Motivating Future Interconnects: A Differential Measurement Analysis of PCI Latency

David J. Miller[*] Philip M. Watts Andrew W. Moore

Computer Laboratory
University of Cambridge
Cambridge, CB3 0FD, UK

ABSTRACT

Local interconnect architectures are at a cusp in which advances in throughput have come at the expense of power and latency. Moreover, physical limits imposed on dissipation and packaging mean that further advances will require a new approach to interconnect design. Although latency in networks has been the focus of the High-Performance Computing architect and of concern across the computer community, we illustrate how an evolution in the common PCI interconnect architecture has worsened latency by a factor of between 3 and 25 over earlier incarnations.

Keywords

Experimental evaluation, FPGA, interconnects, latency, microbenchmark, measurement, PCI Express

1. INTRODUCTION

Demand for performance is constantly driven by new applications. It applies to all aspects of computer systems, from disk to processor, and the internal interconnect is no exception. Alongside a long-considered possible energy benefit [1], recent photonic advances have provided enabling technologies for both improved interconnects and perhaps also the whole-sale re-evaluation of interconnect design.

Wholly-photonic systems have for some time been promoted as specialist interconnects such as for HPC and even the back-planes of Internet routers [2]. SWIFT [3] and Data Vortex [4] illustrate novel approaches to LAN and HPC interconnects enabled by new photonic-electronic devices and using techniques such as the wave-length striping of data. Each of these systems was architected to embrace the bufferless nature of photonic interconnects, turning that to an advantage.

More recently still, the potential for photonic interconnects further includes high-speed on-chip interconnects [5], and an exciting early-day optical PCI Express experiment [6].

[*]Contact author address: `david.miller@cl.cam.ac.uk`

Aided by advances such as practical polymer waveguides [7, 8] as well as ongoing developments on optical-electronic integration we can reconsider interconnect architectures throughout the spectrum of computer-systems. However, the naïve reuse of existing photonic-systems, such as long distance communications techniques, CRC error detection, and their use of buffers to mediate flow control [1], may compromise the actual performance gained [9, 10].

Understanding the change in latency across a number of evolution-points for the PCI interconnect provides the motivation for this work. We focus on PCI Express, the latest incarnation of the PCI family.

A trend noted five years ago observes that throughput improves significantly faster than latency [11]. Accordingly, we show in this paper that instead of merely improving more slowly, PCI Express actually made latency much worse compared with its antecedents.

Traditionally, there has been a substantial difference in performance between peripherals (notably disk and network) and the local interconnect used to attach them to a host. This is now no longer always the case. 10 Gb/s Ethernet to a server is not unusual, and throughputs of 10 Gb/s have been in use by the HPC community since early this decade. Where peripheral characteristics once dominated latency, interconnect latency may now be a significant factor.

Community knowledge[1] as well as research such as Endo *et al.* [12] has shown that certain applications are more sensitive to latency than others — especially those which are interactive or transactional in nature, where responsiveness is dominated by latency. For example, recent novel use of graphics processors has been made in scientific computing [13], the performance of which greatly depends on communications latency, just as it does in parallel applications [14]. Although some of these references are over 10 years old, the issues which they address are no less relevant today, and it seems that very little related work has been done in the intervening time.

Further, the success of algorithmic share trading applications depends heavily on how quickly quotes are received and trades made, and financial markets demand faster link speeds[2] — in part, because of an expectation of lower latencies [15]. Finally, HPC interconnects go to great lengths (such as the use of cut-through routing) to minimize latency [16]. Therefore, it is unfortunate that the PCI in-

[1] "It's the latency, stupid", Stuart Cheshire, May 1996. `http://rescomp.stanford.edu/~cheshire/rants/Latency.html`
[2] "Stock exchanges start thinking in microseconds", Patrick Thibodeau, August 2008. `http://tinyurl.com/Thi08`

	Combinational Logic	Registers
PCI-X mixed mode	2500	1500
PCI Express 4-lane	10500	9000
Factor increase	$\times4$	$\times6$

Table 1: End point resource utilization compared

terconnect which interfaces an InfiniBand adapter will relinquish (hard-won) latency gains by being packet oriented and employing a store-and-forward switching architecture.

This paper presents the results from a study into the relative latency intrinsic to different forms of PCI architecture. The study reveals that latency suffers in PCI Express — just at the time when the importance of latency has become significantly greater.

2. LATENCY STUDY

The radical changes PCI Express made solved throughput and physical implementation issues (notably PCB layout and skew) using inexpensive existing technologies, such as high speed serial transceivers, 8b10b coding, scrambling and CRC error checking. We show that certain compromises were made, however, which may have adversely affected latency.

Architecturally, PCI Express is a message passing protocol, and shares much in common with Gigabit Ethernet. If not directly derived from Ethernet, it was certainly inspired by it — to the extent that it even uses the same coding and CRC error checking mechanisms.

The parallel PCI standards all use parity to detect transmission errors. This means that a PCI target can begin decoding, allocating and scheduling a transaction as soon as the transaction arrives because it can be confident that the header is error-free. PCI Express must wait until the whole packet has arrived, which means the packet must be buffered in scratch space until the receiver can verify its integrity. Conversely, the transmitter must *also* buffer the packet, in case retransmission should be required.

Where PCI-X targets could choose between split and immediate completions (described in Section 4.2), in PCI Express all transactions must be split into separate messages so buffering and serialization overheads are doubled.

As also described in Section 4.2, high speed serial transceivers, or multi-gigabit transceivers (MGTs) can potentially add significant latency over and above serialization delay. Clock domain crossing, coding and equalization (if any) can each also add delay. Latency in modern MGTs is better than in earlier MGTs, but the aforementioned synchronous components mean that an MGT can never be lower latency than conventional I/O.

PCI Express uses a layered protocol stack (described in Section 4.2). The longer pipeline formed by the protocol processing logic (*i.e.*, greater complexity) demands more complex logic which, aside from greater power dissipation, potentially means greater processing delay. Table 1 shows that a typical FPGA based PCI Express implementation uses around 5 times more logic than that of a mixed mode PCI-X implementation. The trend in general is for PCI Express devices to require a heat sink, where the older PCI-X equivalents didn't. For example, compare the Intel 82546 PCI-X and Intel 82571 PCI Express Gigabit Ethernet controllers: the data sheet for the latter demands a heat sink, where the former does not [17, 18]. Higher power dissipation may suggest increased overhead and therefore latency associated with the implementation of a PCI Express end point.

