WP350 (v1.2) October 28, 2014

Understanding Performance of PCI Express Systems

By: Jason Lawley

PCI Express® technology, which is a serialized point-to-point interconnect protocol, provides a high-bandwidth scalable solution for reliable data transport. Although PCI Express system transfer rates can seem extremely fast compared to those of its predecessor, PCI, users should understand the difference in raw bit rate movement versus data transfer performance.

This white paper explores factors of the PCI Express technology that can impact performance. It also provides guidance on how to estimate the performance of a system. This white paper includes information on PCI Express Gen 3, available in Xilinx UltraScale™ and Virtex®-7 XT/HT devices.

Performance Defined

The specified maximum transfer rate of Generation 1 (Gen 1) PCI Express systems is 2.5 Gb/s; Generation 2 (Gen 2) PCI Express systems, 5.0 Gb/s; and Generation 3 (Gen 3) PCI Express systems, 8.0 Gb/s. These rates specify the raw bit transfer rate per lane in a single direction and not the rate at which data is transferred through the system. Effective data transfer rate or performance is lower due to overhead and other system design trade-offs.

Various types of data, such as memory, I/O, or configuration data, move through a PCI Express system. Most designs focus on moving memory data as efficiently as possible. For the purposes of this white paper, *performance* is defined as the rate that memory data can be moved through the system. Hereafter, *data* means memory data.

While Gen 1 and Gen 2 share many similarities, Gen 3 introduced new concepts in order to continue to double the data rate with each new generation. This white paper looks at Gen 1 / Gen 2 implementations and explains key differences that first appeared in Gen 3.

Data Transfer Overhead

Any data movement through a PCI Express system includes a certain amount of overhead. This section examines the effects of symbol encoding, Transaction Layer Packets (TLP) overhead, and traffic overhead on the performance of a PCI Express system.

Symbol Encoding

The PCI Express Gen 1 and Gen 2 protocols use an 8B/10B encoding scheme on the transmission line to maintain DC balance. Unlike some performance factors, the encoding scheme cannot be adjusted to find an optimal setting; the 8B/10B encoding scheme is a requirement. Table 1 shows encoding examples for selected bytes of data. The entire 8B/10B encoding table is available in *PCI Express Base Specification, v2.0* [Ref 1].

Table 1: **8B/10B Encoding Example**

8-Bit	10-Bit +	10-Bit –
00000000	1001110100	0110001011
00000001	0111010100	1000101011
00000010	1011010100	0100101011
…	…	…
11111101	1011100001	0100011110
11111110	0111100001	1000011110
11111111	1010110001	0101001110

The 8B/10B encoding scheme guarantees a transition-rich data stream so that the receiving device can perform clock recovery on the incoming serial data. *Transition rich* means that for every 20 successive bits transferred, the difference in the number of 1s and the number of 0s cannot be more than two, and there cannot be more than five 1s or five 0s in a row.

To maintain DC balance, each byte transmitted has a positive-disparity 10-bit encoding and a negative-disparity 10-bit encoding. For each byte sent, the transmitter must determine whether to send the positive- or negative-encoded version of the byte. Therefore, to transmit one byte of data, 10 bits are transmitted on the serial link, resulting in a 20% loss in possible throughput due to the encoding. The result is generally referred to as the theoretical bandwidth of the link (Equation 1).

$$Theoretical\ Bandwidth\ (Bytes) = \frac{2.5\ Gb/s \times 2\ (two\ directions) \times Lane\ Width}{10\ bits/byte} \quad \textit{Equation 1}$$

The Gen 3 transfer rate was set to 8.0 Gb/s. To double the effective bandwidth from the Gen 2 rate of 5.0 Gb/s, a decision was made to use 128B/130B encoding, which has a 2% loss characteristic, instead of 8B/10B (20% loss characteristic).

Table 2 shows the theoretical bandwidth for various PCI Express link widths and speeds.

Table 2: **Theoretical Bandwidth, Full-Duplex (Gb/s)**

	Link Width			
	x1	**x2**	**x4**	**x8**
Gen 1	0.5	1.0	2.0	4.0
Gen 2	1.0	2.0	4.0	8.0
Gen 3	2.0	3.9	7.9	15.8

To summarize this section: Although Gen 1 transmission lines can operate at 2.5 Gb/s, 8B/10B encoding reduces effective bandwidth to 2.0 Gb/s per direction per lane. Similarly, Gen 3 can operate at 8.0 Gb/s using 128B/130B encoding, which reduces the theoretical bandwidth to 7.9. The actual system bandwidth is slightly lower due to packet and traffic overhead, producing an overall system efficiency 7.8 Gb/s per direction per lane.

Transaction Layer Packet Overhead

A PCI Express system transfers data in the payload of TLPs. Memory data is transferred in Memory Write TLPs and Completion with Data TLPs, which are responses to memory read operations. Figure 1 and Figure 3 show typical TLPs for Gen 2 and Gen 3, respectively.

