PCI Express* (PCIe*) 3.0 Accelerator Features

Jasmin Ajanovic, Intel Corporation

Introduction

In recent years there has been a trend towards offloading compute intensive workloads to specialized devices referred to as "accelerators". These accelerators are used to improve platform performance in key computation (financial services), visualization (advanced gaming and workstations), content processing (Crypto and XML acceleration), and intelligent I/O applications. Two notable changes have contributed to this trend. First, the accelerators have benefited from Moore's law which has given rise to extremely powerful engines which are increasingly multi-core/multi-thread/multi-card devices. Second, the accelerators are becoming increasingly programmable.

> ***What is an accelerator?***
>
> It's a device that is optimized to enhance the performance or functionality of a computing system. Normally, processors operate sequentially, executing instructions one by one. Various techniques are used to improve performance; hardware acceleration is one of them. The main difference between hardware and software is concurrency, allowing hardware to be much faster than software. Hardware accelerators are designed for computationally intensive software code.

Although the development of specialized application accelerators had been going on for some time, there was no common standard hardware attach point and no common software architecture or programming model. The solution space was very fragmented and as the number of applications started increasing an industry framework that economically and efficiently enables specialized acceleration became highly desirable. This had been recognized by Intel and IBM who, in mid-2006, started an initiative called "Geneseo". Its goal was to enhance PCI Express[1], which was already an industry mainstream IO interconnect standard, with capabilities needed by accelerators. Intel and IBM developed a proposal for a set of extensions to the PCI Express* (PCIe*) Specification Rev 2.0. Although the initial requirements that drove the proposal were derived from accelerator applications, the extensions are general in nature and fit very well within the scalable architecture framework of PCIe. This proposal has now been adopted by the PCI SIG as a base for PCIe 3.0.

Note that Intel's strategy for supporting accelerator applications extends beyond PCIe to include system interconnects i.e. FSB (short term) and QPI (long term) as well as to address platform software infrastructure through a program initiative called QuickAssist.

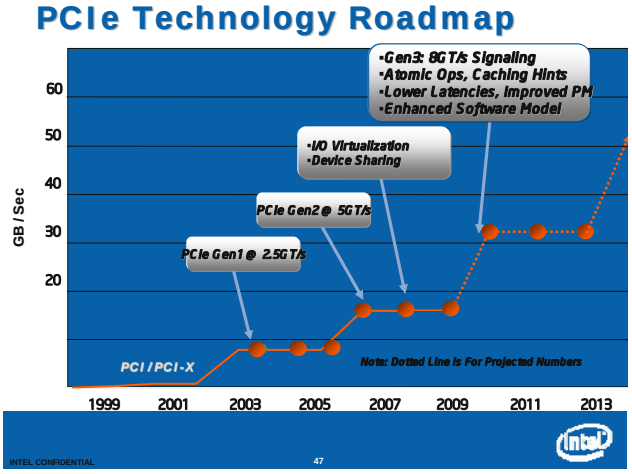
---

[1] *For background on PCI Express see text box at the end of this document*

PCI Express Technology Evolution

The PCI Express architecture is well established and being widely deployed in a variety of Intel Architecture (IA) and non-IA platforms. It evolved through two generations of spec/product deployment and work on the third generation is in progress. The following is summary of all new PCIe capabilities that are under development for Rev 3.0 of the specification:

- **Increased Throughput**
  - 8GT/s speed improvement
  - Signaling/Encoding Scheme – B/W efficiency improvement through replacement of 8b/10b encoding–with-scrambling scheme
- **Latency Reduction**
  - Data Re-use Hints – Mechanism for efficient and lower latency access
  - ID-based Ordering – Transaction level attribute to optimize ordering within Root Complex and memory subsystems
- **Software Model Improvements**
  - Atomic Read-Modify-Write – Mechanism to reduce synchronization overhead between accelerator and host CPU
  - IO Page Faults– Mechanism to notify device of page faults and request for faulted pages to be made available
- **Configuration and RAS Enhancements**
  - BAR renegotiation– Mechanism for dynamic change of device BAR sizes
  - Internal Error Reporting– Extend the AER to report internal errors
- **Power Management**
  - Dynamic Power Allocation - Support for dynamic performance/power operational modes through standard configuration mechanism
  - Latency Tolerance B/W Reporting
  - Opportunistic Buffer Flush & Fill
- **Accelerator Aggregation/Embedded Applications Support**
  - Multicast – Capability for efficient multi-party communication
- **Scalable Architecture Improvements**
  - TLP Prefix Architecture – Increased scalability of packet protocol for future uses



PCIe goes through constant improvements

PCIe 3.0 Features for Accelerators

This section of paper will focus on technical details and key benefits of PCIe 3.0 capabilities that can specifically improve accelerator applications.

Gen3 Signaling: *Improved Bandwidth*

Every three years, the PCIe interconnect performance has usually doubled. Generation 1 PCI Express ran at 2.5 Giga-transfers per second, but with its 8b/10b encoding scheme (10 bits for encoding 8 bits of information to maintain enough "1" to "0" transitions required for an embedded clock), which represents a 20% overhead, its effective transfer rate is 2 Giga-transfers per second. Today, PCIe 2.0 runs at 5 Giga-transfers per second using same 8b/10b encoding scheme, effectively doubling the transfer rate of Gen1. PCIe 3.0 is going to double the data rate again but rather than going to 10GT/s signaling, while using the 8b/10b encoding scheme, it is going to run at 8 Giga-transfers per second and use a new encoding scheme that will present close to 0% overhead.[2]

Most accelerator applications directly benefit from higher aggregate bandwidth, particularly applications that use wide (e.g. x16) PCIe implementations such as Graphics and GPGPU cards. For illustration purposes the following table shows theoretical maximum bandwidth rates achievable with different generations of PCIe based on x16 wide interface.

	Raw Bit Rate	Link BW	BW/lane/way	BW x16
PCIe 1.x	2.5GT/s	2Gb/s	~250MB/s	~8GB/s
PCIe 2.0	5.0GT/s	4Gb/s	~500MB/s	~16GB/s
PCIe 3.0	8.0GT/s	8Gb/s	~1GB/s	~32GB/s

Data Reuse (Caching) Hints: *Provide stronger coupling between Host Cache/Memory hierarchy and IO*

One of the most important aspect of accelerator's (as well as in general of an IO device) behavior within the system is interaction with the host including CPU and cache/memory. However, in current systems exchange of information such as control structures (headers & descriptors) as well as data payloads is not optimized for temporal reuse within cache/memory hierarchy. For an example: IO/accelerator writes data to a host memory and later host CPU reads the data out of the memory. This sequence that includes two accesses to memory/DRAM can be optimized by IO writing data directly to a cache and CPU reading out of cache. This optimization would improve performance (reduced access latencies) and significantly reduce consumed bandwidth and power of system resources (e.g. of DRAM as well as of cache-coherent system interconnect such as QPI).

However, it is important to note that in many cases it is not desirable to cache all IO traffic. For an example, modern operating systems typically implement a speculative read-ahead buffer for the disk (i.e. storage controller writes to host memory). Caching such data, which may easily be many megabytes, would needlessly pollute system caches with data that may never be used. This is again where explicit hints from IO/accelerator can help to optimize behavior on a per-transaction basis.