The rest of this paper presents a comparative investigation into the latency characteristics of PCI Express links, parallel PCI buses and their respective implementations.

2.1 Apparatus

A modern PC typically consists of memory, processor(s) and peripherals connected together with a chipset. Intel chipsets (as used here) typically consist of two multifunction devices known as the MCH (Memory Controller Hub, aka North Bridge) and the ICH (I/O Controller Hub, aka South Bridge). The MCH integrates a small number of higher performance device controllers, typically memory, CPU and a link to some sort of high performance disk or graphics accelerator. The ICH integrates a larger number of lower-speed devices, typically individual disks, USB, sound, network, and input devices.

PCI forms a hierarchy of buses, each attaching up to 32 devices, each of which contains up to eight independently addressable PCI targets known as 'functions'. In PCI terminology, a bridge is a function that attaches to exactly two buses. Bridge chips, such as the PXH ([PCI Express to] PCI-X Hub), frequently contain multiple PCI bridges.

The motherboard used in this study is a Super Micro X7SBE, which is based on the Intel 3210 (rev 01) + ICH9R (rev 02) chipset. PCI-X bus connectivity is provided by a 6700PXH (rev 09). The CPU is an E8400 (3.0 GHz core, 1333 MHz FSB, stepping 0a). The operating system is 32-bit Linux Ubuntu 8.10 running kernel 2.6.28.3. Two FPGA based end points are used (one PCI-X and one PCI Express) to provide a measurable target on the external buses and links.

Figure 1 shows the CPU, main memory (SDRAM), and ICH connected to the MCH. The link between the MCH and ICH is called the Direct Media Interface (labeled D), and is a (proprietary) variant of a 4-lane PCI Express v1.1 link. The MCH provides two 8-lane PCI Express v1.1 links (labeled E_a and E_b). As modern Intel chipsets provide native support only for PCI Express, E_a is used to provide PCI-X connectivity by means of a PXH bridge chip. PCI-X can operate at 133 MHz (labeled X_h), 66 MHz (labeled X_l) and in basic (conventional PCI) mode at 33 MHz (not shown). Note that, although the PCI-X bus is nominally 64 bits wide, bus control phases are always 32-bit, and only data phases can be 64 bit [19, 20]. Since this study involves DWORD reads, throughput calculations consider the bus to be 32 bits wide.

The MCH contains two 8-lane PCI Express bridges (labeled Ma and Mb). The SDRAM controller (labeled Ms) — the only other addressable device within the MCH — is included to complete the picture of timing relationships between MCH devices. Devices in the ICH present to software as being attached to the same logical bus as devices in the MCH so, together, they provide timing information for the near side of the DMI link (labeled D).

There are two bridges within the PXH (attached via Ma and E_a), but only one is used and shown (labeled Bb). The FPGA based PCI-X target attaches via Bb, and can run in three modes — 133 MHz PCI-X (labeled Tx_h), 66 MHz

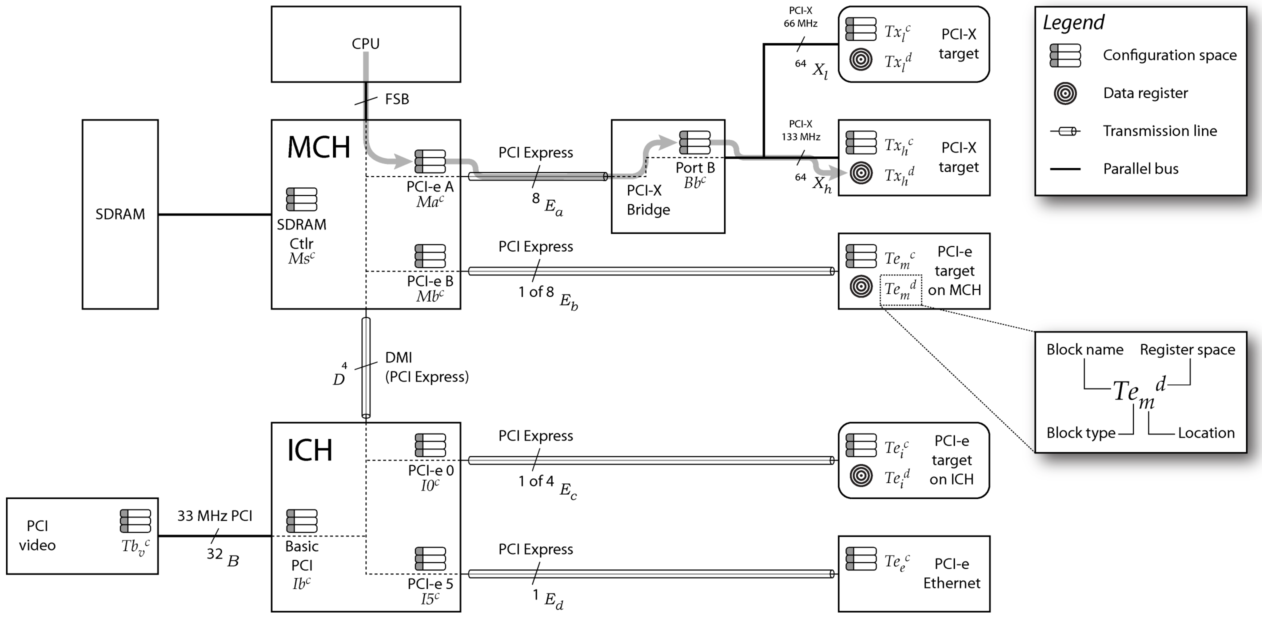

Figure 1: Block diagram of experimental apparatus

PCI-X (labeled Tx_l), and 33 MHz basic PCI (not shown, but referred to as Tb).

