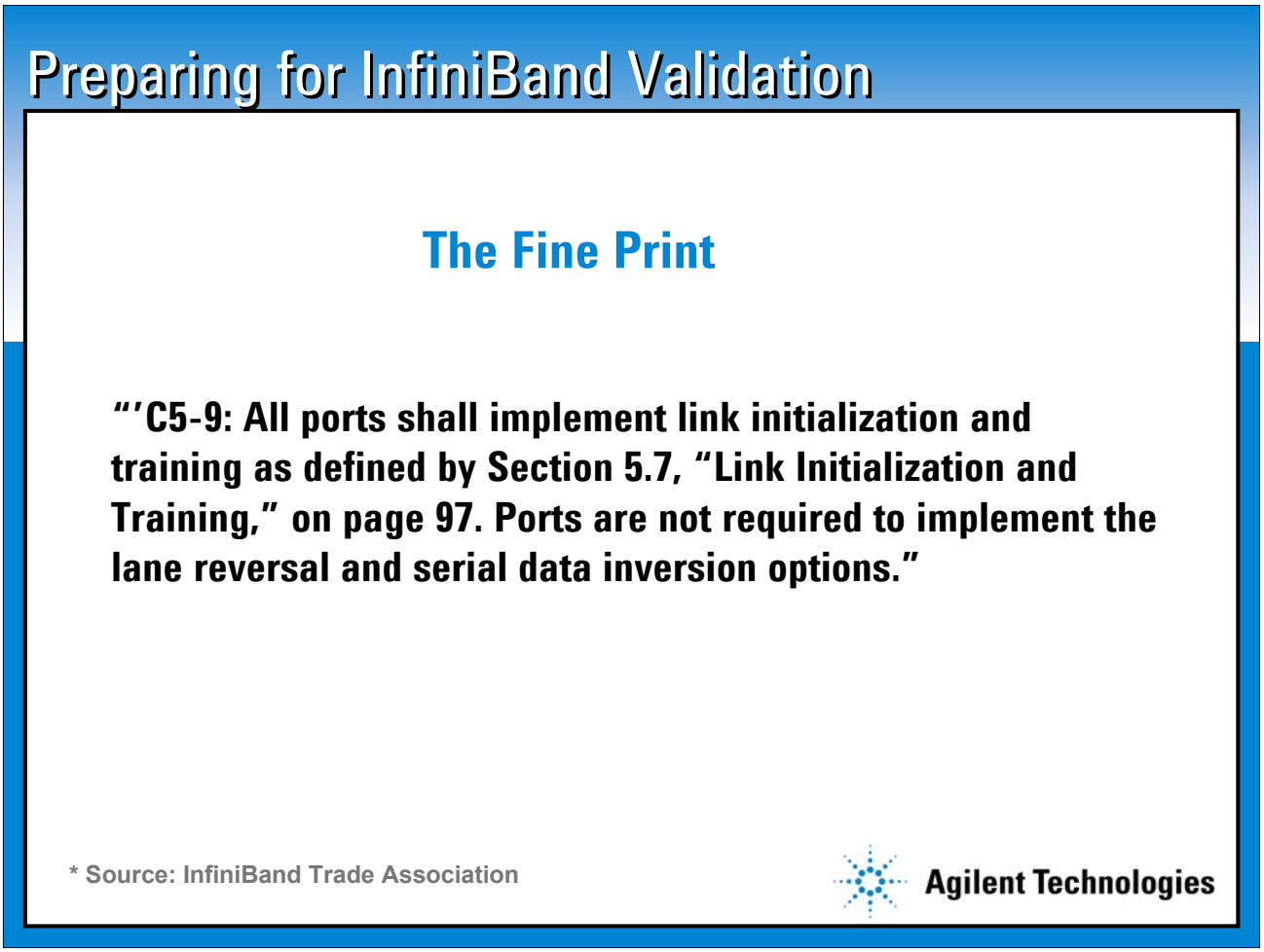
Preparing for InfiniBand Validation and Debug

Once you've selected your tools or have an idea of what tools to choose, you need to start putting together a validation plan for your InfiniBand based system.

It turns out that the InfiniBand specification actually provides a roadmap for putting this together in the form of the specifications and checklists developed in the standard and run by the compliance and operability working group. If you look in volume 1 and 2 of the standard, you'll see towards the end of the document a checklist that looks very much like this; there are line items here, and the lines marked with C are items that are required to be compliant; you have to comply with those in order to be compliant with the InfiniBand specifications.

The ones marked with an O are optional. These are things you can add and say, we add extra value, we support these optional features in the InfiniBand spec.

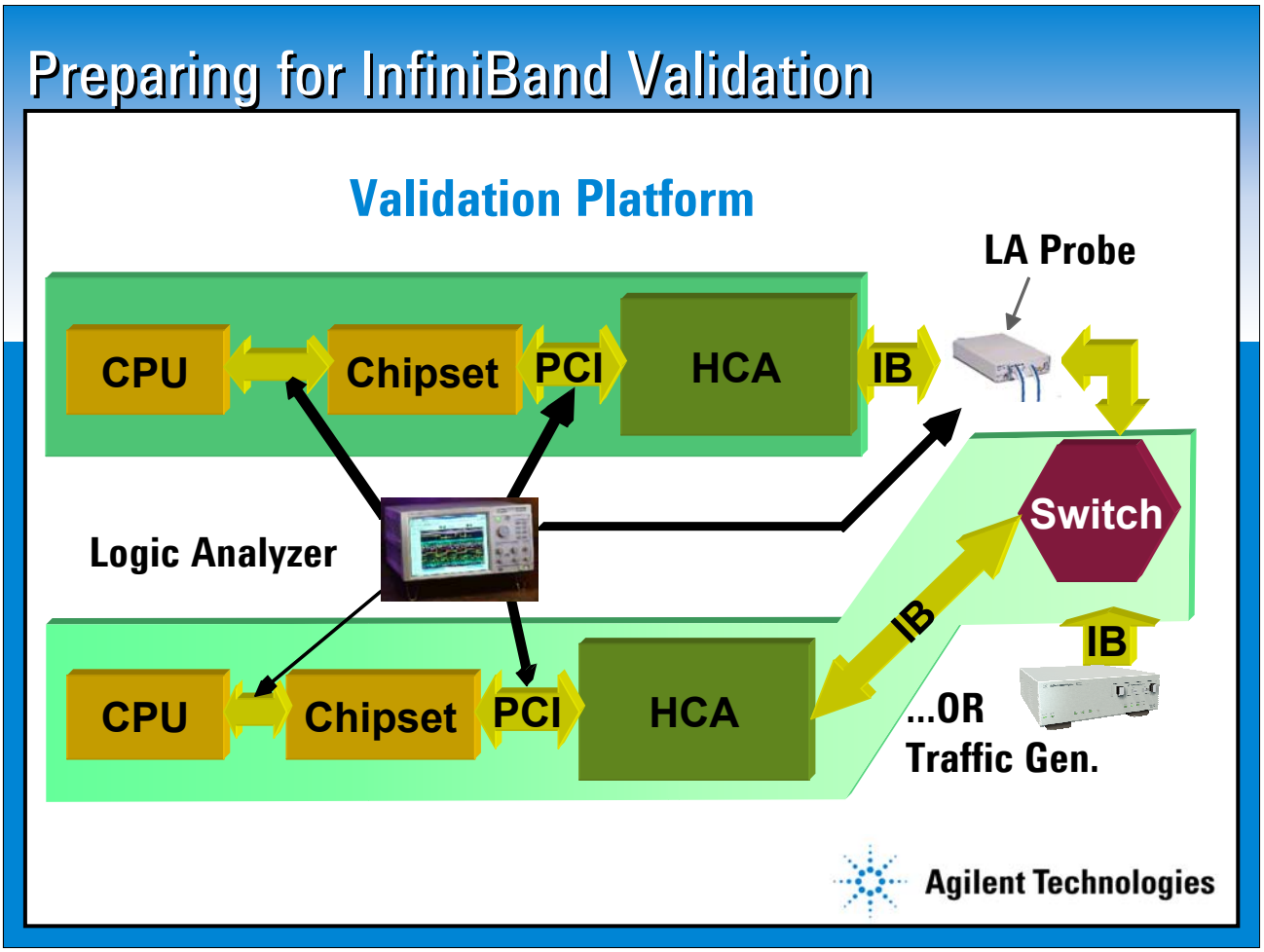
You can go into the spec and start to put together a checklist used to develop a structure for a test plan. Behind each one of these items is a more detailed explanation of what that item really means.



Preparing for InfiniBand Validation and Debug--The Fine Print

In putting a test plan together you really need to read the fine print so that you have your checklist and then you read the fine print to help you define a specific test case and a specific test procedure to validate your system against that part of the InfiniBand spec.

This is just a typical example, in fact, this one not only amplifies the hot line item in the checklist but also refers to other sections of the spec that will provide even more detail on precisely what it means to comply with that part of the specification.



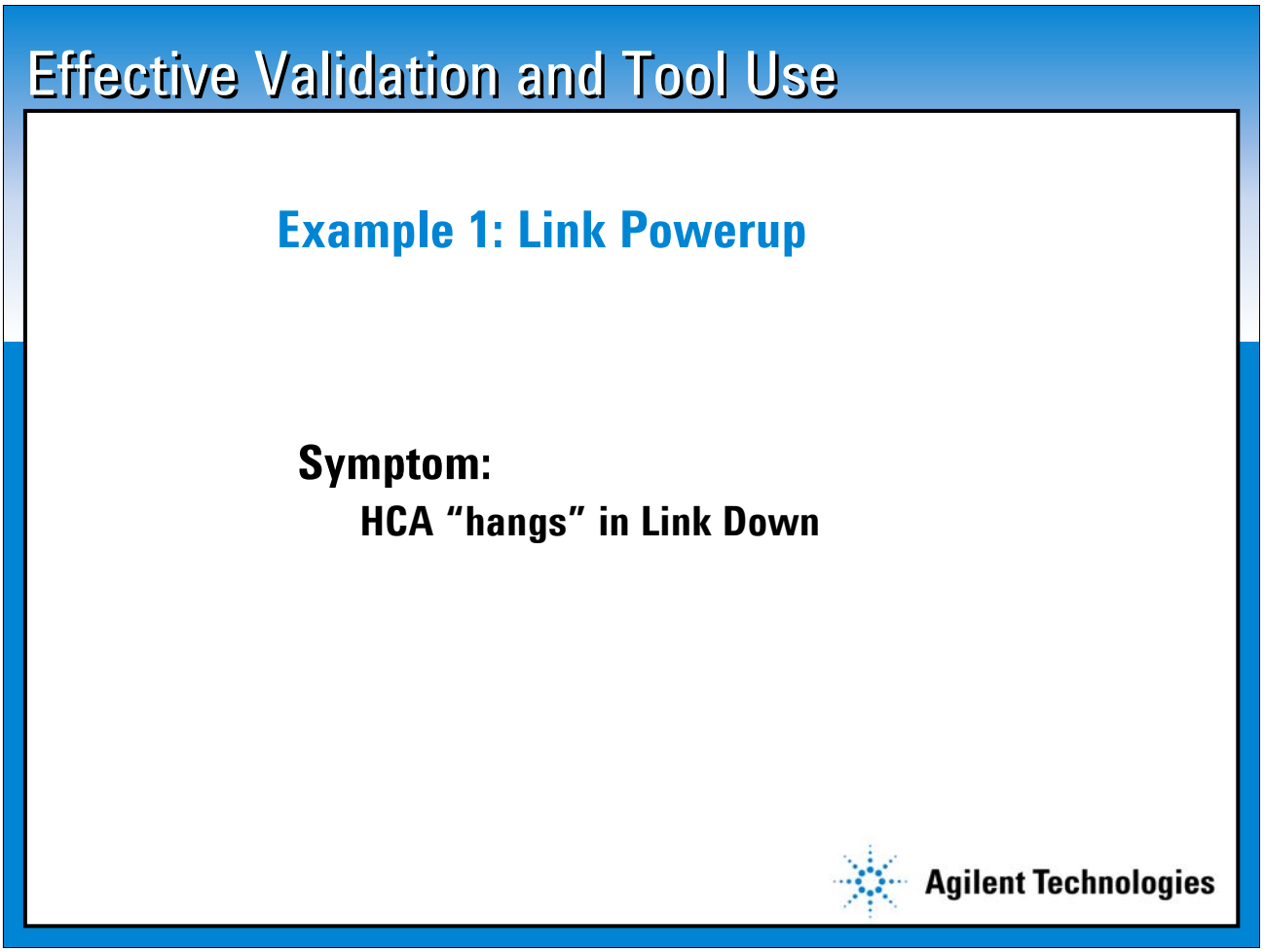
Validation Platform

The last thing you need to do prior to system validation is put together a validation platform. This is an example of a typical validation platform. We'll be using this platform throughout the rest of the paper. This platform is a server system with a CPU, chipset, and InfiniBand host channel adapter. It is representative of the first InfiniBand implementations available on the market.

The first InfiniBand systems that have come out are mostly PCI cards that plug into a PCI slot with InfiniBand ports out the back. In the future, engineers will be migrating these HCA functions right into the chipset to lower costs and provide improved performance. So this validation platform is representative of the kind of platforms that you can set up today.

On one side of the platform is an HCA and then on the other side, something for it to talk to; a switch, another HCA, or real world environment for your validation. If you need a deterministic set of stimulus, you can use an InfiniBand Traffic Generator and just have it talk directly to the HCA. At various points in this system, you'll probe the different channels and busses so that as you find problems and trace them to root cause, you've got visibility into the different parts of the system that you need to get the problem debugged. In this picture, we're probing CPU front-side bus, PCI bus, InfiniBand channel.

Once you assemble the validation platform, you have a test plan, and you have all the proper tools together, then it's really time to put the power on. See if the thing works, and start working through the checklists.

