Appendix A
Test, Debug and Verification of PCI Express® Designs

by Gordon Getty, Agilent Technologies

Scope

The need for greater I/O bandwidth in the computer industry has caused designers to shift from using parallel buses like ISA, PCI and PCI-X to using multi-lane serial interconnects running at Gigabit speed. The industry has settled on PCI Express® technology as the key I/O technology of the future, as it delivers on the higher bandwidth requirements, helps to reduce cost for silicon vendors and leverages the software environment from the pervasive PCI/PCI-X® technology. While the change from parallel buses to multi-lane serial buses sounds like a small step, it presented a whole set of new debug and validation challenges to designers.

Serial technology requires a different approach to testing, starting from the physical layer and moving up through the transaction layer. In many cases, the parallel bus had several slots connected to the same physical lines, which allowed you to connect test equipment to the same bus and monitor other devices. With the point-to-point nature of serial technologies, this is no longer possible, and with the speed moving from the megahertz range to the gigahertz range, probing of the signal becomes a real challenge.

The second generation of PCI Express, known as PCI Express 2.0 (PCIe™ 2.0), is based on PCI Express 1.0 principles, but it supports speeds of up to 5 GT/s. Preserving backwards compatibility with PCI Express 1.0 presents its own set of challenges. Also, new and extended capabilities related to energy saving - including active state power management (ASPM) and dynamic link width negotiation - make achieving interoperability between devices more challenging, especially if these features are implemented incorrectly. Careful design and validation processes can help you avoid costly chip re-spins to fix interoperability issues.

This chapter will guide you through overcoming the challenges faced when you debug and validate your PCI Express devices.

Electrical Testing at the Physical Layer

PCI Express specification requires devices to have a built-in mechanism for testing the electrical characteristics of the devices, such as exists on motherboards and systems. When the transmit lanes of a device are terminated with a 50-ohm load, the transmit lanes are forced into a special mode known as compliance mode.

When a device is in compliance mode, it automatically generates a specific pattern known as the compliance pattern. Two different de-emphasis modes are introduced with the 5.0 Gb/s transfer rate. All add-in cards should be tested at the 2.5 Gb/s speed with -3.5 dB de-emphasis (Figure A-1), and at 5.0 Gb/s (Figure A-2) with both -3.5 dB de-emphasis and -6 dB de-emphasis.

Figure A-1: 2.5-GT/s PCIe Compliance Pattern

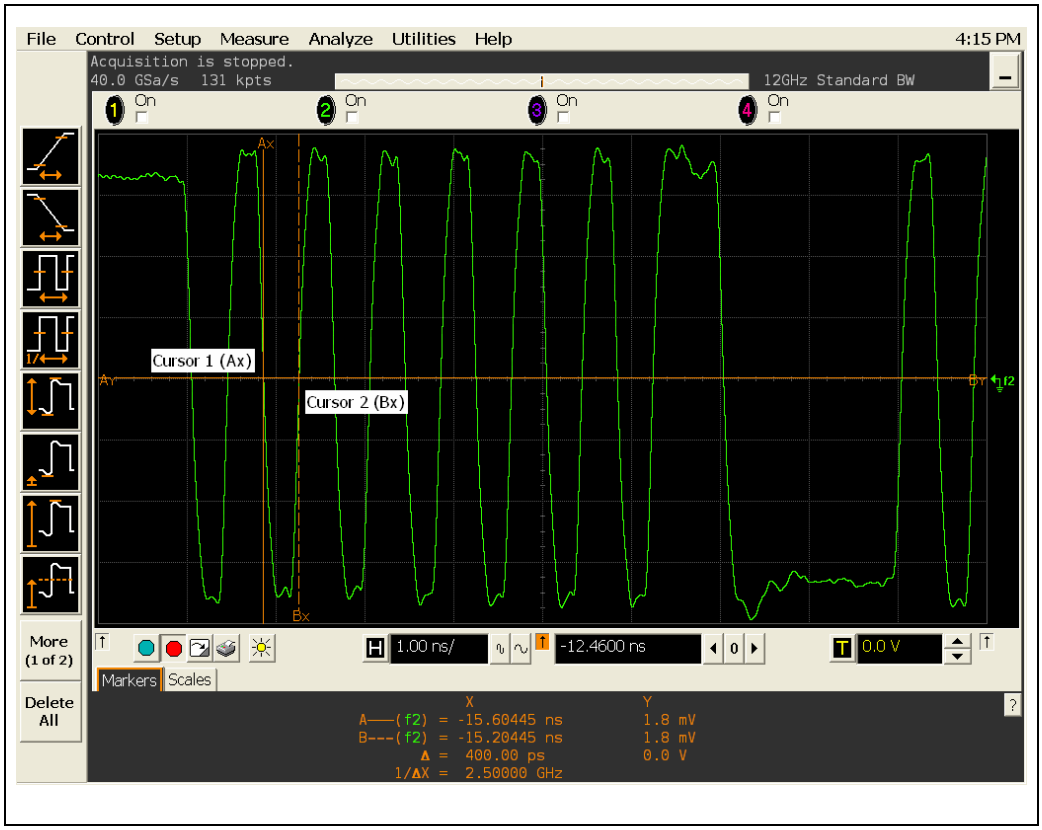
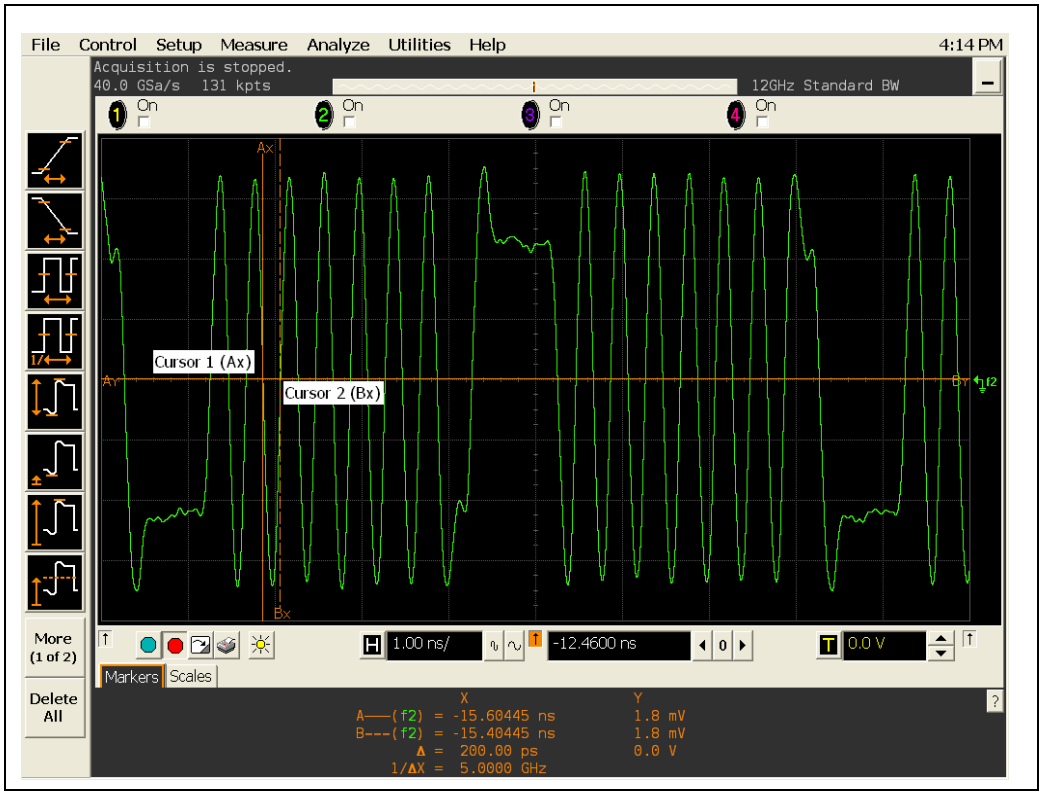


Figure A-2: 5-GT/s PCIe Compliance Pattern



The equipment required to carry out electrical testing on PCIe 2.0 devices includes a high-performance oscilloscope such as the Agilent Technologies DSO81304B 13-GHz Infiniium scope and a board into which you can plug an add-in card to provide a load on its transmitters. Alternatively, you can use a load board that plugs into a system and forces its transmitters into compliance mode ensuring that the device is generating a measurable signal.