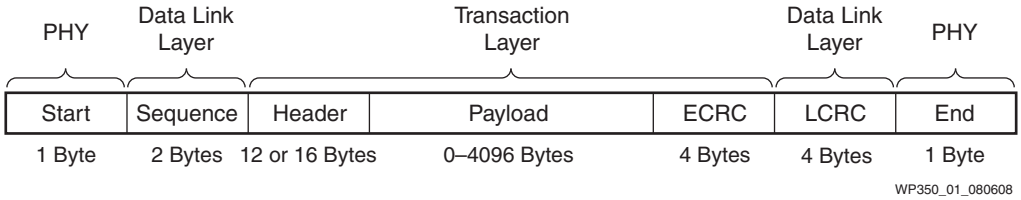

Figure 1: **Gen 2 Transaction Layer Packet**

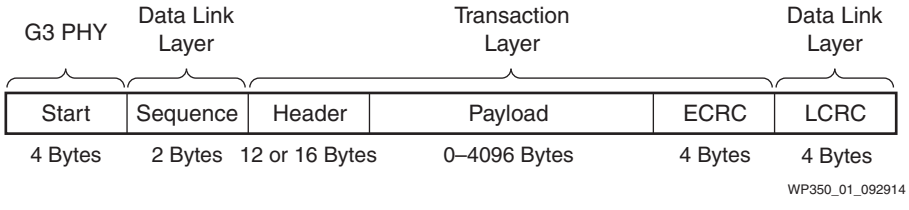

Figure 2: **Gen 3 Transaction Layer Packet**

Memory Request 32-Bit Addressing Header

+0								+1								+2								+3							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	X	0	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Requester ID																Tag								Last DW BE				First DW BE			
Address [31:2]																														00	

Memory Request 64-Bit Addressing Header

+0								+1								+2								+3							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	X	1	0	0	0	0	0	0	TC			0	0	0	0	TD	EP	Attr		0	0	Length									
Requester ID																Tag								Last DW BE				First DW BE			
Address [63:32]																															
Address [31:2]																														00	

WP350_03_092914

Figure 3: **Memory Request Header Format**

The transaction layer, data link layer (DLL), and physical layer (PHY) add overhead to each TLP, thereby reducing the effective data transfer rate. The transaction layer attaches the packet's header and optional end-to-end cyclic redundancy checksum (ECRC). The DLL adds the sequence number and Link Layer CRC (LCRC) to the packet to guarantee successful transmission across the link. The PHY adds information to mark the beginning and end of the packet.

TLP overhead varies between 20 to 30 bytes depending on Gen 2 or Gen 3 speed, the use of 32-bit or 64-bit addressing, and optional ECRC. Memory read or write TLPs can use either 32-bit or 64-bit addressing. The 64-bit addressable TLP header is 16 bytes (as opposed to 12 bytes for 32-bit addressing) and requires an additional 4 bytes of information to be exchanged in the packet. Figure 4 shows the memory request TLP Header format for 32-bit and 64-bit addressing.