The ICH provides a conventional 32-bit 33 MHz PCI bus (labeled B, via bridge Ib), which attaches the on-board video adapter (labeled Tb_v), and six PCI Express lanes which can be combined according to need. This motherboard provides one 4-lane link (labeled E_c, via bridge $I0$) to a slot for add-in cards, and the remaining two links attach Intel 82573 EtherExpress NICs (Network Interface Controller), one of which is shown (labeled Te_e, via link E_d and bridge $I5$). The FPGA based PCI Express target can be attached via Mb (where it is labeled Te_m) or $I0$ (where it is labeled Te_i). The rounded rectangles for targets Tx_l and Te_i represent alternative configurations for Tx_h and Te_m respectively, not physically separate devices. Two additional targets are measured, but not shown in Figure 1: an 8-lane PCI Express Areca ARC-1220 SATA RAID adapter attached in place of Te_m and a 4-lane PCI Express Intel 82571 EtherExpress dual-port NIC (very similar silicon to the on-board NICs) attached in place of Te_i.

The enlargement describes the nomenclature used for labeling measurement targets. *Block name*s of M, I, B and T respectively represent the MCH, ICH, PXH PCI-X bridge and target end points. *Block type*s of e, x and b respectively represent PCI Express, PCI-X and basic (conventional) PCI devices for end points, or the port number for bridges. *Location*s of m, i, h and l respectively represent targets on the MCH, ICH, 133 MHz PCI-X bus, and 66 MHz PCI-X bus, where e and v represent the integrated Ethernet and VGA adapters. *Register space*s of c, di and ds respectively represent configuration registers, data registers with immediate completions and data registers with split completions.

2.2 Measurements

The objective of these measurements is to determine the marshaling delay inherent in each link: *i.e.* protocol and all other overheads as distinct from serialization delay. We consider serialization delay to be of less interest than marshaling delay because it is purely a function of signaling speed and link width. Marshaling delay, on the other hand, invites more opportunity for improvement: although there are mechanisms that (depending on application) can mitigate latency, such mechanisms don't alter the existence of that latency.

DWORD transfers (the smallest unit of data in the PCI protocol) are therefore used to characterize the round-trip time (RTT) across a given link or bus because they represent the smallest possible transaction and therefore contain the minimum component of serialization delay. Since, in PCI, memory-mapped write operations are posted (*i.e.* no acknowledgment of completion is made), non-posted read operations must instead be used for probe transactions.

We initiate each measurement from the processor, and time each read using the processor's TSC (Time Stamp Counter) register. The TSC, incremented once for every core clock cycle, provides a useful clock reference as sampling the TSC register can be done with a single processor instruction. Therefore, the latency incurred by the act of sampling the time is lower than would be any other source of time information available within the PC.

The accuracy of the TSC can be affected if Intel SpeedStep (a power saving technology) is active; Intel SpeedStep was disabled for this work. Additionally, we operated the machine with only one CPU core as coherence among the TSC registers of each CPU core is not guaranteed.

Without building new hardware, it is not possible to measure individual links directly. Instead, we estimate the RTT across a given link by taking the difference between measure-

Target	PCI-X (imm)		PCI-X (spl)		PCI Express	
Method	**kernel**	**user**	**kernel**	**user**	**kernel**	**user**
min_val	399	399	471	477	2169	2169
mean	403.6	403.7	493.6	493.6	2193	2193
std_dev	3.710	3.734	3.829	4.001	13.01	13.13

Table 2: Software sampling validation results (nanoseconds)

ments of devices on either end of that link. This is repeated until all links between the CPU and leaf node devices (end points) described above have been measured.

PCI specifies separate address spaces for configuration registers (which control how the device responds to transactions), memory-mapped I/O registers and legacy I/O registers. Every device, including bridges, is required to support a minimum set of configuration registers. End points implement data spaces as required to support their application, whereas bridges implement no data spaces of their own; rather, they forward data space transactions across to the bridge's subordinate bus. Bridge RTTs can therefore only be measured using configuration reads.

The path to each end point is measured by timing reads to a configuration register (denoted with a superscript suffix ^{c} in Figure 1, *e.g.* Ma^c) in every bridge between the CPU and the end point inclusive. In the example in Figure 1 (shown by the arrow), Ma^c is measured, then Bb^c and Tx_h^{di}. In the FPGA based targets — in which we can design logic to minimize handling time, and control how the transaction is terminated — samples are taken of a memory-mapped I/O register with immediate (denoted with superscript suffix ^{di}) and split completions (denoted with superscript suffix ^{ds}, and described further in Section 4.2).

The RTT for each target is taken to be the minimum from a set of one million samples. Each sample is comprised of certain irreducible components of delay (serialization and marshaling overhead) that represent the latency intrinsic to the protocol, plus some variable amount of error attributable to small variations between samples in the protocol overhead component (owing to implementation detail), and to ordinary measurement error. The data-set minimum represents the best approximation of the irreducible component (the smallest variation and error). Figure 2 displays cumulative distribution functions of one million measurements of the RTT in two cases. This demonstrates that there is a well defined minimum RTT in both cases. All other measurements showed similar distributions. We realize that the minima do not represent performance on average; however, our intention throughout has been to identify base-level measurements of interconnect latency.

The latency of each individual link or bus (including associated buffering and processing) can then be estimated by taking the difference between the minima measured at the devices on either end of that link.

2.3 Validation

In a system as complex as a computer with a modern chipset, processor and operating system, measurements made from software could be biased by a variety of influences.

2.3.1 Software sampling mechanism

Table 2 compares the aggregate results from two implementations (one kernel mode, the other user mode) of the sampling software used to measure RTTs. The two implementations are very different yet, with interrupts disabled, there is good agreement between samples. We are therefore confident that the software measurement technique produces reliable results.

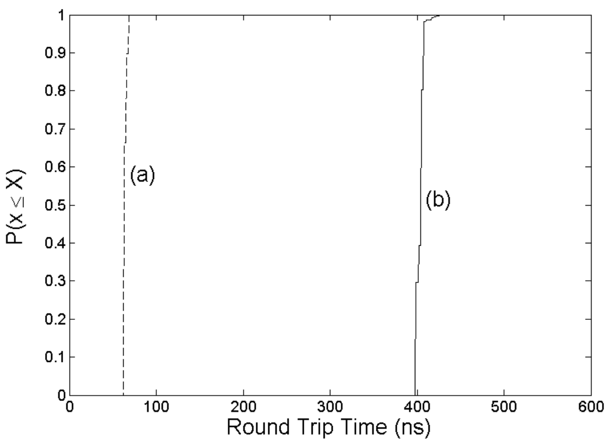

Figure 2: Cumulative Distribution Functions of RTTs (a) for reads to the MCH SDRAM controller, Ms^c, which is representative of all MCH bridges and (b) for reads to PCI-X target, Tx_h^{di}, with immediate completion

The low standard deviations reported in Table 2 provide further evidence that the measurement technique gives repeatable results (c.f. Figure 2).