PCI Express specifications (V1.1 and later) requires you to capture and process one million unit intervals of data to be able to make a valid measurement. The Agilent 81304B scope has a "QuickMeas" (QM) function that provides user-defined macros and data capture functionality intended to meet needs that may be very specific to a given application or measurement.

The PCI-SIG® provides compliance base board and compliance load board to help accomplish these tasks. These boards provide a consistent platform to make electrical measurements. Figure A-3 and Figure A-4 show a typical setup.

Figure A-3: Typical Setup for Testing and Add-In Card

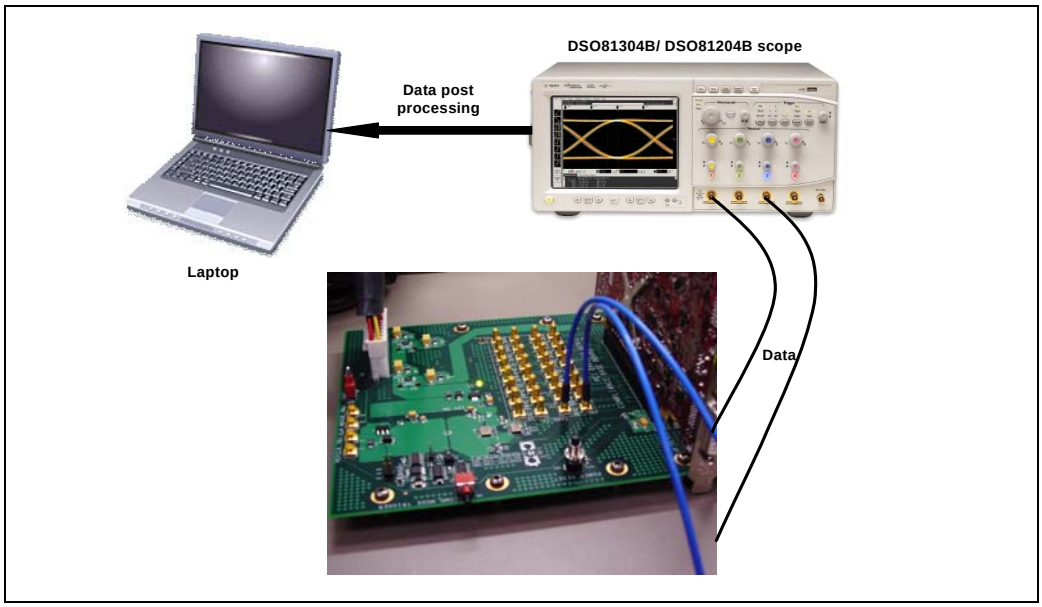
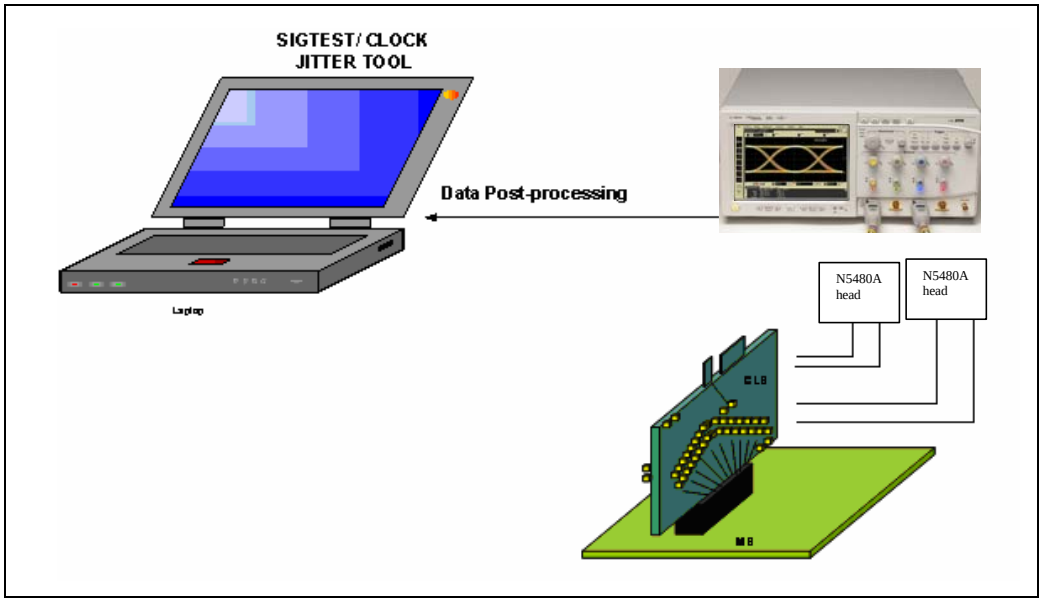
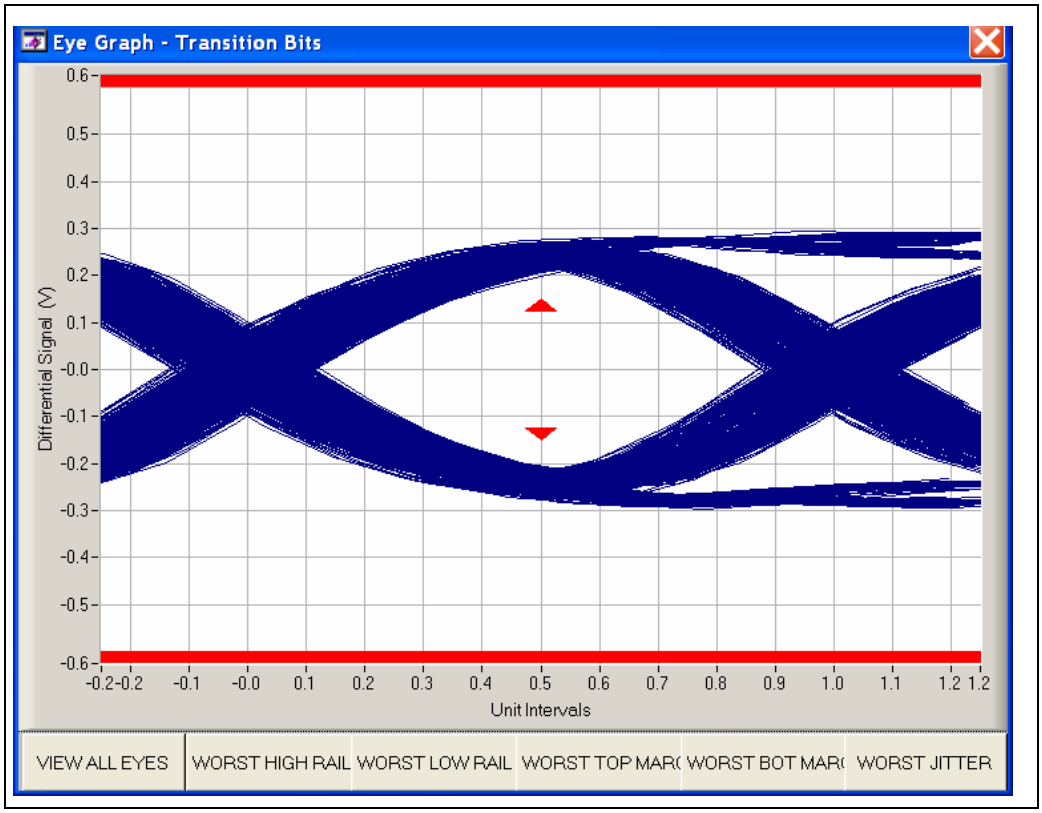


