Overcoming PCI-Express Physical Layer Challenges

Using the Tektronix Logic Protocol Analyzer

PCI Express is a ubiquitous and flexible bus addressing many markets. Unfortunately this flexibility can also cause integration issues that are very difficult to debug. Rate changes, width changes, spread spectrum clocking and advanced power states can all cause or exacerbate integration bugs. Frequency margining a system is a useful method to ensure that a design will still have margins when it moves to high volume manufacturing. The Tektronix Logic Protocol Analyzer is a valuable tool for tracking and providing visibility of link width changes, rate changes, and advanced power states. Additionally this product family enables the user the flexibility to use spread spectrum clocking and to frequency margin their system while still providing visibility. This paper will present how the Tektronix Logic Protocol Analyzer is used to overcome these challenges using powerful triggering and multiple data views.

Individual Lane Visibility During Link Width Changes

PCI Express busses change their width on the fly for several reasons, most notably to trade off power against bandwidth and to enable high reliability systems by down training link width when a lane fails. This enables the link to maintain operation although at a lower bandwidth. A debug engineer working on a high reliability system will not only need to verify that a width change occurs when a lane fails, but may also need to characterize or optimize how long it takes. The Tektronix Logic Protocol Analyzer allows you to easily trigger on link width changes and view the data on each lane individually and intuitively. Triggering on any link width change is easy: select the Link Event specially designed for this purpose. See Figure 1A.

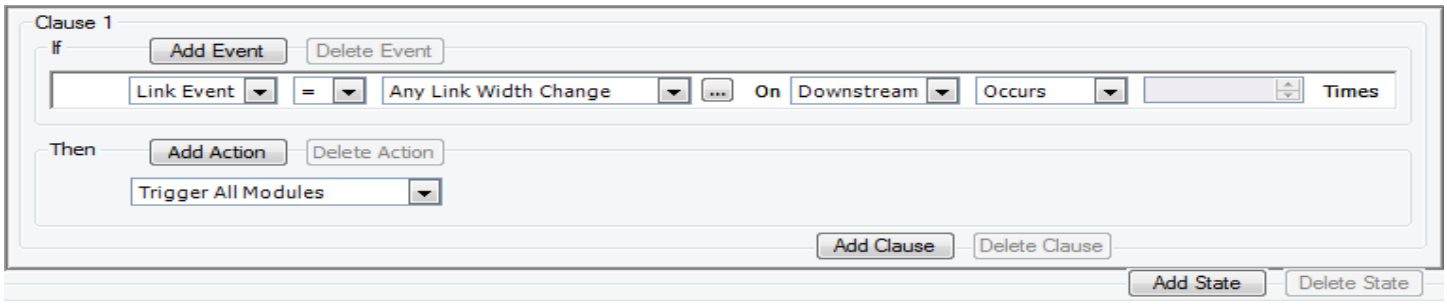


Figure 1A. Simple Link Width Change Trigger

Triggering on a specific width change requires a little more effort but is still fairly straightforward; the approach relies on a trigger resource called symbol sequence recognizers. The symbol sequence recognizers can be configured to trigger on any sequence of bytes or symbols on one or more lanes.

To trigger on a specific width change, create a two-state trigger with the symbol sequence recognizers configured for training sequences. The first state looks for a training sequence with the lane number set to 8. The second state looks for a training sequence with the lane number on lane 8 set to PAD. This indicates a link width change from a width of 8 or more to a width of less than 8. Figure 1B shows the two states along with the individual symbol sequence recognizer definitions.