It is interesting and encouraging to note that our technique is sensitive enough to detect a small difference in the compiled output between the user mode and kernel mode sampler. In an earlier iteration of this validation experiment, the results from the user mode results were consistently 3 ns greater than the kernel mode results. Careful examination revealed that this was because a missing compiler flag for the user mode sampler caused the call to the `rdtsc` instruction not to be in-lined.

2.3.2 Effect of cross traffic

Modern architectures make considerable use of sophisticated scheduling and pipelining techniques. Although PCI access rules are strict, and re-ordering and combining are forbidden within uncached I/O space, it is conceivable that the implementation of the chipset could influence the results collected depending on what cross-traffic is present.

We tested the effect of cross traffic by running an experiment in which known cross traffic is injected in a variety of combinations, while making measurements in the manner described in Section 2.2.

We present in Table 3 the minima and standard deviations of data recorded under artificially-generated cross traffic. With an otherwise idle system, read operations alternate between a pair of data registers in the FPGA targets. Different combinations of target device and page are tested in order to establish the independence of the target on the effect of the cross traffic generated. The column labeled *Reference* shows the result from the software validation experiment *user* case for comparison.

The six cases tested were:

1 Two registers within the same page of the PCI-X target using immediate completions (Tx_h^{di})

2a Two registers within the same page of the PCI-X target using split completions (Tx_h^{ds})

2b Two registers from different pages within the PCI-X target using split completions (Tx_h^{ds})

3a Two registers within the same page of the PCI Express target (Te_m^{ds})

3b Two registers from different pages within the PCI Express target (Te_m^{ds})

4 One register from each of the PCI-X (Tx_h^{di}) and PCI Express (Te_m^{ds}) targets

Since all measurements are made with interrupts disabled, and since only one processor core is active in the system, we make the assumption that the effect of our artificially-generated, interleaved cross-traffic is as significant as any external "natural" source of cross-traffic. In other words, although the artificial cross-traffic is synchronous with respect to itself, the lack of any other asynchronous source of read transactions means that all other traffic not generated by the cross-traffic experiment must also be synchronous with the measured traffic.

We note that the interleaved read transactions in the combinations described above show no appreciable difference in observed minima — *i.e.*, there is good agreement between the reference and cross traffic measurements. We therefore conclude that the presence of cross traffic is not a factor in the results we present next.

3. RESULTS

Table 4 presents the main results in this paper. The *Source Target* and *Dest. Target* columns show the minima as measured at the location indicated (refer to Figure 1). The difference column shows the estimate of the latency for that link based on those RTTs.

For parallel bus measurements, the estimated RTT is also expressed in bus clock cycles. Since the RTT includes processing time within the PCI logic as well as time on the

Case	Reference	Address 1	Address 2
1	399.0 ns (±3.734 ns)	399.0 ns (±3.700 ns)	399.0 ns (±3.742 ns)
2a	477.0 ns (±4.001 ns)	477.0 ns (±3.830 ns)	489.0 ns (±3.838 ns)
2b		486.0 ns (±3.954 ns)	489.0 ns (±4.014 ns)
3a	2169.0 ns (±13.13 ns)	2169.0 ns (±12.93 ns)	2169.0 ns (±12.85 ns)
3b		2169.0 ns (±13.28 ns	2169.0 ns (±12.96 ns)
4	As per cases 1 and 3	399.0 ns (±3.876 ns)	2169.0 ns (±13.06 ns)

Table 3: Minimum RTTs from the Cross traffic validation test (±1σ)

PCB, the absolute values are only an approximation, but their relative values are in proportion, as described in Section 3.1.

For comparison, the serialization period (actual time spent on the printed circuit board) is shown as calculated and also expressed as an efficiency (percentage of the estimated time). Note that for parallel PCI, the exact period of the transaction depends on how quickly the intended target decodes the transaction. For PCI Express links, the serialization delay can be calculated exactly and includes frame delimiters, serial number, and CRC with the transaction packet itself.

Transactions destined for the ICH pass through the MCH, but there is no bridge in the MCH which handles ICH traffic. However, the three measurable targets (Ma^c, Mb^c and Ms^c) within the MCH all return the same RTT; this value (63 ns) is used to calculate the delay of D.

3.1 Parallel PCI results

A parallel PCI device may be designed to claim a transaction after one, two or three cycles after a transaction begins as suits implementation constraints. A device is described as fast, medium or slow devsel respectively.

Column *bus clocks* in Table 4 illustrates the effect of this decode latency. The VGA adapter and the FPGA target are known to be medium and slow devsel devices respectively, from which we predict that DWORD bus transactions with the VGA adapter should be one cycle faster than with the FPGA target. The data show this to be the case.

Similarly, the effect can be seen in the relationship between the RTTs for split and immediate completions. Split transactions involve two bus transactions. The target of the transaction which completes a split request is the PCI-X bridge, which in this case is a fast devsel device (two cycles faster than the FPGA target). The relationship can be expressed as:

$$T_{split} = 2 \times T_{imm} - 2 \times T_{cycle}$$

As highlighted in Table 4, latency in the parallel buses reduces linearly with increasing bus speed. These observations further validate the results presented here.

We note that at 84 ns, the 133 MHz PCI-X bus (X_h) is three times faster than the 8x PCI Express link (E_a) which feeds it (252 ns, as noted in Table 4). This is despite E_a having nearly eight times the throughput. Evidently, the advantage that PCI Express has in serialization delay is lost to the packet buffering and processing as described in Section 4.2, which is not present in PCI-X.

3.2 PCI Express results

In addition to measuring E_a, the 8-lane PCI Express link which attaches Bb, we also measured an 8-lane PCI Express SATA RAID adapter attached via E_b. Both return RTTs of 252 ns, further confirming the validity of the bridge measurements.