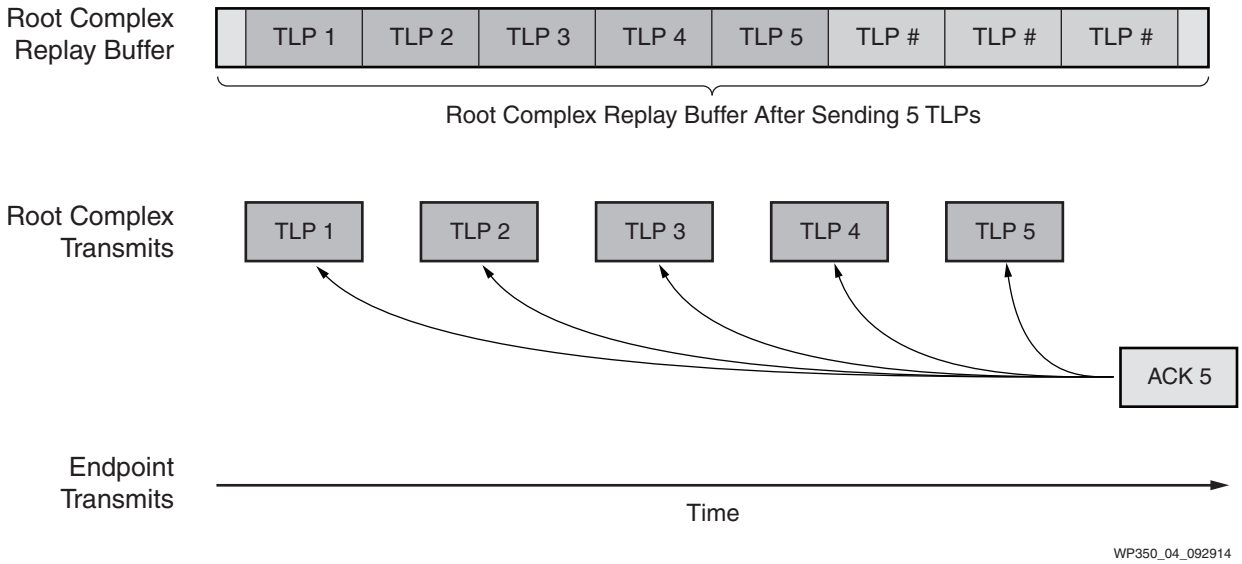

Figure 4: **Five Data TLPs and a Single Acknowledgment**

The transfer of large amounts of data requires multiple TLPs, each having its own overhead. Although each TLP contains a given amount of overhead, larger multiple TLP transfers increase link efficiency. The maximum payload size (MPS) setting, assigned to the communicating devices, determines the maximum TLP size. Increasing the MPS does not necessarily lead to a corresponding increase in link efficiency because as individual TLPs become larger, other factors such as traffic overhead begin to impact link performance.

Traffic Overhead

The PHY and the DLL both introduce traffic overhead. When the link reaches its normal operating state (L0), the PHY inserts skip-ordered sets to compensate for any difference in bit rates between the two communicating ports. Skip-ordered sets are four bytes in length for Gen 1 and Gen 2 speeds. For the Gen 3 speed, the skip-ordered set is typically 16 bytes in length. For Gen1 and Gen2, skip-ordered sets are scheduled for insertion at intervals between 1,180 and 1,538 symbol times. A symbol time is the amount of time it takes to place one byte (10 bits due to 8B/10B encoding) onto the serial lane. For Gen 3, skip-ordered sets must be scheduled to occur within 370 to 375 blocks. The PCI Express specification does not allow for ordered sets or other packets to be inserted into the middle of TLPs. This means that skip-ordered sets and other link management packets from the DLL can be transmitted only during gaps in TLP traffic. This requirement is one of the reasons why increasing the MPS does not result in an equivalent increase in link efficiency. As TLPs become larger, gaps between TLPs increase for link management activities.

The purpose of the DLL is to maintain reliable data transport between two link partners. To accomplish this goal, the PCI Express specification defines data link layer packets (DLLPs) that originate in and are consumed by the DLL layer. Various types of DLLPs are used for link management. The acknowledgment (ACK) DLLPs, non-acknowledgment (NAK) DLLPs, and flow control (FC) DLLPs have the most effect on performance.

Link Protocol Overhead

For every TLP sent from one device to another, an ACK or NAK must be returned to the sender of the TLP to indicate successful or unsuccessful reception of the packet. The transmitting device must hold the TLP in its replay buffer until an ACK is received for that TLP. This allows for the packet to be sent again if some problem occurred during the original transmission, ensures that no data is lost, and allows for a highly reliable link. Given the ACK/NAK nature of the protocol, when large numbers of TLPs are sent, significant numbers of ACK/NACK DLLPs are also generated, thus reducing the bandwidth of the link.

The protocol allows for multiple DLLPs of the same type waiting for transmission to be collapsed into single DLLPs as shown in Figure 5. For example, if five transmitted TLPs are all successfully received and if the receiver acknowledges the fifth TLP, it is assumed by the sender that all five TLPs were properly received. There are both pros and cons to collapsing ACKs on the link. Collapsing ACKs or NAKs reduces the amount of traffic overhead on the link. However, if ACKs are not sent frequently enough, the transmitter can throttle back on packet transmission while packets wait to be cleared from the replay buffer. The mechanism for determining when or when not to collapse multiple DLLPs into one is part of the design of the individual device's data link layer. If packets are not acknowledged quickly enough, the transmitting device's replay buffer fills and does not allow new packets to be sent until the older packets are acknowledged. This halts transmission of data on the line.