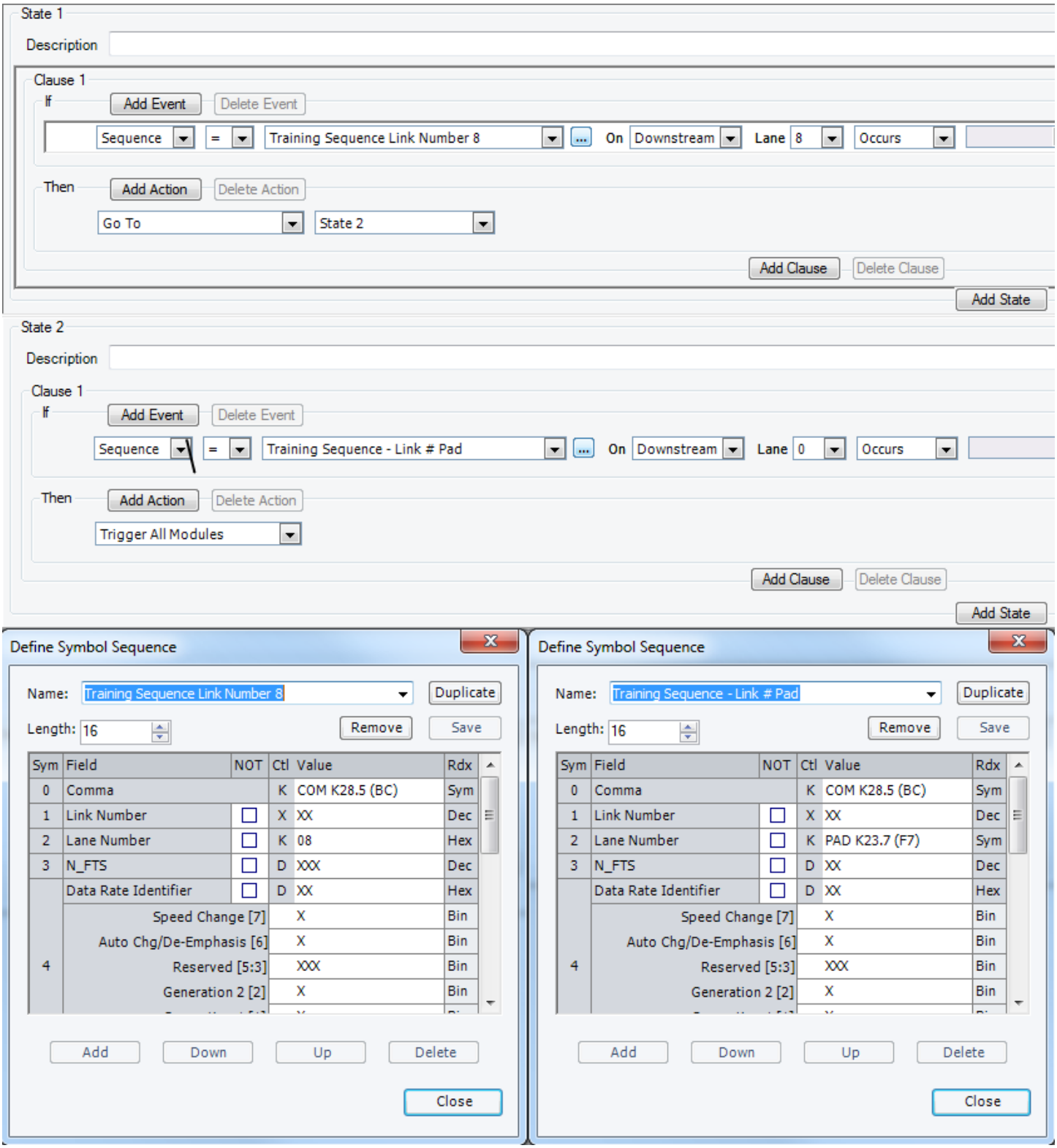
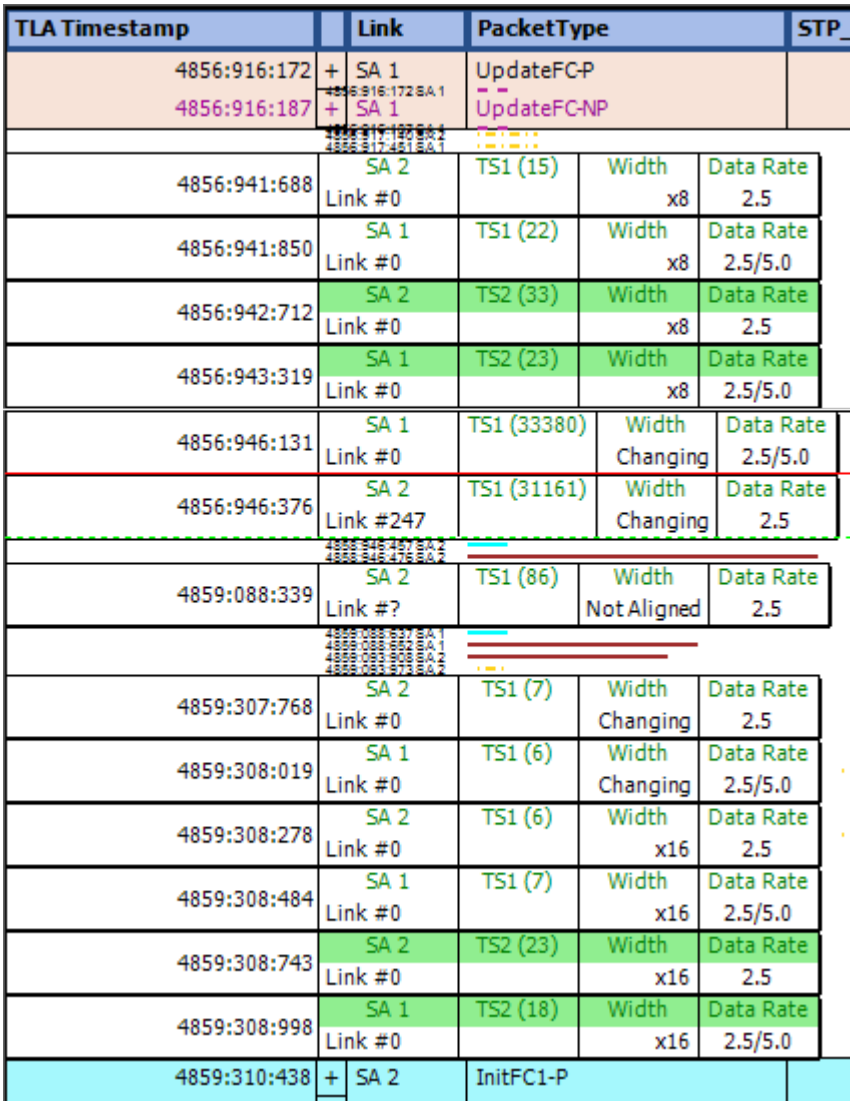


Figure 1B. Advanced Link Width Change Trigger

Triggering on the width change is the first requirement; seeing the width change is equally important.

To debug link width changes it is crucial to look at individual bytes on a given lane. Using the LPA's listing window is the most effective tool to display individual bytes on each and every lane. However, width change can potentially take many milliseconds to complete and this can result in pages and pages of listing window data

What is needed is a way to see a high level view of the width change and then easily drill down into the lane by lane, byte by byte information.



