# InfiniBand System-Level Debugging

Application Note 1382-1

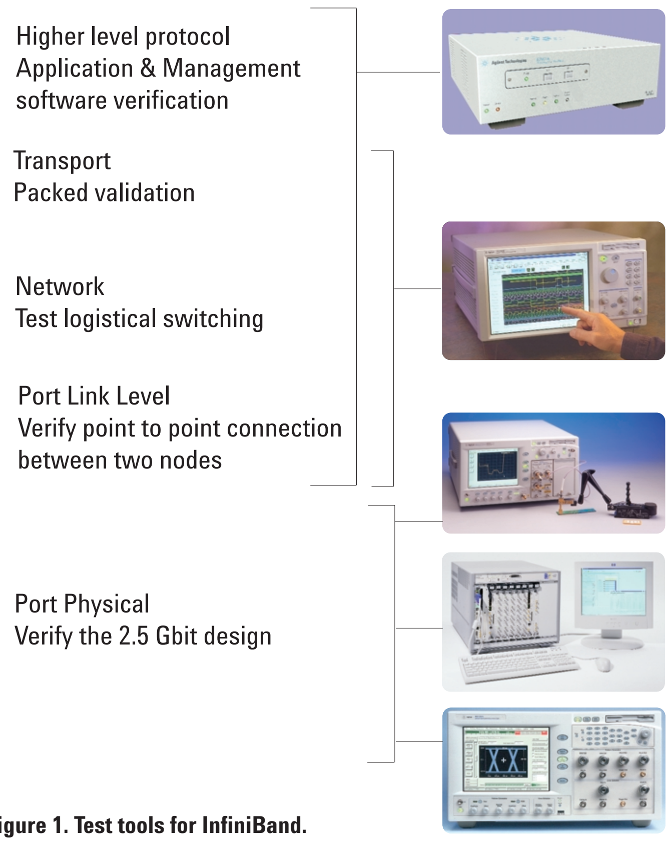
## Who Should Read this Application Note?

This application note is written for R & D engineers developing InfiniBand components, as well as InfiniBand system designers and integrators. It covers key concepts underlying system-level debug and validation of InfiniBand systems. Specific examples using logic analyzers and protocol analyzers to debug these systems are presented.

A huge range of technologies is required to support InfiniBand from the lowest physical layers through the higher layers of its protocol stack. For physical-layer debug and validation you should use tools such as wide-bandwidth oscilloscopes, Bit Error Rate Testers (BERTs), vector network analyzers (VNAs), and Time Domain Reflectometers (TDRs).

This application note focuses on "functional" validation of InfiniBand systems, for which logic analyzers and protocol analyzers are the most useful tools. (Figure 1).

**Table of Contents**



Higher level protocol
Application & Management software verification
Transport
Packed validation
Network
Test logistical switching
Port Link Level
Verify point to point connection between two nodes
Port Physical
Verify the 2.5 Gbit design
E2950 Series
• E2951A protocol analyzer
• E2953A traffic generator
16700 Series Logic Analysis System
• N4206A software tool
• N4207A logic analysis probe
86100A Infiniium DCA
81250 ParBERT
86130A BitAlyzer

**Figure 1. Test tools for InfiniBand.**

Agilent Technologies

# Introduction

## The Benefits of InfiniBand

InfiniBand is an architecture and specification for communication between processors and I/O devices that promises greater bandwidth and almost unlimited expandability in tomorrow's computer systems. This architecture also promises increased reliability, better sharing of data between clustered processors, and built-in security.

InfiniBand interconnect technology uses a switched-fabric, point-to-point architecture to deliver new levels of scalability, availability, and performance. Scalability is delivered in part by its switched network interconnect, its automatic reconfiguration of the network upon addition or removal of devices.

To improve availability, InfiniBand technology utilizes point-to-point interconnects, allows for redundant paths between devices with subsequent failover capability, and includes integrated error detection and correction. Data is transferred between InfiniBand fabric elements via copper cables or optical fiber. Beyond the raw data transfer rates, system-level performance is improved by the concept of intelligent channels, with computing intelligence distributed into the I/O system to offload the task of controlling I/O from processors.

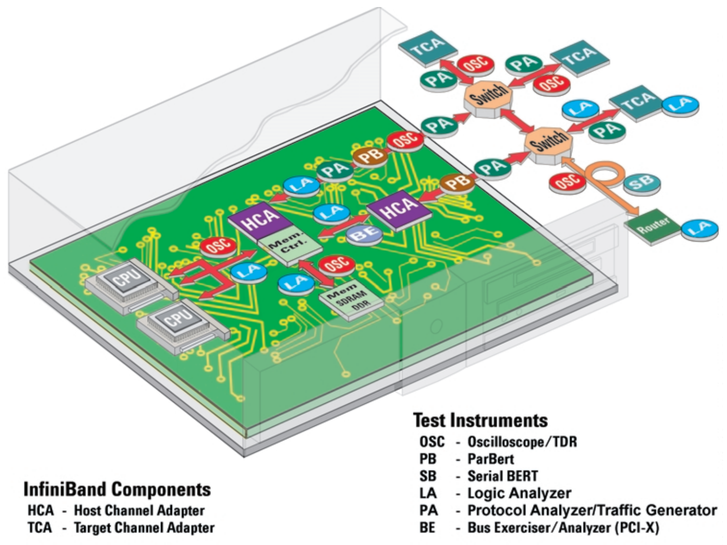


**Figure 2. A typical InfiniBand system showing interconnects to various components and appropriate test equipment.**

## An InfiniBand System

Figure 2 shows a diagram of a typical InfiniBand system. The lid of the box is removed to show the CPU, chip set, and an InfiniBand HCA (host channel adapter). Outside the box is the InfiniBand fabric. The arrows represent the different links that connect the various components. The dots represent the different types of test equipment you might need to connect to those links in order to debug the complete system, as follows:

- OSC = Oscilloscope
- PA = Protocol Analyzer/Traffic Generator
- LA = Logic Analyzer
- BE = Bit Error Rate Tester

# System-Level Debugging

## Key Requirements for System-Level Debugging

There are three main concepts to keep in mind to do effective system-level validation: broad visibility, cross-correlation, and stimulus diversity.