[2] *The reason for this is because based on initial feasibility studies of Intel and other companies within PCI SIG it has been found that 10GT/s would create substantial implementation challenges and would require significant changes to existing PCB (FR4) materials and connector design. Such changes might have compromised PCIe electromechanical backwards compatibility and in addition would have significantly reduced channel length from 20" max that was supported by Gen1/Gen2.*

To implement a successful cache management policy in relation to IO traffic it is necessary to have information about the intended use of data. This information exists at the device (IO/accelerator) and can be carried to host CPU/cache complex via PCIe using Data Reuse/Caching Hints (DRH) mechanism. DRH mechanism uses up to 10-bits of information within PCIe MemRd/MemWr and AtomicOps transaction headers to indicate:

- Temporal Re-Use indication ie. whether and how data will be re-used
- Association of CPU/core with Requests from specific devices ie. which socket will use the data
- Location/Level of a targeted cache ie. which level of cache is appropriate for caching IO
- Allocation hints ex. whether IO data should have higher/lower priority in eviction

DRH mechanism opens the doors for significant improvements in a number of new applications: Communications adapters in HPC clusters, Database System Clusters & Computational Accelerators.

Atomic Read-Modify-Write Transactions: *Extend inter-processor synchronization mechanisms to IO*

Read Modify Write (RMW) operations are hardware assisted operations that atomically update a variable at memory location. These operations have a long history and find wide usage in a variety of synchronization algorithms Ex. semaphores, mutex conditions and barriers to coordinate operations across different processor cores/threads.

There is a need for similar type of coordination between host processor subsystem and IO devices that has not been adequately addressed with current IO interconnects including PCIe. Currently used mechanisms rely on interrupts and a higher-level interlock schemes that are very inefficient from communication overhead point of view e.g. acting upon an interrupt signaling in current mainstream OS environments can take many microseconds. Emergence of accelerator applications that requires tighter and more efficient coordination with software tasks running on the Host CPU(s) creates a need to extend current inter-processor communication mechanism such as Atomic RMW to IO subsystem. PCIe 3.0 addresses that requirement by providing set of operations shown in the following table.

AtomicOp	Description
FetchAdd	Data(Addr) = Data(Addr) + AddData
Swap	Data(Addr) = SwapData
CAS (Compare & Swap)	If (CompareData == Data(Addr)) then Data(Addr) = SwapData

These operations support 32-, 64- and 128-bit operand sizes and require target adresses to be naturally aligned to operation size. This requirement ensures that operands are contained within as single cache-line which allows very efficient processing of Atomic RMW within a Host cache hierarchy. (In this manner a need for a “Bus-LOCK” type of operations that impose significant overhead is removed in the case where Atomic RMW is initiated from IO side.)

PCIe Atomic RMW is defined as a symmetric capability i.e. it supports operations: IO→HostMem, HostCPU→IO, IO→IO. Protocol allows multiple outstanding Atomic RMWs. With a rich library of proven algorithms in this area, newly defined PCIe operations can be used for important applications such as:

- FetchAdd & Swap: Lock-Free Statistics e.g. counter update
- CAS: implementation of critical sections and non-blocking queuing algorithms

ID-Based Transaction Ordering: *Reduces transaction latencies in the system*

Conventional transaction ordering in PCIe is very strict (limits transactions bypassing older transactions) to ensure correctness of the producer-consumer model. For an example: Reads can not go around Writes to guarantee that resulting operations use correct and not stale data. This requirement has a root in a very simplified and restricted platform model (from an early era of PC) where there was a single Host CPU coordinating a work of an IO subsystem that used serialization of interconnect operations. Strictly following this requirement in modern systems can cause fairly long transaction stalls of 100's of nanoseconds affecting system performance. Going forward with multi-core/multi-thread Hosts connected to more sophisticated/complex IO devices via PCIe fully split-transaction protocol, there is an opportunity to optimize flows between unrelated transaction streams. For an example: graphics card initiated traffic may not have any direct relation to network controller traffic. Currently available mechanisms in PCIe to mitigate these restrictions are based on notion of Virtual Channels as well as on so called Relaxed Ordering transaction attribute. However, both of these have been defined from server usage point of view and carry inherent cost making it less attractive for mainstream use. To address this deficiency, PCIe 3.0 defines ID-Based Transaction Ordering mechanism that relies on current PCIe protocol mechanism for differentiating traffic sourced by different devices (i.e. Requestor ID and Target ID). New mechanism enables PCIe devices to assert an attribute flag (on a transaction basis) that allows relaxation of transaction ordering within PCIe fabric and within Host subsystem (including cache/memory hierarchy). This mechanism can be effectively applied to unrelated streams within:

- Multi-Function device (MFD) / Root Port Direct Connect
- Switched Environments
- Multiple RC Integrated Endpoints (RCIEs)

resulting in improvement of :

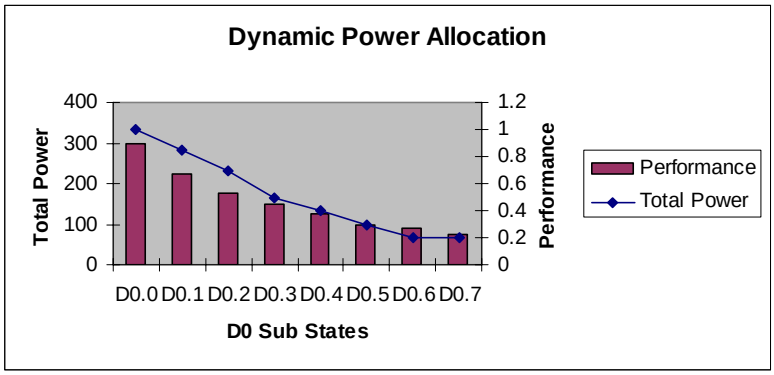
- Access latencies of individual transactions
- Improved system performance
- Reduced utilized bandwidth/power within Host Memory (DRAM)
- Improved IO Virtualization (e.g. Intel's VTd) assisted hardware.

Dynamic Power/Thermal Control: *Allows New Platform Level Flexibility in Thermal/Power Resource Management*

There is number of PCIe adapters such as high-end Graphics that are drawing more watts with each generation, which requires attention to power and cooling. (PCI SIG is presently defining an electro-mechanical solution for 300W add-in card.) This clearly requires good platform Power Management (PM) capability. Currently defined PCI Express PM capabilities do not comprehend ability to dynamically manage power allocated to adapter. PCIe 3.0 defines extensions that would allow IO devices and accelerator to operate at a different

power/performance levels while still in a fully active state (i.e. so called "D0" state from PCIe PM specification point of view).

As illustrated in the following figure, this new capability will allow different power and performance operational modes to enable dynamic and fine-grained power management based on the platform's changing workloads or the operating system's power management policies.



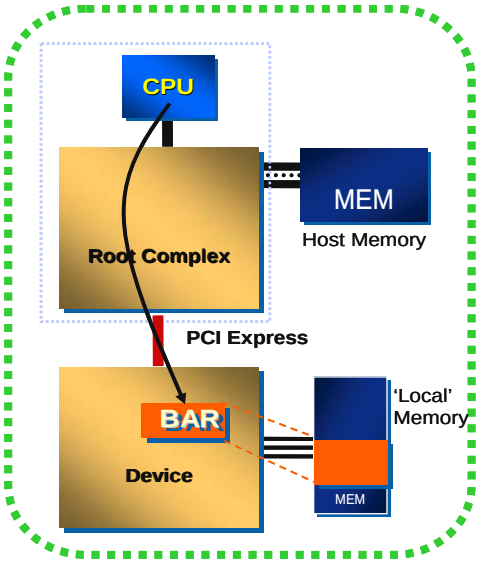
It is expected that this new capability will result in the following benefits:

- Over-provisioning of power delivery resources no longer required resulting in platform cost savings
- Allow Battery (Mobile)/ Power (Desktop/Server) optimizations by enabling User/OS defined profiles for "best performance", "lowest power", etc.