TLA Timestamp	Link	PacketType	STP_
4856:916:172	SA 1	UpdateFC-P	
4856:916:187	SA 1	UpdateFC-NP	

4856:941:688	SA 2 Link #0	TS1 (15)	Width x8	Data Rate 2.5
4856:941:850	SA 1 Link #0	TS1 (22)	Width x8	Data Rate 2.5/5.0
4856:942:712	SA 2 Link #0	TS2 (33)	Width x8	Data Rate 2.5
4856:943:319	SA 1 Link #0	TS2 (23)	Width x8	Data Rate 2.5/5.0
4856:946:131	SA 1 Link #0	TS1 (33380)	Width Changing	Data Rate 2.5/5.0
4856:946:376	SA 2 Link #247	TS1 (31161)	Width Changing	Data Rate 2.5
4859:088:339	SA 2 Link #?	TS1 (86)	Width Not Aligned	Data Rate 2.5
4859:307:768	SA 2 Link #0	TS1 (7)	Width Changing	Data Rate 2.5
4859:308:019	SA 1 Link #0	TS1 (6)	Width Changing	Data Rate 2.5/5.0
4859:308:278	SA 2 Link #0	TS1 (6)	Width x16	Data Rate 2.5
4859:308:484	SA 1 Link #0	TS1 (7)	Width x16	Data Rate 2.5/5.0
4859:308:743	SA 2 Link #0	TS2 (23)	Width x16	Data Rate 2.5
4859:308:998	SA 1 Link #0	TS2 (18)	Width x16	Data Rate 2.5/5.0
4859:310:438	SA 2	InitFC1-P		

Figure 2A. Transaction Window Width Change

The LPA transaction window provides the high level view allowing a user to see the entire width change on one screen; the listing window allows a user to drill down into individual training sequences of interest. Co-scrolling the two windows together makes it easy to keep track of the details within the high level view.

The transaction window capture of a x8 to x16 width change shown in Figure 2A, spans about 2.4ms. By filtering and collapsing repetitive ordered sets the user can see the entire width change. When the width change functions correctly, the display provides the necessary information, and the user can move on. For the high reliability system described at the beginning of this section the transaction window quickly shows how long the width change takes and some possibilities for optimization.