It is usually essential to have broad visibility into your system's behavior. If you are fortunate enough to already know the exact location of a particular problem, such as the PCI bus or a particular channel or component of the InfiniBand fabric, then you may be able to avoid cross-bus system validation and use a targeted bus tool. (This could be a protocol analyzer and/or bus exerciser for the PCI bus, or a protocol analyzer and/or traffic generator for the InfiniBand bus). However, if you are attempting to track down system-level problems, you must have broad visibility across a range of buses and links within the system. Perhaps problems detected on the InfiniBand fabric may actually be generated from incorrect traffic on the PCI bus. You may also need to observe behavior in your system on a picosecond time scale, which is especially true for InfiniBand.

Once you have obtained measurements from different parts of the system, you need to be able to correlate all this activity. Cause and effect for a particular system-level problem are not usually isolated to just one part of the system; they can often span the entire system. An event may be detected in one area, but the root cause of the problem may actually be somewhere else. So you need to be able to correlate activity in one part of the system with what is going on in other parts of the system.

## Cross-Correlation

Cross-correlation takes two forms. One is spatial correlation: connecting what is happening at one particular point in the system with something else that is happening at another physical point in the system at the same time. System-level debug requires you to correlate information across two or more different points in the system so you can see the simultaneous behavior at all of those points.

The other form of correlation is temporal correlation: how something that happens in the system at one particular moment in time relates to what happens at another moment. A good example of this would be matching a request packet with its corresponding response.

The final requirement for system-level validation is a variety of stimuli. You need to test your system in a real-world environment because it is impossible to model the real world perfectly or to cover all possible conditions and corner cases your system is required to handle.

## Real World

However, the real world can be very hard to control. Once you find a problem or suspect that there are particular areas that require extra attention, you then need to provide a controlled stimulus to your system at specific points. By injecting controlled stimulus into your system, you can set up a series of tightly controlled tests allowing you to reproduce specific types of problems. Setting up a series of regression tests that exercises the whole system is a good practice. These regression tests should include specific test cases each with a very specific stimulus.

Only with a well-planned variety of stimuli can you cover all cases and validate your system. All of these testing and validation requirements create a need for highly capable tools. You need to be able to work across the entire system, correlate activities from picoseconds to packets, handle the billions of events that occur in the real world, and also provide very focused, directed, and controllable stimulus to your system.

### Tool Selection for InfiniBand System Validation

As you prepare to perform system-level validation, you must select the appropriate tools for your test bench. Three basic types of test tools are most commonly needed for effective debugging and validation of InfiniBand systems at the higher protocol layers. You should be familiar with what each of these tools is optimized for and how they complement other available tools.

The first tool is a protocol analyzer (PA) for InfiniBand, the second is a logic analyzer (LA) that has InfiniBand support, and the third is a traffic generator to create controlled InfiniBand traffic for system validation. Each of these test instruments has different strengths.

A protocol analyzer focuses on providing a comprehensive view of information and data transfer on just the InfiniBand link. This instrument is optimized for protocol measurements, which tend to be hierarchical in nature. For this reason, it provides a hierarchical view of the protocol. For instance, you can click on specific packets or pieces of packets to expand the data for greater detail. The protocol analyzer provides a hierarchical, browser type view of the InfiniBand protocol.

A logic analyzer is optimized for providing cross-bus or multi-bus correlation of information, and provides a consistent way of displaying data no matter which bus is being monitored. It is most useful for looking at levels of the protocol up to the transport layer, including subnet management datagrams, and typically does not display information up to the application layer. Setting up a logic analyzer and looking at measurement results for an InfiniBand system is very similar to the way it has traditionally been accomplished with microprocessor buses and other kinds of buses such as PCI.

The logic analyzer is also a very powerful tool for being able to “reach” inside ASICs that you can’t probe directly. It can obtain measurements across the entire system and allow you to correlate information taken in one part of the system with information taken in another part. This allows you to see how, in time, events in one part of the system affect events in another part of the system, and thus enables you to infer the internal state of your system.

### Validation Plan

Once you have selected your tools, you then need to create a validation plan for your InfiniBand-based system. The published InfiniBand standard (www.infinibandta.org) with its specifications and checklists provides a roadmap for putting the validation plan together. The line items of the checklist marked with a "C" are required for compliance with the InfiniBand specifications; the lines items marked with an "O" are optional.

You can use the specification to develop a structure and put together a checklist for the test plan. Behind each of the checklist items is a more detailed explanation of what that item means. Reading the fine print of the specification will help you define a specific test case and a specific test procedure to validate your system against that particular requirement of the InfiniBand specification.

### Validation Platform

The last thing to do prior to system validation is to put together a validation platform. Figure 3 shows a typical validation platform, which will be used for the examples described below. One part of this platform is a server system with a CPU, chip set, and InfiniBand host channel adapter (HCA). It is representative of the first InfiniBand implementations available.

The other part of the platform contains something for the server to talk to: a switch, another HCA, or a real-world environment for validation purposes. If you need a deterministic stimulus, you can use a traffic generator for InfiniBand and have it talk directly to the HCA.

Once you assemble the validation platform, have a test plan, and have the proper tools, then it's time to turn the power on, see if the system works, and then start working through the checklists. The following section describes three examples for debugging InfiniBand systems with these tools.

