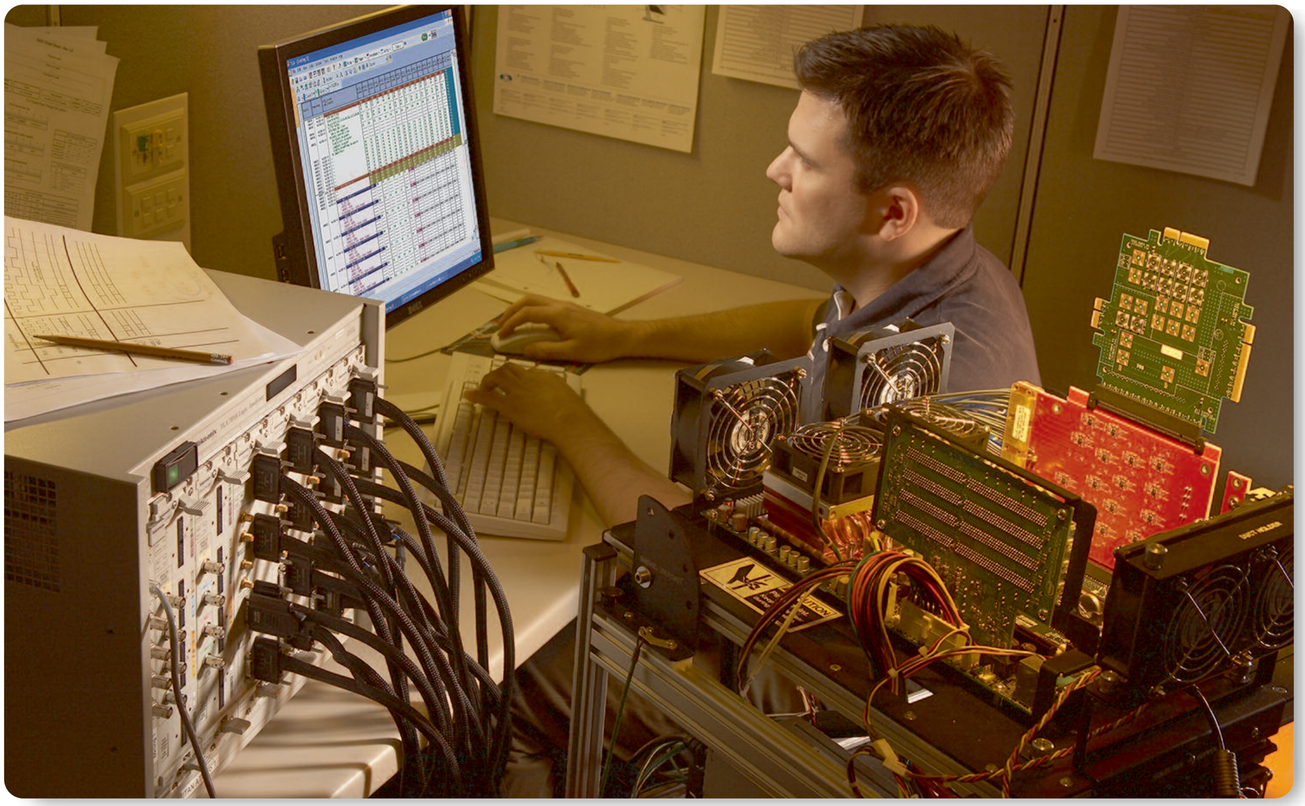
Cross-Bus Analysis Reveals Interactions and Speeds Troubleshooting