Timestamp	Uni_Up Link Details	Uni_Up Up0	Uni_Up Up1	Uni_Up Up2	Uni_Up Up3	Uni_Up Up4	Uni_Up Up5	Uni_Up Up6	Uni_Up Up7
-2.148,892,375 ms	**** TS1 - Not Aligned **********	COM	COM	COM	COM	No_Sig	No_DL>	No_Sig	No_Sig
-2.148,876,446 ms	Link No: Unknown (Not Aligned)	PAD	PAD	PAD	PAD	No_DL>	No_DL>	No_DL>	No_DL>
-2.148,875,509 ms	Lane Ordering: Unknown (Not Aligned)	PAD	PAD	PAD	PAD	No_DL>	No_DL>	No_DL>	No_DL>
-2.148,874,572 ms	N_FTS: 24 Dec	18	18	18	18	No_DL>	No_SL>	No_DL>	No_DL>
-2.148,873,635 ms	Data Rate ID: 06 Hex	06	06	06	06	No_DL>	No_SL>	No_DL>	No_DL>
	Gen 2 rate supported	------	------	------	------	------	------	------	------
	Gen 1 rate supported	------	------	------	------	------	------	------	------
-2.148,861,454 ms	Training Control: 00 Hex	00	00	00	00	No_SL>	COM	No_SL>	No_SL>
	Hot Reset: De-assert	------	------	------	------	------	------	------	------
	Disable Link: De-assert	------	------	------	------	------	------	------	------
	Loopback: De-assert	------	------	------	------	------	------	------	------
	Disable Scrambling: De-assert	------	------	------	------	------	------	------	------
	Compliance Receive: De-assert	------	------	------	------	------	------	------	------
-2.148,860,517 ms	TS1 Identifier	4A	4A	4A	4A	No_SL>	B5	No_SL>	No_SL>
-2.148,859,580 ms	TS1 Identifier	4A	4A	4A	4A	No_SL>	B5	No_SL>	No_SL>
-2.148,858,643 ms	TS1 Identifier	4A	4A	4A	4A	No_SL>	B5	No_SL>	No_SL>
-2.148,842,714 ms	TS1 Identifier	4A	4A	4A	4A	No_SL>	B5	No_SL>	No_SL>
-2.148,841,777 ms	TS1 Identifier	4A	4A	4A	4A	No_SL>	B5	No_SL>	No_SL>
-2.148,840,840 ms	TS1 Identifier	4A	4A	4A	4A	No_SL>	B5	No_SL>	No_SL>
-2.148,839,903 ms	TS1 Identifier	4A	4A	4A	4A	No_SL>	B5	No_SL>	No_SL>
-2.148,827,722 ms	TS1 Identifier	4A	4A	4A	4A	No_SL>	B5	No_SL>	No_SL>
-2.148,826,785 ms	TS1 Identifier	4A	4A	4A	4A	No_SL>	B5	No_SL>	No_SL>
-2.148,825,848 ms	TS1 Identifier	4A	4A	4A	4A	No_SL>	COM!	No_SL>	No_SL>
-2.148,824,911 ms	**** TS1 - Not Aligned **********	COM	COM	COM	COM	COM	K28.6	COM	COM
-2.148,812,730 ms	Link No: Unknown (Not Aligned)	PAD	PAD	PAD	PAD	K28.6	PAD	K28.6	K28.6
-2.148,811,793 ms	Lane Ordering: Unknown (Not Aligned)	PAD	PAD	PAD	PAD	PAD	18	PAD	PAD
-2.148,810,856 ms	N_FTS: 24 Dec	18	18	18	18	18	06	18	18
-2.148,809,919 ms	Data Rate ID: 06 Hex	06	06	06	06	06	00	06	06
	Gen 2 rate supported	------	------	------	------	------	------	------	------
	Gen 1 rate supported	------	------	------	------	------	------	------	------
-2.148,793,990 ms	Training Control: 00 Hex	00	00	00	00	00	AD	00	00
	Hot Reset: De-assert	------	------	------	------	------	------	------	------
	Disable Link: De-assert	------	------	------	------	------	------	------	------
	Loopback: De-assert	------	------	------	------	------	------	------	------
	Disable Scrambling: De-assert	------	------	------	------	------	------	------	------
	Compliance Receive: De-assert	------	------	------	------	------	------	------	------
-2.148,793,053 ms	TS1 Identifier	4A	4A	4A	4A	4A	48	4A	4A
-2.148,792,116 ms	TS1 Identifier	4A	4A	4A	4A	4A	C8	4A	4A
-2.148,791,179 ms	TS1 Identifier	4A	4A	4A	4A	4A	38	4A	4A
-2.148,778,998 ms	TS1 Identifier	4A	4A	4A	4A	4A	24	4A	4A
-2.148,778,061 ms	TS1 Identifier	4A	4A	4A	4A	4A	62	4A	4A
-2.148,777,124 ms	TS1 Identifier	4A	4A	4A	4A	4A	EC	4A	4A
-2.148,776,187 ms	TS1 Identifier	4A	4A	4A	4A	4A	F4	4A	4A
-2.148,764,006 ms	TS1 Identifier	4A	4A	4A	4A	4A	27	4A	4A
-2.148,763,069 ms	TS1 Identifier	4A	4A	4A	4A	4A	F5	4A	4A
-2.148,762,132 ms	TS1 Identifier	4A	4A	4A	4A	4A	COM	4A	4A
-2.148,761,195 ms	Not Aligned	COM	COM	COM	COM	COM	K28.6	COM	COM
-2.148,749,014 ms	Not Aligned	PAD	PAD	PAD	PAD	K28.6	PAD	K28.6	K28.6
-2.148,748,077 ms	Not Aligned	PAD	PAD	PAD	PAD	PAD	D8	PAD	PAD
-2.148,747,140 ms	Not Aligned	18	18	18	18	18	12	18	18
-2.148,746,203 ms	Not Aligned	06	06	06	06	06	B2	06	06
-2.148,730,274 ms	Not Aligned	00	00	00	00	00	AD	00	00
-2.148,729,337 ms	Not Aligned	4A	4A	4A	4A	4A	48	4A	4A
-2.148,728,400 ms	Not Aligned	4A	4A	4A	4A	4A	C8	4A	4A
-2.148,727,463 ms	Not Aligned	4A	4A	4A	4A	4A	38	4A	4A
-2.148,715,282 ms	Not Aligned	4A	4A	4A	4A	4A	24	4A	4A
-2.148,714,345 ms	Not Aligned	4A	4A	4A	4A	4A	62	4A	4A
-2.148,712,471 ms	Not Aligned	4A	4A	4A	4A	4A	EC	4A	4A
-2.148,700,290 ms	Not Aligned	4A	4A	4A	4A	4A	F4	4A	4A
-2.148,699,353 ms	Not Aligned	4A	4A	4A	4A	4A	27	4A	4A
-2.148,698,416 ms	Not Aligned	4A	4A	4A	4A	4A	F5	COM	4A

Figure 2B. Listing Window Width Change

If something interesting did occur and some of the ordered sets need to be analyzed in more detail, then the listing window shown in Figure 2B, provides decoded lane level data. By “locking” the transaction to the listing window, the user can quickly navigate to an area of interest using the transaction window and the system will display the details of the item in the listing window.

Rate Change Capture

The PCI Express specification allows two agents to negotiate speed on the fly. Providing visibility of these successful or unsuccessful speed changes is a strength of the Logic Protocol Analyzer. As with width changes, debug starts with powerful and easy to use triggering capabilities. A user can either trigger on a successful speed change with a simple trigger or the user can use symbol sequence recognizers to trigger on training sequences with the speed change bit asserted as part of a trigger.