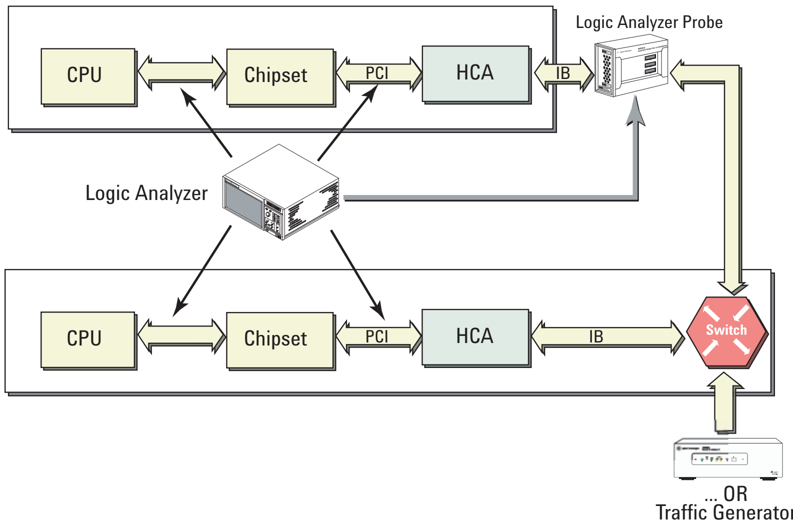


**Figure 3. The validation platform used for the examples.**

### Case Study 1: Link Power-Up Negotiation Failure

The first example of how to use these tools to debug the system is link power-up. This is the first task that you would typically perform after running the "smoke test".

You plug all the cables together and then attempt to power up the system. You expect that the InfiniBand channel will come up, initialize itself, and the link will be up and ready for work. Unfortunately, you discover that the link hangs after you power up the system. All you know at this point is that your system is stuck in the link-down status. Your problem is to figure out why this happens and how to prevent it from happening.

Initially you have little insight into the root causes of the problem. Following a troubleshooting process can help you systematically uncover the source of the problem and develop a way to solve it. The steps for debugging a problem such as this are:

1) Focus on the symptom and obtain more information about the failure condition.
2) Develop a working theory as to why the failure occurred.
3) Confirm or disprove your theory by making additional measurements.
4) Develop and implement a fix for the problem.
5) Confirm that the fix solves the problem.

### Focus on the Symptom

These steps will be described further for the link power-up negotiation failure example. The first step is to focus on the symptom in an attempt to get additional information about why the system failed. In this example, the only thing that you initially know is that the link was down when it should have been up.

The first step is to set up the logic analyzer to look at the PCI bus during link power-up. You should look at traffic going to and from the HCA and simultaneously monitor the InfiniBand link with the same logic analyzer for time-correlated measurements of the PCI bus and the InfiniBand link (figure 4).

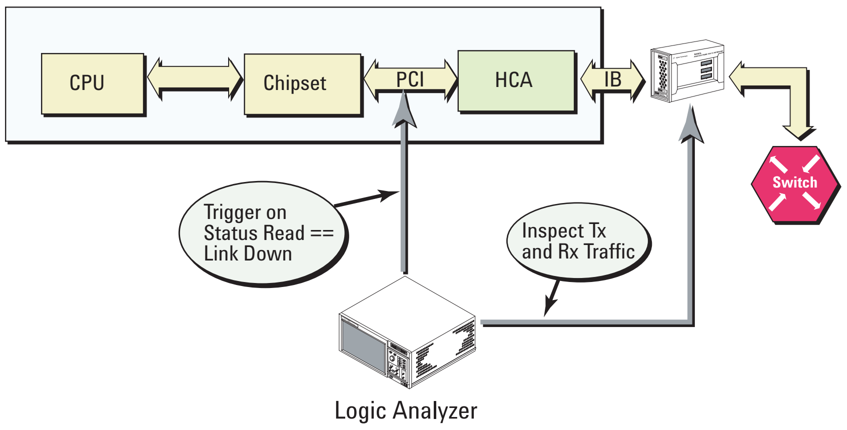
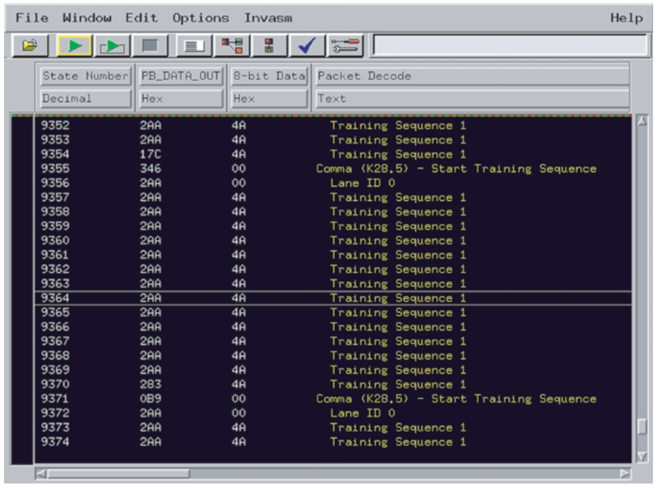

CPU
Chipset
PCI
HCA
IB
Switch
Trigger on Status Read == Link Down
Inspect Tx and Rx Traffic
Logic Analyzer

**Figure 4. Using a logic analyzer to investigate the link power-up negotiation failure.**

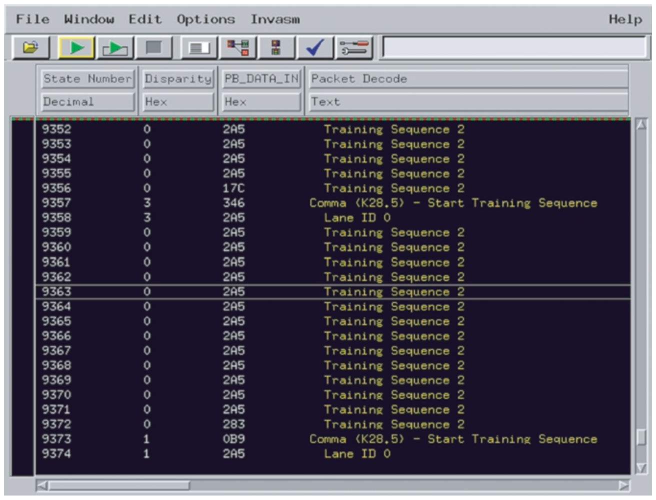
At some point the HCA will return status information to the CPU indicating that the link is down. Knowing this, you can set up the logic analyzer to trigger on this “link down” status and then look at the time-correlated InfiniBand traffic to attempt to discover something that might provide a clue about the problem.