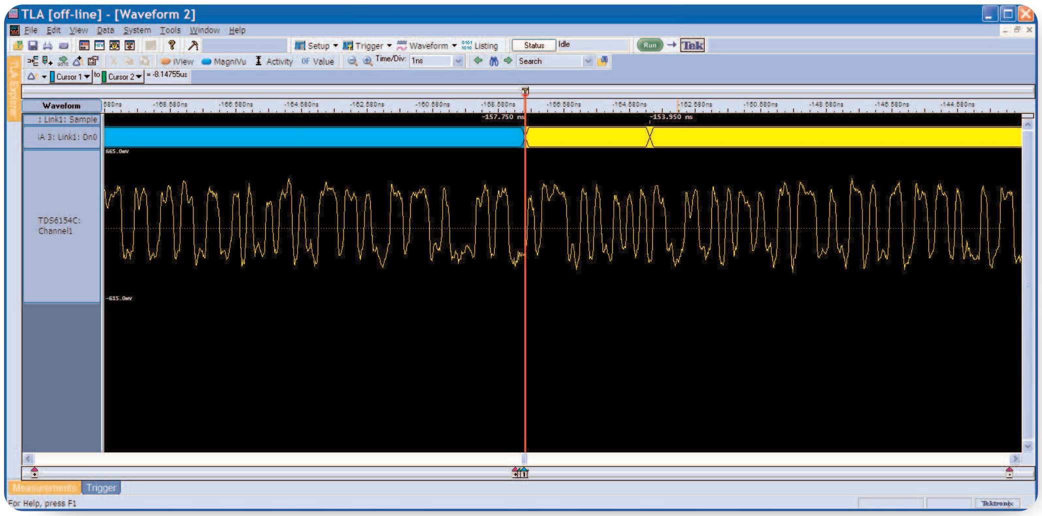


Today's digital systems—from the video game console in the media room to the complex switching elements in a communication network—rely heavily on serial bus technology to do their job. Not surprisingly, a host of application-specific serial buses has emerged. Serial ATA handles communication between chipsets and disk drives. HDMI manages data going from digital A/V sources to display devices. PCI Express (PCIe), designed to connect peripheral devices in the PC environment, now finds itself in a wide range of applications not served by other specialized interfaces. In a given electronic system, it is not unusual to find all of these buses coexisting, and potentially several parallel buses as well.

This trend has intensified the demand for cross-bus troubleshooting solutions that offer a simple integrated way to view logical activity on several different buses at once. A variety of solutions exists. The traditional approach is to pair a standard-specific protocol analyzer with a logic analyzer (LA); the former takes care of the serial acquisition while the LA captures parallel bus data that may pertain to the troubleshooting issue at hand. Another approach is to use an LA with a bus support package that includes an external interface to convert serial data into the parallel data used by the logic analyzer.

Now a third methodology has arrived. Tektronix TLA7000 Series logic analyzers can be equipped with integrated PCI Express serial acquisition modules that plug directly into the LA mainframe just like their parallel counterparts. Users can mix serial and parallel acquisition modules within a single system. With the addition of this serial capability, the TLA7000 family can capture and display time-correlated parallel and serial data as well as analog waveforms from an oscilloscope, all on the same LA screen.

Tektronix
Enabling Innovation


TLA [off-line] - [Waveform 2]
File Edit View Data System Tools Window Help
Setup Trigger Waveform Listing Status Idle Run
iView MagniVu Activity Value Time/Div: 1ns Search
Cursor 1 to Cursor 2 = -8.14755us
Waveform
Link1: Sample
LA 3: Link1: Dn0
TDS6154C: Channel1
-157.750 ns
-153.950 ns
Measurements Trigger
For Help, press F1

▶ **Figure 1.** *Cross-bus analysis often begins with a dual-trace LA display showing the analog waveform plotted against a serial or parallel bus data acquisition. Captured with an oscilloscope and ported to the TLA7000 Series logic analyzer using the iView function, this screen shows that the analog behavior of the serial signal is not causing the errors in the serial signal. The analog waveform is within normal tolerances.*

It is a capability that is destined to simplify digital troubleshooting. Using a combination of PCI Express serial and parallel modules, cross-bus analysis can be performed by one logic analyzer system.