Similarly, measurements for 4-lane links are made of D (the DMI link) and also the Intel 82571 EtherExpress NIC attached via E_c, a 4-lane PCI Express version of the on-board 1-lane NICs. Unsurprisingly, the 4-lane NIC returns an RTT of 498 ns, almost twice the latency of the 8-lane links measured. The DMI link RTT is 366 ns — some 26% faster than the NIC. While this result may be entirely due to proprietary optimizations of the DMI link over a true 4-

Description	Source target (1)	Dest. target (2)	Latency (2) − (1)	clks	Time on PCB	Efficiency
MCH bridge to PXH bridge PCI Express 8x	Ma^c = 63 ns	Bb^c = 315 ns	252 ns	–	22 ns	8.7%
MCH bridge to ICH bridge DMI (PCI Express 4x)	M^c = 63 ns	I^c = 429 ns	366 ns	–	44 ns	12.0%
ATI VGA adapter basic PCI (32 bit, 33 MHz) on ICH	Ib^c = 423 ns	Tb_v^c = 711 ns	**288 ns**	10	180 ns	62.5%
33 MHz PCI target (mapped data immediate completions)	Bb^c = 315 ns	Tb^{di} = 639 ns	**324 ns**	11	210 ns	64.8%
33 MHz PCI target (config space immediate completions)	Bb^c = 315 ns	Tb^c = 639 ns	324 ns	11	210 ns	64.8%
66 MHz PCI-X target (mapped data immediate completions)	Bb^c = 315 ns	Tx_l^{di} = 489 ns	**174 ns**	11	105 ns	60.3%
66 MHz PCI-X target (mapped data split completions)	Bb^c = 315 ns	Tx_l^{ds} = 633 ns	318 ns	20	–	–
66 MHz PCI-X target (config space immediate completions)	Bb^c = 315 ns	Tx_l^c = 549 ns	234 ns	15	–	–
133 MHz PCI-X target (mapped data immediate completions)	Bb^c = 315 ns	Tx_h^{di} = 399 ns	**84 ns**	11	53 ns	63.1%
133 MHz PCI-X target (mapped data split completions)	Bb^c = 315 ns	Tx_h^{ds} = 468 ns	153 ns	20	–	–
133 MHz PCI-X target (config space immediate completions)	Bb^c = 315 ns	Tx_h^c = 429 ns	114 ns	15	–	–
Intel EtherExpress PCI-e 1x on ICH	$I5^c$ = 429 ns	Te_e^c = 1737 ns	**1308 ns**	–	176 ns	13.5%
Dual Intel EtherExpress PCI-e 4x on ICH	$I0^c$ = 429 ns	Te_d^c = 927 ns	**498 ns**	–	44 ns	8.8%
Areca SATA RAID Controller PCI-e 8x on MCH	Mb^c = 63 ns	Te_r^c = 315 ns	**252 ns**	–	22 ns	8.7%
1x PCI-e target on MCH (mapped data split completions)	Mb^c = 63 ns	Te_m^{ds} = 2169 ns	**2106 ns**	–	176 ns	8.4%
1x PCI-e target on MCH (config space split completions)	Mb^c = 63 ns	Te_m^c = 2331 ns	2268 ns	–	176 ns	7.8%
1x PCI-e target on ICH (mapped data split completions)	$I0^c$ = 429 ns	Te_i^{ds} = 2535 ns	**2106 ns**	–	176 ns	8.4%
1x PCI-e target on ICH (config space split completions)	$I0^c$ = 429 ns	Te_i^c = 2679 ns	2250 ns	–	176 ns	7.8%

Table 4: Differential Measurement Results with Best Case Latencies in Each Class Highlighted in Bold

lane PCI Express link, an alternative explanation is made in Section 4.1.

The 1-lane FPGA based targets, Te_m and Te_i, both measure 2.1 μs — approximately 8 times the 8-lane RTTs. We note that the 1-lane on-board NIC does better at 1.3 μs, for which we propose an hypothesis in Section 4.1.

With the exceptions of the DMI and on-board NIC, the PCI Express targets measured all return RTTs with good agreement appropriate to their respective link widths — yet as noted, even the fastest link is still substantially outperformed by the technology which PCI Express was meant to replace. Furthermore, in the most equivalent comparison[3] the 1-lane PCI Express link has between 16 and 25 times longer latency than 133 MHz PCI-X.

3.3 General observations

Reads of the configuration space of the FPGA-based end points are consistently slower than reads of data registers. Completion of configuration transactions is handled by the PCI cores themselves, so we speculate that the user code in each of the targets is more efficient at completing requests than the PCI cores.

4. ANALYSIS

Since access to implementation detail is generally unavailable (or only available under NDA), we can only conjecture as to why PCI Express in general should exhibit latencies so much greater than PCI-X.

[3]Direct comparisons are complicated by the fact that PCI-X is half duplex and PCI Express is full duplex. For these purposes, we compare aggregate throughput in PCI Express (since a read transaction is a bi-directional process) with the half-duplex throughput in parallel PCI. We consider parallel PCI to be 32-bit wide (even when all 64 bits are available) because the upper 32 bits are never used during a DWORD transaction. 133 MHz PCI-X is therefore 133 MHz × 32 bits = 4.3 Gb/s and 1-lane PCI Express is 2.0 Gb/s/lane × 2 = 4.0 Gb/s.

4.1 A latency model for PCI Express

The PCI Express data presented in Section 3 suggest an inverse relationship between total latency and link width. Accordingly, we hypothesize a simplified delay model, for n lanes, of the form:

$$T_{RTT} = 2 \times \left(T_{proc} + \frac{T_{serdes}}{n} + T_{flight} \right)$$

which, together with the data collected, support the perhaps obvious conclusion that latency reduces with increasing link width.

The factor of two allows for both request and completion packets. T_{flight} is negligible at 2 ns per foot. T_{proc} and T_{serdes} represent the work done (for a given transaction size) that, respectively, is fixed and dependent on link width.

Except for D and Te_e^c, the PCI Express RTTs scale almost linearly with link width, suggesting that T_{serdes} dominates measured latency. We hypothesize that this is because packets are processed using a clock derived from the link. For a given internal data-path-width, wider links require a faster clock and therefore not only do packet data arrive faster, packet processing tasks are completed faster as well.