The results of this measurement are shown in figure 5. The logic analyzer display window on the left shows InfiniBand traffic going out from the HCA to the far-end device. The window on the right shows traffic coming back from the far-end device to the HCA being measured.

These displays show that the system is still sending “Training Sequence 1s” (TS1s) while the far-end device is sending “Training Sequence 2s” (TS2s). When an HCA has completed its configuration and is ready to actually do something using the InfiniBand protocol, it will return TS2s, so you know at this point that the far-end device has completed configuration. However, your system is still sending TS1s, indicating that it hasn’t completed configuration yet and is locked in a link-down status. This information shows that the channel is functioning on an electrical level, but there is some reason why the system doesn’t complete its configuration even though the far-end device does complete the configuration process.



```
State Number  PB_DATA_OUT  8-bit Data  Packet Decode
Decimal       Hex          Hex         Text
9352          2AA          4A            Training Sequence 1
9353          2AA          4A            Training Sequence 1
9354          17C          4A            Training Sequence 1
9355          346          00          Comma (K28.5) - Start Training Sequence
9356          2AA          00            Lane ID 0
9357          2AA          4A            Training Sequence 1
9358          2AA          4A            Training Sequence 1
9359          2AA          4A            Training Sequence 1
9360          2AA          4A            Training Sequence 1
9361          2AA          4A            Training Sequence 1
9362          2AA          4A            Training Sequence 1
9363          2AA          4A            Training Sequence 1
9364          2AA          4A            Training Sequence 1
9365          2AA          4A            Training Sequence 1
9366          2AA          4A            Training Sequence 1
9367          2AA          4A            Training Sequence 1
9368          2AA          4A            Training Sequence 1
9369          2AA          4A            Training Sequence 1
9370          283          4A            Training Sequence 1
9371          0B9          00          Comma (K28.5) - Start Training Sequence
9372          2AA          00            Lane ID 0
9373          2AA          4A            Training Sequence 1
9374          2AA          4A            Training Sequence 1
```



```
State Number  Disparity  PB_DATA_IN  Packet Decode
Decimal       Hex        Hex         Text
9352          0          2A5           Training Sequence 2
9353          0          2A5           Training Sequence 2
9354          0          2A5           Training Sequence 2
9355          0          2A5           Training Sequence 2
9356          0          17C           Training Sequence 2
9357          3          346         Comma (K28.5) - Start Training Sequence
9358          3          2A5           Lane ID 0
9359          0          2A5           Training Sequence 2
9360          0          2A5           Training Sequence 2
9361          0          2A5           Training Sequence 2
9362          0          2A5           Training Sequence 2
9363          0          2A5           Training Sequence 2
9364          0          2A5           Training Sequence 2
9365          0          2A5           Training Sequence 2
9366          0          2A5           Training Sequence 2
9367          0          2A5           Training Sequence 2
9368          0          2A5           Training Sequence 2
9369          0          2A5           Training Sequence 2
9370          0          2A5           Training Sequence 2
9371          0          2A5           Training Sequence 2
9372          0          283           Training Sequence 2
9373          1          0B9         Comma (K28.5) - Start Training Sequence
9374          1          2A5           Lane ID 0
```

**Figure 5. Measurement results showing that the system is still sending TS1s while the far-end device is sending TS2s.**

### Develop a Working Theory

To determine why this is happening, you must make additional measurements to gain sufficient insight to put together a theory. You should now focus on all status reads going from the HCA to the CPU. (The HCA sends an interrupt to the CPU giving its current status as it goes through the link power-up negotiation process.) You should still trigger the logic analyzer's acquisition on the link-down status, but instead of looking at all PCI traffic, you should now focus on status reads. You should continue to monitor the InfiniBand traffic to see how the status of the HCA over time correlates with the InfiniBand traffic over this same period of time.

For this measurement, the display of the InfiniBand transmit link shows that the HCA continues to transmit TS1s during the measurement time period. TS1s were coming into the HCA from the InfiniBand receive link at the beginning of the time period, but by the end, TS2s were being received. The trace of the PCI bus shows that the HCA was sending the CPU a time-out status.

This leads to a working theory: If TS2s start coming in while the system is still in a receiver-configuration state, something gets confused and the HCA never leaves link configuration. Figure 6, taken from the published InfiniBand standard, shows graphically what the working theory predicts. First, the HCA goes into the receiver-configuration state and sends out TS1s. When the link is established, it should then start sending out TS2s. The working theory is that for some reason the switch to receiving TS2s instead of TS1s confuses the HCA, and after 150 ms it times out.