Digital Solutions Often Begin With Analog Troubleshooting

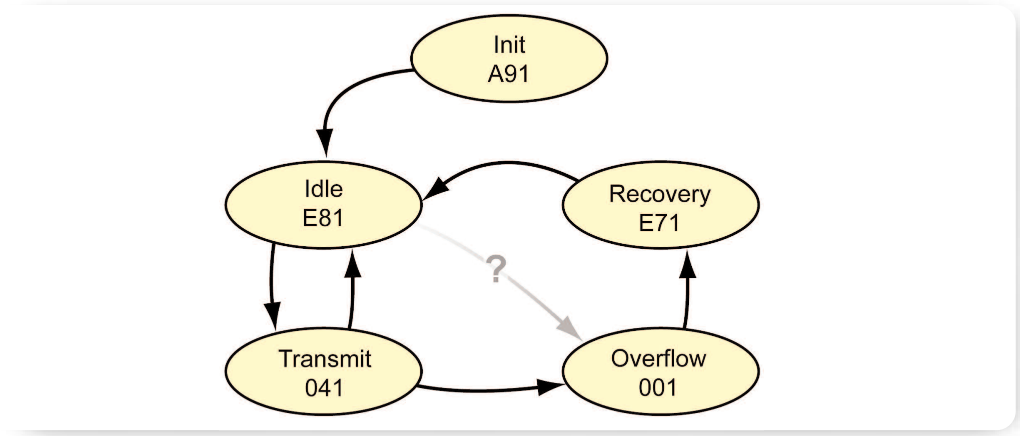
The underlying architecture of a PCI Express serial link is well established. Often embedded as an element within an FPGA, a PCI Express transmitter with a SERDES (serializer-deserializer) at its heart sends 8b/10b encoded information to a receiver elsewhere in the system. Transmission impedances, bit rates, and clock characteristics are explicitly specified and controlled for interoperable operation between diverse manufacturers' PCI Express components.

Though this link is a digital system, errors therein may have either digital or analog origins. Frequently the first step in troubleshooting is to take a "snapshot" of the analog waveforms at the time of the error.

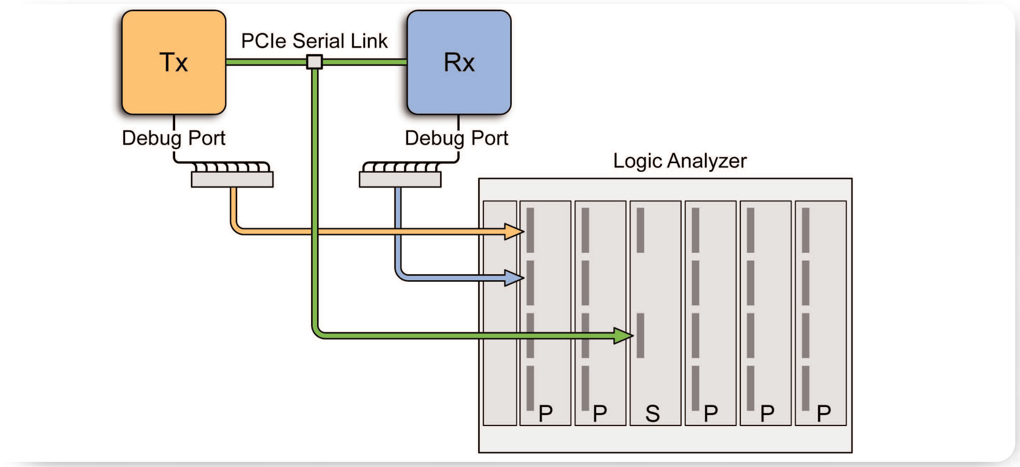
Some logic analyzers include features that enable them to integrate analog acquisitions (waveforms) from a connected oscilloscope into the digital LA display. The analog traces are of course time-correlated with their digital counterparts. This makes it possible to observe analog events such as glitches and runt pulses concurrently with the digital events that may be their cause or consequence. A logic analyzer equipped with parallel modules and serial modules and this analog display capability is an unmatched cross-bus analysis platform.

Figure 1 shows an acquisition taken from a PCI Express serial link. The cursor location marks the cycle in which the link goes to an incorrect packet value. The mnemonics of that state are explained in more detail in the next section of this document. All that matters right now is that there is an error, its occurrence has caused the LA to trigger, and that event has in turn triggered an acquisition by a real-time oscilloscope monitoring the link. The image in Figure 1 was captured by a Tektronix TLA7000 Series logic analyzer equipped with an integrated PCI Express serial module as well as iView (Integrated View) tools, which deliver oscilloscope waveforms to the LA display. The oscilloscope used in this example is a Tektronix TDS6154C real-time instrument. Other models in the Tektronix DPO7000 Series, DPO70000 Series, and the DSA70000 Series also offer the iView capability.