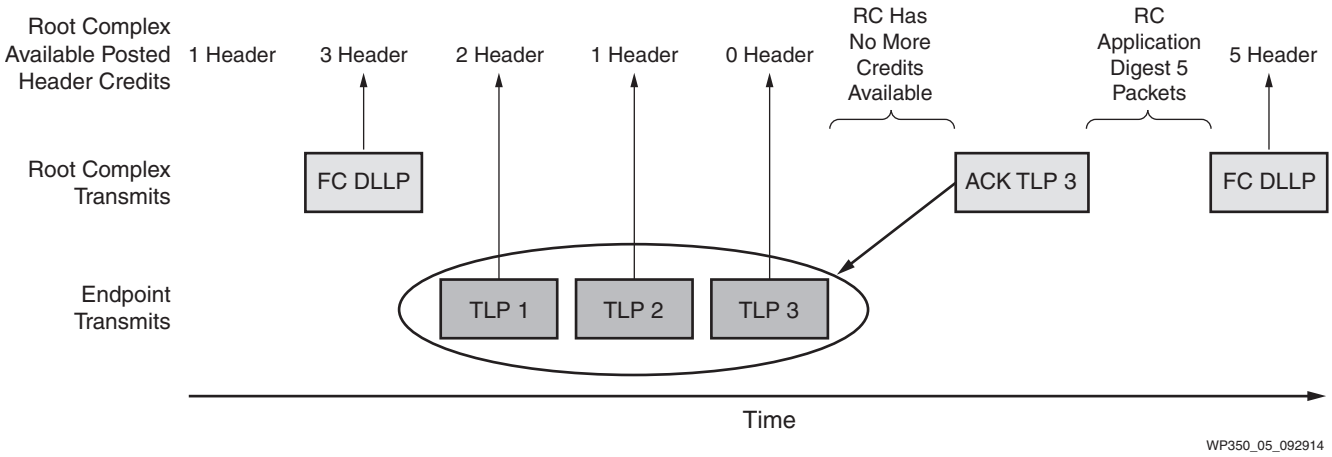

Figure 5: **Flow Control Update**

Flow Control Protocol Overhead

Credit-based flow control eliminates packet discards due to receive buffer overflow. While flow control is necessary and more advantageous than the older "retry" model from legacy PCI, it still has the potential consequence of throttling down the efficiency of the link. FC DLLPs sent by each link partner continually update the status of the device's receiver so that the transmitting device only transmits a packet if it knows that the receiver has sufficient buffer space. The transmitter keeps a running count of the number of credits available at the link partner's receiver and decrements this count each time a packet is sent, as shown in Figure 6. After the receiver processes the packet and frees buffer space, it transmits an FC Update DLLP, notifying the transmitting device of available space. The efficiency with which a device can process and transmit FC Update DLLPs impacts the overall performance of the link. Similar to how a device handles collapsing ACKs/NAKs, the mechanism used by a device to process and

transmit flow control updates is dependent on the device's design and can vary between devices. Again, there are both pros and cons based on the frequency at which flow control updates are sent. Less frequent flow control updates reduce link management traffic overhead. However, the receiver's buffers are larger to sustain reasonable performance.

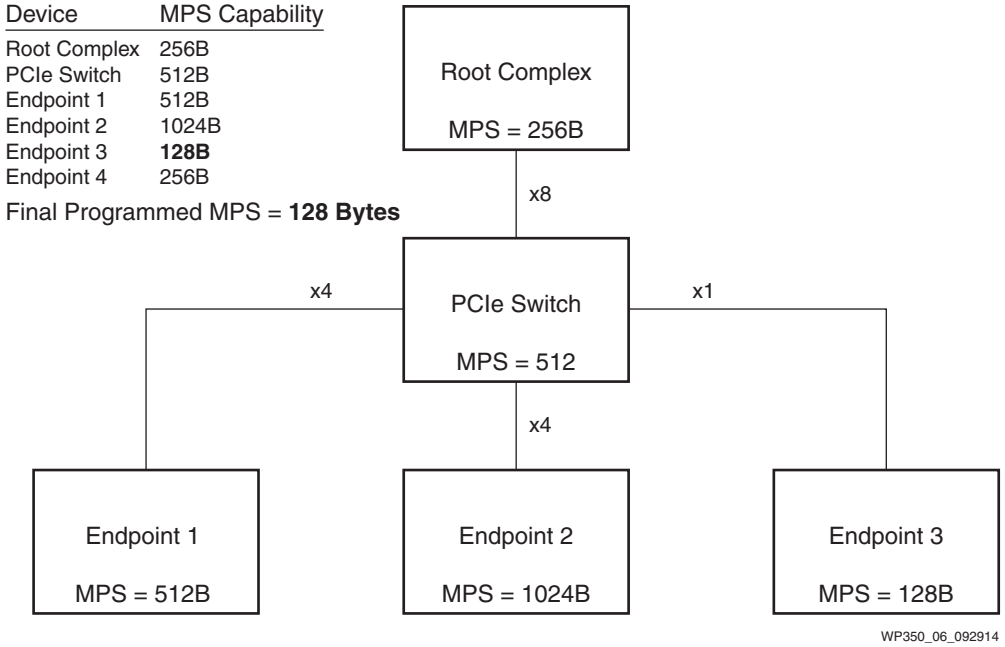

Figure 6: **Setting Maximum Payload Size**

System Parameters Affecting Performance

Aside from protocol and traffic overhead, there are other elements that impact performance in a PCI Express system. Three of these are maximum payload size, maximum read request size, and request type.

Maximum Payload Size

Although the PCI Express specification allows for payloads of up to 4,096 bytes, the specification says: "Software must take care to ensure that each packet does not exceed the Max_Payload_Size parameter of any system element along the packet's path." This means that every device in the hierarchy must use the same MPS setting, and the setting must not exceed the capability of any device within the hierarchy. Therefore, devices with high MPS capability are required to operate at lower MPS settings to accommodate the device with the lowest MPS capability. For example, the MPS of the PCI Express system in Figure 7 is programmed to 128 bytes to accommodate Endpoint 3.

A system's MPS setting is determined during the enumeration and configuration process. Every device in the hierarchy advertises its MPS capability in its Device Capability register, which is located in the device's configuration space. Software probes every device to determine its MPS capability, determines the MPS setting, and programs every device by writing the MPS setting to its Device Control register.

