THE I/O INTERCONNECT SOLUTION

PCI Express Packet Latency Matters

January 15, 2007 **Version 1.0**

1 Introduction

Latency is the delay between starting and completing an action. For a switch, it's the time between the first bit of a packet on an input pin and the first bit of that packet on an output pin forwarded through the switch.

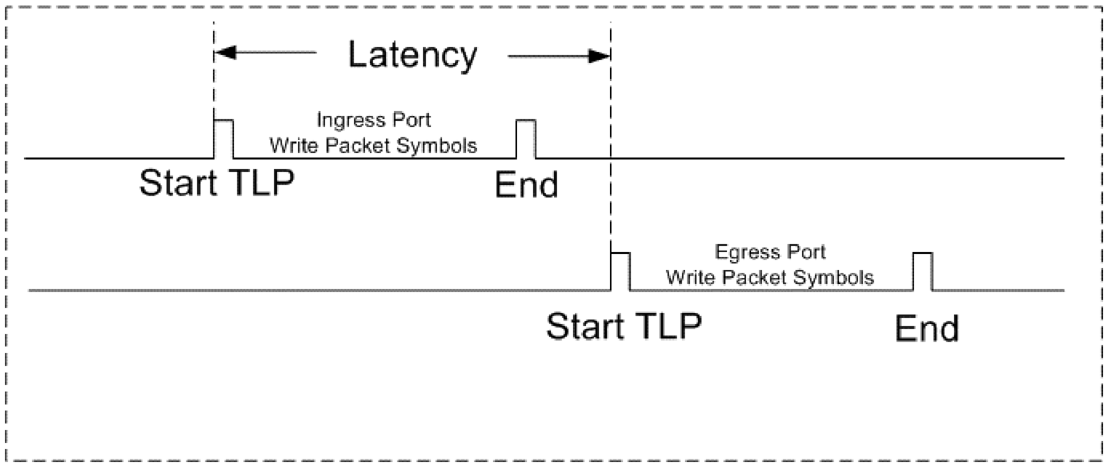

Figure 1: Latency Definition

Latency matters when it affects higher level functional throughput. When looking at the action of forwarding a packet through a switch, one can conclude that high throughput can be achieved in a switch independent of its switch latency. However, sending a packet through a switch is always part of a response to a higher level sequence of actions and the cumulative delay of that sequence may indeed affect the throughput of the higher level operation of which it is a part.

A key example of this is a CPU centric system. In a CPU-centric system, the processor performs a sequence of reads and writes to its memory and I/Os directly or through DMA. One downside of packet latency in this system is that some otherwise productive resource of a system must idle during the latency period, waiting for either data upon which to do work or for instructions as to what work to do.

1.1 Latency Sensitivity of Reads

A read is generally considered to be a blocking operation in that once a read request is initiated no additional instructions in thread of processing can be undertaken until it is completed. Simple applications have the following work flow:

1. Make a read request
2. Wait for data
3. Process the data
4. Loop back to 1

In this simple example, the latency of the read directly affects the throughput. If the read latency is much smaller than the processing time, then latency isn't a problem. When it's not small, users look for ways to mask the latency by doing useful work during it. A multithreaded processor could switch threads, for example, doing some other work during the latency. Optimizing compilers issue the read early to minimize the wait.

Bus interface units usually have an ability to issue multiple read requests before being forced to wait for a completion. If, for example, N outstanding read requests are supported, and the completion to the first read request arrives before the N^{th} read request is sent, then latency is said to have been masked and full throughput can be achieved after that initial waiting period. In practice, devices have varying degrees of ability to mask latency. In a system such as a PC or server where there is no control as to what is plugged into an open slot, latency is always an issue.

1.2 DMA I/O and Read Latency

The DMA I/O subsystem at the heart of PCs and servers is inherently latency-sensitive. I/O is accomplished using a DMA controller in each I/O device to move data between it and main memory located next to the CPU. The DMA controller follows a chain of descriptors located in memory. Each descriptor describes a unit of work assigned to the DMAC, requiring the DMAC to move a block of data from itself to memory or from memory to itself. The DMAC reads a descriptor, then assigns a DMA engine to do the data movement dictated by the descriptor. While the data is being moved, it reads the next descriptor. If the DMA engine completes its assignment before the next descriptor read completes, it is forced to idle for lack of work. Typically, the workload for DMA consists of a mix of short and long data blocks (packets) to be moved to and from memory. When the data block is relatively long, latency is masked. For short blocks, such as those used for Ethernet control packets, descriptor read latency leads to a loss of throughput. To avoid this, a sophisticated DMAC may read several descriptors ahead and maintain a cache of prefetched descriptors. However, there is always a limit to the size of the cache and to the number of descriptors available to be prefetched. A particular device may be capable of masking descriptor read latency when directly attached to a North Bridge (NB) but its throughput may suffer when it is connected to the NB through a switch. System designers are best advised to use the lowest-latency switches available to maximize the performance of their I/O subsystems.

1.3 Accelerators and Switch Latency

An increasingly common usage model is the attachment of multiple accelerators to a processor complex to increase performance in certain applications. Examples are the use of graphics processors and floating-point acceleration. In the accelerator model, the host processor offloads a computation to the accelerators, and then waits for the result. It may or may not have useful work to perform while waiting. The amount of time it waits is the sum of:

1. Synchronization time at start of computation
2. Time for accelerator to read data from memory
3. Time for accelerator to produce the result
4. Time for accelerator to write the result back to memory
5. Synchronization time at end of computation

Each of these operations, except for the computation time itself (#3) includes the interconnect latency. All the usual techniques of masking latency with concurrency apply. Nevertheless, when you consider that typical accelerators operate in the GHz range while interconnect latency is generally greater than 150 nanoseconds; you can see it is necessary to offload a relatively long computation in order to gain throughput. Each incremental reduction in switch or interconnect latency increases the feasibility of using accelerators to enhance the throughput the systems.

1.4 Summary

Given enough time and resources, engineers can usually figure out how to mask any fixed amount of latency. Often this effort consumes most of their development time and contributes significantly to the end cost of their product. No more dramatic example of this exists than the die area consumed by cache memories, cache controllers and support for multiple threads on high-end microprocessor chips.

Efforts to mask latency achieve varying degrees of success. Systems can have varying degrees of latency, which can additionally show temporary dependency to traffic loading upon any given link in the network. In practice, some devices may show latency sensitivity in some slots of some systems, but not in others. These disturbing observations lead to additional engineering effort to identify root cause and develop ways to reduce sensitivity to latency, thus delaying product time to market.

The availability of low latency switches makes the job of Systems Designer and every other component and process reliant on the PCIe infrastructure for data delivery easier. PLX Technology recognizes the important system differences between throughput and latency and why it matters to our customers. With our industry leading low latency of 110ns for PLX's Altair (PEX 8548, PEX 8547, PEX 8533 and PEX 8525) family of switches, PLX should be the first choice of system engineers interested in producing high performance designs.