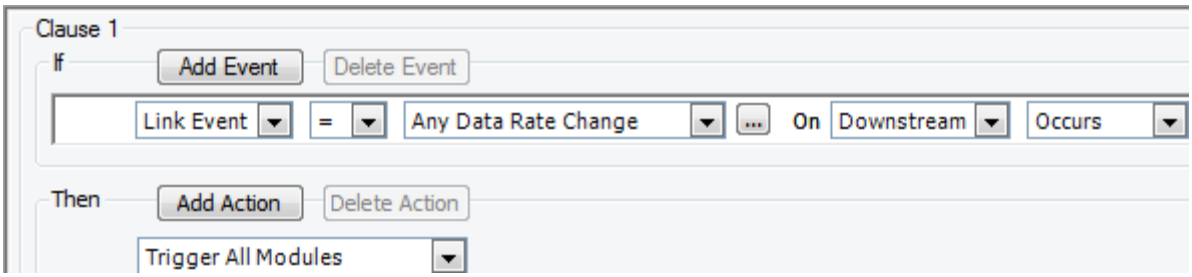


Figure 3A. Simple Rate Change Trigger

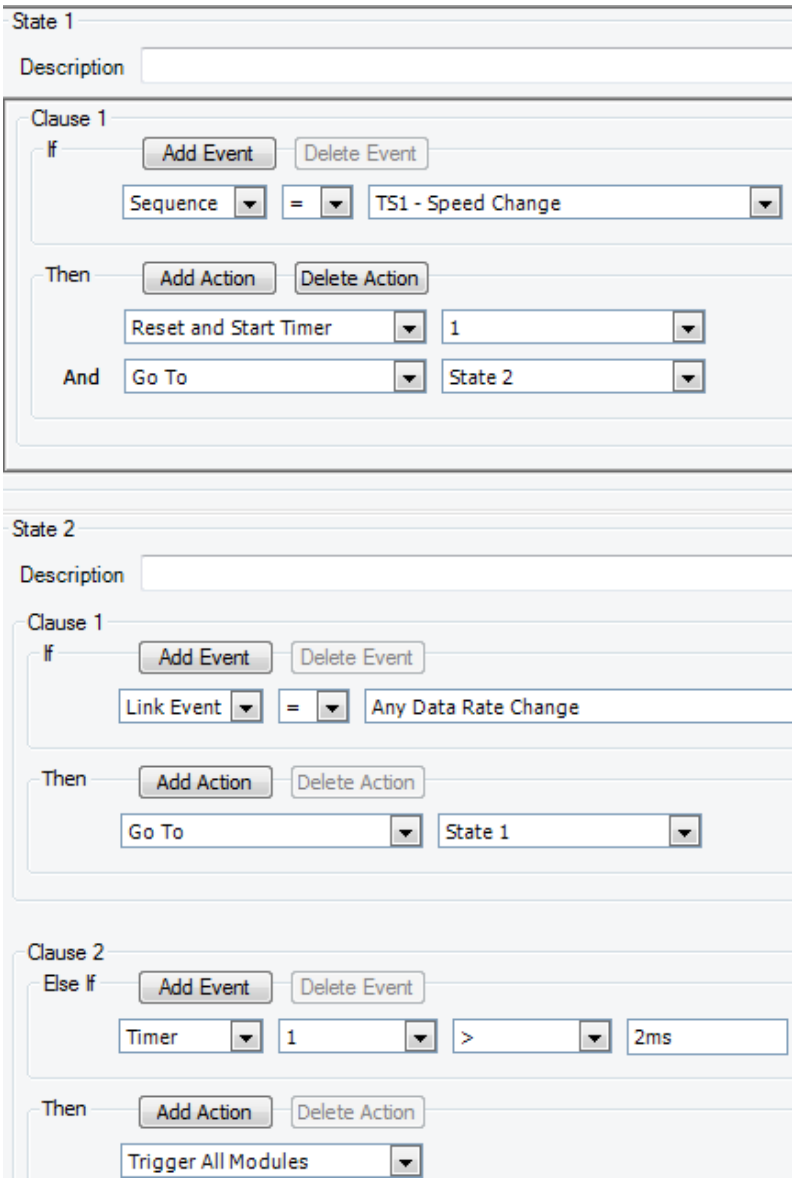
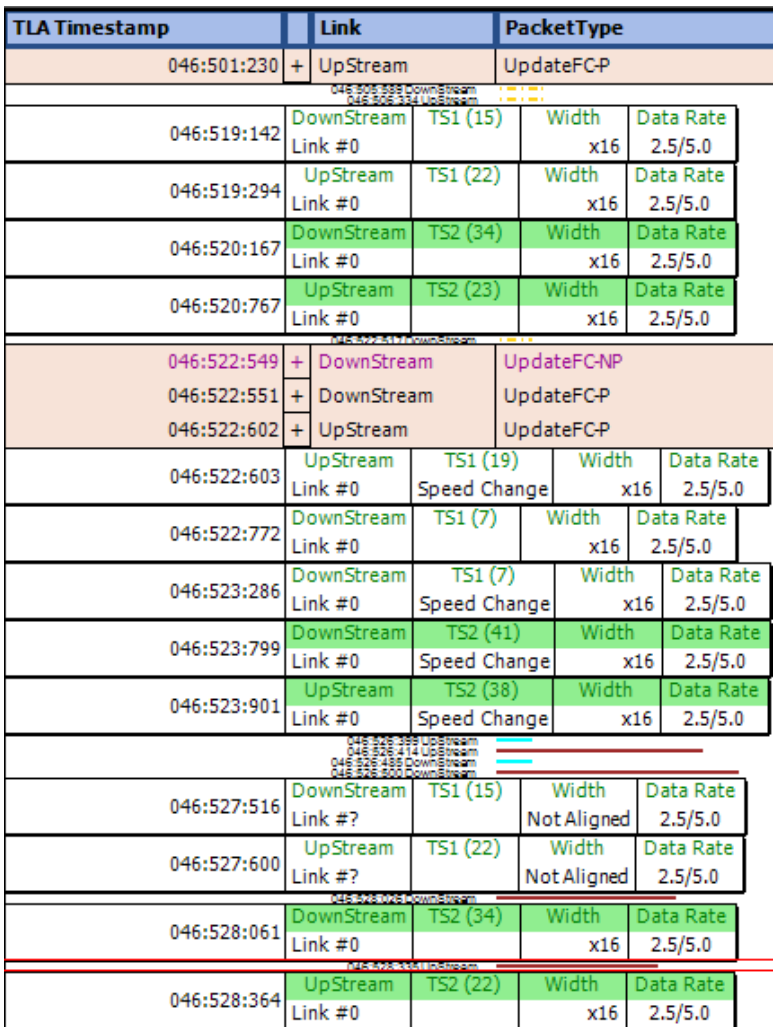