Figure A-4: Typical Setup for Testing a Motherboard



With the setups shown in Figures A-3 and A-4, data is captured on the oscilloscope. Post-processing is used to measure jitter on the reference clock and to measure the random and deterministic jitter on the data lines. In electrical testing, you need to test each individual lane independently, as each lane is likely to have different electrical characteristics. The data is captured and then post-processed to form an eye diagram, such as the one shown in Figure A-5.

Figure A-5: Oscilloscope Eye Diagram



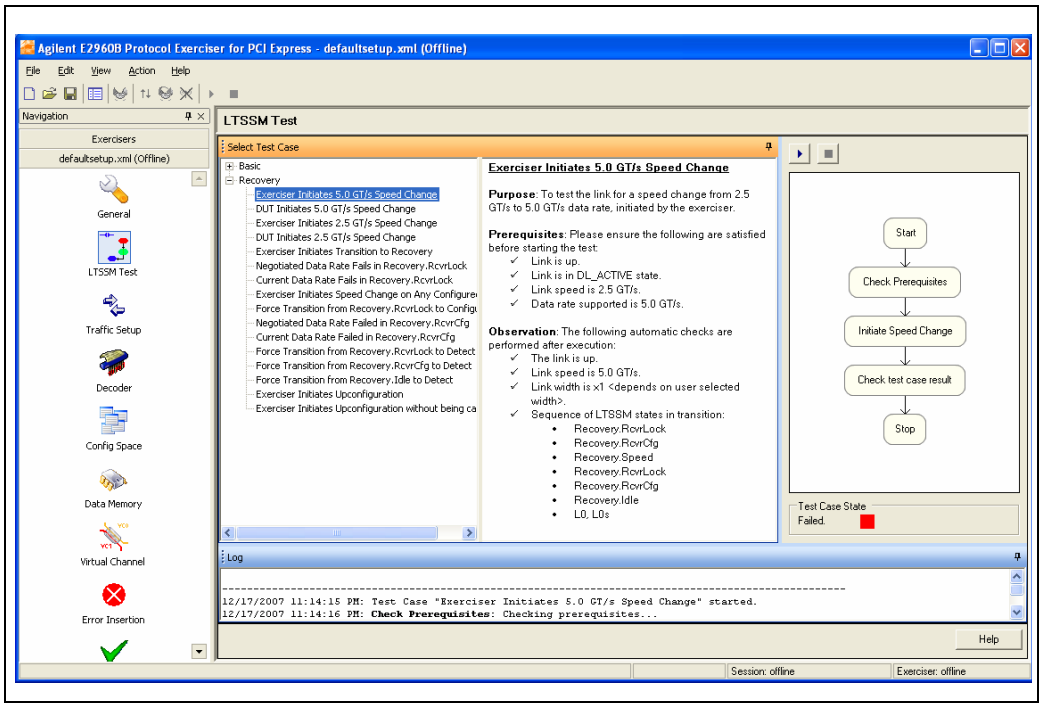
Using the eye diagram, you can measure the tolerances of voltage and jitter against the specification to determine if the device is compliant electrically. If you find the device is not compliant, you have an early indicator that interoperability is a potential issue.

Link Layer Testing

Once you have determined that the electrical characteristics of your device are within the specification, your device should be able to successfully establish a link with another device. For example, you should be able to plug an add-in card into a system motherboard and have the devices negotiate and complete link training. If you have followed the specification for the link training and status state machine (LTSSM) design, the two devices will be able to negotiate the number of lanes and the speed of the link. For PCI Express 1.0, the only speed required is 2.5 Gb/s, whereas in PCI Express 2.0, speeds may go up to 5 Gb/s speed. Speed negotiation occurs during link training.

Figure A-6 shows a test setup screen of a Agilent Protocol Exceriser through which one can test the LTSSM.

Figure A-6: Testing the LTSSM with the Agilent N5309A Exerciser for PCIe 2.0



PCI Express has a more robust set of error checking mechanisms than conventional PCI. It is possible to recover from certain types of errors that may be caused by conditions such as a marginal signal quality at the electrical layer. There are cases where you will want to test the device against error conditions to ensure the robustness of your design and to ensure that your device will continue to operate correctly in a real operating environment. This is especially important for data integrity reasons. In conventional PCI, a simple parity check was applied to the data by using a dedicated line on the bus. Should a parity error occur, the system would typically signal it by asserting the PERR signal and potentially the SERR signal, which would most likely cause a "blue screen" scenario where the system would halt.

The data link layer (DLL) in PCI Express is designed to ensure that data transferred over the physical layer reaches its destination intact without errors. This is done using an acknowledge (ACK) or not acknowledge (NAK) message protocol. The DLL adds an LCRC (CRC= cyclic redundancy code) to the packet that is checked at the receiving end for integrity, and if the calculation turns out to be different, it is considered an error at the DLL and it potentially signals that the data being transferred was somehow changed and is now not valid. In conventional PCI, such an error would result in a system halt. In PCI Express, the correct behavior would be to send a NAK back to the originator, indicating that it should resend the same packet since there was an error in the transmission of the packet the previous time it was sent. The originator would hold the packet in its replay buffer until it has received some kind of acknowledgement from the other end of the link.

PCI Express also has a message generation mechanism defined in the specification that allows the devices to report that an error has occurred but it has been noticed and fixed. A correctable error (ERR_COR) message is sent in this particular case and the relevant bit set in the configuration space.

When a packet is sent, a timer is also started, and if no acknowledgement (ACK or NAK) is sent by the time this timer expires, the device resends the packet stored in the replay buffer. If this happens four times in a row, the DLL forces the physical layer to retrain the link, since there is potentially a problem that cannot be resolved without retraining or re-initializing the link.

It is very difficult to create the scenarios described above using real devices since the condition technically is not built into the design. However, to ensure your devices behave properly if such a condition arises, you can use a protocol test tool such as the Agilent E2960B exerciser for PCI Express 2.0. This stimulus tool has the capability to generate traffic and non-standard conditions and test that the results are correct.