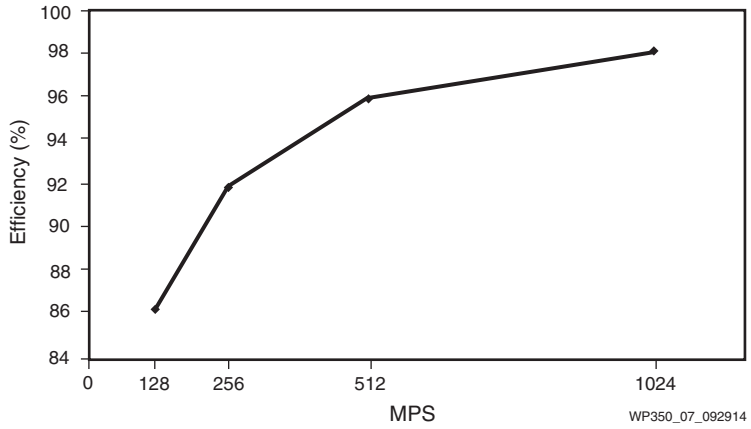

Figure 7: **Packet Efficiency as a Function of MPS**

The MPS setting determines the number of TLPs required to transfer a given amount of data. As the MPS increases, fewer TLPs are needed to transfer the same block of data. However, as Table 3 and Figure 8 show, increasing payload size does not increase efficiency at the same rate.

Equation 2 defines the packet efficiency.

$$Packet\ Efficiency = \frac{Maximum\ Payload\ Size}{Maximum\ Payload\ Size + Overhead} \qquad \text{Equation 2}$$

Table 3 shows packet efficiency calculations for four MPS settings. These examples assume a TLP with a 3 DW or 12-byte header and no ECRC, resulting in 20 total bytes of overhead.

Table 3: **Packet Efficiency Calculations**

MPS (Bytes)	Calculation	Packet Efficiency (%)	% Increase
128	128 / (128 + 20) = 86	86	-
256	256 / (256 + 20) = 92	92	6
512	512 / (512 + 20) = 96	96	4
1024	1024 / (1024 + 20) = 98	98	2

The theoretical maximum data throughput is the packet efficiency as a percentage of the theoretical bandwidth shown in Table 2. For example, with an MPS of 128 bytes, 86% of the bits transmitted on the link could potentially be bits of data. However, due to the traffic overhead discussed in "Traffic Overhead," page 5, this is not the case.

Figure 8 is a graphical representation of the packet efficiency calculations in Table 3.

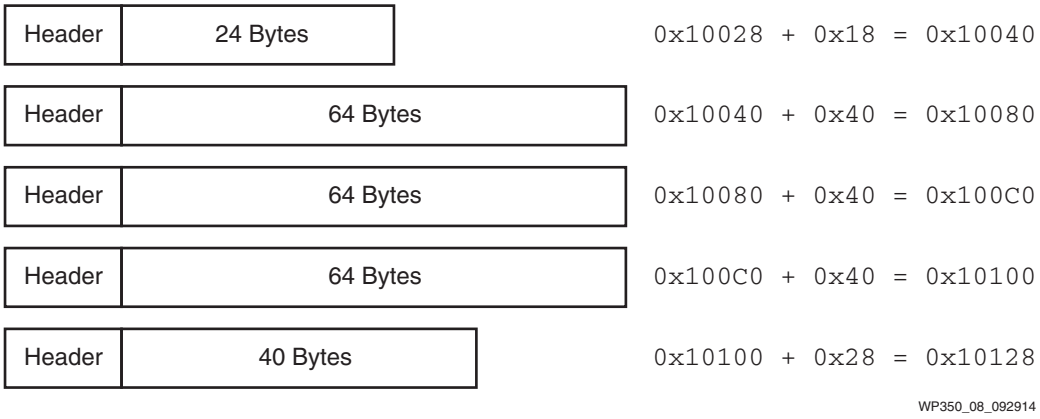

Figure 8: **Read Completions with the RCB Set to 64 Bytes**

Figure 8 shows that packet efficiency increases as MPS increases, but not at the same rate. For example, when MPS increases from 128 bytes to 256 bytes, packet efficiency increases 6%. When MPS increases from 512 bytes to 1024 bytes, packet efficiency only increases 2%. For most systems currently available, MPS tends to be set at 128 bytes or 256 bytes. Packet efficiency is fixed regardless of the transfer size for a given MPS.

For example, using a 128-byte MPS setting, a transfer of any size has a packet efficiency of 86%. For a x8 link, the half-duplex write throughput efficiency using the values in Table 2, page 3 is 1720 MB/s (0.86 × 2000 MB/s). Table 4 shows the link efficiency, also referred to as the theoretical maximum data throughput.