The analog waveform depicts the exact same data as the serial busform display above it. The two views are time-correlated (synchronized). Theoretically it is possible to hand-decode the actual binary waveform data to confirm this. Looking at the analog waveform in the vicinity of the cursor, it is apparent that the error is not caused by analog-domain problems such as runt pulses or glitches. The pulses in the error packet are consistent in amplitude, duration, and aberrations with those that precede and follow them. Thanks to the



▶ **Figure 2.** *The Debug Port state machine of the PCI Express receiver.*



▶ **Figure 3.** *The logic analyzer acquires parallel data from the debug ports simultaneously with serial data from the PCI Express link. Acquisition modules marked with a "P" are parallel while the unit marked with an "S" is a dedicated PCI Express serial module. All data traces are time-correlated when an integrated logic analyzer is used.*

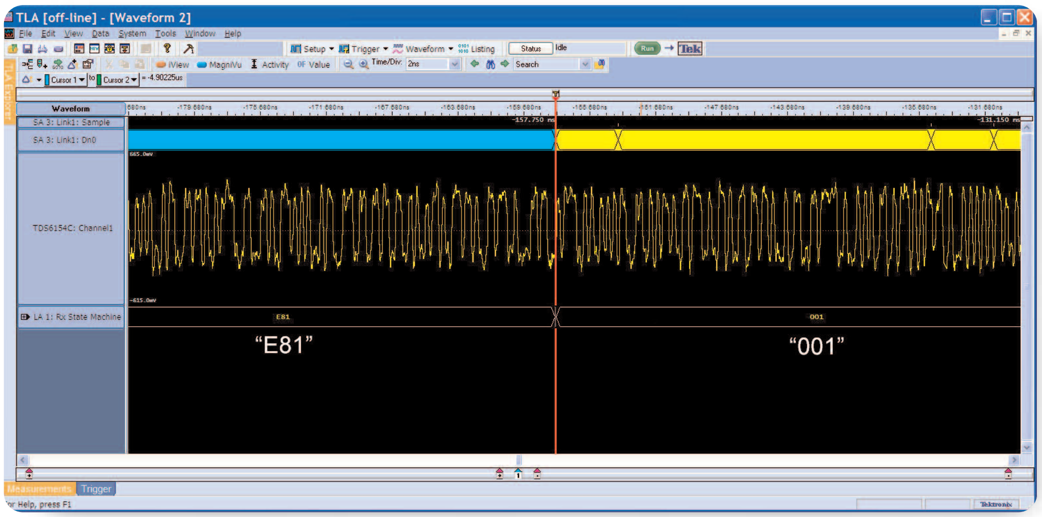
accurate time correlation of the two views, one can conclude that the packet error does not stem from an underlying analog problem. If there had been such a problem, the next troubleshooting steps would rely on an oscilloscope triggered by the logic analyzer to track down the root cause. But the findings in Figure 1 strongly imply a digitally-based issue deriving from a timing problem or other digital conflicts. The logic analyzer is of course the tool of choice to complete the job.

Debug Port's Parallel Data Offers a Different View of Serial Link Activity

PCI Express transmitter/receiver pairs, common in consumer electronic products, telecommunication systems, and many other applications, often include not only a serial link, but also a built-in "debug port." This parallel output delivers real-time data summarizing the transactions occurring within the device. With debug ports on both the transmitter and the receiver, developers can monitor the health of the transmission link and localize many types of problems to either the transmit or the receive side.

Figure 2 represents a state machine that might be found within a PCI Express serial receiver. The simplified interactions shown here symbolize a routine link procedure, with the black arrows indicating legal state transitions.

Figure 3 is a block diagram showing a test setup for the PCI Express serial link and its transmitter and receiver state machines. Assume that this is a troubleshooting routine designed to locate the origin of garbled data appearing on the serial link. The debug ports are of course connected to a parallel acquisition module, while the PCI Express link connects to a serial module.



▶ **Figure 4.** *The serial errors (the yellow portion of the serial busform trace) coincide with an incorrect state change in the Debug Port state machine. This implies a timing problem within the SERDES, which may stem from errors in the FPGA synthesis process.*

Figure 4 is a screen image from the logic analyzer acquisition. This view adds the parallel data stream captured from the receiver's Debug Port. The new logic analyzer busform trace includes the hexadecimal values shown in the state machine diagram (bottom waveform).

Look closely at both busform envelopes at the point where the red cursor line crosses them. Here the link enters the Overflow (001) state. Something has gone wrong. The routine has jumped directly from Idle to Overflow, which is impossible if the state machine is circulating properly through its instructions. The gray arrow in Figure 2 indicates this "error" step.