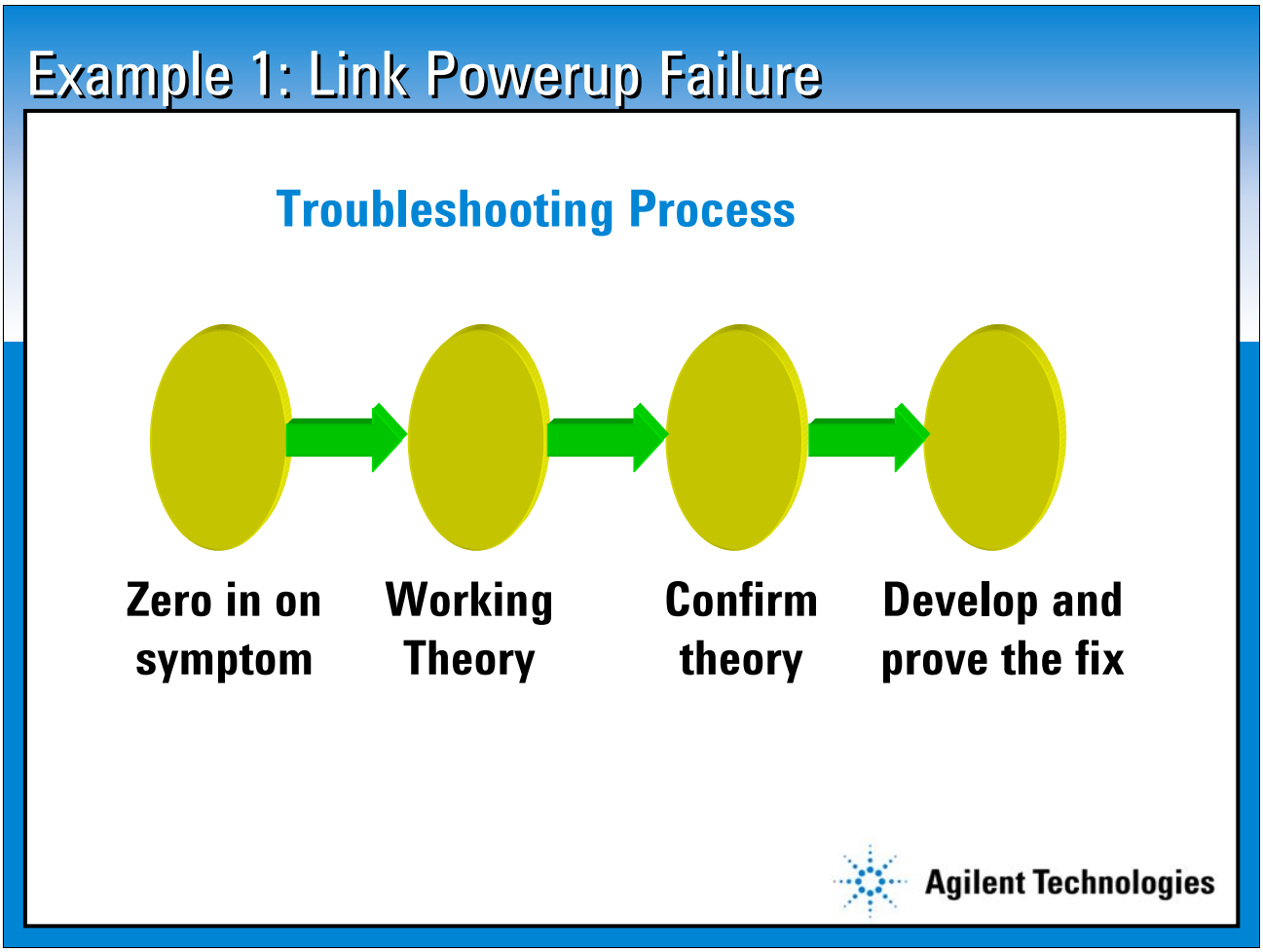


Effective Validation and Debug Tool Use

Example 1: Link Powerup

This is the first specific example of how to actually use these tools in this platform to debug your system. This example is the very first thing that you typically do after you run the smoke test. You plug all the cables in, each box powers up by itself, you plug all the cables together and you power it up.

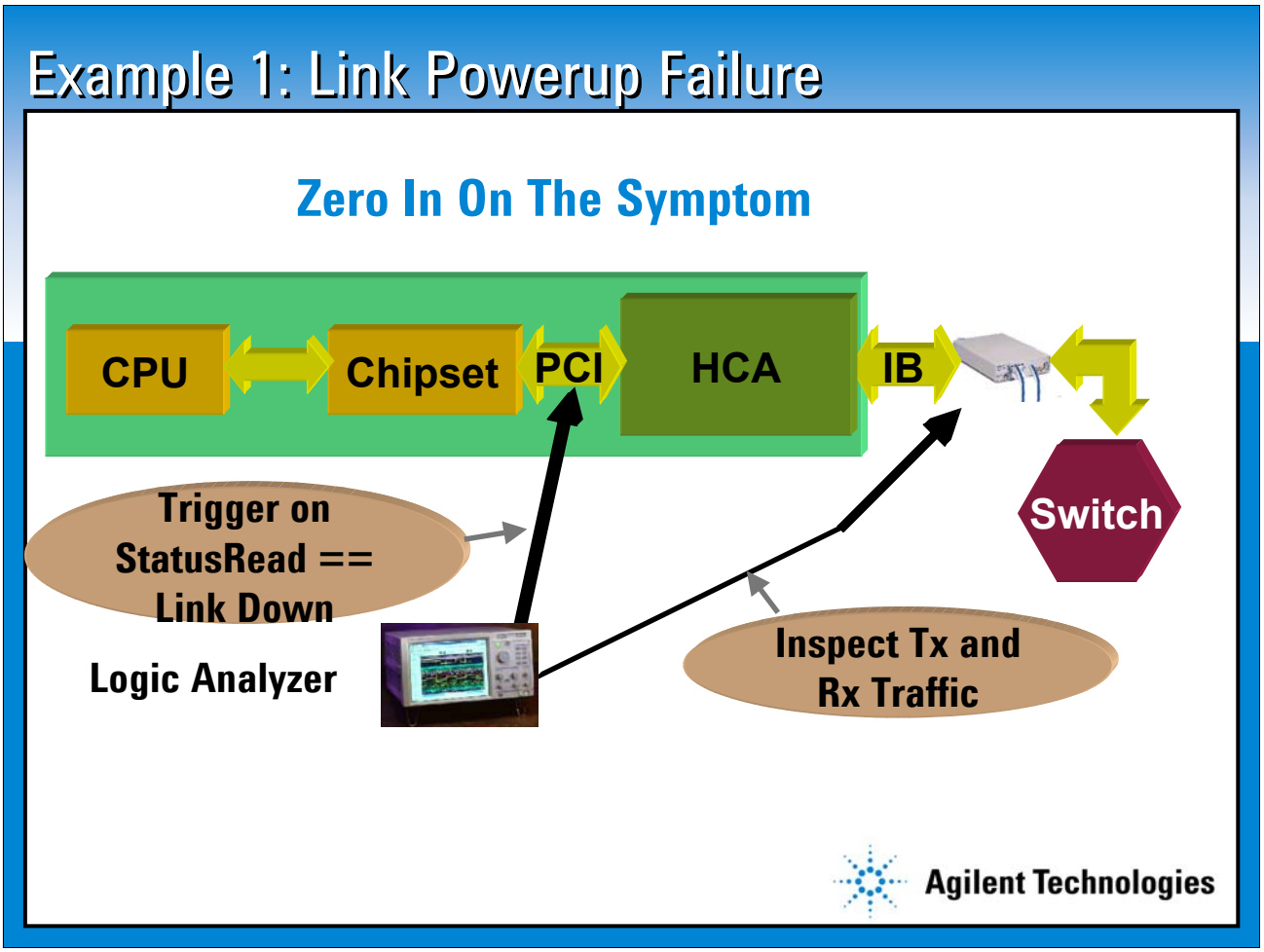
You're expecting that the InfiniBand channel will come up, initialize itself, and the link will be up and ready for work. Unfortunately, what you find out is that after you power the system up, the link hangs. So you expected the link to come up, but instead it's hung in link-down status, and at this point that's all you know. The problem then is to figure out why did that happen and how do we keep that from happening. What is the reason why the system is failing.



Example 1: Link Powerup Negotiation Failure

Troubleshooting Process

This is a summary of a somewhat idealized but typical troubleshooting process for debugging a problem like this. I'm sure everybody's gone through this kind of thing before, where you know very little to start so you want to zero in on the symptom and try to get more information about what really might be going wrong, so that you can develop a working theory that gives you some idea of what might be going wrong, and then you can make some additional measurements to either prove or disprove that that theory really explains what's happening and then go ahead and implement the fix. We'll walk through each of these steps in this example.



Example 1: Link Powerup Negotiation Failure

Zero in on the Symptom

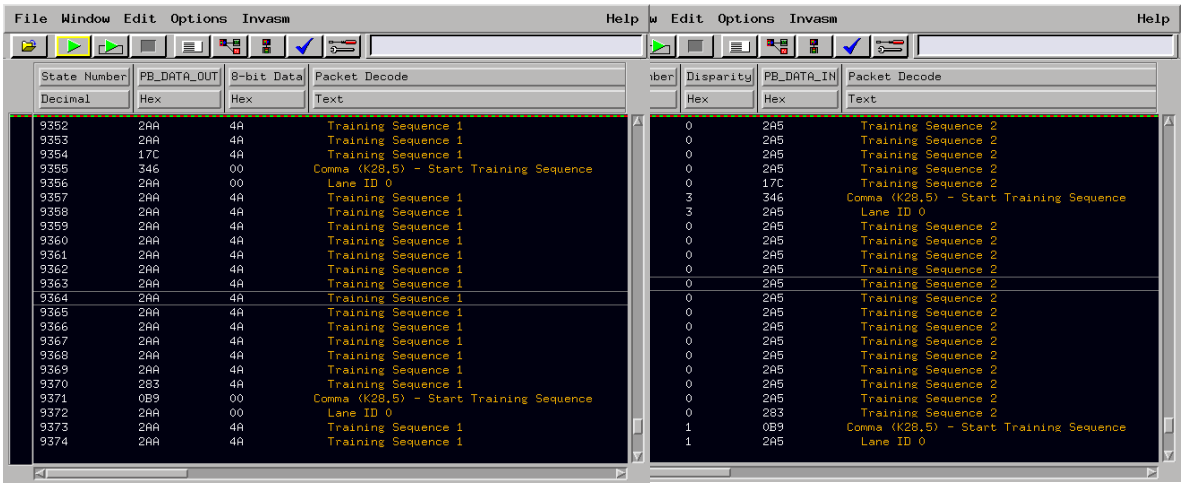
The first step is to zero in on the symptom, try to get more information to tell you what is really going on in your system and why it failed. Here we've set up a relatively simple measurement. All we know right now is that the link is down when it should have been up.

The simple thing to do is to setup the Logic Analyzer to look at the PCI bus, we want to look at traffic going to and from the HCA. We also want to look at the InfiniBand link with the Logic Analyzer for time correlated measurements.

The HCA at some point is going to return to the computer status that says the link is down. What we want to do is trigger on the event when the link is actually reported as being down and then take a look at the time correlated InfiniBand traffic and see if we can discover something that might give us a clue about what's going wrong.

Example 1: Link Powerup Failure

Initial Traffic Traceback



We didn't complete powerup. **....But they did!**

Agilent Technologies

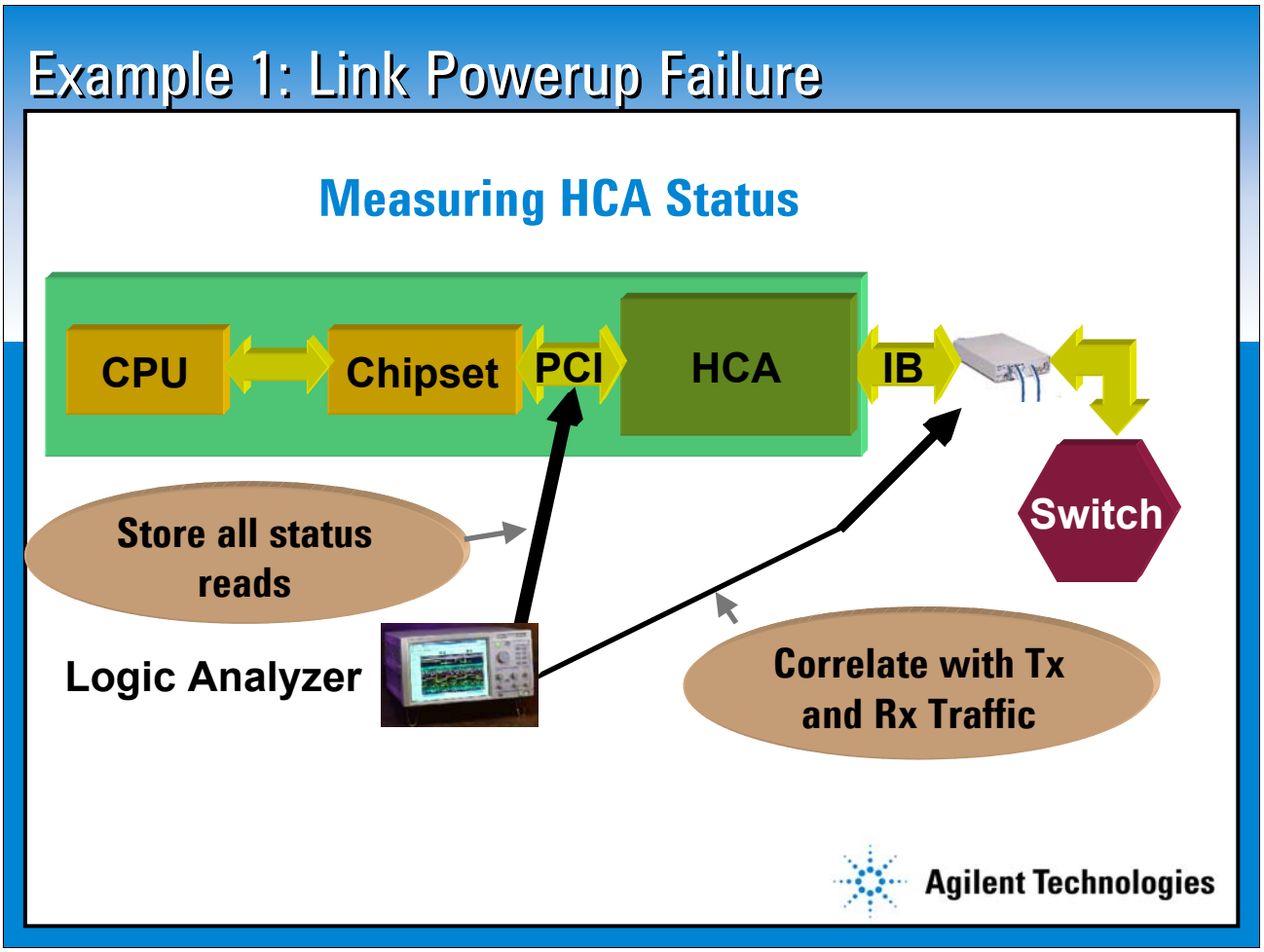
Example 1: Link Powerup Negotiation Failure

Initial Traffic Traceback

The PCI and InfiniBand measurement was taken and look at the results. You can see this window on the right shows InfiniBand traffic going out from the HCA to the far end device. The window on the left shows traffic coming back from the far end device to our HCA that we're measuring.