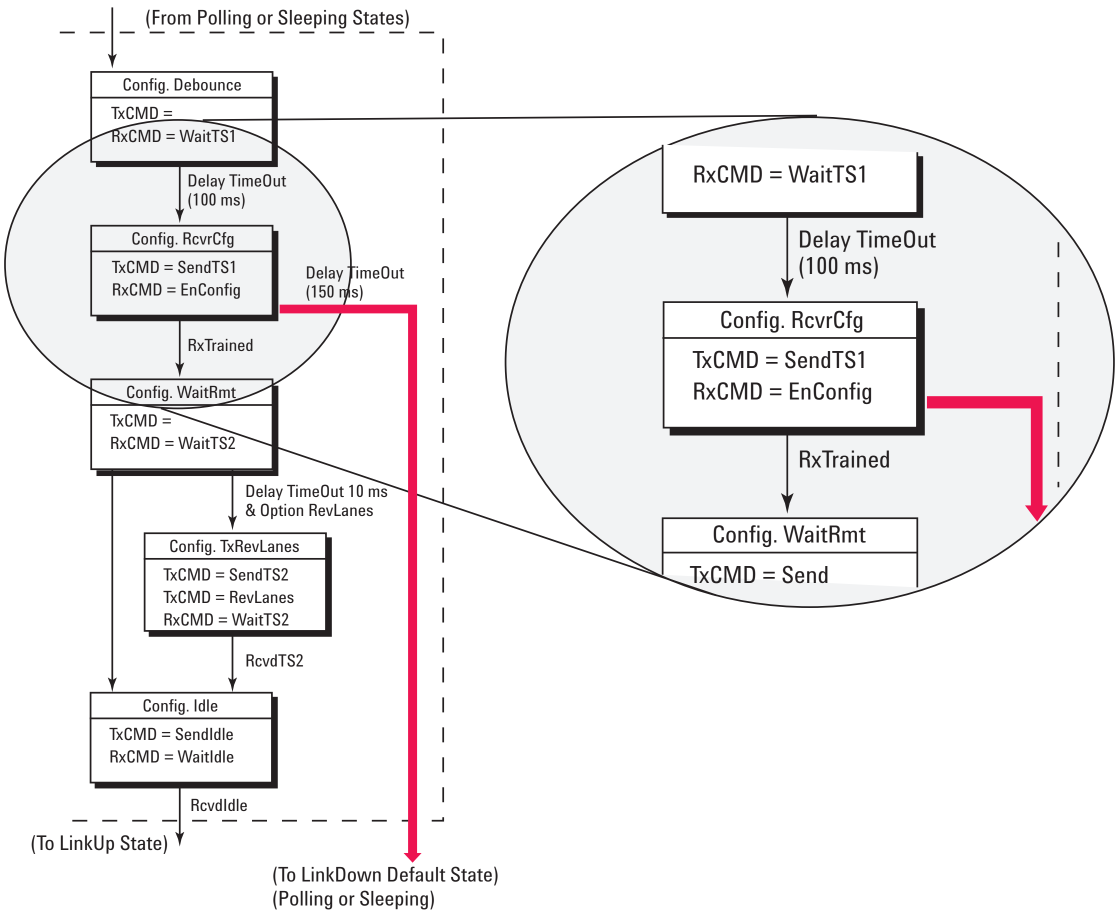


**Figure 6. The InfiniBand standard showing where the system may be getting stuck in the link configuration state.**

### Confirm Your Theory

You would like to know whether the system comes in at the top of this state machine and then immediately goes to the timeout, or goes through some other path. (There are three other time-out conditions that can occur, each 2 ms long.) Knowing this will help you figure out where in the state machine the problem is located.

To determine this, you need to set up four logic analyzer measurements, each of which requires monitoring activity in several parts of the system at the same time. You need to set up a logic analyzer trigger condition that looks for a timeout greater than 150 ms while the system is receiving TS2s and still transmitting only TS1s. If the analyzer triggers, then you know that the bold path in figure 5 was the path that was followed. If the logic analyzer does not trigger, then a different path was followed.

To verify that none of the other paths was followed, you can set up the logic analyzer to trigger on one of the other three possible conditions. If the working theory is correct, none of them should cause the logic analyzer to trigger. But if one of them does, then you know something is wrong with the working theory and you need to look somewhere else.

To set up a measurement that triggers on the link power-up failure, you need to look at the receive and transmit traffic on the InfiniBand channel, and the PCI bus, so cross-bus triggering is essential. You need to look for the point in time when the system starts receiving TS2s instead of TS1s, so the logic analyzer is going to monitor the receive channel. When the system begins to receive TS2s, the logic analyzer will set a global flag, allowing every other analyzer to know that the transition from TS1s to TS2s has occurred.

When it detects the flag, the analyzer that is monitoring the PCI bus will start a timer and begin looking for timeout status. In other words, it is looking for a time-out status that occurs more than 150 ms after the switch from TS1s to TS2s. If this happens, the PCI bus analyzer will set a flag indicating that the 150-ms time out has occurred. In the meantime, another analyzer is looking at the InfiniBand transmit link. When it detects the second flag, it verifies that TS1s are still being transmitted. If so, then all three conditions are satisfied and the analyzer will trigger. This process is diagrammed in figure 7.

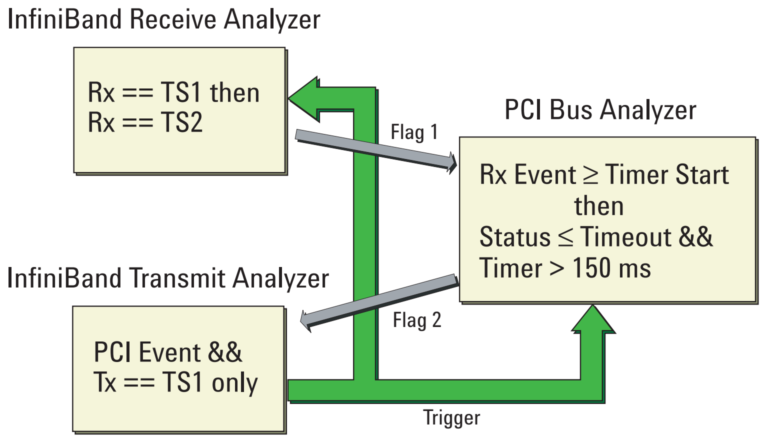
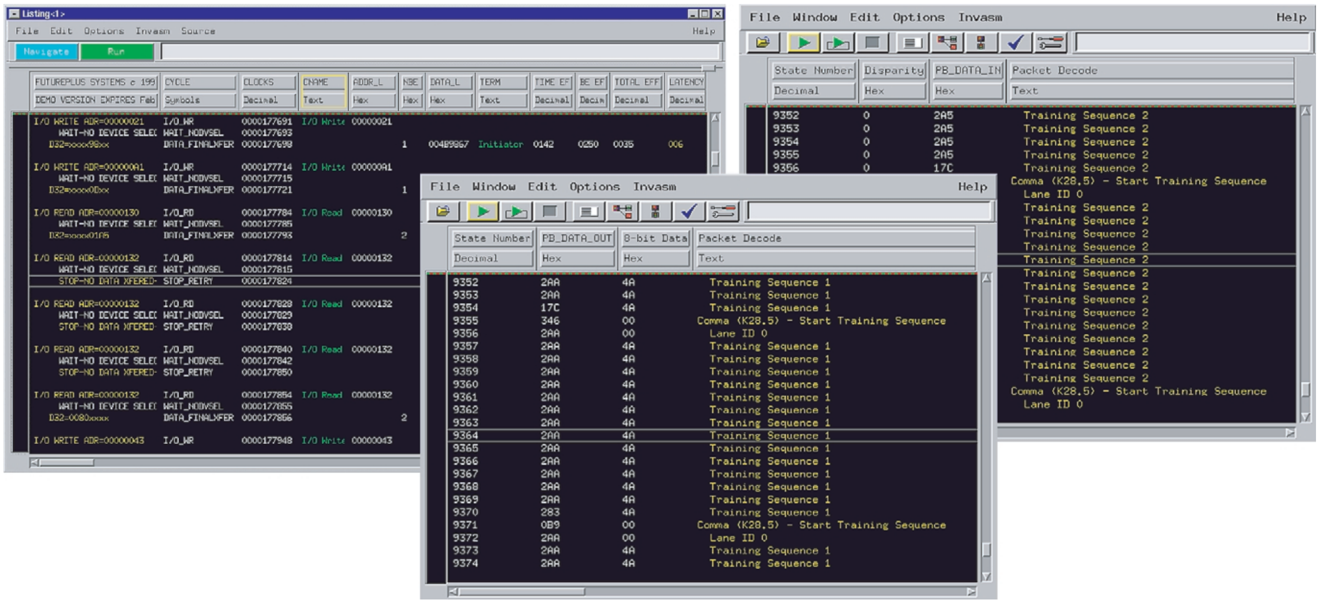