Table 4: **Theoretical Maximum Data Throughput in Mb/s**

Link Width	Write	Read
x1	1720	1520
x4	6880	6080
x8	13760	12160

Table 4 assumes an MPS of 128 bytes for writes. For reads, a Read Completion Boundary (RCB) of 64 bytes is assumed, meaning that the majority of the completion packets will contain 64 bytes of data. This example assumes that all reads are satisfied with a multiple of 64-byte completion with data packets. The theoretical maximum data throughput for reads is based only on the completion traffic. There is some associated latency with the initial read request, but because the read requests to move large blocks of data are generally sent in a stream, the initial latency is not much of a concern. In effect, the reads are pipelined and queued up in the host as the memory controller processes and returns the completions. The read efficiency is found by a similar formula to those in Table 3: 64 / (64 + 20) = 76%.

Maximum Read Request Size

During configuration, the software also programs the maximum read request size into each device's control register. This parameter sets the maximum size of a memory read request, which can be set to a maximum of 4096 bytes in 128-byte increments.

The maximum read request size can be larger than the MPS. For example, a 512-byte read request can be issued to a device with a 128-byte MPS. The device returning the

data for the read request limits the size of the Completion with Data TLP to 128 bytes or less, which requires multiple completions for one read.

The system uses the maximum read request size to balance the allocation of bandwidth throughout the topology. Limiting the maximum amount of data that a device can read in one transfer prevents it from monopolizing system bandwidth.

The maximum read request size also affects performance because it determines how many read requests are required to fetch the data, and read requests are 100% overhead because they do not contain any payload. Reading 64 KB of data using a maximum read request size of 128 bytes requires 512 Memory Read TLPs (64 KB / 128 bytes = 512) to request the data from memory. For efficiency when moving large blocks of data, the size of the read request should be as close as possible to the maximum read request size to reduce the number of reads that must be transmitted.

Read Completion Boundary

Completion with Data TLPs return data in response to a read request. The Read Completion Boundary (RCB) allows data from single read requests to be serviced by multiple completions. The data is divided into 64-byte or 128-byte completions that are naturally aligned with address boundaries. This means that the amount of data returned by some completions can be less than the RCB setting, depending on the location of the next address boundary. Typically, most root complexes set the RCB at 64 bytes and return data in 64-byte completions instead of what might be allowed by the MPS. Figure 9 shows how 256 (`0x100`) bytes read by an endpoint from address `0x00010028` are probably returned by a root complex with an RCB setting of 64 bytes.

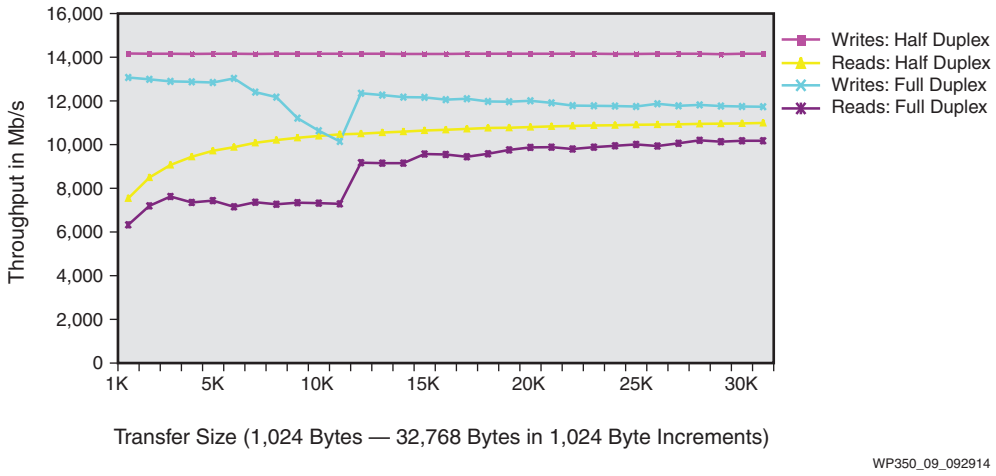

Figure 9: **Dell PowerEdge 1900 Server (Intel E5000P Chipset)**

Five Completions with Data are required to return 256 bytes of data. Each data TLP has a 3 DW header, so the packet efficiency of a single 64-byte packet is 76% (64 / (64 + 20) = 76%).

Many root complexes use a 64-byte RCB even though the MPS setting is 128 bytes or higher, which means that read completions are not only impaired by the time it takes to transmit the request, but also by the fact that the completions are divided into the smallest packages allowed by the RCB setting. See the example calculations in "Read Transaction Throughput," page 13 to better understand how the RCB affects bandwidth when other system parameters are considered.

Posted and Non-Posted Transactions

Non-posted transactions are requests that return a Completion TLP from the Completer, indicating that the transaction was successfully processed. The Completer is the device that is the ultimate destination of the request, not an intermediate device along the packet's route such as a switch.

These requests are non-posted transactions:

- Memory Read
- Memory Read Lock
- I/O Read
- I/O Write
- Configuration Read (both Type 0 and Type 1)
- Configuration Write (both Type 0 and Type 1)

For read requests, the Completer returns a Completion with Data TLP. If the Completer cannot satisfy the request, the Completer returns a Completion without Data TLP with a message indicating that the transaction was not successful. For non-posted write requests, the Completer returns a Completion without Data TLP that indicates whether the write was successful.