Figure 3B. Failed Rate Change Trigger

A user can also define a trigger to look for failed rate changes, one example of which is shown in Figure 3B. The trigger uses the rate change link event trigger shown above combined with a timer and a symbol sequence recognizer. This kind of trigger is very powerful when used in combination with automated test scripts to find issues that occur infrequently.

This particular trigger is a simple example using symbol sequence recognizers. Combining packet or ordered set triggers with multiple states, counters, and timers lets the user find a variety of complex issues inlcuding missed LTSSM transitions and timeouts among many others.



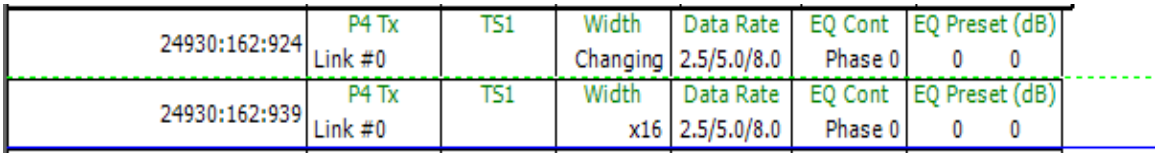
As with the width change example above, triggering on an event is the first requirement; seeing the data is equally important both at the highest level and at the lowest lane by lane level.

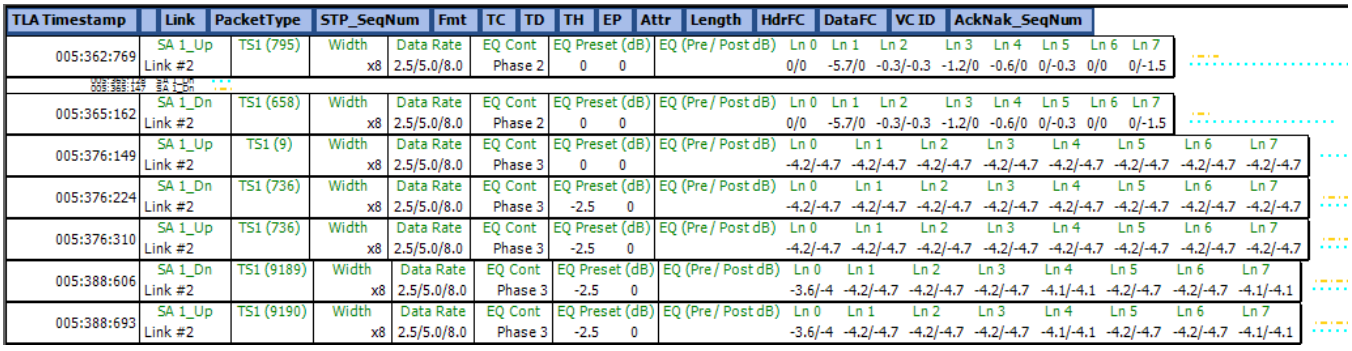
For many speed change issues, the transaction window shown in Figure 4 displays the level of detail needed to see the entire speed change. There are some problems (such as a failed speed change) that require lane level visibility and are better viewed in the listing window.

The complexities of link equalization negotiation introduced into PCIE 3.0 increases the relevance of lane elements such as preset hints. Actual cursor settings on each lane become critical pieces of information. Below is shown the listing window view of the link equalization transitioning from Phase 1 to Phase 2.

If the system under test is failing the rate change to 8Gb/s due to the equalization negotiation, a user may need to cross trigger an oscilloscope and verify the equalization settings are actually being applied.