You can see that we're still sending TS1's , Training Sequence 1's, while the far end is sending Training Sequence 2's. What that means is that the far end has completed its link configuration, which is what Training Sequence 2's mean. When a channel adapter has completed its configuration and is ready to actually do things with the InfiniBand protocol, it will send TS2's back. So the far end is configured, but we are still sending TS1's which means that we haven't configured ourselves yet.

For some reason we've locked up in a link down and have not completed configuration. This tells us a few things. One is that the channel at a certain electrical level is functioning, and secondly, that there is some sort of functional failure; some reason why we're not completing our configuration even when the other side is completing.



Example 1: Link Powerup Negotiation Failure

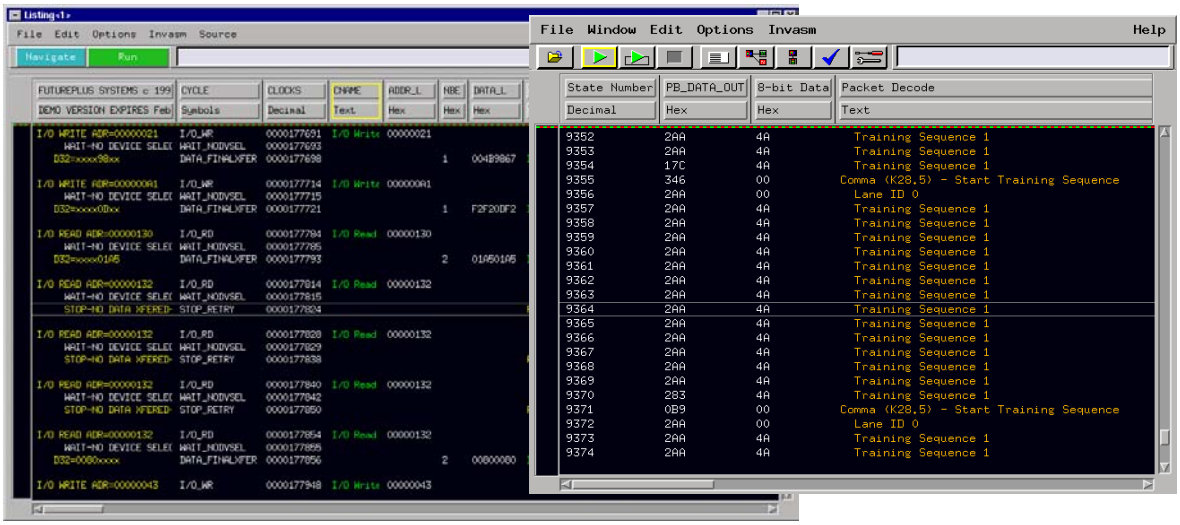
Measuring HCA Status

Next we need figure out why that might be happening. We want to make some additional measurements to see if we can get enough insight to put a theory together. The next thing to do is to focus on all status reads coming back from the HCA. The HCA is going to be sending an interrupt back to the CPU saying 'here's the current status as I'm going through the link power up negotiation'. We will still trigger on link-down, but instead of looking at all PCI traffic, focus on status read to zero in on a specific type of transaction.

Keep an eye on the InfiniBand traffic and see how the status of the HCA over time corresponds to the InfiniBand traffic that's going on over time.

Example 1: Link Powerup Failure

HCA Status Trace



A link timeout status was returned.

Agilent Technologies

Example 1: Link Powerup Negotiation Failure

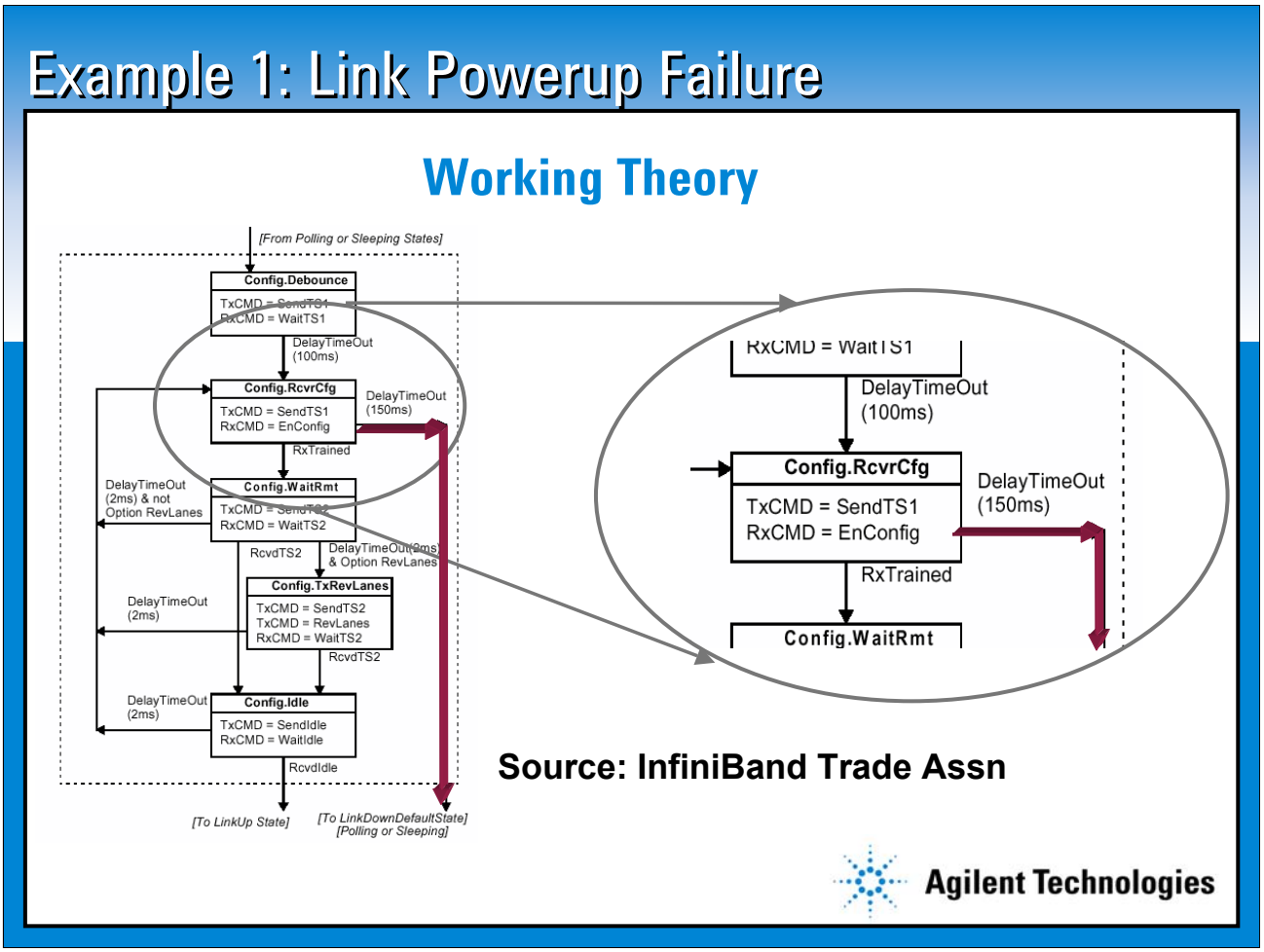
HCA Status Trace

Here's the result of such a measurement. The right window shows that the HCA is still transmitting Training Sequence 1's, and the left window, (a trace of the PCI bus) shows that the HCA was returning a time-out status. So, for some reason we didn't power up, the HCA timed out and said 'I give up'.

Now, it's not shown on this display, but if we look at the InfiniBand receive link, you would see that during that time lap period, TS1's were coming in and by the time we left, TS2's were coming in. That tells us that sometime during this time-out period, the far end of the channel switched from sending TS1's to TS2's. That leads us to a working theory.

Working Theory:

If TS2's start coming in while I'm still in link configuration state, something gets confused and I never leave link configuration.



Example 1: Link Powerup Negotiation Failure

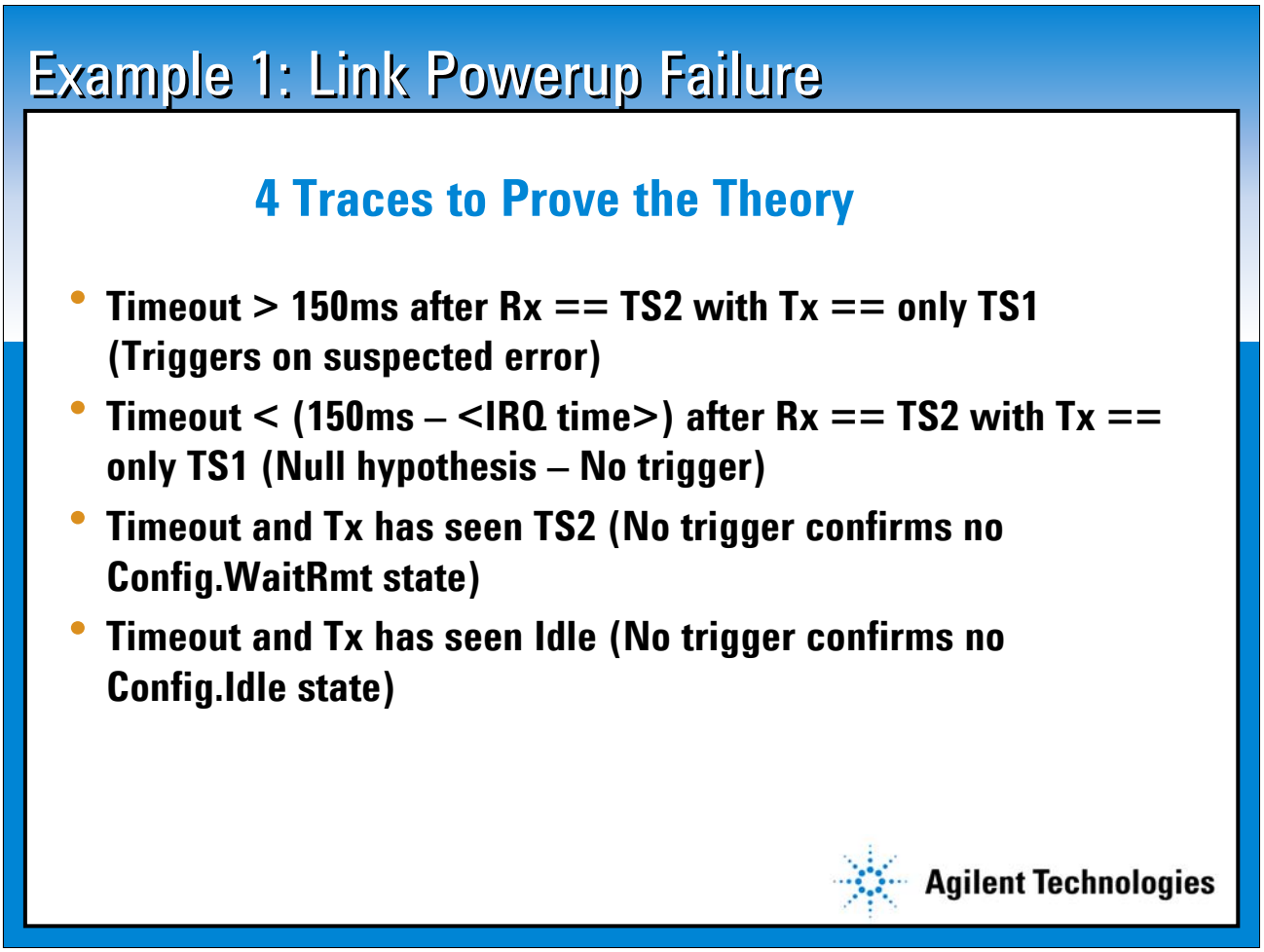
Working Theory

Going back to the standard, this diagram is taken from the InfiniBand spec and it shows graphically what this working theory is predicting. First the HCA goes into receiver configuration state, sends TS1's out. When the link is trained it will start sending TS2's out.