Resizable BAR: *Allows new level of platform resource management that addresses existing problems with graphics/accelerators*

Memory resources attached to PCIe devices are mapped into a system address space by OS configuration software using Base Address Register mechanism as illustrated in the following figure:

A high-end PCIe adapters such as Graphics/GPGPU may include large amounts of locally attached memory (typically 100's of MB but going >=1GB). Due to OS resource management limitations typically only a small fraction of this memory get mapped into system address space via BAR. However, application SW running on the Host side may be required to access much larger amount of PCIe adapter's memory. Current applications use all sort of workarounds (e.g. address "windowing") that come with significant software implementation complexity and performance overhead and in some cases applications may not work at all since allocated resources are not adequate.



BAR == Base Address Register – PCI mechanism for mapping device memory into sys. address space

PCIe 3.0 solves this problem by defining new Resizable BAR Capability structure that allows PCIe devices to advertise multiple supported sizes for device Memory Space resources that can be programmed into device's BAR. This mechanism is backwards compatible with exiting OS resource allocation software. Using new

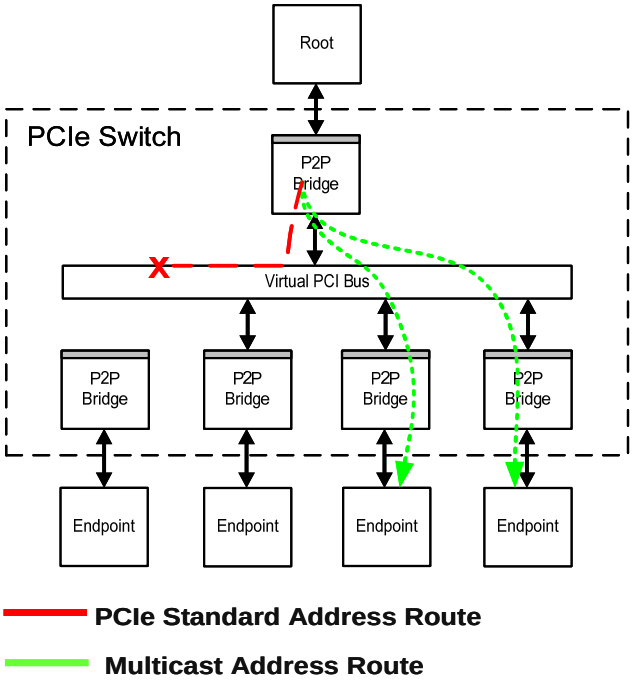
scheme, improved resource allocation software will attempt to program the largest acceptable size for each device BAR. This will result in higher performance as well as in enabling new usages/applications.

Multicast Transactions: *Provides performance scaling of existing apps (e.g. multi-Graphics) and opens new usages for PCIe*

There are new usages of PCIe in embedded communications platforms as well as in a mainstream PCs (e.g. multiple graphic/GPGPU cards) that can benefit of mechanism where single transaction (typically a Write) can be routed to multiple recipients simultaneously. This type of mechanism is called Multicast. A single Multicast transaction can replace multiple individual transactions resulting in lower system overhead (reduced utilization of bandwidth/power of memory and/or interconnect) and improved performance.

PCIe 3.0 defines Multicast Capability structure to configure PCIe components (Root Complex, Switches and Endpoints) for Multicast address decode and routing. PCIe Multicast supports only Posted, address-routed transactions (e.g., Memory Writes) and allows both RCs and EPs to be targets and initiators. The key mechanism associated with this new capability is a new Multicast BAR (Base Address Register) that is used to configure Multicast target address space.

The following figure depicts an example where Host CPU (connected to PCIe Root) generates a single Multicast Memory Write that "hits" Multicast BAR within the PCIe Switch and spawns into multiple individual Memory Writes targeting PCIe Endpoints connected below the Switch.