Figure 4. Complete Rate Change in Transaction Window





Uni_Dn Link Details	Uni_ Dn0	Uni_ Dn1	Uni_ Dn2	Uni Dn3
**** TS1 ****************************>	1E	1E	1E	1E
Link No: 0 Dec	00	00	00	00
Lane Ordering:	00	01	02	03
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15	---	---	---	--
N_FTS: 94 Dec	5E	5E	5E	5E
Data Rate ID: 0E Hex	0E	0E	0E	0E
Gen 3 rate supported	---	---	---	--
Gen 2 rate supported	---	---	---	--
Gen 1 rate supported	---	---	---	--
Training Control: 00 Hex	00	00	00	00
Hot Reset: De-assert	---	---	---	--
Disable Link: De-assert	---	---	---	--
Loopback: De-assert	---	---	---	--
Disable Scrambling: De-assert	---	---	---	--
Compliance Receive: De-assert	---	---	---	--
EQ TS1: 39 Hex	39	39	39	39
Equalization Control: 01b	---	---	---	--
Reset EIEOS Interval: De-assert	---	---	---	--
Transmitter Preset: 0111b	---	---	---	--
Use Preset: De-assert	---	---	---	--
FS: 3E Hex	3E	3E	3E	3E
LF: 15 Hex	15	15	15	15
TS1 EQ Control: 8D Hex	8D	8D	8D	8D
Post-Cursor Coefficient: 0D Hex	---	---	---	--
Reject Coefficient Values: De-assert	---	---	---	--
Parity: Assert	---	---	---	--
TS1 Identifier	4A	4A	4A	4A
TS1 Identifier	4A	4A	4A	4A
TS1 Identifier	4A	4A	4A	4A
TS1 Identifier	4A	4A	4A	4A
TS1 Identifier	4A	4A	4A	4A
DC Balance	08	08	08	4A
**** TS1 ****************************>	1E	1E	1E	1E
Link No: 0 Dec	00	00	00	00
Lane Ordering:	00	01	02	03
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15	---	---	---	--
N_FTS: 94 Dec	5E	5E	5E	5E
Data Rate ID: 0E Hex	0E	0E	0E	0E
Gen 3 rate supported	---	---	---	--
Gen 2 rate supported	---	---	---	--
Gen 1 rate supported	---	---	---	--
Training Control: 00 Hex	00	00	00	00
Hot Reset: De-assert	---	---	---	--
Disable Link: De-assert	---	---	---	--
Loopback: De-assert	---	---	---	--
Disable Scrambling: De-assert	---	---	---	--
Compliance Receive: De-assert	---	---	---	--
EQ TS1: 3A Hex	3A	3A	3A	3A
Equalization Control: 10b	---	---	---	--
Reset EIEOS Interval: De-assert	---	---	---	--
Transmitter Preset: 0111b	---	---	---	--
Use Preset: De-assert	---	---	---	--
Pre-Cursor Coefficient: 06 Hex	06	06	06	06
Cursor Coefficient: 2B Hex	2B	2B	2B	2B
TS1 EQ Control: 8D Hex	8D	8D	8D	8D
Post-Cursor Coefficient: 0D Hex	---	---	---	--
Reject Coefficient Values: De-assert	---	---	---	--
Parity: Assert	---	---	---	--
TS1 Identifier	4A	4A	4A	4A
TS1 Identifier	4A	4A	4A	4A
TS1 Identifier	4A	4A	4A	4A
TS1 Identifier	4A	4A	4A	4A
TS1 Identifier	4A	4A	4A	4A

Figure 5. EQ Negotiation Phase 1 to Phase 2

The transaction window shows the link equalization negotiation in a concise view. Because a particular training sequence may be repeated 65536 times, the transaction window allows a user to collapse these ordered sets. In this way, an entire negotiation can be viewed in a single row, reducing the need to page through thousands of rows of data.

At times, though, a platform may have a problem lane for any number of signal integrity reasons. In this case the system must provide the user access to a detailed view of the different lanes to view the link equalization negotiation. Under these circumstances, the user takes advantage of the detailed, lane by lane display in the listing window, as shown in Figure 5.

The Tektronix Logic Protocol Analyzer allows a user to view the data in a variety of correlated views. From the lowest level waveform view with correlated oscilloscope captures all the way up to a summary of the entire 16GB capture, the user can choose which views meet the needs of their application. This is especially important for PCIe3 which has blurred the lines between protocol and physical layers.

Advanced Power Management Support

The PCI Express specification contains several power saving modes. These modes are increasingly important for all types of systems. Servers are looking to cut power costs and mobile applications need to save battery life. As a result, system designers are continuously improving and optimizing the use of these power saving modes. At the same time, these modes are increasingly the source of bugs and compatibility issues. Designers and debug engineers need a tool providing visibility into these issues to take full advantage of and debug power saving modes.

At the outset of any ASPM investigation, the tool should provide a high level understanding of the system behavior. Finding out which power states are being used by which agents and how often is very simple using the LPA summary profile window. This window shows a count of all packets and ordered sets in the acquisition. The summary profile window also displays small graphs (“sparklines”), plotting the total number of each event over the time of the acquisition. Sparklines help a user understand the overall efficiency of the system or which part of a very large acquisition may be interesting to drill down into.

Figure 6 shows a summary profile window capture of a system running through L0s and L1. This indicates L0s and L1 are both occurring on both sides of the link, and that L0s is happening more frequently than L1. This is shown by the counts and timing of power management DLLPs, training sequences, fast training sequences, and electrical idle ordered sets. The viewfinder allows the user to analyze any section of the trace and jump to the first packet or ordered set of any type in that section.

Protocol Element	In Viewfinder		In Total		Overview			
	Up	Dn	Up	Dn	Max Up	Up	Max Dn	Dn
Errors	0	0	0	0	0		0	
TLPs	829	838	1994	1577	217		223	
DLLPs	991	1353	2346	2477	199		381	
Ack	630	545	1444	991	192		186	
Nak	0	0	0	0	0		0	
PM	336	152	840	304	84		38	
PM_Enter_L1	0	0	0	0	0		0	
PM_Enter_L23	0	0	0	0	0		0	
PM_Active_State_Request_L1	336	0	840	0	84		0	
PM_Request_Ack	0	152	0	304	0		38	
InitFC	0	0	0	0	0		0	
UpdateFC	25	656	62	1182	7		195	
Vendor Specific	0	0	0	0	0		0	
Ordered Sets	1151	1332	3094	2939	258		263	
TS1	139	142	368	301	35		40	
TS2	228	228	593	490	68		57	
FTS	695	858	1895	1924	162		199	
EIOS	43	51	116	113	9		11	
EIEOS	0	0	0	0	0		0	
SKP	46	53	122	111	10		10	
SDS	0	0	0	0	0		0	

Figure 6. Summary Window View of L0s and L1 Traffic

Overcoming PCI-Express Physical Layer Challenges

Using the Tektronix Logic Protocol Analyzer

Once an area of interest is identified a user can view that area in increasing levels of detail using the transaction, listing and waveform windows. Figure 7 is an example of the LPA transaction window identifying a system making very inefficient use of L1 and L0s. In this view it is easy to see which agent requested the L1, how long the system spent in L1, Recovery and L0. The system exits L1 long enough to perform one memory write. This may be necessary on this system or it may be an opportunity for optimization. The images below demonstrate the strengths of both views: the listing window reveals any lane-level issues (such as possible errors), while the transaction window provides excellent data density to rapidly navigate to areas of interest.