The working theory is that while we are in this first state, TS1's that are coming in turn to TS2's. For some reason that switch from TS1's to TS2's confuses the HCA when it's in this state and after a 150 milliseconds, it times out.

It turns out that there are actually three other time-out conditions that can occur that are 2 milliseconds long each.

This timeout is 150 milliseconds long. What we would like to know is: Do we come in at the top and go right out to a timeout, or are we going through some other path in this state machine? That helps us decide what parts of the state machine are causing the problem.



Example 1: Link Powerup Negotiation Failure

4 Traces Prove the Theory

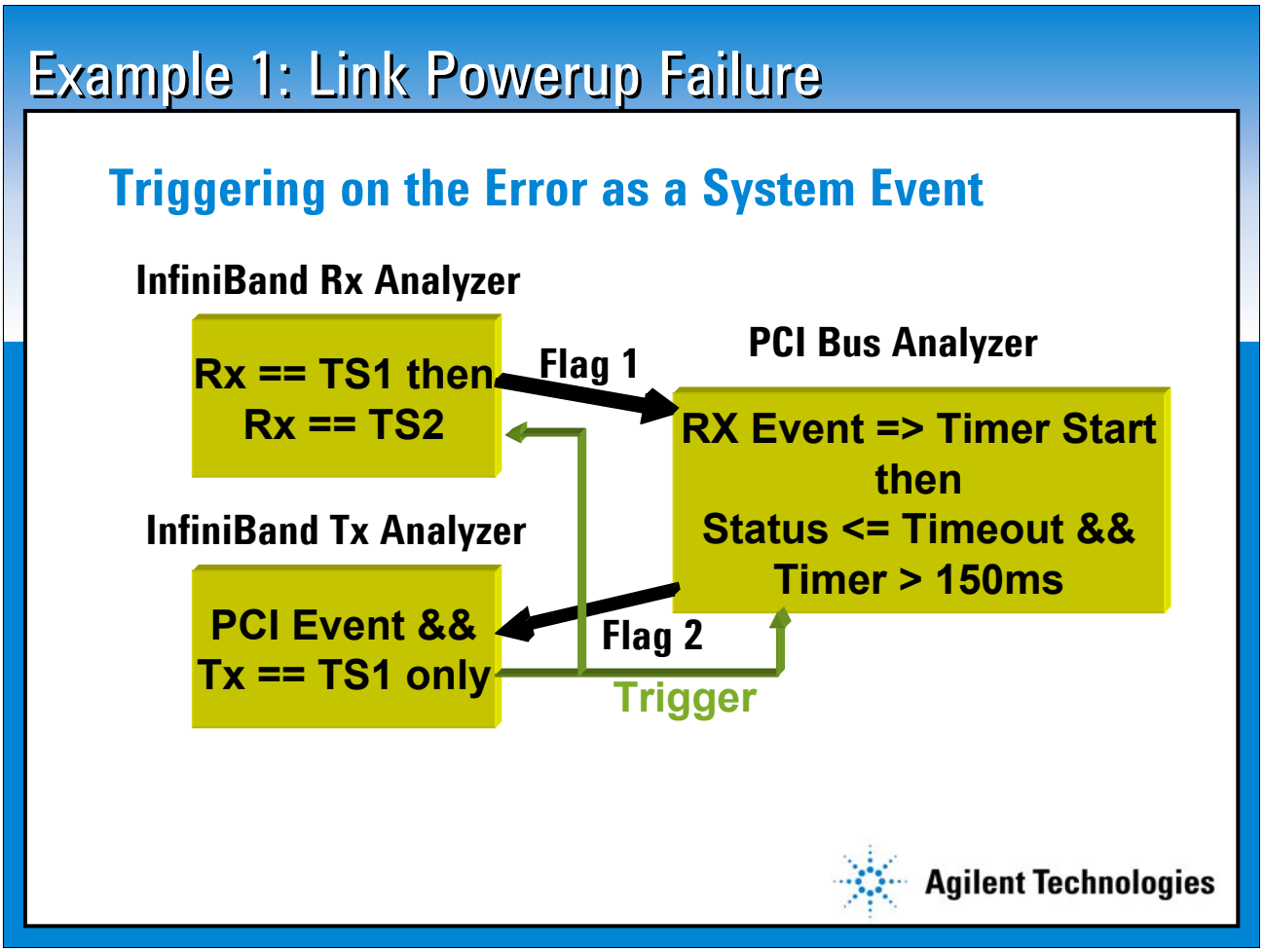
To answer that we can set up four measurements with the Logic Analyzer to try and confirm whether our working theory is correct or not. These are all measurements that involve looking at activity in several parts of the system at the same time.

If we setup the trigger that says if the time-out is greater than 150 milliseconds and we're receiving TS2's and we're still only transmitting TS1's, and the Logic Analyzer triggers, then that tells us that that path outlined in red in the previous slide was the path that we took.

If it doesn't trigger, then there is some other path that we took.

We can set up these other three measurements and if our theory is correct, none of them should trigger. But if one of them does, then that will tell us that there's something wrong with our working theory and maybe we need to look somewhere else.

Cross bus triggering is essential because we're looking at transmit and receive traffic on the InfiniBand channel and we also have to look at PCI traffic to detect a time-out.

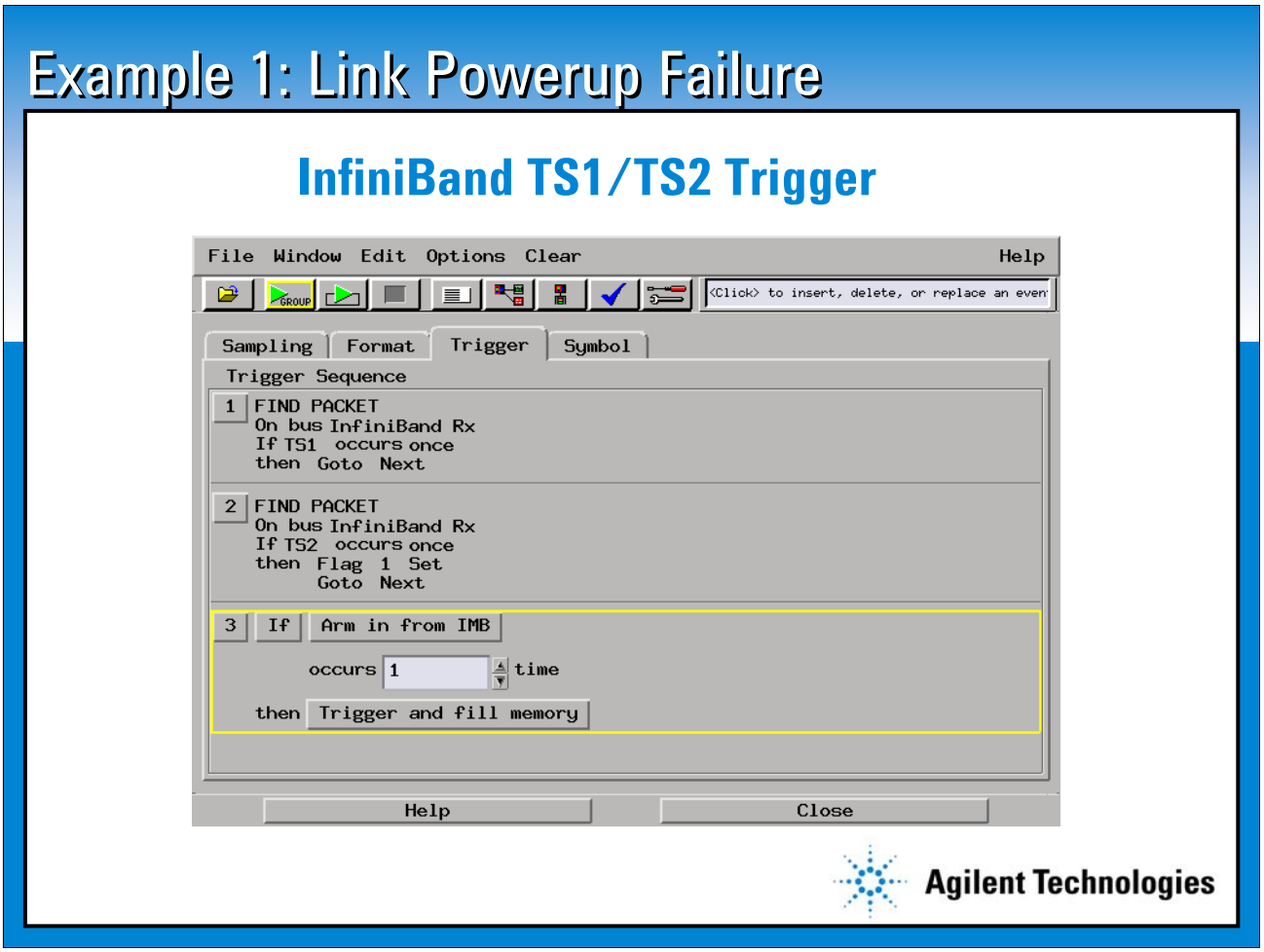


Example 1: Link Powerup Negotiation Failure
Triggering on the Error

This is specifically how you set up a measurement to trigger on the Link Powerup Negotiation failure. We need to look at the receive traffic on the InfiniBand channel, the transmit traffic, and the PCI bus. Walking through, we're going to look for the point in time when we switch from the receiving TS1's to start receiving TS2's. The Logic Analyzer is going to look at the receive channel and when it sees it's starting to receive TS2's , it's going to set a flag--basically a global event, to say 'we've made this transition and now every other analyzer is able to know that we've made the transition from TS1's to TS2's.'

When that occurs, then the analyzer that's looking at the PCI bus will start a timer, and wait for 150 milliseconds. It will also be looking for time-out status. So, basically, what it's doing is looking to see if we achieve a time-out status more than 150 milliseconds after we switch from TS1's to TS2's. That tells us that we're going out that final branch. If that happens, then we'll set another flag which says ‘we think we're exiting out this branch’. Then there's a final check -- another analyzer is looking at the transmit status, the InfiniBand transmit link, and if we're still sending TS1's, then all three conditions are satisfied and the analyzer will trigger.

This measurement is built from an understanding of states that are occurring on different parts of the system. You cannot make this measurement without looking at several parts of the system.

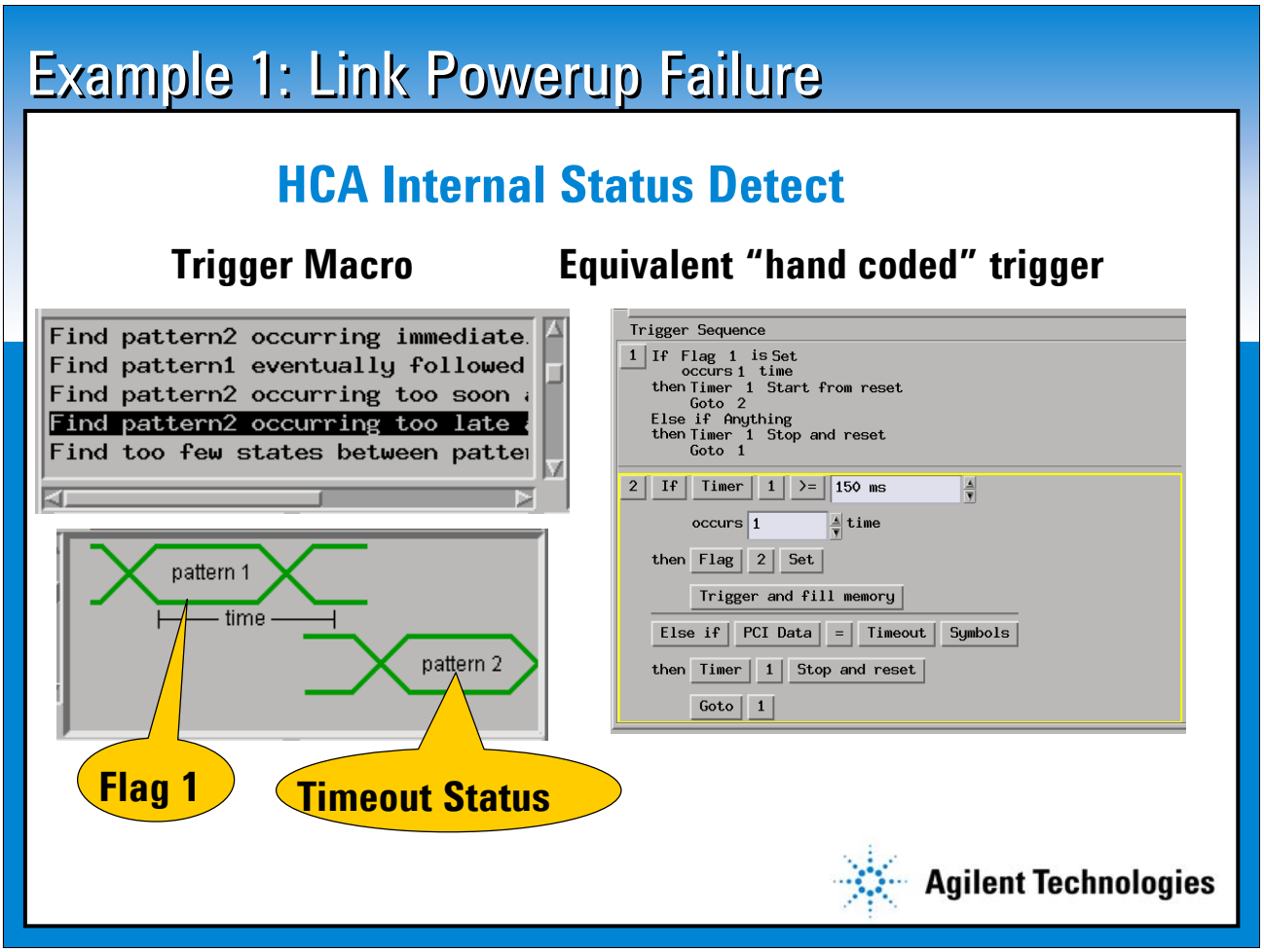


Example 1: Link Powerup Negotiation Failure

InfiniBand TS1 Trigger

This is an example of how you'd set up the trigger on the transmit channel. We want to find a packet where you have a TS1, so you wait until you start seeing TS1's, the link has come up initially and then you wait until you start seeing TS2's, and then as soon as you see a TS2, set that flag. Then once all three busses have found their conditions, then you'll trigger and fill memory. That means take the measurement, complete the trace.