For example, the target logic in Te_m and Te_i (the FPGA based PCI Express targets) is driven with a 31.25 MHz clock derived from the link. The TLP handling code returns a response in around 5 cycles which, at 31.25 MHz, is 160 ns — almost as much as the serialization delay for that 1-lane link.

The impact of T_{serdes} can be mitigated by using a fast clock to drive processing logic, but introduction of extra clock domains increases complexity of design and testing, and clock domain crossing is therefore generally avoided where possible.

Where this design choice is made, however, T_{proc} becomes more significant, and overall latency can be reduced. We speculate that this is the case in D and Te_e, or else a narrow internal data path is used to increase the rate at which

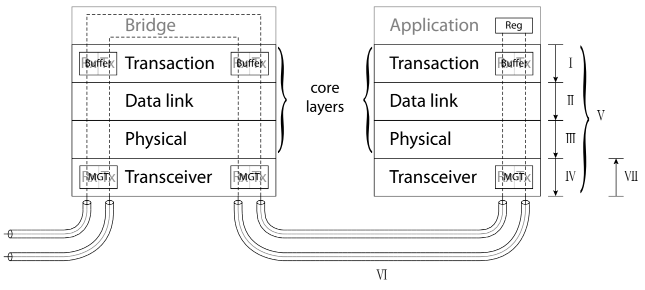

Figure 3: PCI Express layer model (Roman numerals correlate with Table 5 and Figure 4)

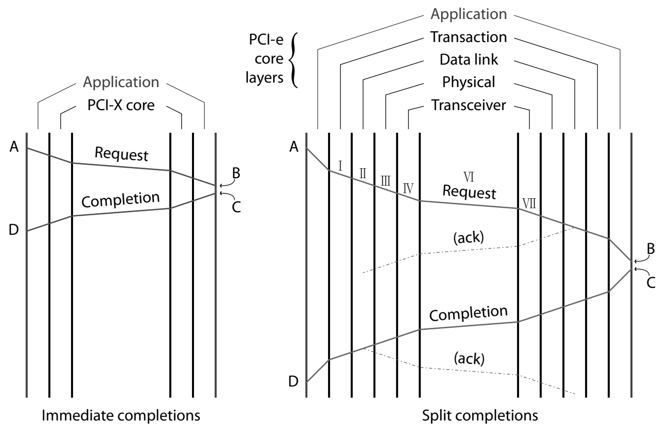

Figure 4: Immediate and split completions compared (Roman numerals correlate with Table 5 and Figure 3)

processing logic is driven, and for this reason the RTTs measured for D and Te_e don't scale strictly with link width.

4.2 Analysis of a PCI core

Parallel PCI architecture multiplexes address and data on one 32 or 64 bit bus, and uses about a dozen extra signals to control the progress of a transaction. The IP cores that drive them can therefore almost be as simple as a few pipeline stages on the multiplexed address and data bus along-with sufficient logic to sequence bus transactions.

PCI Express follows a layered communications stack (shown in Figure 3) resembling the OSI model — more processing is done in PCI Express than in PCI-X. Each of these layers require more logic to implement the standard, and each contribute to the higher total delay seen in the PCI Express results.

	Component	Delay	% of total
I	Tx Transaction Layer	1056 ns	71%
II	Tx Data Link Layer	336 ns	22%
III	Tx Physical Layer	32 ns	2%
IV	Tx Transceiver	72 ns	5%
V	Tx Total	1496 ns	100%
VI	Time of flight	< 1 ns	
VII	Rx Transceiver	183 ns	

Table 5: Delay breakdown by layer (Roman numerals correlate with Figures 3 and 4)

The precise breakdown by layer of the total latency is challenging to deduce without access to the high level RTL (Register Transfer Level) of a PCI Express core. However, an estimate may be made by analyzing the post-synthesis netlist of a PCI Express core in simulation.

Table 5 presents approximate figures for the transmit path in the PCI Express core used for the FPGA based PCI Express target in this study. This core buffers packets in its transaction layer, and the transaction layer accounts for approximately three quarters of the latency in the transmit path. This highlights the impact that buffering within the end-point can have on end-to-end latency.

The latency in the receiver half of the transceiver is included for comparison with the transmit half. Time of flight on short traces is negligible, and serialization delay (176 ns, as noted in Table 4) is dwarfed by the other components. From this we conclude that it is not the principle of serialized links, but the protocol overhead which dominates performance.

Technological developments can help reduce latency, but as Table 5 shows, most of the latency arises from how the protocol itself works, and the greatest performance gains will be made in a protocol designed for low latency.

Figure 4 shows the events of an immediate completion on a PCI-X bus compared with those of a split completion on a PCI Express link. (A split completion on a PCI-X bus would look similar to that of the PCI Express case shown, with the following differences: PCI-X has no need to acknowledge transactions and isn't layered like PCI Express, so the overhead will be less. Conversely, PCI Express cannot support immediate completions).

In each case, a request originating at A traverses the cores and arrives at target B. After a short delay, while the requested data are being retrieved (C), the immediate completion returns the data in the one, unbroken transaction (D). When the target splits the request, a new transaction is initiated at some later time to carry the response back to the requester.

In PCI Express, acknowledgment packets are returned to free buffers in the transmitter.

5. LIMITATIONS

Certain sources of latency have been left uncharacterized in our analysis.

PCI Express supports sliding windows with cumulative acknowledgments similar to TCP, and therefore pending acknowledgments don't prevent packet transmission for as long as there are sufficient flow control credits available in each receiver [21].

Certain conditions (*e.g.* a large number of posted write operations) can exhaust flow control credits, which will introduce latency. Because the probe requests used to measure latency in this study are serialized (only one measurement is made at a time), flow control credit starvation cannot happen here.

Multi-lane PCI Express links have the potential to introduce an additional source of latency. Although most individual transceivers can perform CRC calculation and checking automatically, they cannot do so when the packet data are interleaved across multiple lanes, because no one transceiver sees the whole packet. As a result, PCI Express implementations must implement their own CRC logic in a separate stage. PCI Express soft core end points, for example, implement CRC logic in their link layers — even in the 1-lane instance which could otherwise have used the dedicated CRC logic in the transceivers.