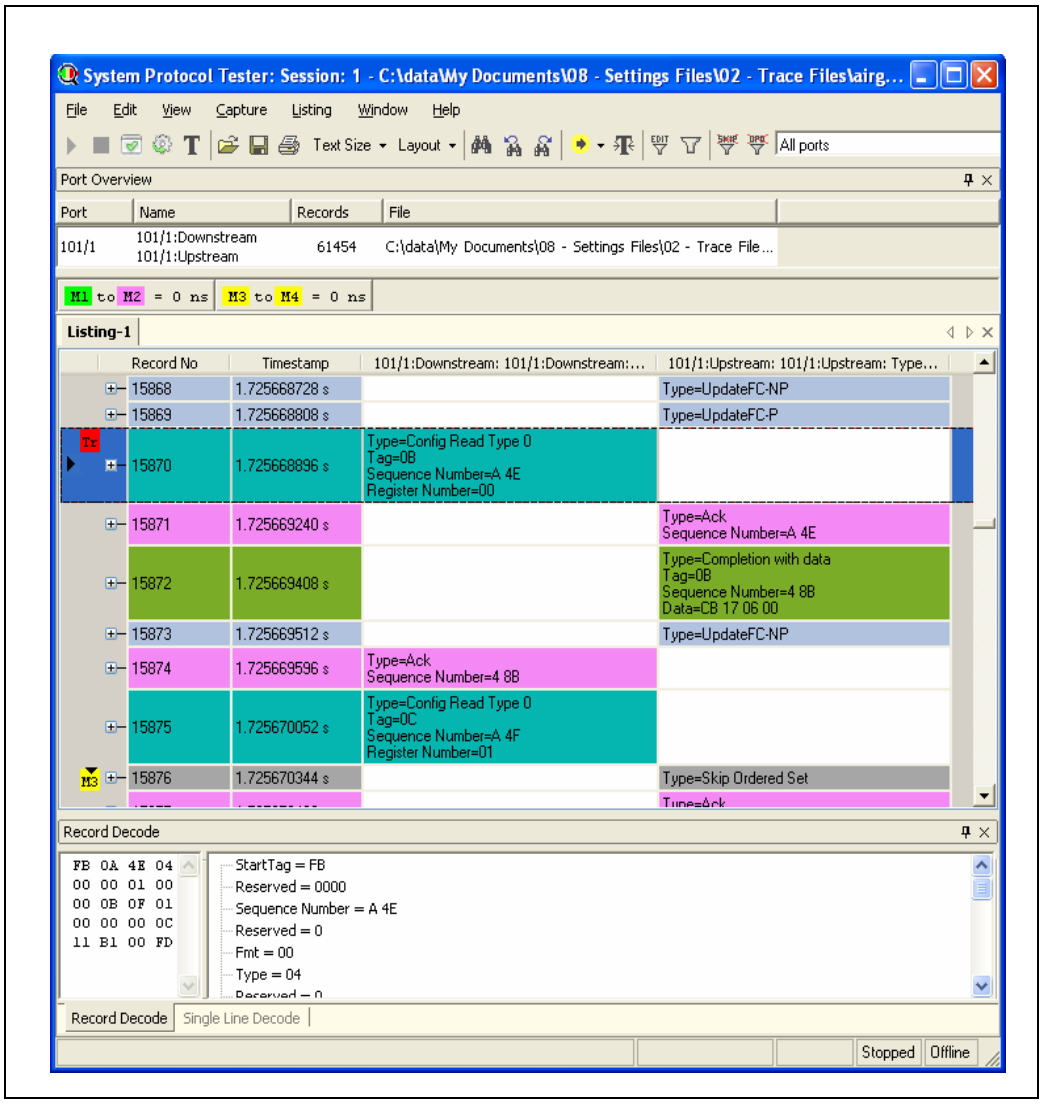
The protocol analyzer gives a view of the link at the data link and transaction layers while also providing logical physical layer information in the form of the 8b/10b symbols transmitted on the wire. See Figure A-7. No electrical signal information is provided. The protocol analyzer has the intelligence to identify and trigger on complex conditions on the bus consisting of combinations of symbols in the form of packets and, at a higher level, transactions. The protocol analyzer is connected between the two devices on the link and observes real traffic between two devices. You can probe the link, either using an interposer card that fits into a slot or a mid-bus probe that uses a predefined layout on the PCB and brings the signals to the surface of the board where they can be probed. Other probing options, especially for embedded PCI Express applications, include flying leads, which can be used when no slot or midbus footprint is available.

The exerciser and protocol analyzer tools are quite different from physical-layer tools, such as the oscilloscopes discussed earlier, in that they operate under the assumption that the physical layer is working properly. However, physical-layer errors may show up at the data link layer and transaction layer in the form of CRC or disparity errors. These types of errors can be emulated by the exerciser because it has a programmable PCI Express interface. The PCI Express exerciser is actually a fully functioning PCI Express device that can behave either as a root complex or an end point and has additional deterministic behavior capabilities on the link, including the injection of errors. The exerciser is designed to establish a link based on parameters such as link speed, link width, and scrambling enabled or disabled, among others. In addition, particularly for PCIe 2.0, the de-emphasis level can be set to off, -3.5 dB or -6 dB.

The PCIe 2.0 specification introduced additional features. Similar features have already shown interoperability problems with existing PCI Express 1.0a and 1.1 devices. During link training, a PCIe 2.0 device advertises the speed capability in the training control register of the training sequence. The bit used to indicate this was previously a reserved bit in the 1.0a and 1.1 specifications. When some devices that implement PCIe 1.0a and 1.1 are plugged into a 5-Gb/s capable system, these reserved bits are not set to zero, and you may not be able to successfully establish a link. Using the Agilent exerciser, it is possible to easily test the behavior of a PCI 1.x device when the higher speed class is advertised.

PCIe 2.0, makes extensive use of the recovery state of the LTSSM, both to allow backwards compatibility with PCIe 1.0 and to allow the negotiation of the higher speed. Testing the LTSSM again provides assurance that devices will be able to link and operate properly in a real environment.

Figure A-7: Representation of Traffic on a PCIe Link Using an Agilent Protocol Analyzer



You would use an LTSSM test, for example, when one end of the link attempts to initiate a speed change to the higher speed and fails - after exiting the Rec.Speed substate, one device remains at the lower speed while the other is at high speed. This scenario is easily tested using the Agilent exerciser in combination with the Agilent protocol analyzer.

A very important role of the DLL is to manage flow control. This is a credit-based mechanism that allows each end of the link to determine the size of the buffer for receiving packets and data at the other end of the link. Immediately after the link has trained, the flow control credits are initialized. Each device advertises the number of headers for posted, non-posted and completion packets in addition to the amount of data payload it can handle. When transaction layer packets are being sent back and forth, the DLL is responsible for sending periodic updates of the available credits to ensure no deadlock condition happens on the link. Flow control has a huge impact on the performance of the link. The exerciser can again be used to emulate different flow control scenarios, as it allows you to manually set flow control parameters from advertising no credits to advertising unlimited credits. The exerciser allows you to arbitrarily send out a DLLP, potentially with incorrect flow control data to test the behavior of the device.

Transaction Layer Testing

The transaction layer in PCIe 2.0 is very similar to that of the 1.x specifications, which provides a great advantage in terms of software compatibility between the two specifications. The layered approach separates the physical connection from the upper-layer protocol. However, the new speed also presents different challenges related to performance at the DLL and transaction layers.

The protocol analyzer also allows you to measure characteristics such as actual throughput on the bus and transaction latencies. This information is extremely valuable for optimizing the performance of devices. The protocol analyzer is also a key instrument for finding errors and performing root cause analysis on error cases such as the one shown in Figure A-8. Unlike physical layer errors that may appear at the data link layer as disparity errors or CRC errors, errors at the transaction layer do not necessarily appear in the data link layer; they need to be dealt with by the receiver transaction layer. Figure A-9 shows an analyzer screen through which you can enable/disable error conditions to trigger on.