**Figure 7. Method to trigger on the link power-up negotiation failure.**

Figure 8 shows the logic analyzer output from all three locations (PCI bus, transmit channel, and receive channel) when the trigger is set up as described. You need to carefully examine the output and make sure that it is what is expected to support your working theory. By looking at this data, you can have confidence that the system is really behaving the way you expect it would, given its problem.

## Fix the Problem

The final steps are to fix the problem and confirm that the fix is correct. It often happens, especially when you are debugging ASICs, that in order to fix the problem you need to go back to simulation. You should set up an environment that reproduces the problem in simulation so that when you develop a fix, you can prove that the fix really works. You can close the loop by taking real-world data, adding it to the simulation environment, implementing your fix, and then running the simulation to confirm that the fix solves the problem.



Listing<1>
File Edit Options Invasm Source Help
Navigate
Run
File Window Edit Options Invasm Help

```
FUTUREPLUS SYSTEMS c 199  CYCLE            CLOCKS      CNAME      ADDR_L    NBE  DATA_L    TERM       TIME EF  BE EF  TOTAL EFF  LATENCY
DEMO VERSION EXPIRES Feb  Symbols          Decimal     Text       Hex       Hex  Hex       Text       Decimal  Decim  Decimal    Decimal
I/O WRITE ADR=00000021    I/O_WR           0000177691  I/O Write  00000021
    WAIT-NO DEVICE SELEC  WAIT_NODVSEL     0000177693
  D32=xxxx98xx            DATA_FINALXFER   0000177698                       1    004B9867  Initiator  0142     0250   0035       006

I/O WRITE ADR=000000A1    I/O_WR           0000177714  I/O Write  000000A1
    WAIT-NO DEVICE SELEC  WAIT_NODVSEL     0000177715
  D32=xxxx0Dxx            DATA_FINALXFER   0000177721                       1

I/O READ ADR=00000130     I/O_RD           0000177784  I/O Read   00000130
    WAIT-NO DEVICE SELEC  WAIT_NODVSEL     0000177785
  D32=xxxx01A6            DATA_FINALXFER   0000177793                       2

I/O READ ADR=00000132     I/O_RD           0000177814  I/O Read   00000132
    WAIT-NO DEVICE SELEC  WAIT_NODVSEL     0000177815
    STOP-NO DATA XFERED-  STOP_RETRY       0000177824

I/O READ ADR=00000132     I/O_RD           0000177828  I/O Read   00000132
    WAIT-NO DEVICE SELEC  WAIT_NODVSEL     0000177829
    STOP-NO DATA XFERED-  STOP_RETRY       0000177830

I/O READ ADR=00000132     I/O_RD           0000177840  I/O Read   00000132
    WAIT-NO DEVICE SELEC  WAIT_NODVSEL     0000177842
    STOP-NO DATA XFERED-  STOP_RETRY       0000177850

I/O READ ADR=00000132     I/O_RD           0000177854  I/O Read   00000132
    WAIT-NO DEVICE SELEC  WAIT_NODVSEL     0000177855
  D32=0080xxxx            DATA_FINALXFER   0000177856                       2

I/O WRITE ADR=00000043    I/O_WR           0000177948  I/O Write  00000043
```

```
State Number  Disparity  PB_DATA_IN  Packet Decode
Decimal       Hex        Hex         Text
9352          0          2A5            Training Sequence 2
9353          0          2A5            Training Sequence 2
9354          0          2A5            Training Sequence 2
9355          0          2A5            Training Sequence 2
9356          0          17C            Training Sequence 2
                                     Comma (K28.5) - Start Training Sequence
                                        Lane ID 0
                                        Training Sequence 2
                                        Training Sequence 2
                                        Training Sequence 2
                                        Training Sequence 2
                                        Training Sequence 2
                                        Training Sequence 2
                                        Training Sequence 2
                                        Training Sequence 2
                                        Training Sequence 2
                                        Training Sequence 2
                                        Training Sequence 2
                                        Training Sequence 2
                                        Training Sequence 2
                                     Comma (K28.5) - Start Training Sequence
                                        Lane ID 0
```

```
State Number  PB_DATA_OUT  8-bit Data  Packet Decode
Decimal       Hex          Hex         Text
9352          2AA          4A             Training Sequence 1
9353          2AA          4A             Training Sequence 1
9354          17C          4A             Training Sequence 1
9355          346          00          Comma (K28.5) - Start Training Sequence
9356          2AA          00             Lane ID 0
9357          2AA          4A             Training Sequence 1
9358          2AA          4A             Training Sequence 1
9359          2AA          4A             Training Sequence 1
9360          2AA          4A             Training Sequence 1
9361          2AA          4A             Training Sequence 1
9362          2AA          4A             Training Sequence 1
9363          2AA          4A             Training Sequence 1
9364          2AA          4A             Training Sequence 1
9365          2AA          4A             Training Sequence 1
9366          2AA          4A             Training Sequence 1
9367          2AA          4A             Training Sequence 1
9368          2AA          4A             Training Sequence 1
9369          2AA          4A             Training Sequence 1
9370          283          4A             Training Sequence 1
9371          0B9          00          Comma (K28.5) - Start Training Sequence
9372          2AA          00             Lane ID 0
9373          2AA          4A             Training Sequence 1
9374          2AA          4A             Training Sequence 1
```