Triggering on this part of the InfiniBand protocol is a pretty straightforward thing for the Logic Analyzer.



Example 1: Link Powerup Negotiation Failure

HCA Internal Status Detect

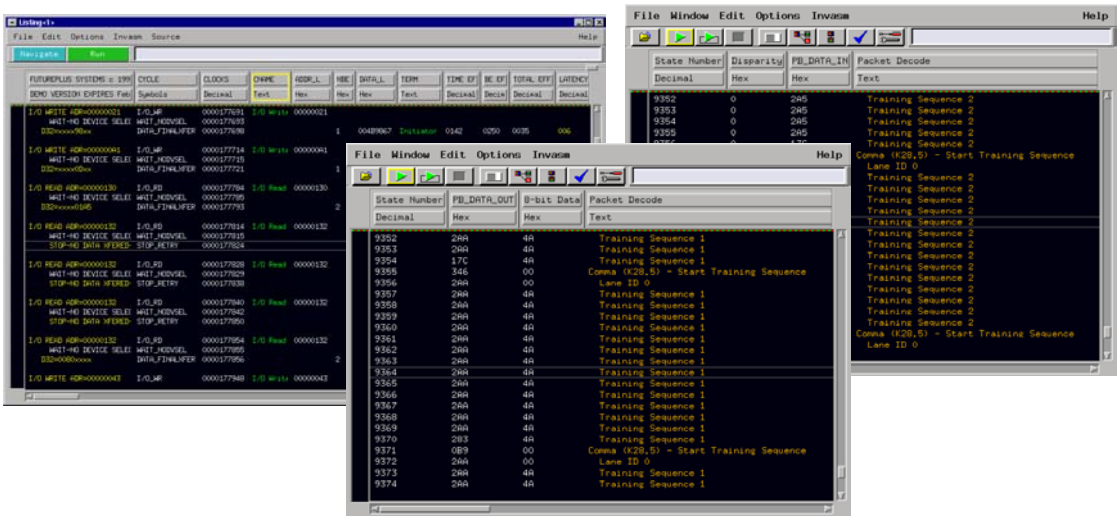
This is an example of how you'd set up the PCI trace. The simple way of stating this is that we're looking for a pattern which is flag 1 being set from the previous Logic Analyzer module. Then we're going to wait a certain amount of time and make sure that pattern 2 occurs only after that amount of time has transpired.

Basically, we wait until flag 1 is set, start the timer, wait 150 milliseconds and make sure that the time-out occurs only after 150 milleseconds and, if that is true, then you set flag 2 and say, okay, time-out has occurred greater than a 150 milleseconds. If it is less than 150 and the time-out occurs, then it will reset the timer and start over.

This is the macro that we used to set up this measurement and this is how that macro actually expands to program the analyzer.

Example 1: Link Powerup Failure

Pulling It All Together

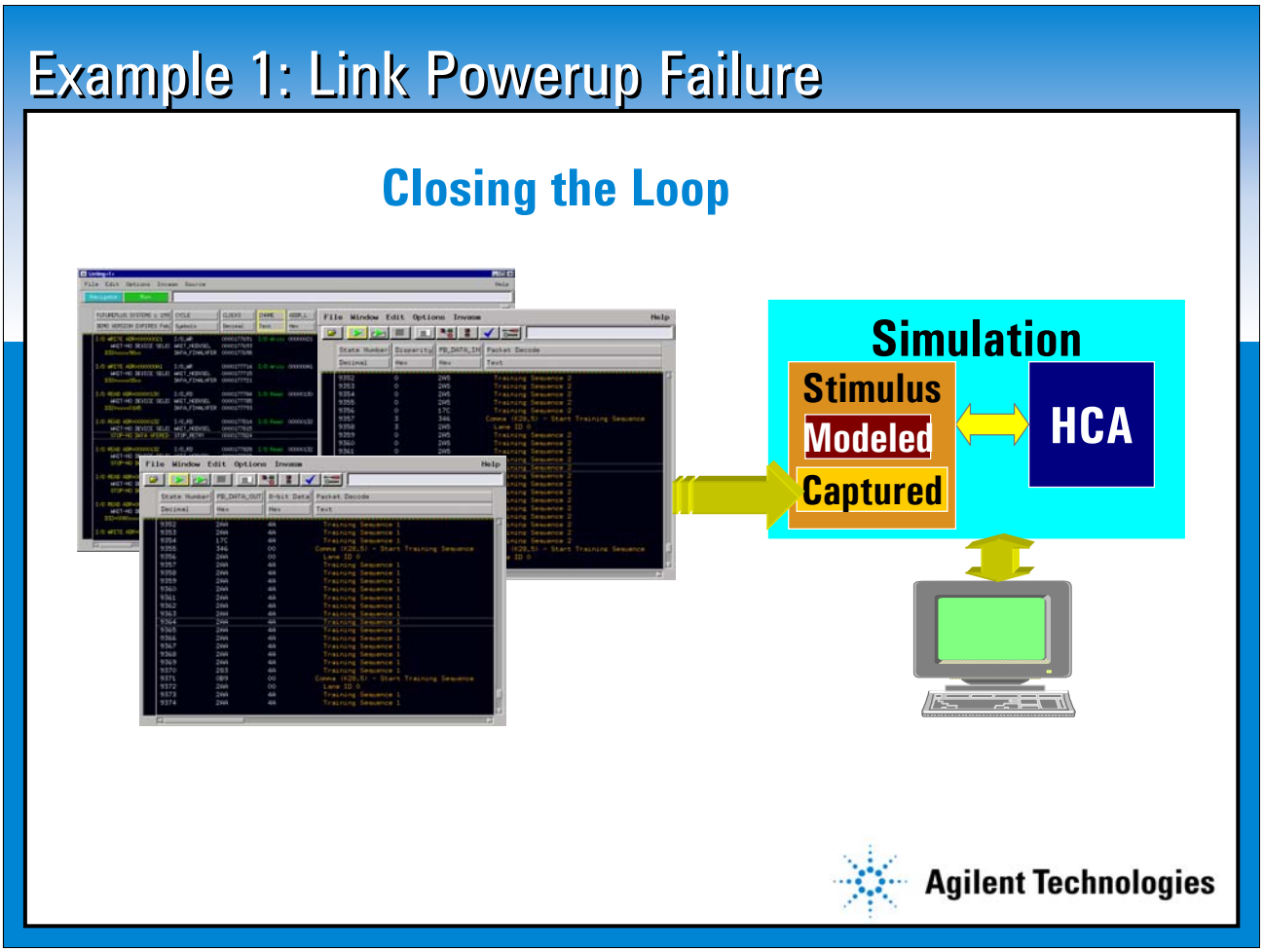


Agilent Technologies

Example 1: Link Powerup Negotiation Failure

Pulling It All Together

We set up the measurement and ran it, and here we see traces from all three parts -- the PCI bus, the transmit channel, the receive channel on the InfiniBand. We want to take a look at these and make sure that they are really telling us what we expect to see. In this case, since this is an example, our working theory is right and we say okay, somehow when we switch from TS1's to TS2's and receive a config state, things get confused and we time-out. Then by looking at these traces, we can build confidence that the system is really behaving the way we think it should behave, given this problem.

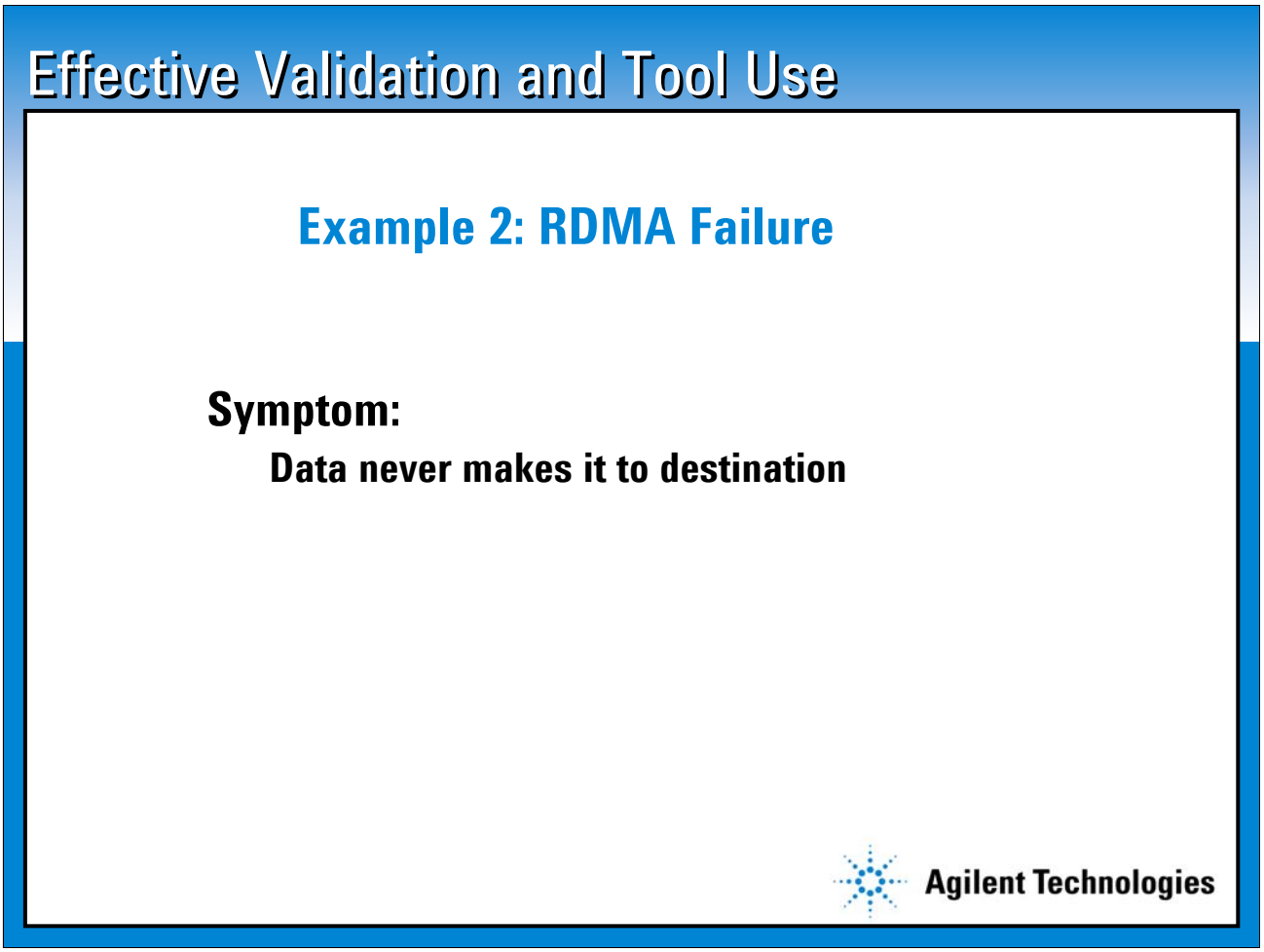


Example 1: Link Powerup Negotiation Failure

Closing the Loop

The last step is to fix the problem and prove that the fix is correct. It's often the case that especially when you're debugging ASICS, in order to fix the problem you want to go back to simulation. (If you had come up with a test case simulation that covered this condition, this problem probably wouldn't have crept into the ASIC in the first place.) Close the loop by taking real world data that was acquired by the measurement tools and then feeding the results into the simulation to reproduce that test case. Perhaps consider adding it to your regression test.

Set up an environment that reproduces the problem in simulation so that when you develop a fix, you're able to prove that the fix really works. You can close the loop by taking real world data , adding it to your simulation environment, making your fix, and proving that the fix is going to work.



Effective Validation and Debug Tool Use

Example 2: RDMA Failure

Now we'll go on to a second example which moves up the protocol stack a little bit. This example is the failure of a remote DMA (Direct Memory Access) transfer.

Again, we start out with very little information. We've set up a remote DMA transfer and for some reason or another all we know is that the data on this end never made it to the other end or it didn't wind up where it was supposed to. And that's all we know.