Figure A-8: Finding a Particular Condition or Sequence of Events Using a Trigger Sequencer

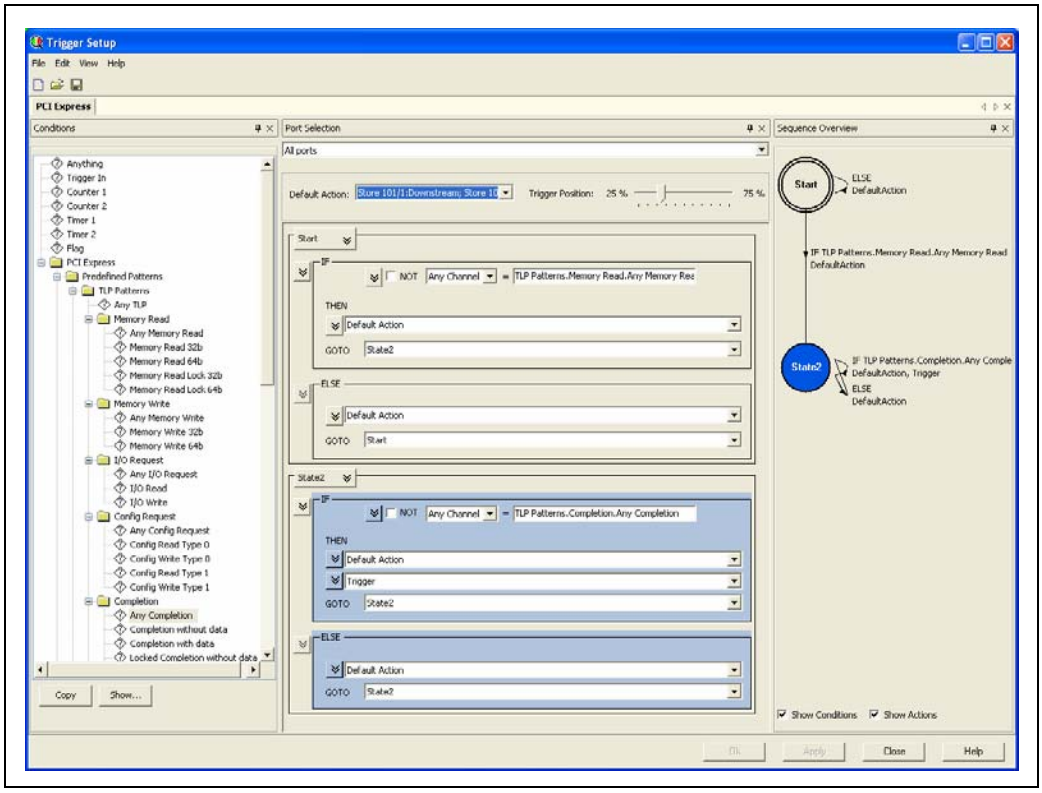
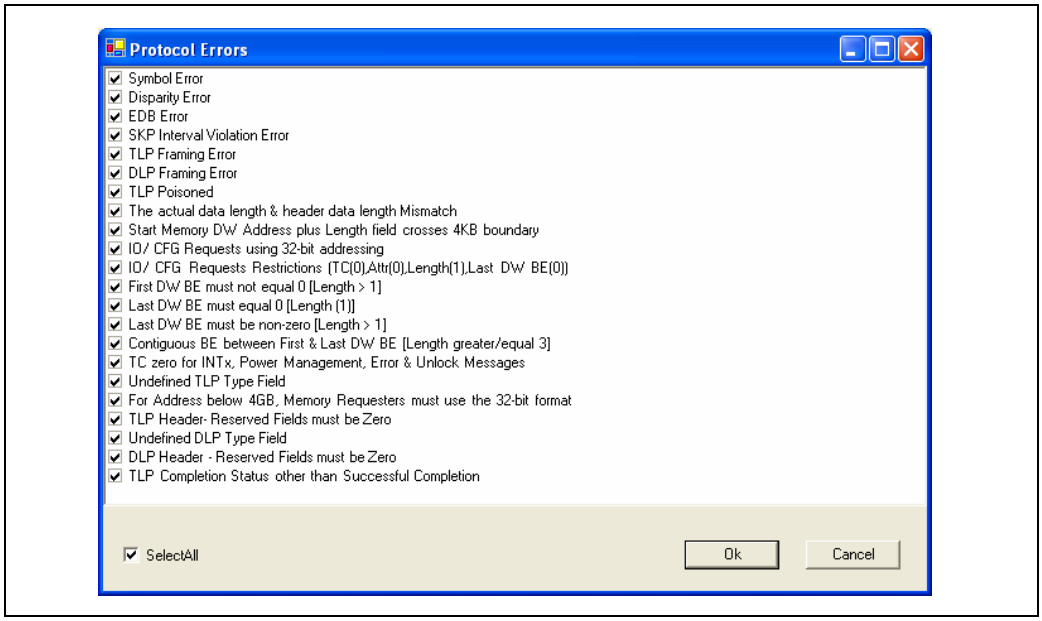


Figure A-9: Typical PCI Express Error Conditions Triggerable on Agilent Protocol Analyzer



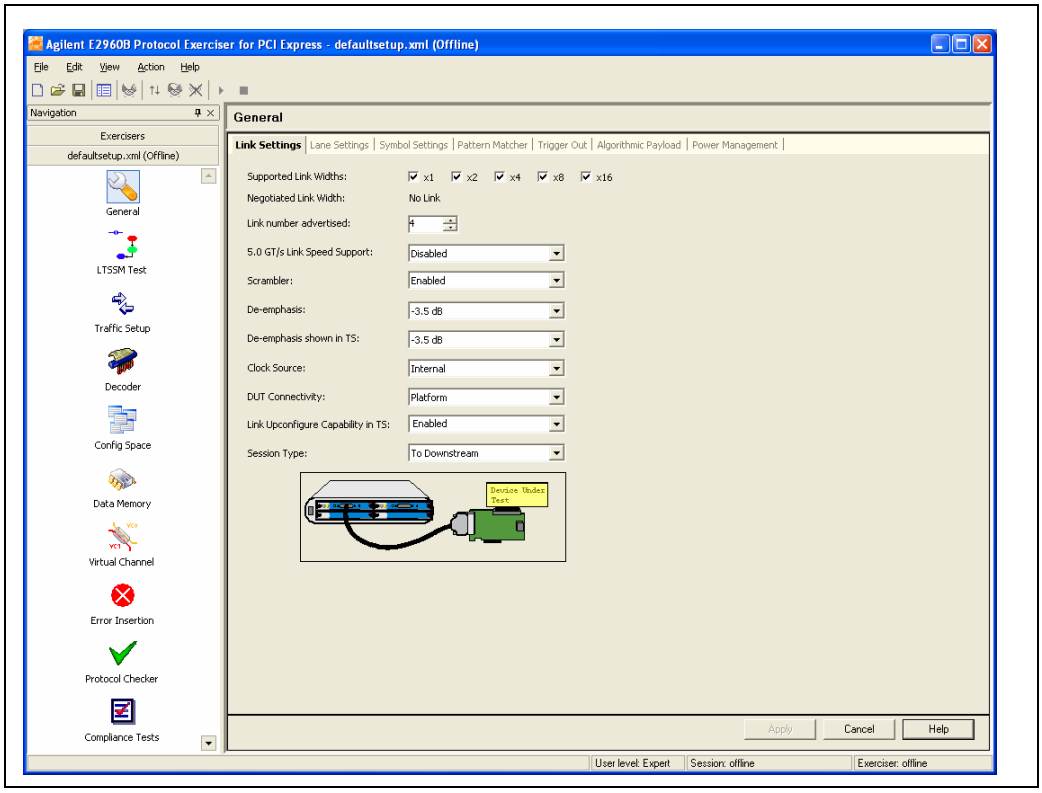
It is strategically important for you to know the performance of a device during the design stages. It is common, especially in the early stages of a technology, to have performance testing capabilities.