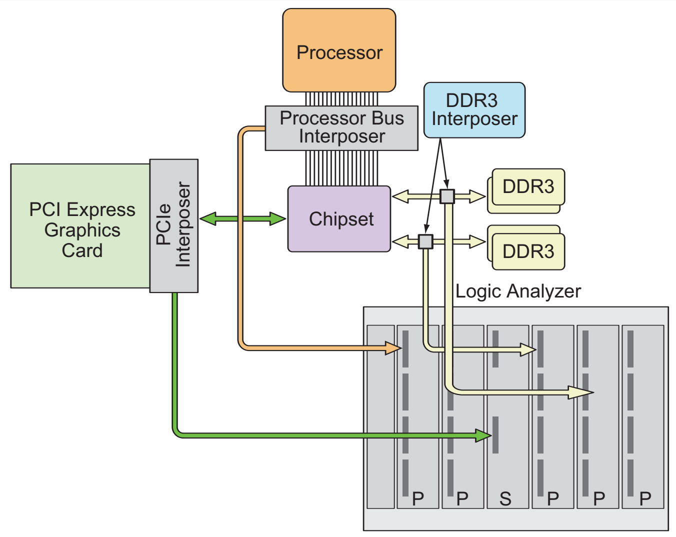
All three traces in Figure 4 are time-correlated thanks to the tightly integrated serial and parallel acquisition modules operating within the same logic analyzer mainframe. In some cases the serial bus transition may lag behind the Debug Port output due to latency, that is, the time required for the serial buffer to flush its contents after the state has changed. In such instances the timing differential visible in the cross-bus view will reflect this latency accurately.

In Figure 4 the traffic on the serial bus is so dense that individual cycles cannot be displayed at the current resolution. But it is clear that the yellow portion of the serial trace coincides with the 001 state on the State Machine trace. The blue portion of the serial trace's timing matches up correctly with the E81 Idle state on the debug port. The link is operational and communicating but it is not following its intended routine.

Because the serial data errors coincide with the Overflow state on the debug port, and because the serial data is driven by the SERDES it is reasonable to assume that the problem is timing-related and originates within the SERDES. At this point there may be several potential troubleshooting strategies, influenced by architectural considerations or other debug findings.

Most commonly, serial link features of the kind discussed here are incorporated into an FPGA. This type of device is designed to transform itself into functional elements defined by the programmer. This "transformation" process is known as synthesis, since it literally synthesizes the desired functions using its internal gates. Knowing this, the astute designer will troubleshoot the error first by double-checking the FPGA synthesis results to make sure the timing of all state machine transitions is correctly implemented.

If that doesn't reveal the problem's source, a second pragmatic step is to route other signals to the debug connector to trace the device's behavior. For example, after evaluating the Current State data as shown in Figure 4 the FPGA might be reprogrammed to deliver the "Next State" data to the debug port. This could reveal issues that are not seen in the Current State, and of course there are even more states that can be investigated beyond that.



▶ **Figure 5.** *Simultaneous acquisition of serial and parallel bus data speeds the search for a memory Write error. As in Figure 3, "P" modules are parallel, while the "S" denotes a PCI Express serial module.*

Monitoring Three Buses At Once To Track Down Errors

The PC motherboard is an example of today's cross-bus troubleshooting environments. It is a complex, sophisticated piece of electronic design; diverse high-speed serial buses transport signals among IC components, between on-board subsystems, and out to peripherals and storage media. A problem on any one of these buses can manifest itself as an error on an entirely different bus.

In the simplest terms, the logic analyzer must trigger on an error occurring on one bus while viewing the error's origins in a different subsystem or bus and its consequences on a third. During the development of a motherboard, interactions and dependencies among buses that are not electrically connected can reveal much about the stability of the emerging product.

Consider the following example: a prototype for a motherboard has arrived after fabrication. During the design validation process, it frequently encounters problems in its routine functional exercises. In the worst case, the device freezes and must be re-booted. At other times the device seems to operate normally but the display is garbled and unintelligible. A block diagram of this motherboard, connected to the logic analyzer for simultaneous serial and parallel acquisition, is shown in Figure 5. The "interposers" are simply probing attachments that plug into existing board-mounted connectors to extract the desired signals.