This capability opens a new application opportunities such as:

- Communications (e.g. route table updates, support of IP Multicast)
- Storage (e.g., mirroring, RAID)
- Aggregation of accelerators e.g. multiple GFX
- PCIe used in place of backplane Ethernet

IO Page Faulting Mechanism: *Critical for future IO Virtualization application scaling*

Emerging usages of Platform Virtualization, including IO Virtualization, increase pressure on memory resources making memory pages pinning (fixed allocation of memory for IO) increasingly expensive. There is a need to improve memory management capabilities that

PCIe defines for IO Virtualization. Currently PCIe supports Address Translation Services where PCIe devices can request upfront from System Address Translation Agent an address translation in order to mitigate potential performance bottlenecks during heavy traffic loads. PCIe 3.0 extends definition of ATS to support IO Page Faulting capability that allows system software to allocate pages requested by IO as swappable (i.e. pages can be swapped between host memory and system storage). This capability may not be applicable for all IO applications (e.g. NIC attach) but there is a range of emerging applications where accelerators such as GPGPUs can use this mechanism to achieve significant performance scaling and simplify software development.

PCIe Products Transitioning

Backwards Compatibility

As with the transition from PCI to PCIe, and then from PCIe 1.0 to 2.0, PCIe 3.0 will minimize the cost and complexity of transition by leveraging existing hardware and allowing operating systems to boot without any changes. It will rely on industry-standard, high-volume system components already in use, from boards to connectors to chips. In addition, it is expected that extended PCIe 3.0 features will be optional—meaning that designers only need to incorporate those features that provide performance gains for their specific systems.

PCIe 3.0 will support all the form factors currently supported by prior generations of PCIe—servers, desktops, mobile computers and embedded devices—and all stand to gain from its improvements. By supporting multiple types of cutting-edge accelerators, PCIe cuts across industry barriers to provide better performance for all. Its elements will be stable, scalable and extensible, providing the power and flexibility necessary for tomorrow's acceleration strategies.

Summary

The “Geneseo” program initiated work on enhancing PCIe as architecture of choice for accelerator attachment while focusing on the following main aspects:

- Enabling higher bandwidth and lower latency attach
- Improving interaction between the host CPU and application accelerators
- Providing a comprehensive range of design options for hardware developers

Applications that will benefit include:

- Visualization, such as complex weather modeling
- Math and physics, such as data-intensive financial applications or seismic analysis for gas and oil exploration
- Content processing, such as the encryption and decryption of communications infrastructure data

At the time when this paper is written (August 2008) Intel and the PCI SIG are on track for completing subsequent drafts of the PCIe Specification Rev 3.0 that will enable initial products delivery in 2010.

Existing PCIe specification documents as well as drafts of PCIe 3.0 Spec can be obtained at: www.pcisig.com

For any additional information contact author at:

jasmin.ajanovic@intel.com

PCI Express Background

PCIe: The Scalable I/O System Interconnect

PCI Express (PCIe) was developed as the next generation I/O system interconnect after PCI, designed to enable advanced performance and features in connected devices while remaining compatible with the PCI software environment.

The PCIe definition is layered to allow connection to copper, optical and emerging physical signaling media. Its versatility as an attach point is readily apparent: it supports chip-to-chip applications, adapter cards, and graphics I/O bandwidth enhancements, as well as interfacing with other interconnects like USB 2.0, InfiniBand™ Architecture and Ethernet. Its embedded clocking scheme enhances frequency scaling, provides advanced features and enables new physical configurations.

Additional features of PCIe include:

- High per-pin bandwidth, which reduces cost, streamlines board design, increases signal integrity and creates the possibility of smaller physical device forms.
- The ability to scale bandwidth through frequency and/or interconnect width.
- A PCI-compatible software model, meaning that PCIe can be implemented with no operating system, platform configuration or device driver interface changes.
- Advanced features like aggressive power management, QoS, isochrony, hot attach/detach, RAS, and hot-pluggable ExpressModules™ for enterprise.