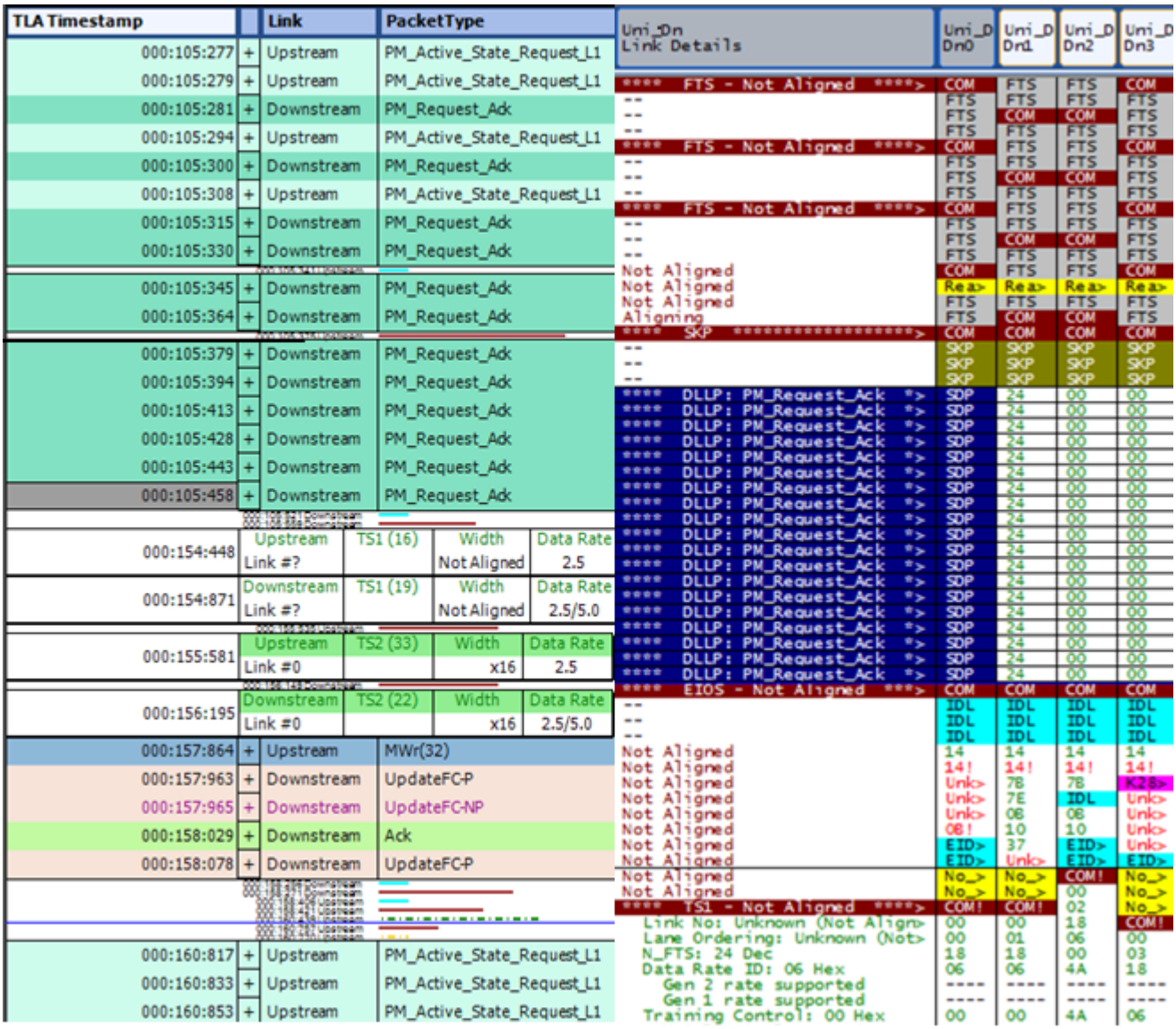


Figure 7. Transaction and Listing Window Views of L1 and L0s

Why is the Physical Layer Important to Me?

Selecting test equipment for your PCI Express application is a very important decision. Silicon designers need to validate potential physical layer issues like power management, rate changes, or width changes. Some designers may need to margin the reference clock in order to stress test the system. Some designers need to validate with spread spectrum enabled. Other users who may not be validating the physical layer at all still need a tool that captures physical layer information when the root cause of bugs stems from the physical layer. Even users debugging higher protocol level issues like transaction latency or flow control, need test equipment supporting all of the physical layer features their system enables. In this way they can quickly identify and dispense with bugs outside their area of responsibilities.

As demonstrated in this application note, the Tektronix PCIe Logic Protocol Analyzer supports all of these features in order to enable debug all of the way from the physical layer up to the transaction layer.

9/2011 54W-27364-0