Example 2: RDMA Transfer Failure

IBTA Requirements*

C9-16	RDMA WRITE - DMA Length Limits	Page 212
C9-17	RDMA WRITE Segmenting/Reassembly	Page 212
C9-18	Multi-packet RDMA WRITE Rule	Page 214
C9-19	RDMA WRITE Request - Req'd Headers	Page 214
C9-20	RDMA WRITE Resp - Req'd Headers	Page 215
C9-21	RDMA READ Response Segments/Reassembly	Page 215
C9-22	RDMA READ DMA Length Limits	Page 215
C10-53	Physical buffer alignment & length reqs	Page 406
C10-54	General reqs wrt local accesses to Regions	Page 406
C10-55	Local access rights checking against Region	Page 406
C10-56	General reqs wrt remote accesses to Regions	Page 407
C10-57	Remote access rights checking against Region	Page 407
C10-58	Deregistration of overlapping Memory Regions	Page 407
C10-59	Deregistration while access in progress	Page 407

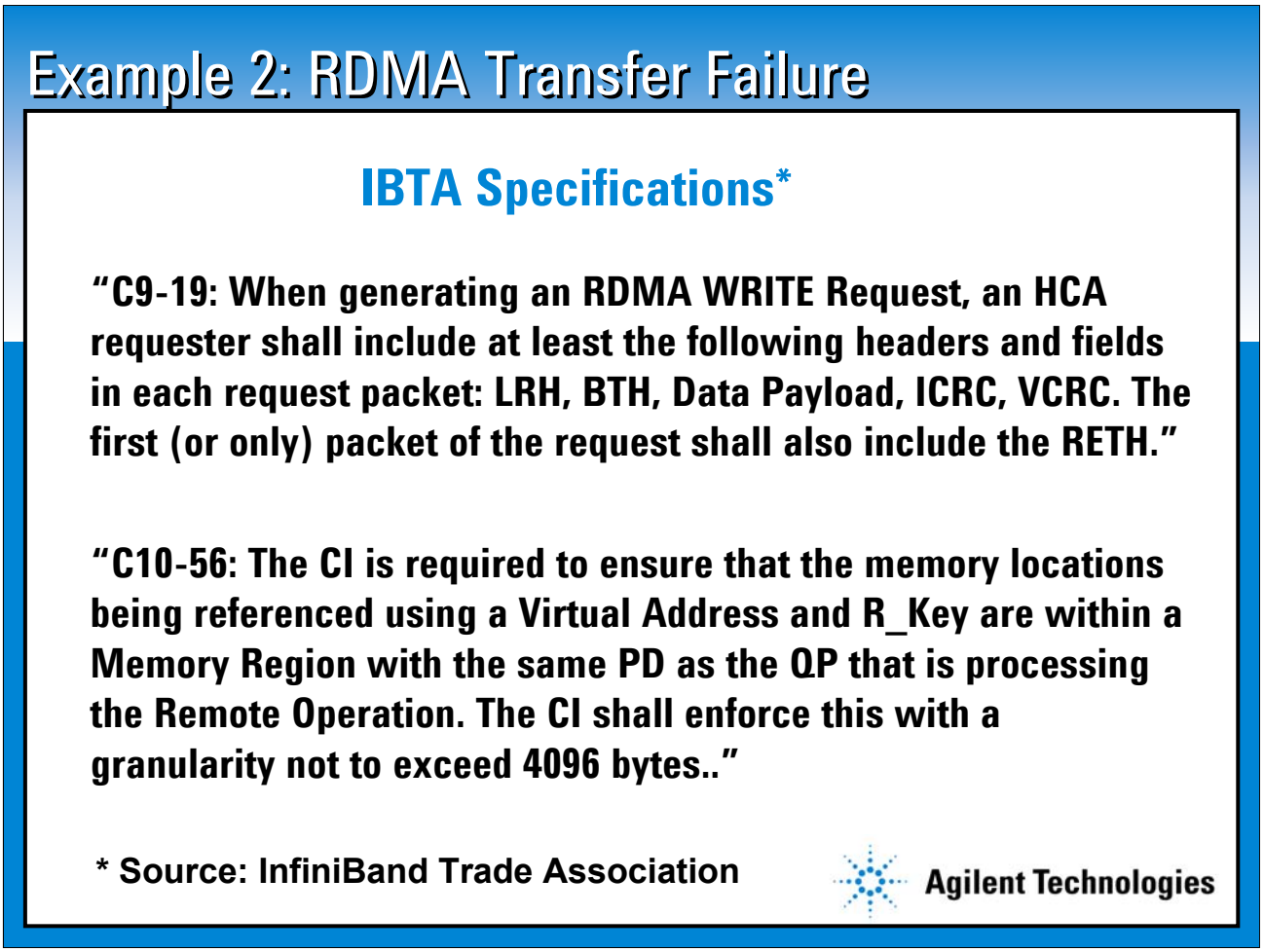
*** Source: InfiniBand Trade Association**

Agilent Technologies

Example 2: RDMA Transfer Failure

IBTA Requirements

When you run into a problem like this, it's a good idea to go back to the spec. In this case, there are two parts of the spec that really govern this kind of a remote DMA transfer and they're highlighted here in blue--they talk about how you do the WRITE requests and how you actually set up the memory to receive the requests.

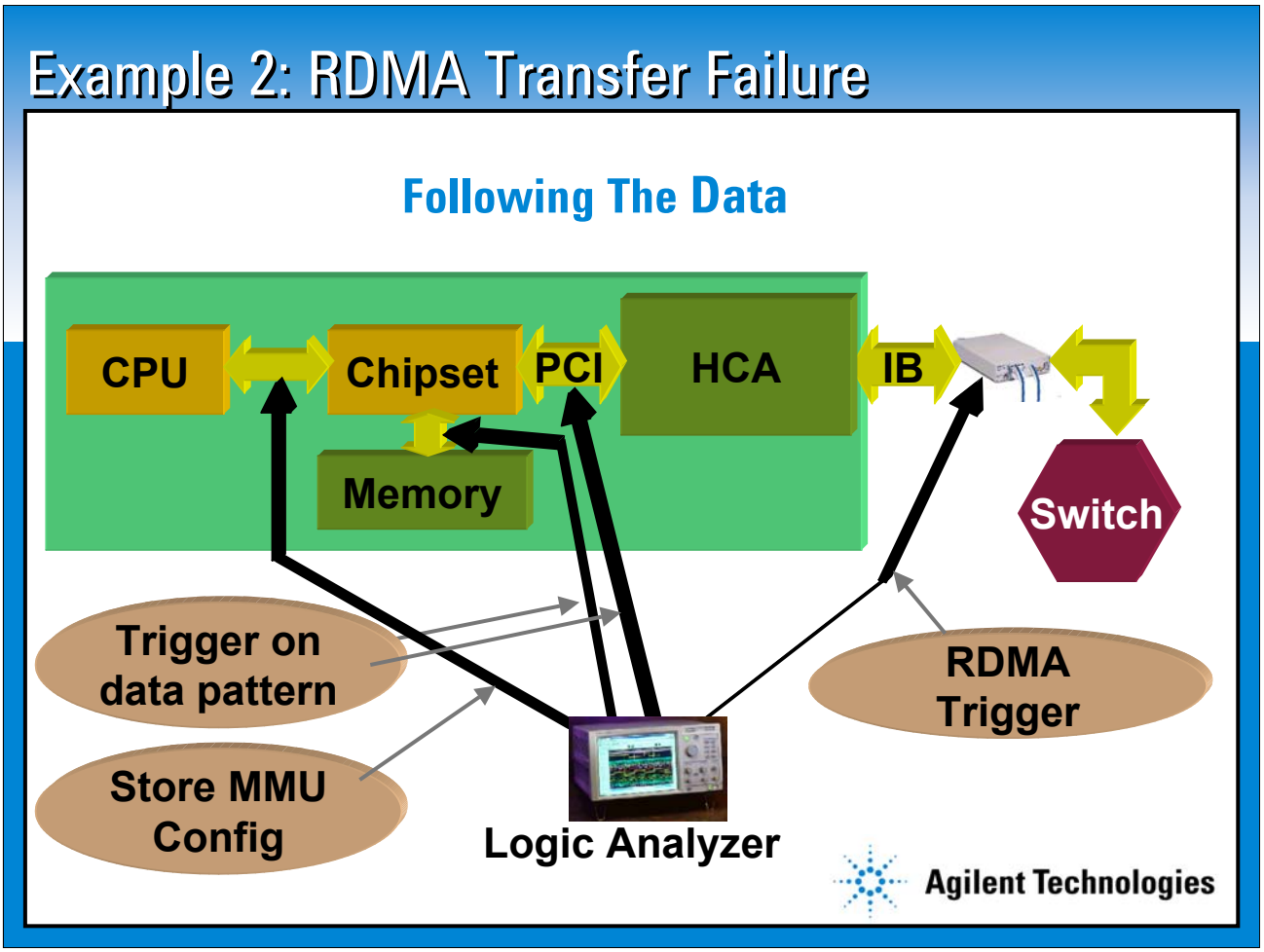


Example 2: RDMA Transfer Failure

IBTA Specifications

Again, here's the fine print. Basically, what this is saying is that an interesting thing about InfiniBand is that all DMA's occur in the virtual address space. DMA's aren't specified in terms of physical addresses as traditional DMA transfers, say on a PCI bus might be. So, this requirement here is basically saying that when you're doing a DMA, you need to give the destination a key that says here is where I want this DMA transfer to end up and this part of the spec is basically saying that that key refers to a virtual address.

Basically the way this is done is you request permission to access a remote area of memory in the virtual address base and the far side of the channel then, if it sends you back a key to access that part, it has granted your request. You use that key whenever you make a DMA transfer to say this is where I want it to go.



Example 2: RDMA Transfer Failure

Following the Data

One way to really debug a problem like this is to use the old tried and true mechanism of signal tracing. This is a different strategy than was used in the previous example. In the previous example we were looking at a bunch of different states of the system, putting them all together to make an inference about something that was going on inside the HCA.

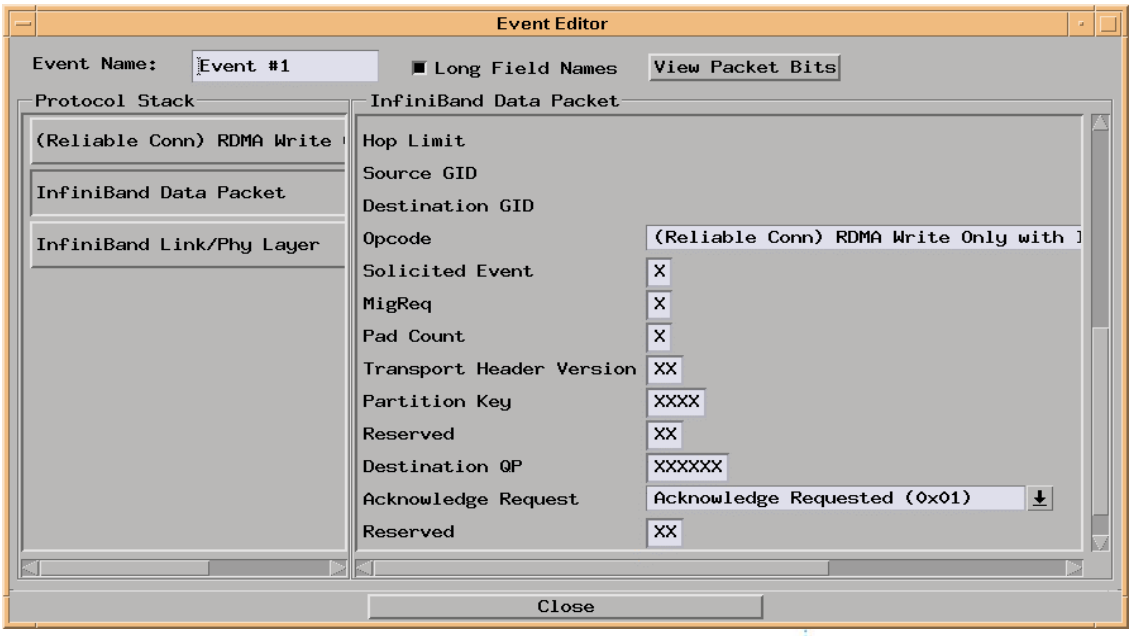
Signal tracing is actually a more straightforward approach where you know that you're starting out with a DMA transfer, and for some reason it's not showing up where you want. The easiest way to debug a problem like this actually is just to follow the data through the system. Like moving your scope probe from point to point through the circuit. Here we'll set up the analyzer to look at the InfiniBand traffic and trigger on the RDMA operation, we want to follow the data as it goes through the PCI bus, into the memory system; and, also take a look at the CPU to Chipset communication because that's probably how the MMU's get configured and programmed. Remember, InfiniBand DMA transfers are in the virtual memory space so the MMU's involved.

If you have a traffic generator, which is not show here, you can generate the RDMA request, and tell the traffic generator to use a very specific data pattern in the DMA buffer. Something that makes it easy to follow that signature as it flows through the system.

Trigger the InfiniBand logic analyzer on the DMA request and trigger the PCI and Memory analyzers on the data pattern. We don't know if it's making it out to the PCI bus or the memory bus, but if it is, we'll find it and trigger. So you can see where it went, and when it went. We'll also look at the CPU bus so we can get some insight into how the MMU is programmed.