The cross-traffic validation experiment (Section 2.3.2) depends on the assumption that with only one processor core enabled, the artificial cross-traffic is comparable with background traffic, and that there aren't other patterns of cross-traffic that might affect the results gathered. A more complete investigation, perhaps based on an SMP-enabled operating system is warranted.

It is possible that a bridge might in some way deprioritise configuration transactions (in much the way an IP router deprioritises ICMP traffic) however, the PCI protocol requires that bridges maintain transaction order — including the order of reads, since read operations can have side effects in PCI devices. Given that ordering must be strictly maintained, there is a strong incentive to retire configuration requests as quickly as possible, so that subsequent transactions can be serviced as quickly as possible. From this, we assert that bridge configuration registers are a reliable indication of latency to that bridge.

6. PROPOSED SOLUTIONS

Higher latency in PCI Express is no reason to abandon high speed serial interconnects — especially when they can scale throughput far beyond that of a parallel interconnect, and we certainly make no case for a return to them. Nor is there any fundamental reason why a PCI Express-like protocol or other backwards compatible protocol should not be low latency.

Serial interconnects can deliver the throughput demanded by emerging applications without compromising latency principally by low latency protocol design, and also by means of emerging interconnect technologies such as photonic switching.

6.1 Low latency protocol design

It should not be assumed that long distance communications techniques are appropriate for short-haul interconnects. For example, instead of a single check-sum code at the end of the packet, separate check-sums for header and payload would permit a receiver to allocate and configure a path while payload is still being received.

Relatedly, interactions between master and target in parallel standards were much more closely coupled than they are in packet-oriented protocols such as PCI Express, and there is no particular reason why this should be so. The sooner transaction correctness can be verified, the sooner the transaction can be allocated and executed. We therefore conclude that there is some merit for abandonment of the packet model employed by PCI Express.

The abstraction afforded by a layered protocol stack can simplify design, but the benefits are outweighed if such an abstraction is made at the expense of latency.

6.2 Photonic interconnect

Intermediate buffering increases latency. A reduction of reliance on buffering — both in I/O structures (such as the MGT block) and in queuing — will likely yield the greatest gains in latency [10]. To this end, we claim that optical interconnects offer the greatest prospects (especially when there are serially-connected bridges involved) because an optical signal can be transmitted through multiple photonic switches before signal integrity issues force an O-E-O (optical-electronic-optical) conversion where buffering might be required. Optical bandwidth scales far beyond electrical bandwidth, important because PCI Express 3 is already at the limits of FR-4 printed circuit board. Possible solutions include optical chip-to-board packaging such as IBM's TeraBus [22] and polymer waveguides which can be incorporated into printed circuit boards such as [8]. Optical loss characteristics mean that a signal can cross a board without need of a repeater. Further, since random access optical memory doesn't (yet) exist, optical architectures must perforce avoid buffering.

Wavelength division multiplexing provides the equivalent of additional lanes without need of additional physical channels and, since optical channels are bidirectional, flow control and acknowledgment feedback (as done in SPINet [23]) require no additional switching overhead.

7. RELATED WORK

The performance of interconnects has maintained the constant interest of a subsection of the computer-architecture community[4] including the analysis of PCI interconnect architectures. For example, Liu *et al.* [24] showed that Mellanox PCI Express adapters consistently outperformed its own PCI-X adapters, both in throughput and in latency. It analyzes end-to-end performance of the respective adapters, taking in operating system, protocol stack, adapter and network performance as a whole. It concludes that for builders of HPC clusters, the Mellanox PCI Express HBA provides lowest latency, but does not attempt to attribute the specific role that the PCI Express link itself plays in the overall performance.

As pointed out by Liu *et al.*, the PCI-X case attaches through an extra bridge, which is itself attached via the same 8-lane PCI Express link which attaches the PCI Express case. Without evaluating the contribution to latency of each component in the PCI hierarchy, measurements will include latency due to both the PCI-X bus, and the PCI Express link which supplies it.

The PCI-X bus on the motherboard used in our study uses a similar arrangement. Their measurement technique times the RTT of an InfiniBand ping message, each of which involves traversing an HBA and associated bus or link four times. If the data collected in this study are representative of typical PCI-X buses and PCI Express links, the contributions of the HBA's host interface to RTT latency are respectively 7% and 27% of the best reported RTTs (4.8 μs and 3.8 μs).

The use of programmable logic for the measurement of high-speed systems has a long heritage, for example, Moll *et al.* [25] demonstrates the use of simple FPGA based hardware for the profiling of software applications including an analysis of PCI bus performance.

[4]For example, see Hot Interconnects Symposium various.

Additionally, the use of CPU counters for the differential analysis of timing has been a technique employed by a number of previous researchers; use of the TSC register for micro-benchmarking is described in Sottile *et al.* [26], and for time stamping in Veitch *et al.* [27], while Hall *et al.* [28] makes a differential analysis of operating system latency in OSF/1, using the Alpha's Program Cycle Counter.

The use of micro-benchmarks has long been established as a useful method for evaluation of specific aspects of performance. Martin *et al.* [14] uses micro-benchmarks to evaluate the effect of various forms of latency, overhead and bandwidth. Similarly, Ousterhout [29] describes micro-benchmarks as applied to profiling operating system performance, and Liu *et al.* [30] as applied to MPI implementations.

Finally, Endo *et al.* [12] establish the primacy of latency to performance in interactive and transactional application software, and describes micro-benchmarking for analysis of performance by latency and encourage further, application-specific study of PCI interconnect latency impact.

8. CONCLUSIONS

In this paper we have described an approach that measures PCI interconnect latency through the differential analysis of transaction delays. Where intuition might suggest that newer, higher throughput interconnects should have lower latency than older interconnects, in fact PCI Express consistently shows higher latency than PCI-X.

The fastest PCI Express link we measured (8-lane) has three times longer latency than the fastest PCI-X bus — despite having nearly eight times the throughput for DWORD reads. We present data that demonstrate that PCI interconnect implementations can have a dramatic effect on the latency measured in PCI Express links: the slowest PCI Express links (single lane), comparable in throughput with 133 MHz PCI-X, have between 15 and 25 times higher latency.