Memory Writes and Messages are posted transactions. These transactions do not require a Completion TLP from the packet's ultimate destination. Instead, the transmitting device assumes that the posted request is successful because the ACK/NAK DLLP protocol guarantees successful transmission of the packet through the system. If the Completer encounters an error when processing a posted request, the requester is not informed through a Completion TLP. However, in most cases, the Completer generates an error message informing the root complex that something has gone wrong.

Calculating Bandwidth

Considering the factors discussed so far, it is possible to calculate or at least get reasonable estimates of a system's performance. In general, bandwidth is the total amount of data transferred divided by the time to transfer the data. Overall bandwidth considers both writes and reads and is the sustainable value over time and not peak performance (Equation 3).

$$Bandwidth\ (BW) = [(Total\ Transfer\ Data\ Size)/(Transfer\ Time)] \times GB/s \quad \textit{Equation 3}$$

When doing calculations, it might be necessary to make reasonable assumptions about some of the parameters, such as how often Update FC DLLPs are sent. This information might be available in the device's literature or can be acquired using link analyzers to study the actual link traffic.

The three examples in this section do not consider the receiver buffer size or the transmitter replay buffer size, which factors in how often packets are transmitted on the link. The goal of these examples is to illustrate some of the parameters that users should consider when attempting to evaluate their performance expectations.

Write Transaction Throughput

This section shows two examples that calculate write transaction throughput. Values used in these calculations are not representative of any one system.

Example 1

In this example, 200 posted writes are transmitted at 128 bytes (MPS) each on a x8 link.

Step 1: Establish known parameters

These parameters are already known:

- The PCIe® link is running at 2.5 Gb/s

  1 byte is transmitted every 4 ns (Symbol Time)
- There are 8 lanes, meaning packets are distributed across each lane

  8 bytes can be sent every 4 ns

Step 2: Calculate TLP and DLLP transfer times based on known parameters

- 128-byte, 32-bit addressable Memory Write TLP transfer time:

  [(128 bytes payload + 20 bytes overhead) / 8 bytes/clock] * [4 ns/clock] = 74 ns
- DLLP transfer time:

  [8 bytes / 8 bytes/clock] × [4 ns/clock] = 4 ns

Step 3: Make reasonable assumptions about system parameters

These assumptions are made regarding system parameters:

- The TLP to ACK ratio is 1 ACK for every 5 TLPs.
- An FC update is issued every 4 TLPs.

Step 4: Calculate bandwidth

To calculate the bandwidth using Equation 3, the total bytes transferred and the transfer time are needed.

- Total bytes transferred:

  200 posted writes × 128 bytes/posted write = 25,600 bytes
- Transfer time:

  (200 × 74 ns) + (40 × 4 ns) + (50 × 4 ns) = 15,160 ns
- Bandwidth:

  [25,600 bytes / 15,160 ns] × [1 GB / 1s] = 1.689 GB/s or 1,689 MB/s

Example 2

In this example, 50 posted writes are transmitted at 512 bytes (MPS) each on a x8 link.

Step 1: Establish known parameters

These parameters are already known:

- The PCIe link is running at 2.5 Gb/s

  1 byte is transmitted every 4 ns (Symbol Time)
- There are 8 lanes, meaning packets are distributed across each lane

  8 bytes can be sent every 4 ns

Step 2: Calculate TLP and DLLP transfer times based on known parameters

- 512 byte, 32-bit addressable Memory Write TLP transfer time:

  [(512 bytes payload + 20 bytes overhead) / 8 bytes/clock] × [4 ns/clock] = 266 ns
- DLLP Transfer time:

  [8 bytes / 8 bytes/clock] × [4 ns/clock] = 4 ns

Step 3: Make reasonable assumptions about system parameters

These assumptions are made regarding system parameters:

- The TLP to ACK ratio is 1 ACK for every 5 TLPs.
- An FC update is issued every 4 TLPs.

Step 4: Calculate bandwidth

To calculate the bandwidth using Equation 3, the total bytes transferred and the transfer time are needed.

- Total bytes transferred:

  200 posted writes × 128 bytes/posted write = 25,600 bytes
- Transfer time:

  (50 × 266 ns) + (10 × 4 ns) + (13 × 4 ns) = 13,392 ns
- Bandwidth:

  [25,600 bytes / 13,392 ns] × [1 GB / 1s] = 1.912 GB/s or 1,912 MB/s

Summary of Examples 1 and 2

Examples 1 and 2 have the same transfer size but the MPS is increased by a factor of 4. The overall bandwidth does not scale by a factor of 4 because as the packets grow larger, it takes more time to transmit and process them. This assumes that the ACK and FC Update, transfer and processing times do not change. In reality, as the packets grow larger, DLLPs are sent less often because the protocol does not allow injecting DLLPs or ordered sets in the middle of TLPs. So as the packet grows larger, the inter-packet gap also increases to allow for DLLPs and skip ordered set transmission.