Example 2: RDMA Transfer Failure

RDMA Packet Logic Analyzer Trigger

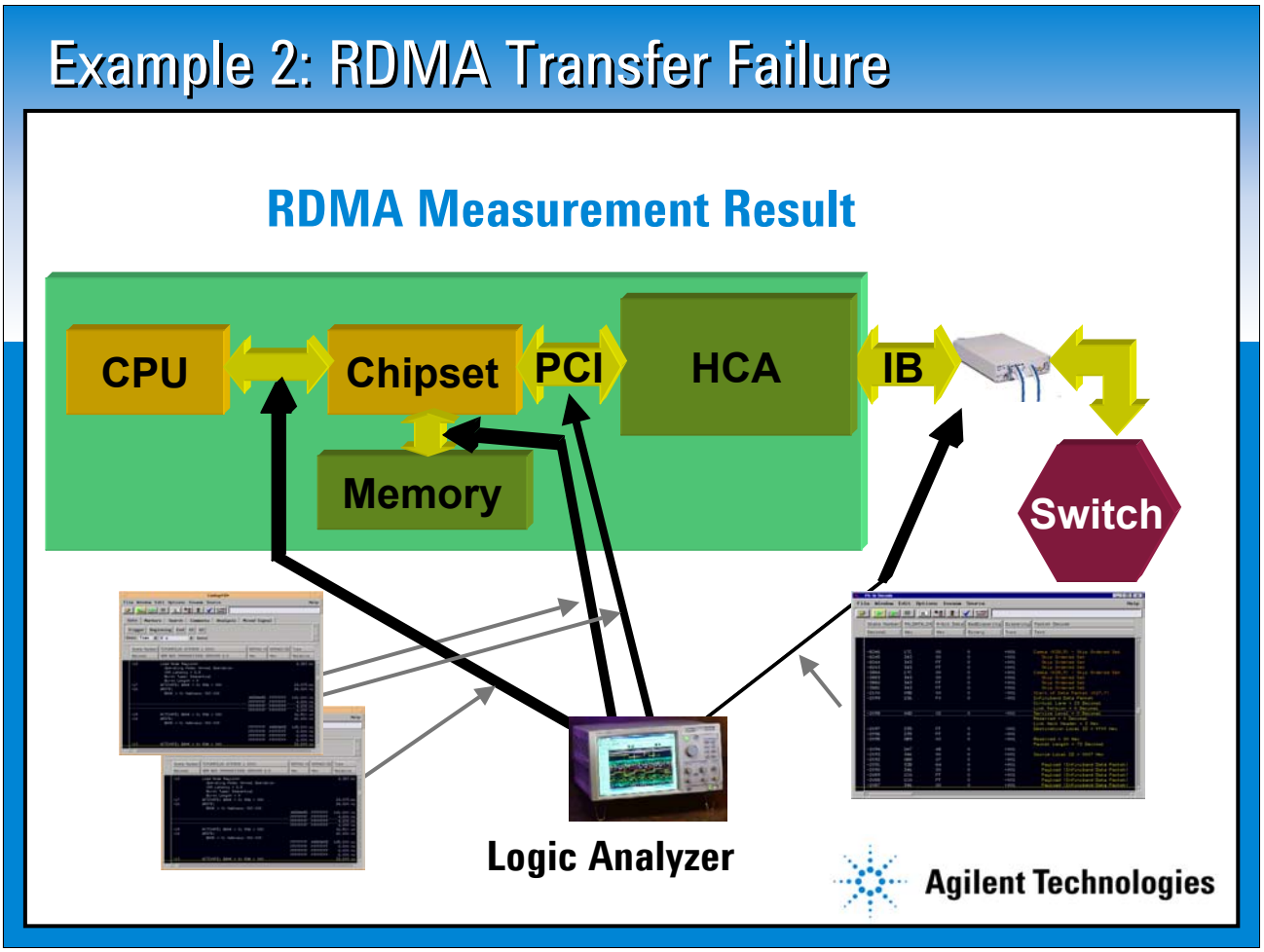


Agilent Technologies

Example 2: LinkPowerup Negotiation Failure

RDMA Packet Logic Analyzer Trigger

This is an example of how the logic analyzer can be set up to trigger on this RDMA packet. And here you simply see different fields in the InfiniBand data packet. This is the out-code field and this happens to be the specific kind of DMA request. So, again, setting up triggers at the network layer in the logic analyzers is a relatively straightforward operation.



Example 2: RDMA Transfer Failure

RDMA Measurement Results

Here's the result of the measurement. The InfiniBand data packet with the RDMA request, memory traffic, and PCI bus traffic. What this measurement shows is that the data buffer actually makes it all the way through into memory, but when you look at the memory addresses that it goes into and compare it against how the MMU tables were set up, you find out that it was sent to the wrong physical memory location.

By making this measurement, you're able to prove that the data buffer actually showed up somewhere, and you're able to figure out where it really showed up because you triggered on the data pattern and you were able to look at the addresses that that data pattern was written into; you can then compare that to how the MMU is programmed.

You use that to help figure out a working theory that says, “why might it have gone to the wrong destination”. These measurements give you enough information that you can then follow through the rest of the troubleshooting process and trace the problem to root cause.

Effective Validation and Tool Use

Example 3: Switch Failure

- Symptom: Raw Packets not filtered properly
- Possible Cause: Virtual port 0 never got configuration command (was configuring the wrong port)
- CIWG Item: C18-16 V1 p818, 639

Agilent Technologies

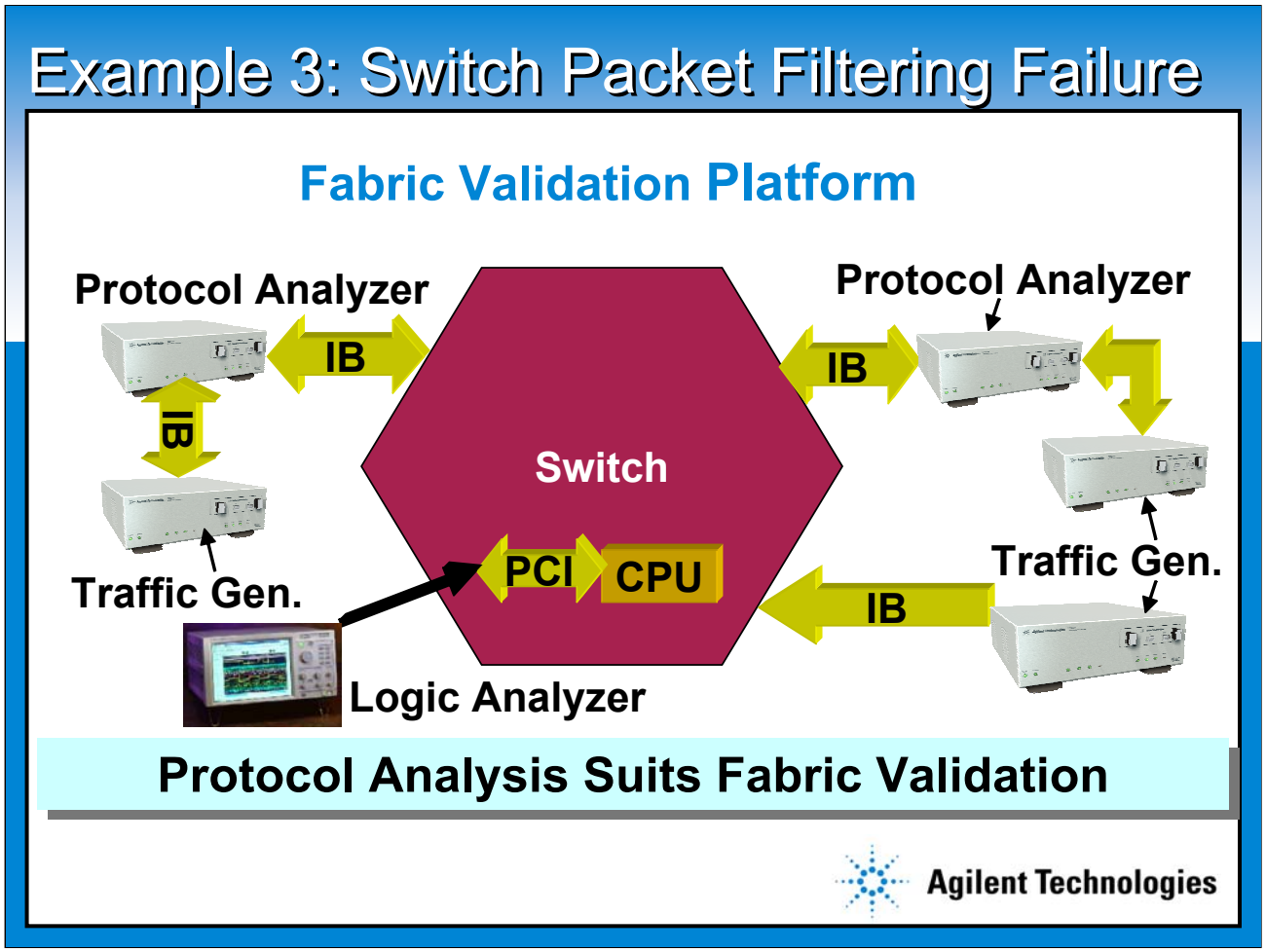
Switch Failure Example

A third example, moving again up the stack a little bit, this is a switch failure example.

Here are symptoms of raw packets not being filtered properly. InfiniBand switches have the property that you can configure them to filter out raw packets optionally. So, if a raw packet comes in on one port, you can tell a specific port just drop it in the bit bucket and not send it out.

The one possible cause for this is that perhaps virtual port 1 which is where all the configuration information for switches gets sent into, maybe the configuration command never went to virtual port 1, or maybe it ended up configuring the wrong port on the switch.

It turns out that there are specific points in the spec the standard you can refer to to describe this kind of behavior.

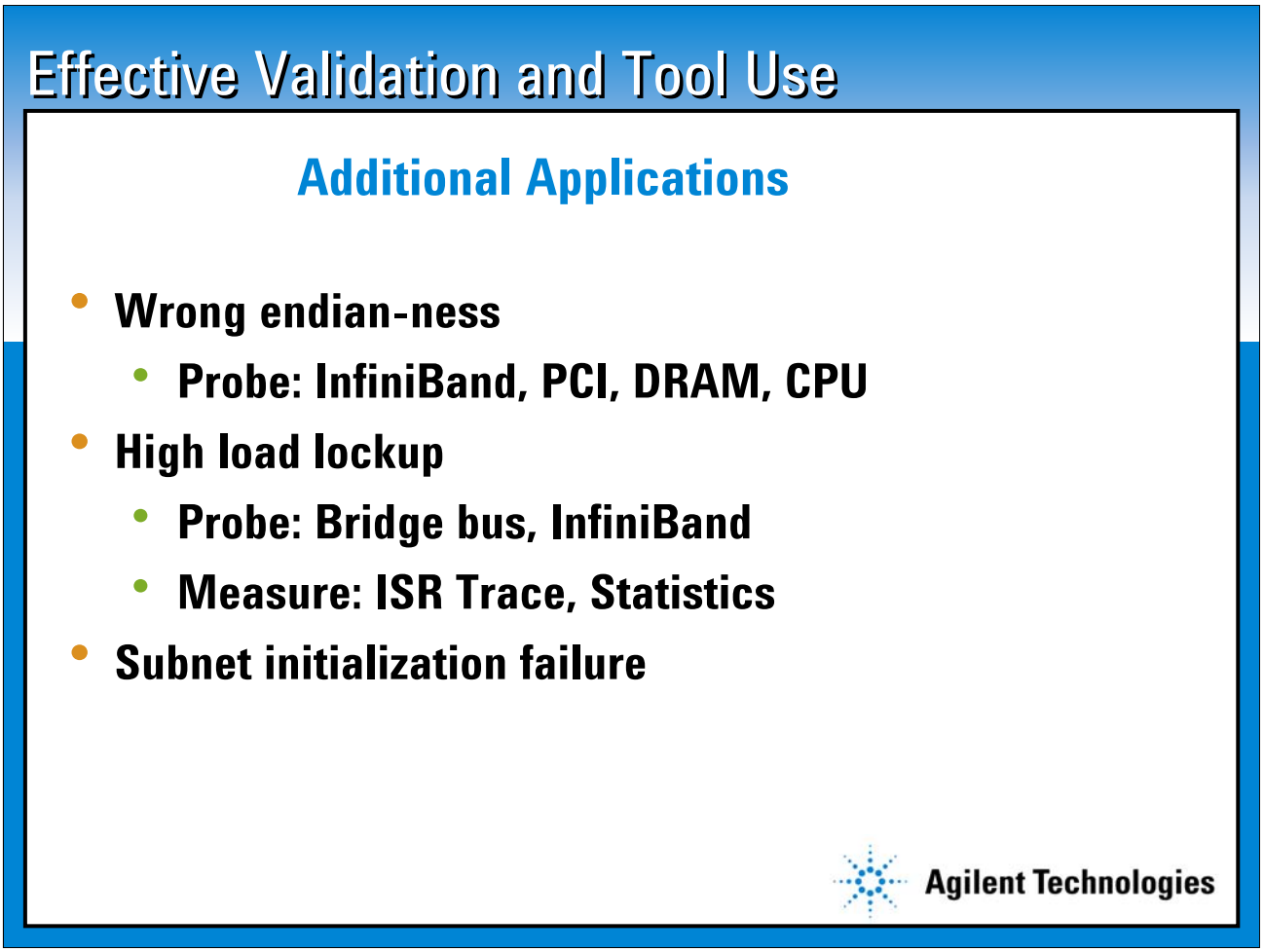


Fabric Interconnect Problem

This is a fabric interconnect problem; packets going through the fabric, packets coming in one part of the switch, going out the other part of the switch, they may also be going through routers and other devices. This is an example of a kind of a problem where you really want to set up a validation platform for fabrics. It is the area where the protocol analyzers and traffic generators really shine, because they're really optimized for looking at InfiniBand traffic at higher layers of the protocol. Here you have your switch, a protocol analyzer on various ports to the switch, with traffic generators generating very specific traffic to make it easy to reproduce the problem.

There might be a logic analyzer involved because often in the case of switches, virtual port 1 doesn't really exist physically. It's called virtual port 1 because semantically it's port 1 of the switch, but physically it may not actually have an InfiniBand connector connected to it. It may actually be implemented as a PCI or other kind of side channel from the CPU that's controlling the switch into the switch circuitry itself. You might need to have a coordinated measurement between the different protocol analyzers and logic analyzers so that you've probed enough parts of the system to really debug this.

With a setup like this, you can generate raw packets going in, find out if the raw packets are coming out, or find out which port they need to be coming out of and then also look inside the switch to try and figure out what is it about how the switch was programmed that could have caused this problem.