A test is created that incorporates a series of READ and WRITE operations, among others. The test proceeds:

– First the CPU issues a WRITE command and sends data 13FF (hexadecimal, as are all values in this discussion) to a particular address location (00100000) in the DDR3 SDRAM memory.

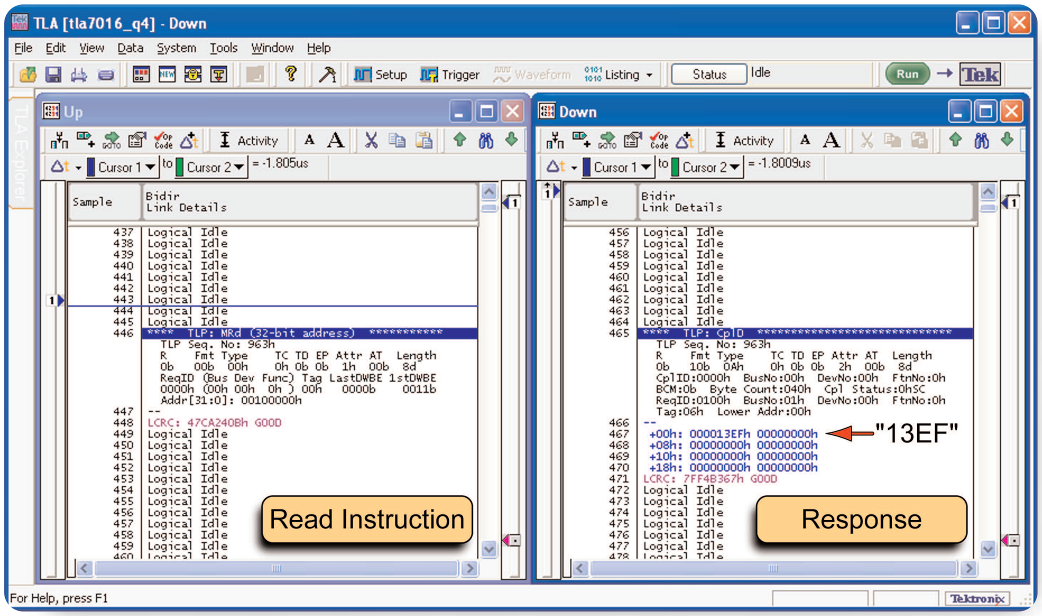
In Figure 5, the instruction passes from the CPU through the processor bus to the chipset, and ultimately to the DDR3 SDRAM.

– Next the graphics card issues a READ instruction to the same address. The command goes over the PCI Express bus, through the chipset, and to the DDR3 SDRAM.

– Lastly the CPU issues a second WRITE command and sends new data to the same location in the DDR3 SDRAM.

Since no other instruction should have modified the data before the graphics card READ, the result of the query should be 13FF—exactly the same data that was written during the first cycle.

But the result is 13EF. The PCI Express graphics card reports an error. What could be causing this problem?



```
Sample   Bidir
         Link Details
   437   Logical  Idle
   438   Logical  Idle
   439   Logical  Idle
   440   Logical  Idle
   441   Logical  Idle
   442   Logical  Idle
   443   Logical  Idle
   444   Logical  Idle
   445   Logical  Idle
   446   ****  TLP: MRd (32-bit address)  ***********
           TLP Seq. No: 963h
           R     Fmt Type    TC TD EP Attr AT  Length
           0b    00b  00h    0h 0b 0b  1h  00b  8d
           ReqID (Bus Dev Func) Tag LastDWBE 1stDWBE
           0000h (00h 00h  0h ) 00h   0000b    0011b
           Addr[31:0]: 00100000h
   447   --
   448   LCRC: 47CA240Bh GOOD
   449   Logical  Idle
   450   Logical  Idle
   451   Logical  Idle
   452   Logical  Idle
   453   Logical  Idle
   454   Logical  Idle
   455   Logical  Idle
   456   Logical  Idle
   457   Logical  Idle
   458   Logical  Idle
   459   Logical  Idle
   460   Logical  Idle
```

```
Sample   Bidir
         Link Details
   456   Logical  Idle
   457   Logical  Idle
   458   Logical  Idle
   459   Logical  Idle
   460   Logical  Idle
   461   Logical  Idle
   462   Logical  Idle
   463   Logical  Idle
   464   Logical  Idle
   465   ****  TLP: CplD  ***************************
           TLP Seq. No: 963h
           R     Fmt Type    TC TD EP Attr AT  Length
           0b    10b  0Ah    0h 0b 0b  2h  00b  8d
           CplID:0000h  BusNo:00h  DevNo:00h  FtnNo:0h
           BCM:0b  Byte Count:040h  Cpl Status:0hSC
           ReqID:0100h  BusNo:01h  DevNo:00h  FtnNo:0h
           Tag:06h  Lower Addr:00h
   466   --
   467    +00h: 000013EFh 00000000h
   468    +08h: 00000000h 00000000h
   469    +10h: 00000000h 00000000h
   470    +18h: 00000000h 00000000h
   471   LCRC: 7FF4B367h GOOD
   472   Logical  Idle
   473   Logical  Idle
   474   Logical  Idle
   475   Logical  Idle
   476   Logical  Idle
   477   Logical  Idle
   478   Logical  Idle
```