We propose a model to explain why PCI Express latencies are so much longer than PCI-X, and provide possible implementation related explanations. We note that link width has a profound impact on latency, much greater than the influence of serialization delay alone. We speculate that the reason for this in some cases is that implementation considerations amplify the contribution of link width to latency.

Accepting that high speed serial links are more scalable than parallel buses, we suggest how such links could be employed in a low-latency interconnect. We observe that latency hiding techniques can, to some extent, ameliorate the impact of intrinsic interconnect latency, yet there are obvious advantages to an interconnect that avoids latency by design.

Interconnect latency is only one of the many sources of latency in a computer system. It is more or less a constant for a given architecture, and it imposes a lower bound on overall latency. By addressing interconnect latency, improvements made elsewhere are more likely to show greater returns.

Lastly, we make the case that a bufferless, all-optical interconnect offers the potential for scalable throughput with low latency.

Future work

Recognizing the contributions of application-specific latency studies such as that of Endo *et al.* [12] and Hall *et al.* [28], we are currently working to quantify the impact of interconnect latency on a variety of applications. Alongside this, we are exploring alternative architecture and protocol design for the PCI interconnect.

9. ACKNOWLEDGMENTS

David Miller would like to thank the Cambridge Commonwealth Trust, the Cambridge Angels and Endace Ltd for their financial support. Philip Watts would like to thank the Royal Commission for the Exhibition of 1851.

The authors thank Steve Hand, Jon Crowcroft, Simon Moore, David Greaves, the Systems Research Group of the Cambridge Computer Laboratory and anonymous reviewers for their helpful and constructive input to this paper.

10. REFERENCES

[1] R. S. Tucker. The role of optics and electronics in high-capacity routers. *Lightwave Technology, Journal of*, 24(12):4655–4673, Dec. 2006.

[2] I. Keslassy *et al.*. Scaling internet routers using optics. In *SIGCOMM '03:*, pages 189–200, 2003.

[3] M. Glick *et al.*. SWIFT: a testbed with optically switched data paths for computing applications. *Transparent Optical Networks, 2005*, pages 29–32 Vol. 2, July 2005.

[4] A. Shacham *et al.*. A fully implemented 12×12 data vortex optical packet switching interconnection network. *J. Lightwave Technology*, 2005.

[5] A. Shacham *et al.*. Photonic NoC for DMA Communications in Chip Multiprocessors. In *HOT Interconnects '07*, pages 29–38, Aug. 2007.

[6] H. Wang *et al.*. Experimental demonstration of end-to-end PCI-Express communication over a transparent all-optical photonic interconnection network interface. In *OFC*, March 2009.

[7] N. Bamiedakis *et al.*. Cost-effective multimode polymer waveguides for high-speed on-board optical interconnects. *Quantum Electronics, IEEE Journal of*, 45(4):415–424, April 2009.

[8] J. D. Ingham *et al.*. Multimode siloxane polymer waveguides for robust high-speed interconnects. In *Lasers and Electro-Optics and 2006 Quantum Electronics and Laser Science Conference*, pages 1–2, May 2006.

[9] Nick McKeown. Buffers: How we fell in love with them, and why we need a divorce. In *Hot Interconnects*, 2004.

[10] M. Enachescu *et al.*. Part III: routers with very small buffers. *SIGCOMM Comput. Commun. Rev.*, 35(3):83–90, 2005.

[11] David Patterson. Latency lags bandwidth. *Communications of the ACM*, 47(10), 2004.

[12] Y. Endo *et al.*. Using latency to evaluate interactive system performance. *SIGOPS Oper. Syst. Rev.*, 30(SI):185–199, 1996.

[13] Q. Huang *et al.*. GPU as a general purpose computing resource. In *9th PDCAT*, pages 151–158, Dec. 2008.

[14] R. Martin *et al.*. Effects of communication latency, overhead, and bandwidth in a cluster architecture. *24th ISCA, 1997*, pages 85–97, Jun 1997.

[15] Aite Group. Algorithmic trading 2006: More bells and whistles.

`http://www.aitegroup.com/reports/200610311.php`, November 2006.

[16] N.J. Boden *et al.*. Myrinet: a gigabit-per-second local area network. *Micro, IEEE*, 15(1):29–36, Feb 1995.

[17] Intel Corporation. *82546GB Dual Port Gigabit Ethernet Controller*, October 2005.

[18] Intel Corporation. *82571 & 82572 Gigabit Ethernet Controller*, December 2006.

[19] PCI-SIG. PCI local bus specification, 2.3, March 2002.

[20] PCI-SIG. PCI-X addendum to the PCI local bus specification, 1.0a, July 2000.

[21] PCI-SIG. PCI Express base specification, 1.1, March 2005.

[22] J. A. Kash *et al.*. Terabus: a chip-to-chip parallel optical interconnect. *Lasers and Electro-Optics Society, 2005. LEOS 2005. The 18th Annual Meeting of the IEEE*, pages 363–364, Oct. 2005.

[23] Assaf Shacham et al. A scalable, self-routed, terabit capacity photonic interconnection network. *Hot Interconnects 13*, 2005.

[24] J. Liu *et al.*. Performance evaluation of InfiniBand with PCI Express. In *HOT Interconnects'04*, pages 13–19, 2004.

[25] L. Moll and M. Shand. Systems performance measurement on PCI Pamette. In *FPGAs for Custom Computing Machines*, pages 125–133, Apr 1997.

[26] M. Sottile and R. Minnich. Analysis of microbenchmarks for performance tuning of clusters. In *Cluster Computing, IEEE International Conference on*, pages 371–377, Sept. 2004.

[27] D. Veitch *et al.*. Robust synchronization of software clocks across the internet. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 219–232, 2004.

[28] J. Hall *et al.*. Counting the cycles: a comparative study of NFS performance over high speed networks. In *Proceedings of the 22nd Conference on Local Computer Networks (Minneapolis, MN)*, pages 8–19, 1997.

[29] John Ousterhout. Why aren't operating systems getting faster as fast as hardware. In *Proceedings of the Summer USENIX Conference*, pages 247–256, 1990.

[30] J. Liu *et al.*. Performance comparison of MPI implementations over InfiniBand, MyriNet and Quadrics. *Supercomputing Conference*, pages 58–58, Nov. 2003.