Stressing the system by generating back-to-back packets, using different packet lengths and types and injecting errors is an ideal way to identify potential problems early that could eventually lead to costly redesigns. Stressing the system also helps you avoid competitive disadvantages.

Testing the system's data integrity is also critical for the success of a device. Running write/read/compare tests with known data and deterministic patterns over a long period of time allows you to test corner cases thoroughly in a short timeframe, giving you confidence that interoperability will not be an issue for the device.

The Agilent exerciser and analyzer are invaluable tools for evaluating and optimizing PCI Express designs. In the past, validation tools were limited. You could use off-the-shelf devices, test tools to change static link parameters and pattern generators. New test tools from Agilent integrate testing the high complexity and dynamic nature of PCI Express devices using one platform driven by a user-friendly GUI interface, as shown in Figure A-10.

Figure A-10: Agilent PCIe 2.0 Exerciser Provides a Powerful and Flexible Validation Platform

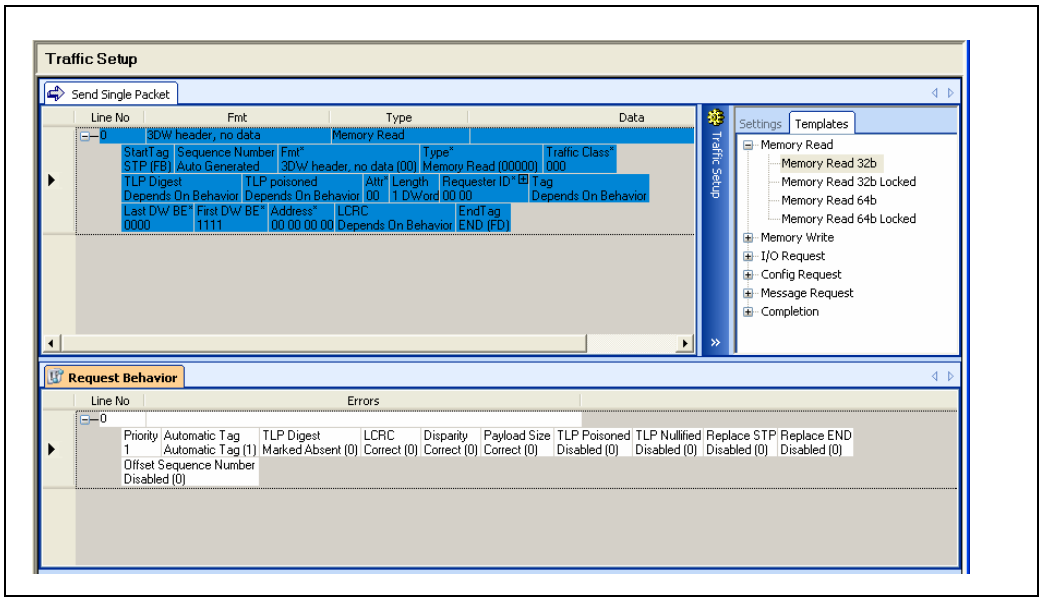


The exerciser is a standard-size PCI Express card, as shown in Figure A-11. It is in fact fully programmable from an external host connected through USB. You can control the behavior of the tool independently of the system it is plugged into.

Figure A-11: The Agilent PCIe 2.0 N5309A Exerciser

Via a user-friendly GUI, it is possible to setup PCI Express traffic and have it downloaded via the USB interface to the PCI Express exerciser. The exerciser provides templates as shown in Figure A-12 for creating different types of individual packets or requests that may contain multiple packets. Each of the requests has an associated behavior. This allows you to create traffic patterns via the exerciser deterministically and also create error cases that would not be possible in a real situation - the errors may happen but it may be unpredictable as to when and why they happen.

Figure A-12: Templates for Creating Traffic Using the Exerciser



The exerciser allows you to recreate error scenarios and perform root cause analysis on them. As shown in Figure A-13, it is possible to edit a packet and inject error conditions. The erroneous packet is then generated by the exerciser and error analysis performed with the aid of an Agilent Analyzer.

The exerciser will behave as a requester or master, and also as a target depending on how you set it up. You can also control the behavior of completions. For example, when the exerciser receives a request in the form of a memory read, the default response would be to respond with a successful completion. However, you can also program it to respond in other ways, including unsuccessful completion or a completer abort. Figure A-14 shows a completion packet editor via which you can inject errors into the completer generated completion packet. In addition, it is possible to have the exerciser delay sending the completion by a specific amount of time, which would facilitate the testing of the completion timeout mechanism on the requester.

Figure A-13: Adding Single or Multiple Error Scenarios to each Request

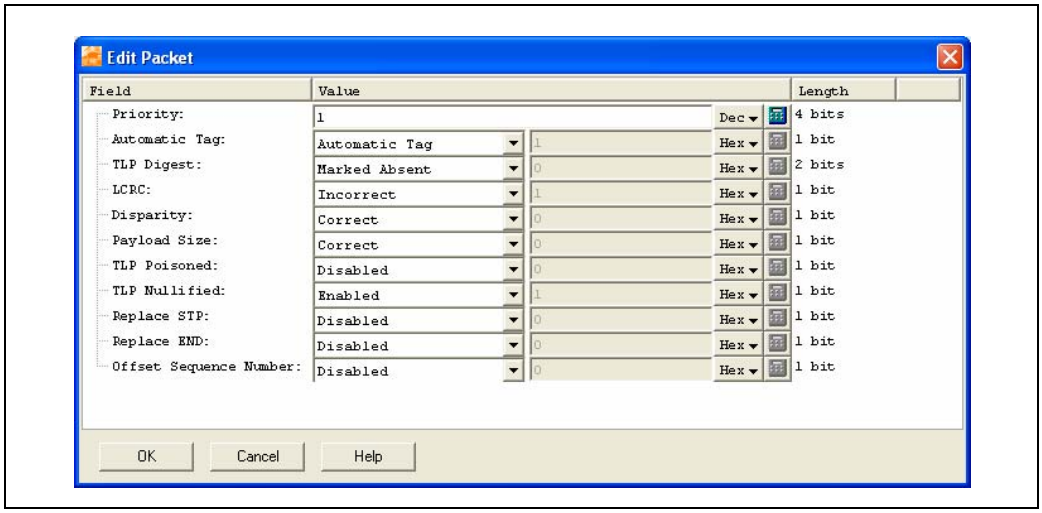
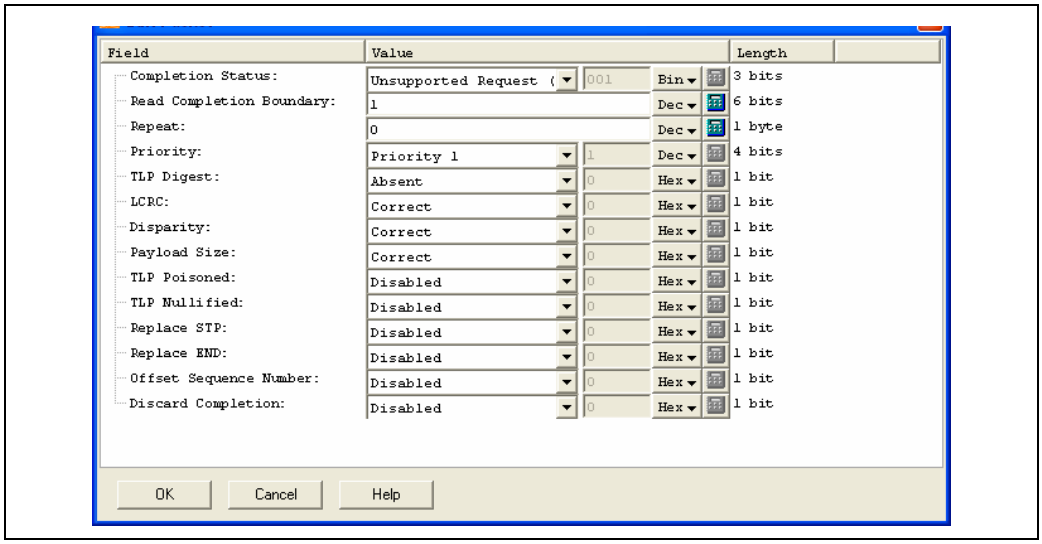