▶ **Figure 6.** *The logic analyzer display shows both the Read cycle and the response from the DUT.*

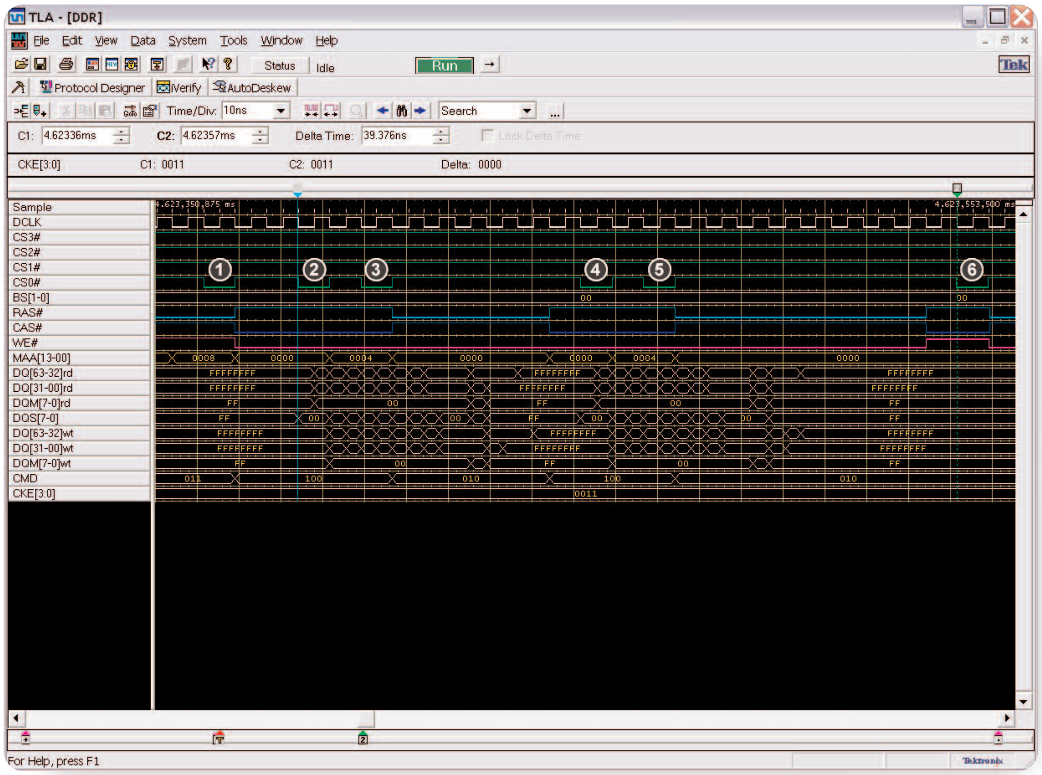
Concurrent monitoring of all three of the buses involved in the transaction proves to be a fast and easy way to track down the problem. In Figure 5, the PCI Express, DDR3 SDRAM and processor bus interposers connect to serial and parallel acquisition modules, respectively.

Looking at the transactions from the PCI Express bus (in their deserialized form, as delivered by the PCI Express acquisition module and shown in Figure 6), it becomes clear that the PCI Express graphics card is indeed receiving the incorrect 13EF data word. It is appropriately reporting flawed data. Neither the graphics card nor the PCI Express bus is the source of the problem.