**Figure 8. The logic analyzer output from the PCI bus (a), transmit channel (b), and receive channel (c).**

### Case Study 2: RDMA Failure

A second example, further up the stack of protocol layers, is the failure of a remote direct memory access (RDMA) transfer. Again, you start out with very little information. An RDMA transfer is set up, but for some reason the data never arrives at its destination. When you run into a problem like this, it's a good idea to go back to the specification. In this case, there are two parts of the specification that govern this type of RDMA transfer: how to do the Write requests and how to set up the memory to receive the requests.

The fine print specifies that for InfiniBand, all DMAs occur in the virtual address space; they aren't specified in terms of physical addresses as are traditional DMA transfers. When you are doing a DMA, you need to give the destination a key that says where you want this DMA transfer to go and that the key refers to a virtual address.

To accomplish this, you request permission to access a remote area of memory in the virtual address space at the far side of the channel. If it returns a key to access that part, it has granted your request. You use that key whenever you make a DMA transfer to specify where you want it to go.

One way to debug a problem like this RDMA failure 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 which different parts of the system were examined to make an inference about what was going on inside the HCA.

Signal tracing is a more straightforward approach when you are starting out with a DMA transfer that does not show up where it is supposed to. The easiest way to debug a problem like this is to follow the data through the system, similar to moving your scope probe from point to point through a circuit. Figure 9 shows how to set up the logic analyzer to look at the InfiniBand traffic and trigger on the RDMA operation. You want to follow the data as it goes through the PCI bus and into the memory system. Since InfiniBand DMA transfers are in the virtual memory space the memory management units (MMUs) are involved. Therefore, you also want to look at the CPU-to-chipset communication as that is probably how the MMUs become configured and programmed.

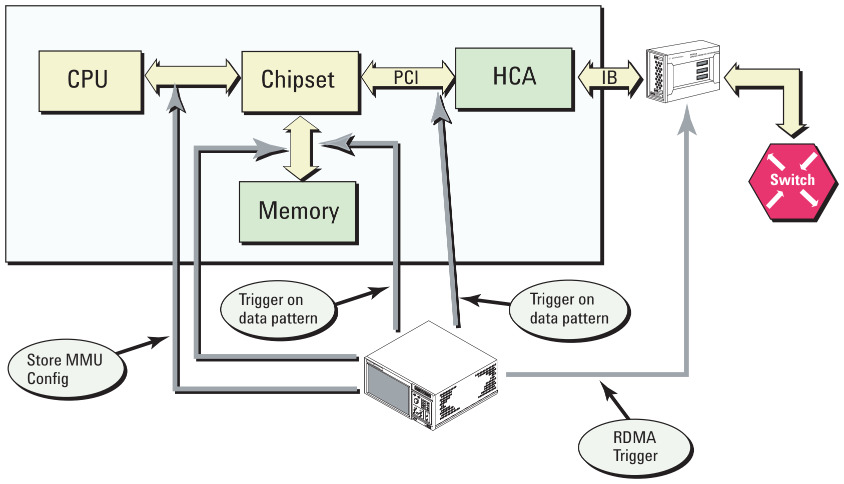


**Figure 9. Logic analyzer setup to investigate the RDMA transfer failure.**

If you have a traffic generator, you can generate the RDMA request and tell the traffic generator to use a unique data pattern in the DMA buffer: a “signature” pattern that is easy to follow as it flows through the system.

Set the logic analyzer to trigger on the DMA request and set the PCI and memory analyzers to trigger on the signature data pattern. This enables you to see the DMA as it passes through each part of the system. You also want to look at the CPU bus to gain insight into how the MMU is programmed.

For this example, the measurement shows that the data buffer actually makes it all the way through into memory, but that it was sent to the wrong physical memory location. This points to a potential problem with how the MMU tables were set up.

By making this measurement, you find out that the data buffer actually did get transferred after all. You can figure out where it went because you triggered on the data pattern and were able to look at the addresses that the signature data pattern was written to. You can then compare these addresses to the expected addresses to determine what the MMU programming error might be.

These measurements give you enough information to establish a working theory as to the root cause of the problem, and then you can follow through the rest of the troubleshooting process.

### Case Study 3: Switch Failure

A third example, moving still further up the protocol-layer stack, is a switch failure problem.

InfiniBand switches can be configured to filter out raw packets: If a raw packet comes in on one port, you can configure that port to "drop it in the bit bucket" and not send it out. The observed symptom of the problem is that raw packets are not being filtered properly.

Virtual port 0 is where all configuration information for switches is sent. Perhaps virtual port 0 never received the switch configuration command, or maybe it configured the wrong port on the switch.