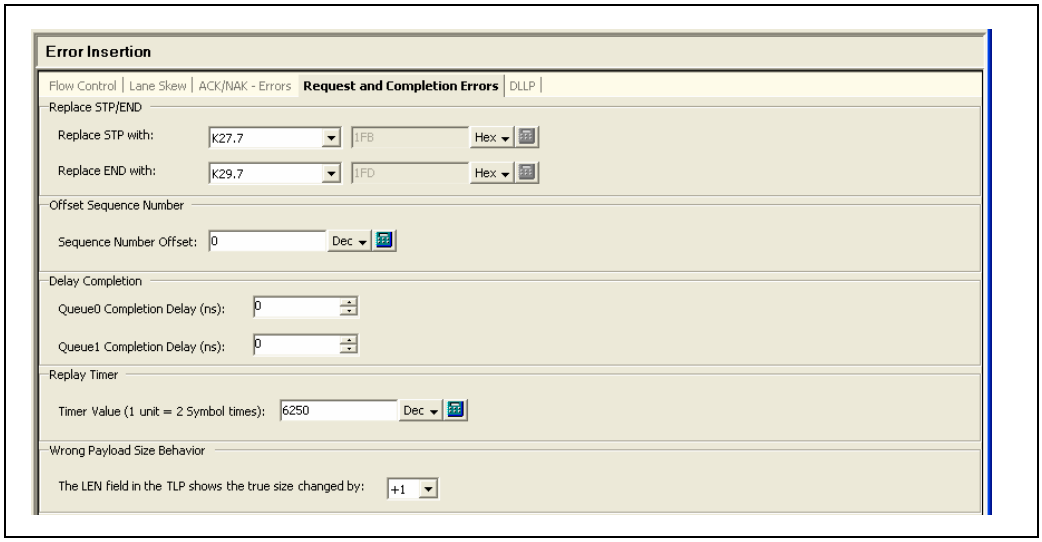
Figure A-14: Programming Completer Behaviors



To create framing errors, you can replace the start-of-transaction-layer packet (STP) symbol and end of packet (END) symbols with other arbitrary values. Also, since the exerciser is a fully programmable device, it is also possible to change values for the replay timer.

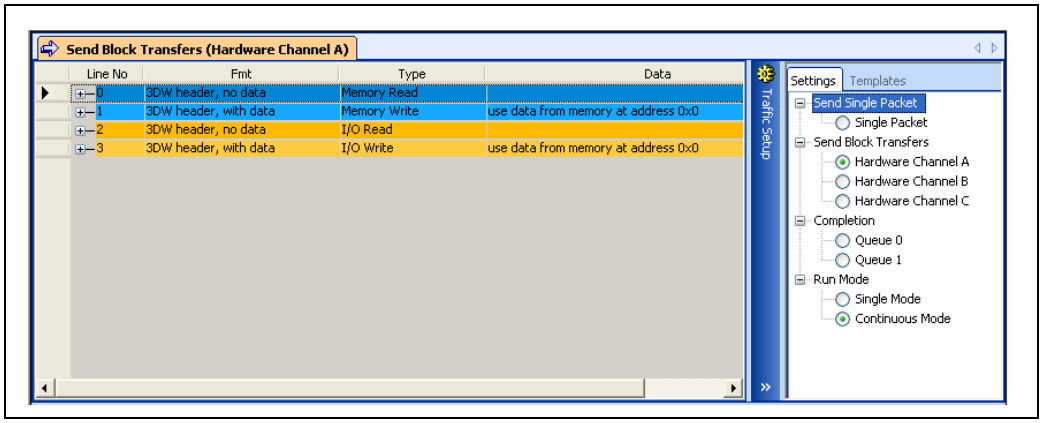
You can create another type of transaction layer error by changing the length field within the TLP to be different from the actual length of the data in the payload. Most likely, this would be treated as a malformed packet. You can easily set up these errors using the exerciser, as shown in Figure A-15.

Figure A-15: Inserting Request and Completion Errors Using the Exerciser



In addition to generating error conditions, you can use the exerciser as a tool to stress the link. This is done by programming the exerciser to do a block of requests, then repeat them continuously. It is possible to have multiple different requests repeated in continuous mode as shown in Figure A-16 and also, if required, add error-case scenarios.