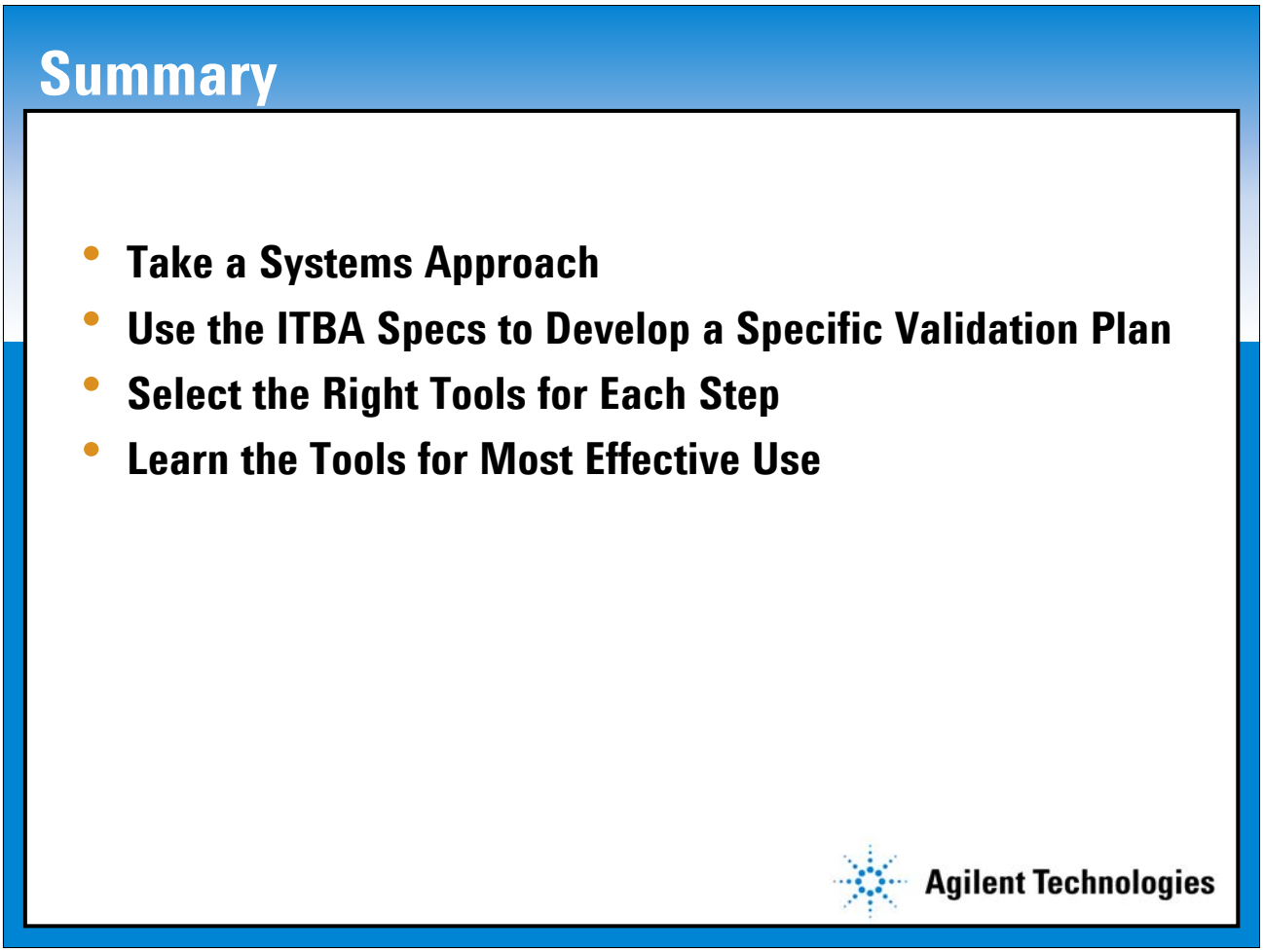
Effective Validation and Debug Tool Use

Additional Applications

There are a lot of different problems that you can run into when debugging and validating InfiniBand systems that require a system view and the ability to look at a lot of different parts of the system. Even something as simple as getting endian-ness wrong, you've got a big endian system on one side and a little endian system on another side and traffic is supposed to go between them and not get all messed up. Maybe it gets messed up in the switch or somewhere else. You probably want to look at the InfiniBand channel, maybe some PCI , DRAM channels, and the CPU bus. Even for something as small, you can often end up having to look in a lot of places.

Maybe you have a problem where you've got a lot of traffic generators hooked up to the fabric to stress the system and all of a sudden the whole fabric just seizes up. That's often a very hard problem to diagnose. You might want to look at mezzanine busses in your system, various InfiniBand channels within the system, or you might want to be making measurements on interrupt service routines. Maybe the lock up occurs because you have a stack overflow somewhere, or some switch just loses its mind because of congestion. You might want to use the protocol analyzer to make statistical measurements -- how much traffic is going through different ports so you can get a view, a sort of density of traffic flow through the system.

Another example might be subnet initialization failure-- trying to create a subnet for some reason, it doesn't happen like the subnet boundaries aren't honored. Protocol analyzers give you really good insight into that. So, there are a lot of problems like this that really demand a system-level view that means you're looking at multiple busses, multiple channels, various points in time, time correlated --

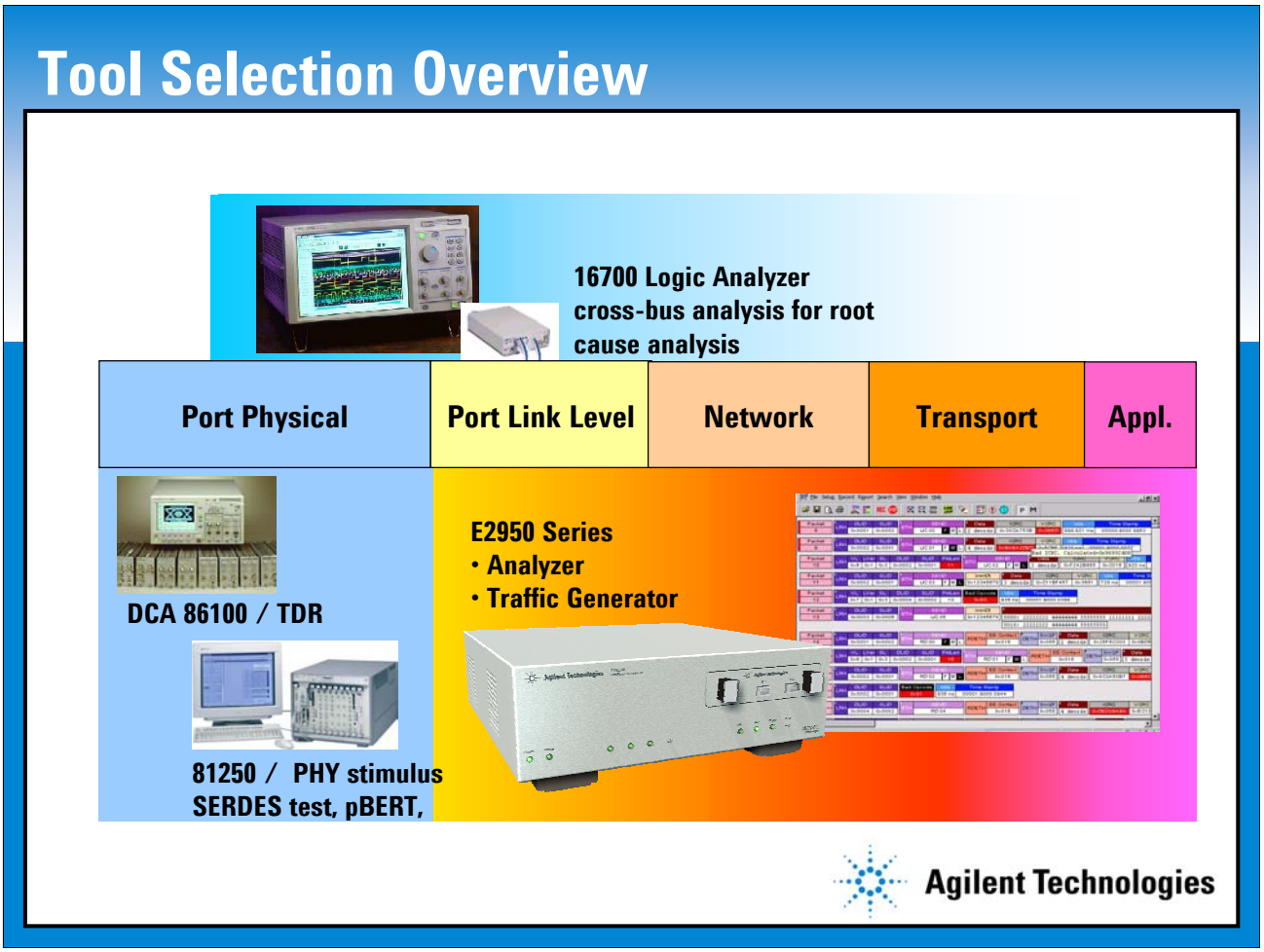


Summary

When you're debugging and validating InfiniBand based systems, you really need to take a systems approach. In general, you're going to have to look at a lot of places in the system and correlate activity in one part to activity in the other parts. Use the specification to help develop your validation plan. The specification outlines all the requirements, and it also tries to organize the requirements into these checklists. You can use the checklist to help organize your validation plan, you can use the details in the spec to produce focused test descriptions for each part of the spec.

Select the right tools, know that the protocol analyzers are really strong at higher level protocol analysis and that the logic analyzers are strong at cross bus analysis, know when to use a traffic generator, when to use real world stimulus, and then learn how to use the tools effectively--understand their strengths.

Because if you can use them effectively, if you understand really how they work and how they can work together, then a lot of these really tough system debug problems can turn out to be fairly straightforward to trace to root cause and fix.

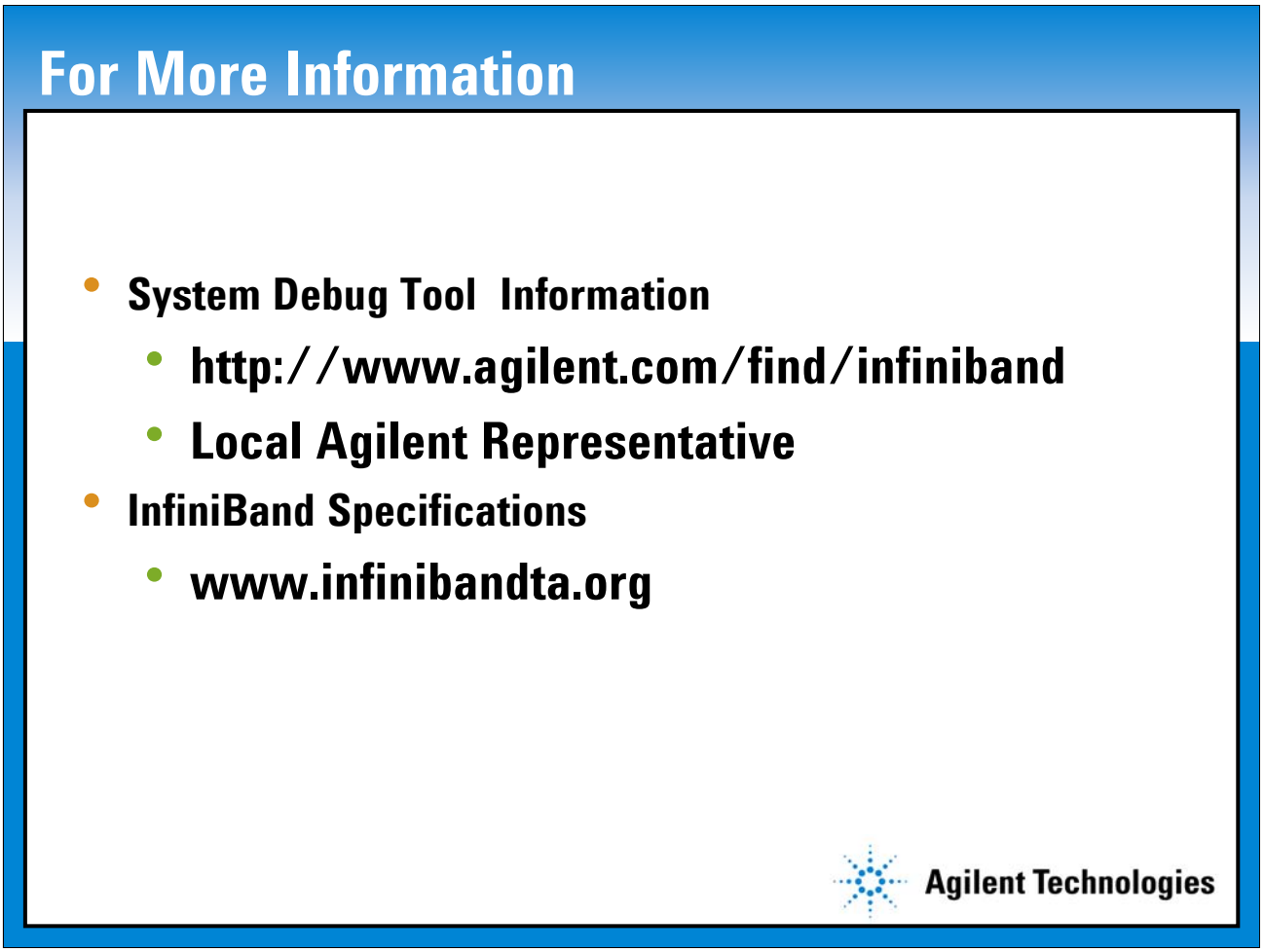


Tool Selection

These are examples of tools that can help you at various levels of the protocol; at the physical layer, TDR, scopes, Bit Error Rate testers, networking analyzers, spectrum analyzers provide insight.

The protocol analyzer works at the 10B and goes all the way up as high as possible.

Logic analyzers straddle the middle, they start working at 10B and then move up into the networking and transport layers. The strength of the logic analyzer is in the ability to perform cross bus analysis. As you move higher up you really want to transition into protocol analyzers as the tool of choice.



For More Information

For more information of you can go to the Agilent website, or call our salesmen. The InfiniBand Trade Association has the spec. It's only 1400 pages long.