This is a fabric interconnect problem, with packets going through the fabric, coming in one port of the switch, going out another port, and possibly also going through routers and other devices. For this kind of problem, you want to set up a validation platform for fabrics as shown in figure 10. This is an area where protocol analyzers and traffic generators really shine, because they are optimized for looking at InfiniBand traffic at higher layers of the protocol. There is a switch, a protocol analyzer on various ports to the switch, and traffic generators creating very specific traffic to make it easy to reproduce the problem.

This validation platform may also require a logic analyzer because for switches, virtual port 0 often doesn't exist physically. It is called virtual port 0 because semantically it is port 0 of the switch, but physically it may not actually have an InfiniBand connector. It may be implemented as a PCI or other kind of side channel from the controlling CPU into the switch circuitry itself. You might need to have a coordinated measurement between the different protocol analyzers and logic analyzers so that you probe enough parts of the system.

With a validation platform like this, you can generate raw packets going in, determine if the raw packets are coming out, or find out which port they should be coming out of. You can also look inside the switch to attempt to determine what part of the switch programming is causing the problem.

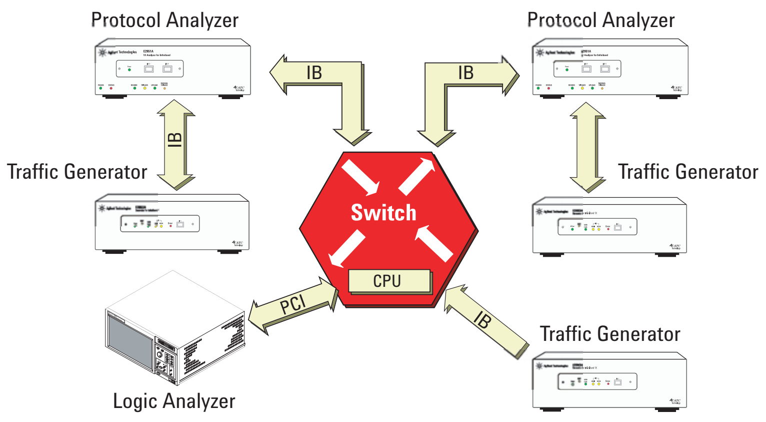


**Figure 10. Fabric validation platform.**

### Additional Applications

There are countless possible problems when debugging and validating InfiniBand systems that require a system-level view and the ability to look at many different parts of the system. Even for something as simple as getting endian-ness (bit order of data) wrong, you probably need to look at the InfiniBand channel, a PCI bus, DRAM channels, and the CPU bus.

Another problem that can occur is when you have many traffic generators hooked up to the fabric to stress the system and the whole fabric suddenly seizes up. That's often a very hard problem to diagnose. You might want to use the protocol analyzer to make statistical measurements of how much traffic is going through different ports so you can get a view of traffic flow through the system.  If you want a single time-correlated view of traffic across the entire fabric, then the logic analyzer can provide that for as many as 80 InfiniBand links at one time.

### Summary

When you are debugging and validating InfiniBand-based systems, you need to take a system-level approach. In general, you must look at many locations in the system and correlate activity in one part with activity in other parts. You should use the InfiniBand specification to help with developing the validation plan. The specification outlines all the requirements and also organizes the requirements into checklists. You can use the checklists to help organize your validation plan, and use the details in the specification to produce focused test descriptions.

Agilent can help you understand the strengths of various test tools for InfiniBand, and which ones to select for a particular task. As a member of the InfiniBand Trade Association, Agilent provides equipment to support compliance tests that cover several parts of the InfiniBand specifications. These range from physical to transport layers of the protocol, and include TDR scopes, logic analyzers, protocol analyzers and traffic generators. Agilent also offers extensive training in both InfiniBand design technology basics, and high-frequency test and system verification.

### Related Literature

| Publication Title | Publication Type | Publication Number |
|---|---|---|
| *Test Tools for InfiniBand* | Color brochure | 5988-2424EN |
| *Passively Probing an InfiniBand System with an Agilent 16700 Series Logic Analysis System* | Product note | 5988-2857EN |

**Agilent Technologies' Test and Measurement Support, Services, and Assistance**
Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Support is available for at least five years beyond the production life of the product. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

**Our Promise**
Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you use Agilent equipment, we can verify that it works properly, help with product operation, and provide basic measurement assistance for the use of specified capabilities, at no extra cost upon request. Many self-help tools are available.

**Your Advantage**
Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and on-site education and training, as well as design, system integration, project management, and other professional engineering services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of your Agilent instruments and systems, and obtain dependable measurement accuracy for the life of those products.

## Agilent Email Updates

**www.agilent.com/find/emailupdates**
Get the latest information on the products and applications you select.

## Agilent Direct

**www.agilent.com/find/agilentdirect**
Quickly choose and use your test equipment solutions with confidence.

**Agilent T&M Software and Connectivity**
Agilent's Test and Measurement software and connectivity products, solutions and developer network allows you to take time out of connecting your instruments to your computer with tools based on PC standards, so you can focus on your tasks, not on your connections. Visit **www.agilent.com/find/connectivity** for more information.

**By internet, phone, or fax, get assistance with all your test & measurement needs**

**Online assistance:**
**www.agilent.com/find/assist**

**Phone or Fax**
**United States:**
(tel) 800 829 4444

**Canada:**
(tel) 877 894 4414
(fax) 905 282 6495

**China:**
(tel) 800 810 0189
(fax) 800 820 2816

**Europe:**
(tel) (31 20) 547 2323
(fax) (31 20) 547 2390

**Japan:**
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

**Korea:**
(tel) (82 2) 2004 5004
(fax) (82 2) 2004 5115

**Latin America:**
(tel) (305) 269 7500
(fax) (305) 269 7599

**Taiwan:**
(tel) 0800 047 866
(fax) 0800 286 331

**Other Asia Pacific Countries:**
(tel) (65) 6375 8100
(fax) (65) 6836 0252
Email: tm_asia@agilent.com

**Product specifications and descriptions in this document subject to change without notice.**

**© Agilent Technologies, Inc. 2004**
**Printed in USA March 17, 2004**

**5988-4225EN**

Agilent Technologies