The next step is to look at transactions on the DDR3 SDRAM bus which, because they produce a display very similar to Figure 6, need not be repeated here. A READ operation confirms that the correct address was written.

That brings the processor into question. Did it send the data it was supposed to send? Monitoring the processor bus establishes that the correct data was written to memory.

All three buses appear to be doing their jobs correctly. The data is being issued correctly and sent to the desired memory location as commanded by the CPU. The only remaining possibility is a timing conflict of some kind. One potential suspect is the READ/WRITE timing. Yet the previous steps have established that the CPU is issuing the WRITE at the expected time.


TLA - [DDR]

Time/Div: 10ns

Status: Idle

C1: 4.62336ms C2: 4.62357ms Delta Time: 39.376ns

CKE[3:0] C1: 0011 C2: 0011 Delta: 0000

For Help, press F1

▶ **Figure 7.** *The DDR3 SDRAM bus. After CS0-1 opens the row, a sequence of WRITE operations occurs, commencing after the cursor. CS0-2 and CS0-3 are the first WRITEs to the memory in this sequence. The paired pulses write to consecutive addresses. CS0-4 and CS0-5 are another pair of WRITEs. CS0-6, the final command to this address location, is a READ that should be occurring before the CS0-4 and CS0-5 pair.*

When timing and synchronization problems are suspected, the logic analyzer's ability to view correlated traces from all three buses is a time-saver. Looking at the memory bus reveals that the READ is preceded by—rather than followed by—the second WRITE cycle, as shown in Figure 7. The PCI Express card receives data stored one operation later than intended. The circled numerals on this timing acquisition correspond to the following steps:

1.	Row Open
2 and 3.	First Writes
4 and 5.	Second Writes
6.	Read (should have occurred between steps 3 and 4)

The time-correlated view of the READ, WRITE, and data values on the respective buses reveals a classic problem: the chipset, which is designed to act as a "traffic director," is not timing the graphics card's READ request correctly. The READ fails to access the memory after the first CPU WRITE cycle as intended. The chipset is source of this problem.

Conclusion

Frequently, tracing a system problem involves much more than just following a glitch back to its source in some logic element. An error on one bus may have its origins—and its impacts—on multiple buses in the system. For this reason, complete cross-bus analysis has become an indispensable troubleshooting methodology. With the advent of integrated tools that bring time-correlated serial, parallel, and even analog events into view on a logic analyzer screen, designers have a powerful new tool in their troubleshooting work. Cross-bus analysis makes it possible to see simultaneous interactions throughout the system, speeding efforts to track down not just errors, but also their root causes.

Contact Tektronix:

ASEAN / Australasia	(65) 6356 3900
Austria	+41 52 675 3777
Balkan, Israel, South Africa and other ISE Countries	+41 52 675 3777
Belgium	07 81 60166
Brazil & South America	(11) 40669400
Canada	1 (800) 661-5625
Central East Europe, Ukraine and the Baltics	+41 52 675 3777
Central Europe & Greece	+41 52 675 3777
Denmark	+45 80 88 1401
Finland	+41 52 675 3777
France	+33 (0) 1 69 86 81 81
Germany	+49 (221) 94 77 400
Hong Kong	(852) 2585-6688
India	(91) 80-22275577
Italy	+39 (02) 25086 1
Japan	81 (3) 6714-3010
Luxembourg	+44 (0) 1344 392400
Mexico, Central America & Caribbean	52 (55) 5424700
Middle East, Asia and North Africa	+41 52 675 3777
The Netherlands	090 02 021797
Norway	800 16098
People's Republic of China	86 (10) 6235 1230
Poland	+41 52 675 3777
Portugal	80 08 12370
Republic of Korea	82 (2) 6917-5000
Russia & CIS	+7 (495) 7484900
South Africa	+27 11 206 8360
Spain	(+34) 901 988 054
Sweden	020 08 80371
Switzerland	+41 52 675 3777
Taiwan	886 (2) 2722-9622
United Kingdom & Eire	+44 (0) 1344 392400
USA	1 (800) 426-2200

For other areas contact Tektronix, Inc. at: 1 (503) 627-7111

Updated 1 June 2007

Our most up-to-date product information is available at: **www.tektronix.com**

9/07 FLG/WOW 52W-21073-0

Tektronix
Enabling Innovation