Read Transaction Throughput

As described in "Posted and Non-Posted Transactions," page 11, memory read transactions use the split-completion model. The requester sends the read TLP. After the completer fetches the data from memory, it sends the data back in the form of Completion with Data TLPs. The size of the completion is determined by a completer's read completion boundary. The requester can transmit multiple read requests in succession before completions start to return. Further, a single read request TLP might result in multiple Completion with Data TLPs, each incurring overhead and processing delay. In fact, it is normal for completions to be returned in groups of 64-byte or 128-byte TLPs depending on the RCB.

How the completions are distributed is a function of the device's design. The size of the payload in the completion TLP cannot be higher than the MPS value set for the device. So if the device is even capable of generating large completion with data TLPs, it still must break off the data into multiple TLPs if the MPS is lower than the device's capability.

Another major factor impacting read performance is the round-trip latency in returning the completion. For example, if the endpoint sends a read request to the memory controller, the time it takes to process the read, fetch the data, and generate the completion back to the endpoint must be considered.

Example 3

In this example, 4096 bytes of data are read on a x8 link where the maximum read request size is 256 bytes.

Step 1: Establish known parameters

These parameters are already known:

- The PCIe link is running at 2.5 Gb/s

  1 byte is transmitted every 4 ns (Symbol Time)
- There are 8 lanes, meaning packets are distributed across each lane

  8 bytes can be sent every 4 ns

Step 2: Calculate TLP and DLLP transfer times based on known parameters

- 32-bit addressable Memory Read TLP transfer time:

  [(20 bytes overhead) / 8 bytes/clock] × [4 ns/clock] = 10 ns
- DLLP transfer time:

  [8 bytes / 8 bytes/clock] × [4 ns/clock] = 4 ns
- RCB is 64 bytes, so the majority of completions are 64-byte TLPs.

  [(64 bytes payload + 20 bytes overhead) / 8 bytes/clock] × [4 ns/clock] = 42 ns

Step 3: Make reasonable assumptions about system parameters

These assumptions are made regarding system parameters:

- The TLP to ACK ratio is 1 ACK for every 5 TLPs.
- An FC update is issued every 4 TLPs.
- The receiver incoming read to completion generation time is 300 ns.

Step 4: Calculate bandwidth

To calculate the bandwidth using Equation 3, the total bytes transferred and the transfer time are needed.

- Total bytes transferred:

  4,096 bytes
- Transfer time:

  (16 × 10 ns) + (16 × 300 ns) + (64 × 42 ns) + (16 × 4 ns) + (20 × 4 ns) = 7,792 ns
- Bandwidth:

  [4096 bytes / 7,792 ns] × [1 GB / 1s] = 0.523 GB/s or 523 MB/s

Summary of Example 3

The calculation in Example 3 results in a throughput of 523 MB/s, which is lower than one might expect. The main reason is due to the assumption that the memory controller is not pipelining the reads and can only process one read at a time, resulting in one completion every 300 ns. Most likely, a real world system design will pipeline

the reads on the memory controller, resulting in completions being returned with greater efficiency. This example emphasizes that the memory controller can have a large impact on the overall performance.

Conclusion

The PCI Express specification defines many great capabilities that allow a link to perform with very high throughput and efficiency. At the same time, however, there are factors that impact performance, such as packet and traffic overhead, and other programmable functionalities, such as MPS. While some aspects of the protocol cannot be changed, such as the serial line encoding and the necessity of DLLPs and ordered sets, other aspects, such as device selection, device capability, and device interaction, can have performance and efficiency impacts. System and application design that considers all of these elements can achieve very high performance given the right expectations.

References

1. PCI Express Base Specification, v2.0
http://www.pcisig.com/specifications/pciexpress.

Related Xilinx Publications

XAPP1052, *Bus Master DMA Reference Design for the Xilinx Endpoint Block Plus Core for PCI Express*.

Additional Resources

The following links provide additional information useful to this document:

- PCI Express Protocol Solutions
http://www.xilinx.com/pcie.
- Kintex UltraScale FPGA KCU105 Evaluation Kit
http://www.xilinx.com/products/boards-and-kits/KCU105.htm
- Kintex-7 FPGA KC705 Evaluation Kit
http://www.xilinx.com/support/
- Virtex-7 FPGA VC709 Connectivity Kit (Gen 3 PCIe)
http://www.xilinx.com/support/
- Artix-7 FPGA AC701 Evaluation Kit
http://www.xilinx.com/support/
- Virtex-7 FPGA VC707 Evaluation Kit
http://www.xilinx.com/support/
- Spartan-6 FPGA Connectivity Kit
http://www.xilinx.com/support/

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
07/28/08	1.0	Initial Xilinx release.
09/04/08	1.1	Clarified the byte size of the Header field in Figure 1, page 3. Corrected the TLP header sizes in the second paragraph under Figure 1.
10/28/14	1.2	Added PCIe Gen 3 content throughout document. Added Figure 2. Moved all performance data to XAPP1052, Bus Master DMA Reference Design for the Xilinx Endpoint Block Plus Core for PCI Express.

Notice of Disclaimer

Automotive Applications Disclaimer