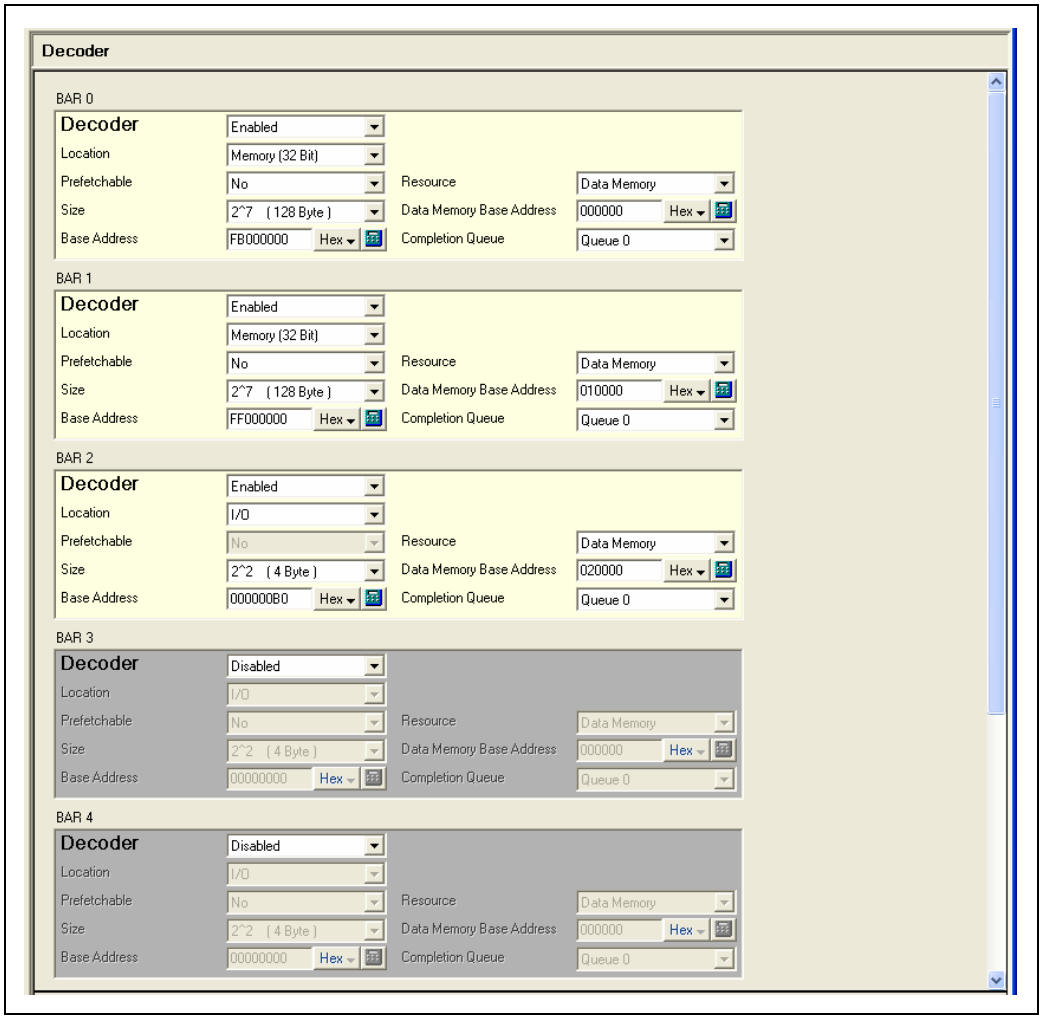
Figure A-16: Using Continuous Mode on the Exerciser Running a Loop of Memory and I/O Transactions



You can program the configuration space of the exerciser to emulate different types of devices. PCIe devices may implement base address registers (BARS) and decoders as required, according to resources needed by that particular device. The BIOS on the system then assigns resources to these devices at system startup time. It is important for BIOS engineers to be able to verify that they can provide the correct resources for any combination of devices that may be plugged into the system. It is critical that address ranges do not overlap.

The exerciser has more than one way of carrying out this testing. Since the exerciser is a real PCIe device, you can manually configure and program the decoders prior to starting up the system. You can also map these decoders to different completion behaviors and priorities and map them to specific areas in the internal data memory of the exerciser. You have complete control of the location, size and type of each of these decoders, as shown in Figure A-17.

Figure A-17: The Exerciser Has Fully Programmable Memory and I/O Decoders

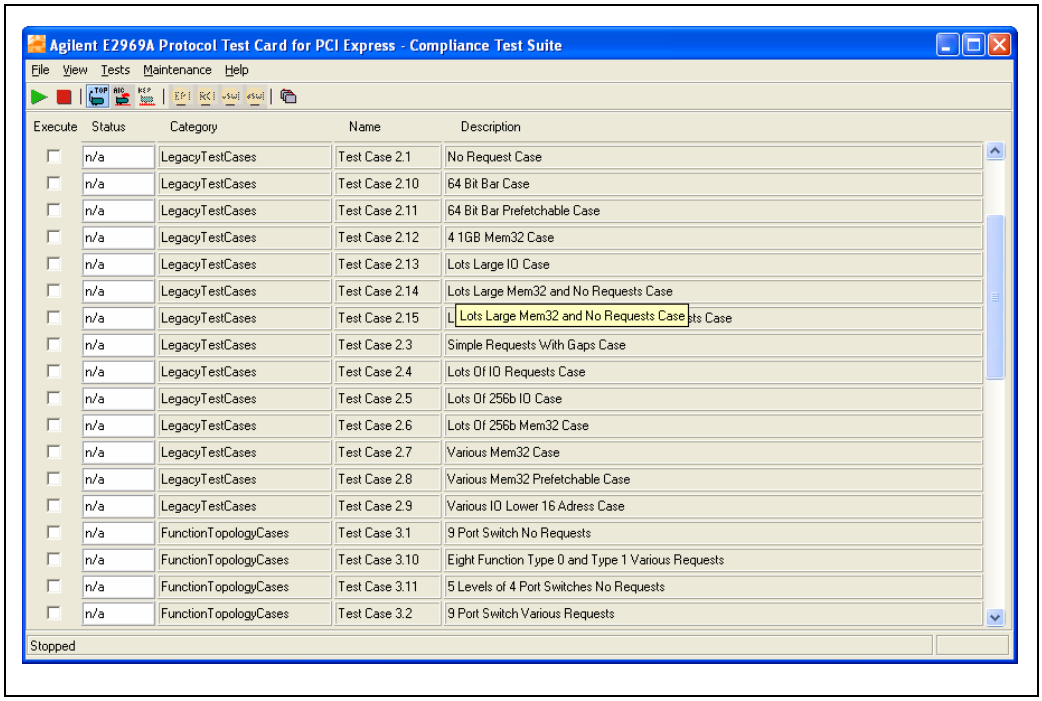


A second method is available for testing system BIOS. This method also provides a way to use the exerciser card as an interface to the PCIe port on the system, but it provides device emulation in software via the USB port. The advantage of this approach is that you can emulate and test many different topologies, including multiple levels of switches or bridges. It can also test the BIOS for different types of devices and resource requirements, such as a bridge device that requests memory resources. It is a valid configuration, but in the past, we've seen certain BIOS systems that will not support this device. It is important that the behavior be correct if an optional feature is not implemented. This method would also be helpful if a device requires more resources than are available on a system. The BIOS should handle this properly and disable the device.

Topology testing has been used since the days of conventional PCI and PCI-X. A PCI or PCI-X exerciser was used to emulate different devices and then check that addresses assigned were correct and not overlapping. The same principle is used for PCI Express testing from the outset, using the Agilent E2969A protocol test card for PCI Express. This method of testing is also ported to the Gen 2 PCI Express exerciser card. The same set of tests used in Gen 1 PCI Express is used. The main difference is that the speed is now 5 GT/s rather than 2.5 GT/s. Since the BIOS is at the application layer, above the transaction layer, the principles of testing it are independent of the physical link width and speed. The same test cases used in the original PCI and PCI-X testing are still quite valid.

Testing can be extended to cover many complex topologies. Figure A-18 is a screen through which you can setup topology tests.

Figure A-18: Topology Tests Using the Agilent E2969A Protocol Test Card



Summary

When you are designing and validating PCI Express designs, it is important to cover all aspects of testing, as problems in the lower layers often result in problems at the upper layers, which ultimately lead to interoperability issues. By designing and testing to the PCIe specification, you can be assured that your devices will work properly and will not face real-world compatibility problems.

Contact Agilent Technologies

For more information on the complete set of test tools from Agilent Technologies, please visit our Web site at: www.agilent.com

For PCI Express applications, please visit www.agilent.com/find/pciexpress

For oscilloscopes, please visit www.agilent.com/find/oscilloscopes

For protocol test tools, please visit www.agilent.com/find/e2960_series

** PCI Express and PCI-X are registered trademarks of PCI-SIG.*

** PCIe is a trademark of PCI-SIG.*